A document recognition system and its applications

by A. Yamashita
T. Amano
Y. Hirayama
N. Itoh
S. Katoh
T. Mano
K. Toyokawa

This paper describes a document entry system called the Document Recognition System (DRS), which facilitates the conversion of printed documents into electronic form. DRS was developed on a personal computer (PC) with an adapter card for recognizing more than 3000 Kanji characters. It provides a flexible framework for object-oriented management of data and processing modules. The framework allows the user to change the combination of processing modules and to select pipelining (parallel processing) or sequential processing. DRS includes processing modules for layout analysis functions such as blob detection, block segmentation, and model matching, and for character recognition functions such as Kanji character recognition, Japanese postprocessing, postprocessing by a user, and error correction through a user interface. The character recognition functions on the card and the other processing-related recognition functions on the PC work cooperatively in the proposed framework. Within the basic framework, we have customized DRS for practical applications. Examples of successful applications—entry into a text database, creation of an electronic catalog, entry of family registration data, and entry of tag data in a manufacturing process—provide evidence of the processing accuracy and robustness of the framework.

Introduction

Although office automation has advanced rapidly, making it possible to create, send, and receive documents via electronic media, we are still surrounded by a flood of printed information.

Conversion of documents from paper to electronic form is therefore still a difficult problem in constructing a document database. In particular, an application using hypermedia and multimedia databases may have to process not only text data, but also image data (such as pictures in a document) and table data, as well as the physical page layout and logical structure of a document.

In the field of document recognition applications, it is not possible to develop a comprehensive entry system to meet all of the requirements of varied applications. Some applications deal with simple reports, while others involve documents that have complicated layouts. Some require every page to be converted, while others require only parts of documents to be converted. Some may attach priority to processing speed, some to high recognition accuracy, and some to both. In particular, when a large number of documents must be entered, it is necessary to consider not only what kind of processing should be applied, but also how to maximize the quality of each type of processing. This variety of requirements means that customization is indispensable in developing any given entry system for practical applications.

0018-8646/96/$5.00 © 1996 IBM

After the development of the first experimental systems for recognizing documents [1–4], integrated systems were devised for practical applications such as recognizing addresses on postcards and reading letters in typical formats [5–7]. Each research project aimed to develop a system for recognizing a specific type of document.

To meet the various requirements of each application, the structure of an entry system should be a combination of building block modules, and its components should be easy to rearrange.

Our Document Recognition System (DRS) is an integrated, workstation-based document entry system with a flexible framework of processing modules [8]. Since processing modules and data objects are designed in an object-oriented fashion, the combination of functions can easily be modified according to the application, and several processes can run concurrently, allowing the computing resources to be distributed effectively.

DRS provides functions for image capture, layout analysis, character recognition with contextual postprocessing, and error correction through a user interface. All of those functions except image capture and character recognition have been implemented in software which runs on the IBM OS/2® operating system.

The rest of this paper is organized in four sections. The first section presents a framework for cooperative work by the functions. The second section briefly describes the configuration of DRS and explains the functions that it supports. Typical examples of applications are presented in the third section, and the last section summarizes the characteristics of the system.

Framework for cooperative work and extension

DRS is implemented on an IBM PS/55 personal computer, which is compatible with the IBM PS/2® line of personal computers.

A binary page image is captured with 400-dpi resolution by an image scanner. The Japanese OCR processor card, hereafter called the OCR card, is dedicated to recognizing about 3000 Japanese characters, including Kanji, Kana, and alphanumerics. This card consists of a Motorola 68040 microprocessor, memory for an image and recognition dictionary (template), and two large-scale integrated circuits (LSIs) for feature extraction and discrimination. The following subsections describe the software architecture of DRS.

- *Design objective*

During the process of document recognition, elementary data—character strings, characters, lines, and picture elements—are extracted from documents. Each of these physical data objects has a position in the page image, a size, and various attributes.

The relationships among those data, such as "part-of," "next-left," and "under," are often specific to a particular kind of document. How the data objects are queried also depends on which processing module is being applied to them at the request of the application. Sometimes character data in a line are queried, sometimes characters whose height is over a threshold level, and sometimes leaf lines in tree-structured data.

The management and structure of data can vary according to the types of queries that are frequently called by the application. Therefore, we defined the DRS framework according to the following criteria:

- Represent a document by using objects of two kinds: elementary data objects ("entities"), and objects that maintain relationships among the data objects ("maps").
- Provide a unified protocol for calling the various functions, so that these functions ("parts") can be put together for specific purposes.
- Place process control information in data objects.

Entities primarily represent document elements. There are five typical entities: a page image; a character; a blob, which represents a rectangle surrounding separated black pixels; a block, which represents a character line and its parents with "part-of" relations; and a content word, which is produced by Japanese postprocessing.

"Maps" manage the relationships among entities. A map includes the structure of the pointers to certain entities that are generated by processing modules. For instance, an object of this type manages the result of layout analysis, which represents a whole page as a set of tree-structured block data. A "block-tree map" can respond to commands such as "Give all the character lines composing the abstract block" and "Delete two lines from the body block."

"Parts" generate entities and set their attributes; for example, a generated character object has the attributes of position, size, recognition results, and contextual postprocessing results. Parts also investigate relationships among entities in order to generate new maps. Typical parts in DRS are layout analysis modules, character recognition modules, postprocessing modules, and user-interface modules. Execution and parameter setting can be requested by sending appropriate messages to the appropriate combination of one or more parts.

An application program can access entities only by sending appropriate messages to a map. The map receiving a message returns pointers indicating the applicable entities. **Figure 1** shows an example of typical objects of three types and their messages. If some application requires specific management for certain entities or a new process exclusive to the application, a

user can design and add a new map and a new part that can accept the appropriate series of messages.

- *Two types of process flow*

The application program can send messages to parts in order to set entities for processing, set parameters, and start execution. Several parts are defined in such a way that they can accept either a pipeline (parallel) flow or a sequential one.

Figure 2 illustrates these two flows. Figure 2(a) represents the sequential flow of the layout analysis of DRS. In sequential flow, all target entities and a start-message are sent to the first part. Since a user-defined message can be sent from a part to any object when the process is completed, an end-message is returned to the program in this flow. The program then sends the data to the second part and starts it.

Figure 2(b) represents the parallel process of character recognition in DRS. The OCR card has its own microprocessor, and the process of recognizing Japanese characters is executed separately from the host PC. All other processes associated with recognition, such as the postprocessing and the display of results through a user interface, are executed on the PC. For effective use of resources, processes should work cooperatively in a pipeline, especially the card's process for recognizing characters and other processes on the PC.

Each entity has a set of flags representing whether each process should be applied, skipped, or completed. In a pipeline flow, the application program first sets the apply-flags of all related parts to the target entities, and sets the entities as targets of the parts. The program then sends to the first part the same number of step messages as there are entities. Whenever the process for an entity is completed, the first part sets a complete-flag to the entity and sends an end-message and a step-message to the next part. The second part investigates which entities are current targets under conditions in which (a) the previous process is complete; (b) the current process is scheduled to be applied; and (c) the current process has not yet been completed. The part then starts the process for the target entities. By this mechanism, step-messages are repeatedly transferred to the next series of parts.

Since the entity units used for processing are not common to all of the processes, an asynchronous pipeline flow should be applied. For example, the Japanese postprocessing actually selects an optimal sequence of morphemes when it encounters a punctuation mark, and therefore generally processes several character lines at a time, whereas the previous process (character recognition) and the next process (displaying the results through a user interface) process text line by line.

The proposed data-driven protocol resolves the problem. Related processes can work cooperatively, and

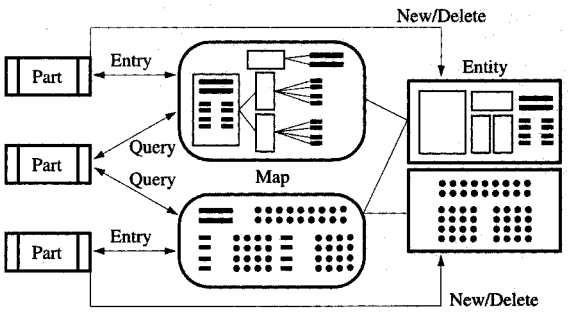


Figure 1

Objects of three types in DRS.

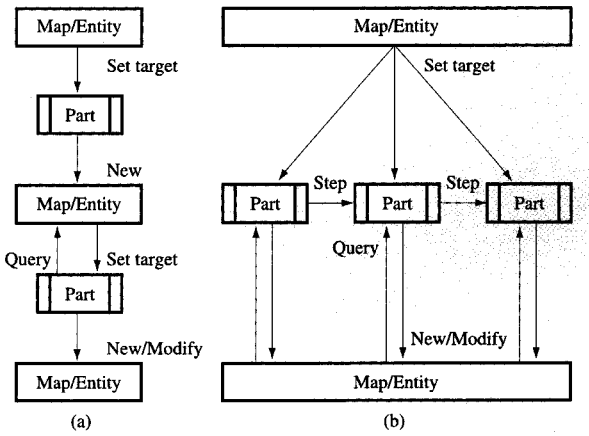


Figure 2

Two process flows of parts in DRS: (a) sequential; (b) pipeline.

each process asynchronously transfers data to the next. The end-message is only a permissive trigger to the next process, which may not in fact start until it has received several end-messages from the previous process. At that time, the next process will execute all possible targets. The protocol has the added advantage that a new part can easily be added and an unused one deleted at any node of the pipeline.

Characteristic parts in DRS

Figure 3 shows the typical process flow of DRS. The processes are roughly separated into two groups: layout analysis and character recognition. In layout analysis, a page image is analyzed so that figure blocks, table blocks,

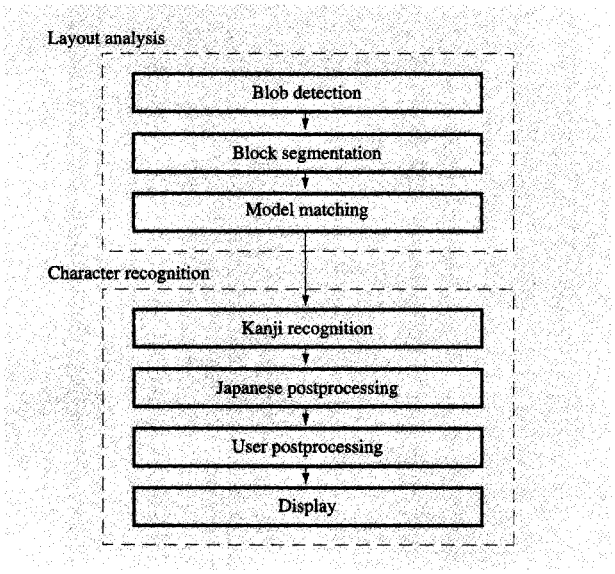


Figure 3

Typical process flow of DRS.

and text blocks can be extracted. When a model for the page is specified, text blocks are further analyzed to extract the block structure defined in the model. The processes work sequentially, as explained in the previous section.

In character recognition, text blocks are recognized and postprocessed, and the results are displayed. All processes work cooperatively in the pipeline, using the proposed mechanism.

Summaries of the algorithms for several parts are given in the following subsections.

- *Blob detection*

A captured image **[Figure 4(a)]** is first segmented into blobs (rectangles on the image) and classified into components of character strings, lines, and picture elements [9]. Later, processing regarding the layout analysis is performed in the domain of these rectangle data.

In the process of raster scanning, smeared run-length data (starting positions and lengths) are generated by replacing short horizontal white runs with black runs. The top and bottom boundaries are detected by comparing run-length data from two vertically consecutive lines. A black run located under a white run is considered to be part of a top boundary. Bottom boundaries can be detected in a corresponding manner. The coordinates of rectangles sandwiched between top and bottom boundaries are calculated and stored in a buffer.

Several adjacent rectangles are integrated into a single component under certain conditions. The smearing technique sometimes mistakenly connects characters to graphics, or to characters in the next line (when an input image is skewed). To avoid such connections, we employ several constraints; for example, rectangles with approximately equal heights can be integrated, while a newly integrated rectangle should not intersect other rectangles. The results of the integration are classified as character lines, horizontal lines, or other objects, according to the height of the rectangle.

- *Block segmentation*

Block segmentation can classify detected blobs into text blocks and figure blocks, and create character lines from blobs [10]. The method does not use any specific model for a page, but automatically segments the page into several blocks with attributes.

First, two histograms are made to represent the height and distance of character strings (character line blobs). The "distance" of character strings means the vertical distance between the baselines of two adjacent character strings.

Since character strings in text areas are arranged regularly, they can be merged into groups by analyzing this regularity. In each histogram, distributed elements have peaks and are classified into several groups. Character lines in a group are expected to have similar height and a regular baseline pitch. Thus, groups are extracted by investigating whether elements have the same peak in the distribution of their distance and height.

The maximum distance and height in a group are used as thresholds for the grouping of character strings. Two adjacent character strings in a group are merged into a text group if their heights and the distance between them are within the threshold. Groups of character strings are then constructed in the image, as shown in **Figure 4(b)**.

In the second step, vertical border lines are drawn along the left and right edges of the text groups. These border lines are extended until they reach the edge of the image or other elements. Horizontal border lines are then drawn at the top and bottom of the vertical border lines. The whole image is segmented into smaller areas by the extended vertical and horizontal border lines. To avoid oversegmentation, if an element of the image lies in two areas, those areas are merged into a single unified block. Thus, the whole image is segmented into several blocks **[Figure 4(c)]**.

If a block includes only character strings, it can be recognized as a text area. However, some blocks include not only character strings but also horizontal lines, vertical lines, and other blobs. A projection-profile method is therefore applied to the unified blocks in order to segment them into text and figure areas. Two kinds of projection—

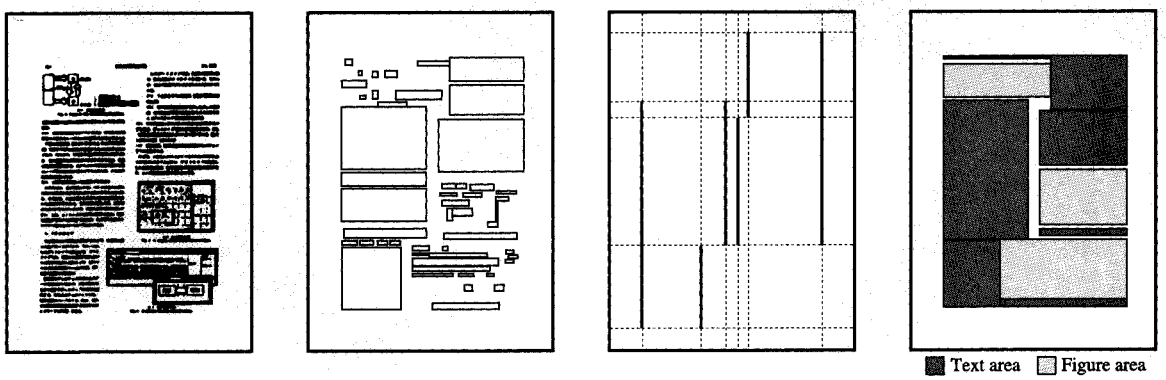


Figure 4

Block segmentation process: (a) sample document image; (b) text groups; (c) a page segmented according to border lines; (d) result of block segmentation.

of character strings (text groups) and of lines and other elements—are created in our approach. From the projection, a unified block is further segmented into text and figure areas. As a result, text areas and figure areas in the columns are detected as shown in **Figure 4(d)**.

- *Model matching*

Form-processing-type applications require that parts of a page image be converted into fields and the specific fields separated for registration. These requirements can be met by adopting a layout model that defines which text blocks should be extracted, and using the model to analyze the page image.

Table 1 shows an example of a layout model for printed forms [11]. The model is declared as a table. The table structure is simple enough to be prepared by using a general text editor.

"Name" is the name of the block. "Num." indicates the minimum and maximum number of lines. "Mark." indicates whether or not a block is defined as a marker. Markers such as running heads and logos are used to detect other target blocks defined in the model. The "X, Y, SX, SY" column shows the x, y-coordinates of a block relative to a marker.

First, marker blocks are extracted; any blobs in the window defined in the model are extracted and integrated into a marker. The absolute coordinates of an extraction window are calculated from the x, y-coordinates of the detected markers and the relative coordinates defined in the model. Character string blobs within the window are then detected, integrated into a character line, and registered as a target block.

Table 1 Example of a layout model.

Nest	*Name*	*Num.*	*Mark.*	*X, Y, SX, SY*
0	Tag	1, 1	N/A	0, 0, 1024, 1024
1	Marker	1, *N*	Yes	200, 100, 800, 80
1	Field-1	1, 10	No	80, 400, 500, 400
1	Field-2	1, 10	No	800, 400, 500, 400
1	Field-3	1, 5	No	160, 800, 800, 200

- *Character recognition*

In response to requests from the PC, characters are recognized by the OCR card. Each character line is first segmented into characters. Each segmented character image is normalized in size to allow extraction of its feature data.

The feature data consist of the distribution of local contour directions and the distance from the frame of the buffer to the nearest black pixel [12, 13].

The local directions are extracted from the normalized 48×48 image by scanning a 2×2 mask pattern. Four directions are assigned according to the mask patterns. The image is divided into twelve partitions in vertical, horizontal, and two diagonal projections. Four local directions are counted in each partition; thus, the number of feature data for local contour directions is 192 (48 partitions $\times$ 4 directions). The local directions are also counted in each direction, and the 192 local directions are reclassified into four summarized feature data.

The image is divided into six partitions in vertical and horizontal projections in order to extract the distance

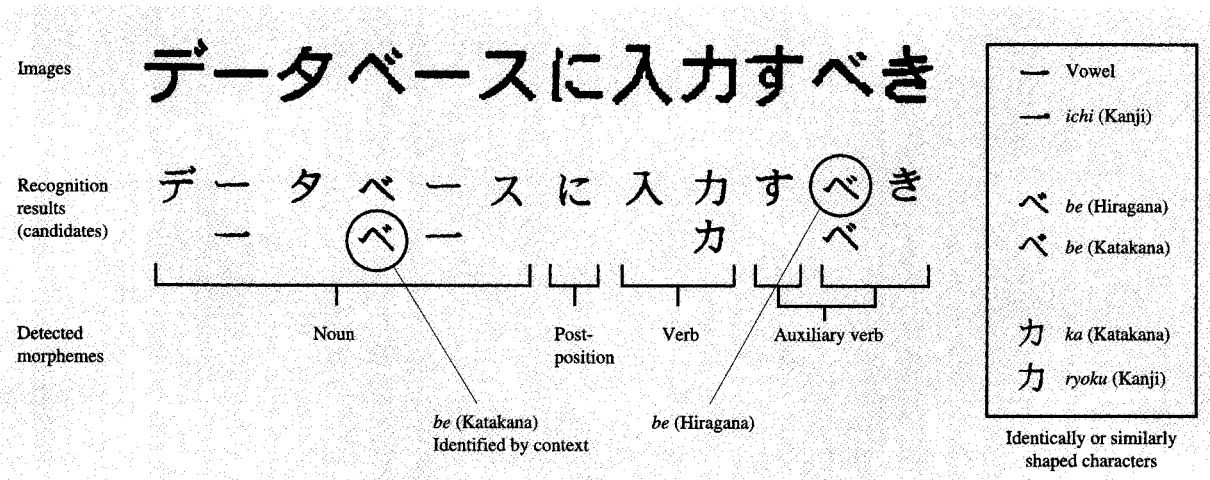


Figure 5

Japanese contextual postprocessing.

feature. The distance from a frame is calculated in a partition, and therefore the number of feature data for the distance is 24 (6 partitions × 4 frames).

The set of extracted feature data is matched with every template in a recognition dictionary, and "city-block" distances are calculated. The four summarized direction features and the 24 distance features are used for preclassification. The 192 local direction features are used to select up to five candidate characters.

- *Japanese contextual postprocessing*

The results obtained by the OCR card are sent to the contextual postprocessing module to improve the recognition accuracy [14]. As shown in **Figure 5**, some Japanese characters have the same shape but different meanings. The character recognition process cannot discriminate among these characters. The postprocessing module selects a probable character sequence from a lattice of recognition candidates by using a Japanese dictionary (containing about 100000 morphemes and constraints on morpheme transitions).

As a result, the two characters in our example are identified as parts of a noun meaning "database" (the first morpheme in Figure 5) and an auxiliary verb meaning "should" (the last morpheme), and the correct character codes are then decided. Through a comparison of the results with the original ones, uncertain characters are detected, and some of them are replaced with candidates that are lower-ranked, but more likely in the context.

In addition, the postprocessing module can detect frequently occurring content words in the selected morpheme sequence. These are good candidates for keywords, which are useful for retrieving documents.

- *User postprocessing and display*

A user can define a specific type of postprocessing, such as a process employing a user dictionary, and apply it to the recognition results in the user postprocessing part. The part obtains the recognition results for a character line, modifies them, and replaces them with new ones. This is a simple process, but it reveals a typical method for adding new parts to the system. If the module can prepare defined messages and can access, via a map, entities such as other recognition modules, it can easily be integrated into the system as a part.

Figure 6 shows a screen on which the recognition results for a page of a report are displayed. Three lines of Japanese are shown in each window. The upper window shows a portion of the captured image (bitmap) and the lower window the related portion of the results (in a particular font). The results are displayed by using outline fonts whose position and size are similar to those of the character images. Thus, the layout of the result with vertical and horizontal spacing is almost identical with that of the original bitmap.

The result in error is displayed by the pop-up correction window, which shows an enlarged character bitmap at the top, the five candidates generated by the recognition part in the middle, and the candidate selected by the postprocessing part at the bottom. DRS notifies the user of which portions should be checked according to the result of the contextual postprocessing (these portions are

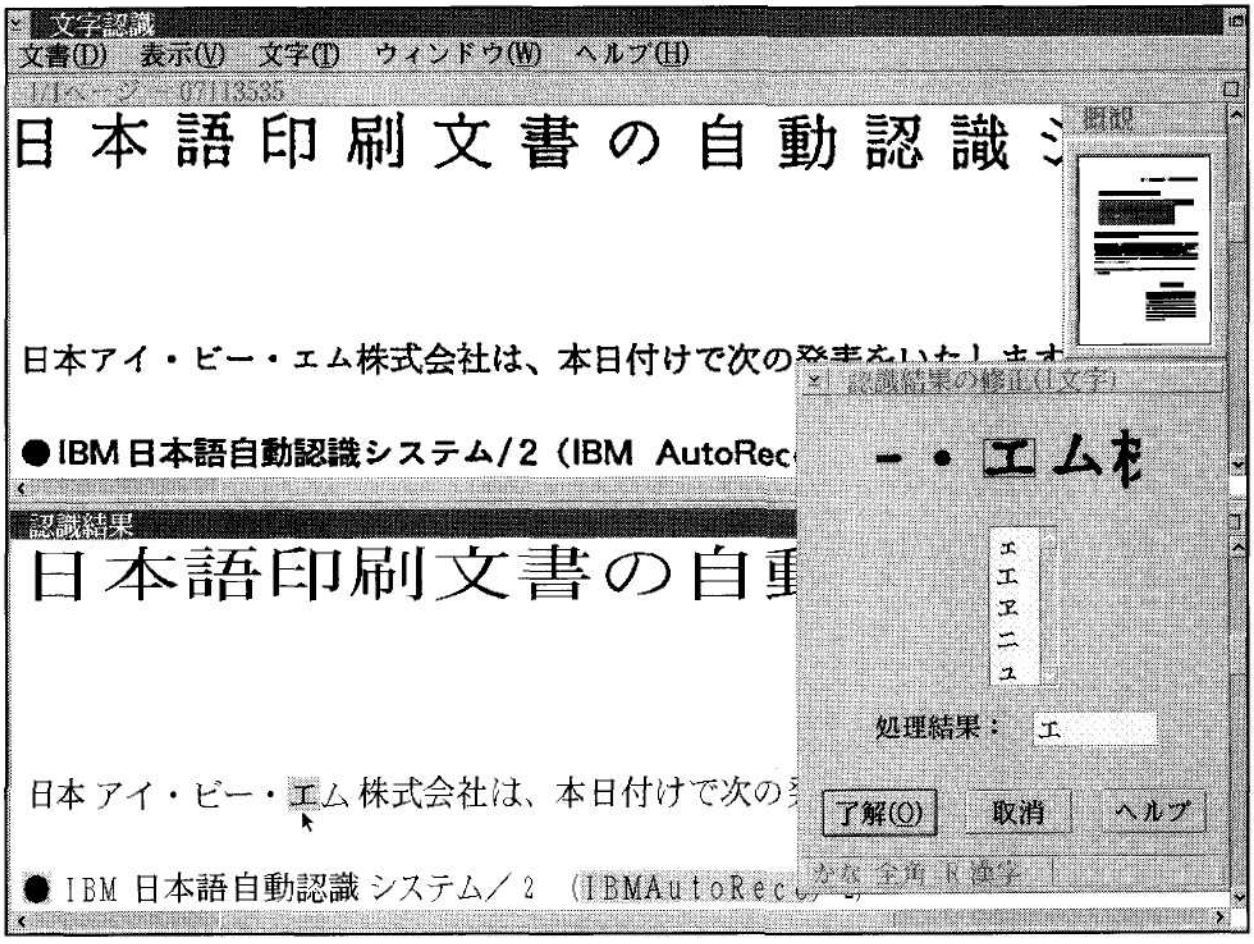


Figure 6

Results on the screen.

indicated by highlighted characters on the screen). In the example, the second candidate is selected as the correct one by the postprocessing part. The user can correct recognition errors in the window by using a mouse or keyboard.

The display-module part can receive step-messages, retrieve processed results, display them on the screen, and send an end-message to the main program. Thus, a user can easily redesign the display part or exchange it for another as necessary, according to the application.

Examples of application

To meet the special requirements of specific applications, one may need to develop new parts or customize existing parts; to accommodate this, the framework of the DRS system is robust and flexible enough to be used for a wide range of applications. We have developed many entry systems based on DRS, which are composed of various combinations of parts on the framework. The following subsections describe several typical systems that offer important proof of the flexibility of the DRS architecture.

Examples of combinations of parts in DRS are shown in **Table 2.** The table shows which parts are used for each application. A circle means that the original general-purpose part is used, a dash that the part is not used, and a description that some appropriate customization is used in the application. The following sections give details of each application.

- *Document database entry*

Creating a document database is the most general application of DRS. Target documents include academic papers, patents, public administration documents, and company reports. In some applications, all of the pages of a document must be converted into text and stored in a text database, while in others, the first several pages,

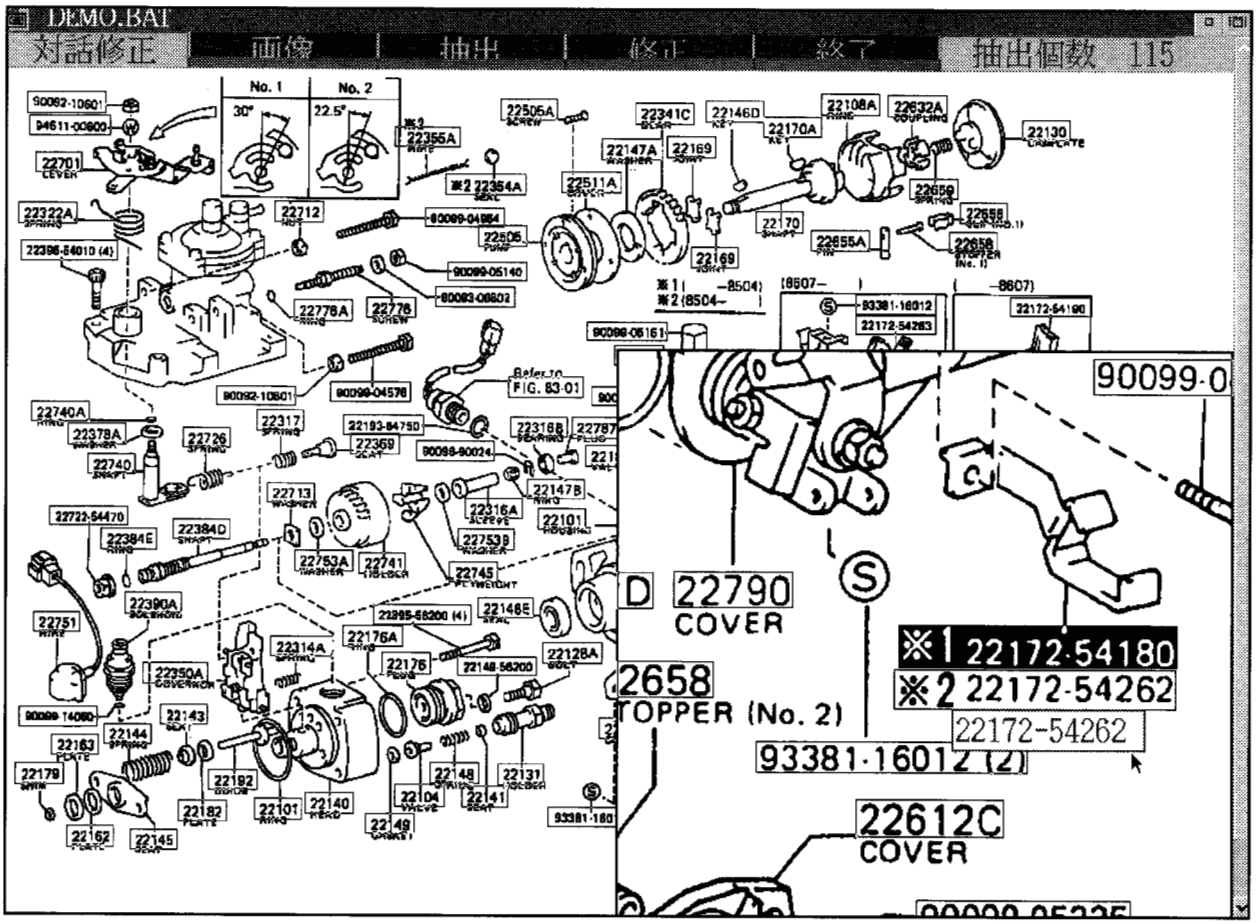


Figure 7

Example from a catalog.

Table 2 Combinations of parts in various applications.

Part	*Application*			
	Document entry	*Electronic catalog*	*Tag recognition*	*Family registration*
Blob detection	○	○	○	○
Block segmentation	○	—	—	—
Model matching	—	—	○	Specific one
Kanji recognition	○	Alphanumeric	Tuned	Tuned, vertical
Postprocessing	Japanese	—	Patterns	City names
User postprocessing	—	Logical check	—	—
User interface	○	Specific interface	—	—

including the index, are converted into text, and all of the page images and some parts of the text to be searched for are stored in an image database.

Since the volume of texts to be recognized is generally large, accurate recognition is a key to success. When many documents of the same kind must be entered, it is effective to customize the template for the fonts used in the documents, and to register frequently occurring "unknown" words in the postprocessing dictionary.

To verify the effectiveness of the layout analysis, we conducted an experiment in which 61 pages from Japanese journals and magazines were tested for block segmentation parts.

The time needed for layout analysis was typically about 10 to 20 seconds. The 61 pages were segmented into 267 text areas and 133 figure areas. On these pages, 249 of 267 text areas (93.3%) and 124 of 133 figure areas (93.2%) were segmented correctly [10].

The character recognition on the card and the postprocessing on the PC were performed at speeds of 45 and 60 characters per second, respectively. Since the processing on the card and on the PC work in parallel, the recognition speed (40 characters per second, including postprocessing) is considerably faster than would be the case for sequential processing (20 characters per second).

- *Electronic catalog creation*

Figure 7 shows an example from an automobile parts catalog (a part of a diagram and, in the inset, the results of processing). A certain motor company has more than 40000 diagrams, which used to be delivered to parts dealers as printed documents. The parts dealers had to search for a necessary part by looking it up in a catalog, then typing the number at an on-line terminal. To make the job easier and to avoid wrong orders caused by mistyping, a new parts-ordering system was developed. In the new system, parts catalogs are delivered as CD-ROMs, which contain compressed diagram images and part numbers associated with their locations. The operator finds the required part in the diagram on the display, and orders it by selecting the part number with the mouse. For these CD-ROMs to be issued, coded data such as part numbers and their locations must be entered with scanned image data. DRS provides semiautomatic entry of the coded data by extracting areas containing part numbers from captured images and recognizing them (extracted part numbers are surrounded by rectangles in Figure 7).

In this case, page structure analysis is not required; instead, the detected character string blobs are recognized and stored. Since part numbers consist of alphanumeric characters, and their arrangement is governed by special rules, the character recognition part is replaced by a software part that can recognize only alphanumeric characters, and special logic is added for checking numbers as a user postprocessing part.

Before actual processing, a test was performed to ascertain the accuracy of part number extraction and character recognition [9]. In the test, 89 actual diagrams created by five different artwork vendors were scanned and tested. The 89 diagrams included 3090 part numbers (each consisting of 6 to 11 alphanumeric characters) of which 2859 (92.5%) were correctly extracted and recognized. Of the remaining 7.5%, 0.9% were not extracted at all, 3.8% were extracted incorrectly, and 2.8% included at least one recognition error. Missing and incorrect extractions of part numbers were caused mainly by discontinuities in smeared runs to be connected. The diagrams include characters of various sizes and pitches which sometimes impinge on graphical part figures, with the result that separated portions of part numbers and part numbers with added noise were neglected or extracted incorrectly.

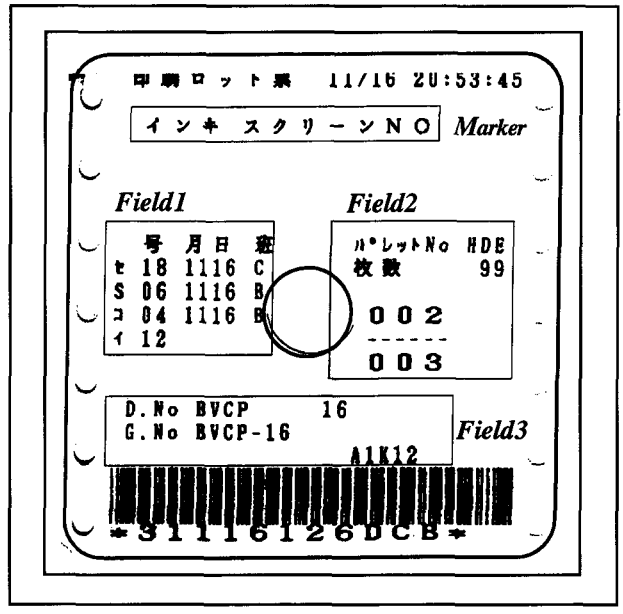


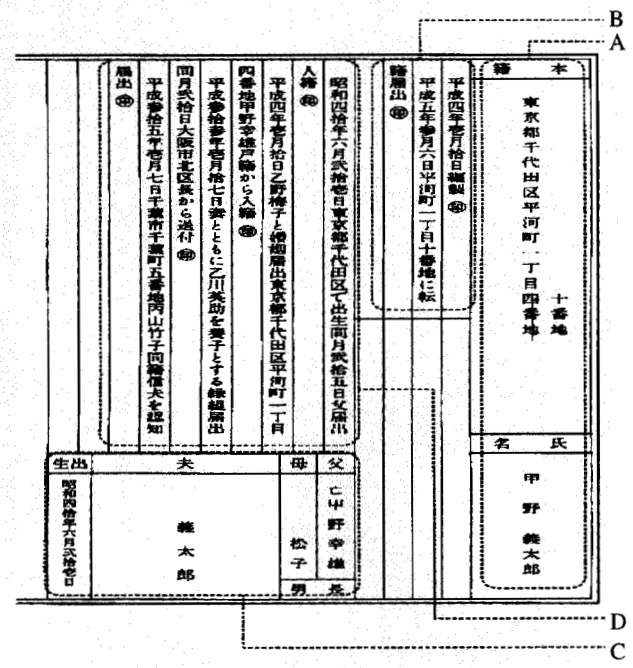
Figure 8

Example of a CD lot-tag.

A character string displayed in reverse video indicates that it does not obey the format rules. The operator can cancel character strings other than part numbers by clicking a mouse, specify a new area for a part number by dragging a mouse, and modify the recognized results by keyboard input.

- *Tag recognition on a manufacturing line*

Figure 8 shows an example of a printed tag attached to a set of music CDs at a certain factory. The tag includes information such as the number of CDs produced, which machines were used, and the customer that ordered them. The set of CDs is transferred together with the tag for subsequent processing, verification testing, label printing, and packaging. In these processes, data necessary for operations are obtained from the tag and the host database. Certain indexes from the tag are re-entered into the computer to query the host database, and a new



A	Title	Form date Legal domicile Head of a family
B	Item (transfer)	Transfer date Old domicile
C	Personal info.	Sex Birth date Relations Father's name Mother's name
	Name	Kana name Kanji name
D	Identity (birth)	Birth place Birth date Reported by

Figure 9

Example from a family register.

printed form indicating the next operation is printed. The manufacturers decided to change from the conventional process to automatic data entry using DRS. DRS recognizes several hundred tags a day and transfers data to the printing program for the new forms. The printing program requests information from the database by using the recognized index in the tag, and both the data recognized by DRS and those requested from the database are merged and printed.

To meet the requirements of the application, we defined a layout model to recognize three portions of a tag. (A marker and three fields are defined in the model, as shown in Figure 8). When DRS is integrated into the flow of daily operations, the number of characters to be recognized is generally not very large; however, high recognition accuracy is indispensable if interactive verification is to be eliminated. We replaced the original Kanji template with one tuned for the printing fonts used in the tags, and replaced the word dictionary with the number patterns appearing in the tags. By using postprocessing with the pattern dictionary, the postprocessor can rectify substitution errors involving similarly shaped characters such as B and 8; O, D, and 0; G and 6.

An experiment was performed for 50 CD lot-tags, which included 150 fields and 550 character strings to be recognized. Of these tags, 80% were correctly processed without any error correction. The errors in the remaining cases were caused by misidentification of fields (8%) and misrecognition of character strings (12%), because of darts and blurred printing. These errors are checked and corrected in subsequent processes on the system.

• *Family registration data entry*

Figure 9 shows a copy of a family register. Each Japanese citizen is legally required to have his or her name entered in a family register. Whenever a birth, marriage, or death occurs, a family member must report it to the local government so that the family register can be updated. The problem until recently was that all registers were handled in the form of paper documents.

In June 1994, the Japanese government amended the law for family registration to permit registers to be handled as electronic documents. We customized DRS to

create an entry system for family registration, which can take microfilms of printed pages, capture image data from microfilms, convert them into character data, parse sentences to extract data items, and register them in a database.

We have developed a new layout model and an analysis module for this application. In the model, a form is defined as a combination of vertical and horizontal lines. Each line is recorded in the model, along with its position and size. Several models can be described in a profile. In this case, six models were defined in the profile. The module first detects lines on the basis of blobs and searches for an appropriate model in the profile, and then extracts fields, which are also defined in the model as rectangles, by using relations between ruled lines and field edges. As shown in the figure, sentences are printed vertically, whereas layout analysis modules assume that character lines are printed horizontally. Therefore, all page images are rotated 90° and input to DRS, which analyzes the layout of the rotated images. Before character recognition, a segmented image of a character is rotated through 270° and re-stored in the original direction. Since ruled lines sometimes touch or intersect characters, they are eliminated before recognition. The template was newly generated for printed fonts, and the word dictionary was also regenerated to include specific words and names of cities.

The results of our experiment showed that the layout was correctly analyzed for 305 of 317 pages (96.2%) and that 5100 of 5527 Japanese characters (92.3%) were correctly recognized [15]. Recognition errors were caused primarily by overlaps between seal impressions and characters, and by handwritten annotations.

After an initial trial in which 10000 registers were entered, all of the data contained in 137000 registers (a register consists of three to four pages on average) in a certain Tokyo ward were entered by the end of 1994. The target registrations included both typed (50%) and handwritten (50%) forms, and about 75% of the typed forms were processed by using the system. The results were registered in the database after several manual correction and examination processes, along with other text data manually entered from handwritten forms.

Concluding remarks

The Document Recognition System that we have developed has a flexible framework for customization. DRS manages data, the relationships among data, and processing modules as independent objects, and the framework allows processing modules to work sequentially or in a pipeline. Consequently, it is easy to replace the original modules with enhanced ones, and to change the combination of modules, the flow, and the style of working according to the requirements of a particular application.

In this paper, we have introduced several specific applications and described how DRS can be used in practical entry systems. These examples demonstrate the robustness of the flexible framework, which allows DRS to be customized for various types of application.

We are continuing to enhance DRS by developing new processing modules, expanding the range of applications, and improving the quality of the existing processing modules.

Acknowledgment

We wish to thank Yuka Tateishi, Yoshiharu Katoh, Yoshinao Kobayashi, and Hiroyasu Takahashi for their advice and cooperation in developing the system. We also thank Toyota Motor Corporation, the Victor Company of Japan, Ltd., and the Toshima Ward Office in Tokyo for their cooperation in providing examples of its use.

References

1. K. Y. Wong, R. G. Casey, and F. M. Wahl, "Document Analysis System," *IBM J. Res. Develop.* **26,** No. 6, 647–656 (1982).
2. H. Kida, O. Iwaki, and K. Kawada, "Document Recognition System for Office Automation," *Proceedings of the 8th International Conference on Pattern Recognition*, IAPR (International Association for Pattern Recognition), Paris, 1984, pp. 446–448.
3. K. Inagaki, T. Kato, T. Hiroshima, and T. Sakai, "MACSYM: A Hierarchical Parallel Image Processing System for Event-Driven Pattern Understanding of Documents," *Pattern Recogn.* **17,** No. 1, 85–108 (1984).
4. I. Masuda, N. Hagita, T. Akiyama, T. Takahashi, and S. Naito, "Approach to Smart Document Reader System," *Proceedings of the Conference on Computer Vision and Pattern Recognition*, IEEE, San Francisco, 1985, pp. 550–557.
5. S. N. Srihari, "From Pixels to Paragraphs: The Use of Contextual Models in Text Recognition," *Proceedings of the International Conference on Document Analysis and Recognition*, IAPR, Tsukuba Science City, Japan, 1993, pp. 416–423.
6. J. Kreich, "Robust Recognition of Documents," *Proceedings of the International Conference on Document Analysis and Recognition*, IAPR, Tsukuba Science City, Japan, 1993, pp. 444–447.
7. T. A. Bayer, "Understanding Structured Text Documents by a Model Based Document Analysis System," *Proceedings of the International Conference on Document Analysis and Recognition*, IAPR, Tsukuba Science City, Japan, 1993, pp. 448–453.
8. T. Amano, A. Yamashita, N. Itoh, Y. Kobayashi, S. Katoh, K. Toyokawa, and H. Takahashi, "DRS: A Workstation-Based Document Recognition System for Text Entry," *IEEE Computer* **25,** No. 7, 67–71 (1992).
9. T. Amano, A. Yamashita, and H. Takahashi, "A Character String Detection Algorithm Using Horizontal Boundaries and Its Application to a Part Number Entry System," *Proc. SPIE* **1452,** 330–336 (1991).
10. Y. Hirayama, "A Block Segmentation Method for Document Images with Complicated Column Structures," *Proceedings of the International Conference on Document Analysis and Recognition*, 1993, pp. 91–94.

11. A. Yamashita, T. Amano, H. Takahashi, and K. Toyokawa, "A Model Based Layout Understanding Method for the Document Recognition System (DRS)," *Proceedings of the International Conference on Document Analysis and Recognition*, IAPR, Saint Malo, France, 1991, pp. 131–138.
12. Y. Nakamura and H. Takahashi, "Character Recognition Apparatus," *IBM Tech. Disclosure Bull.* **28,** No. 9, 3990–3992 (1986).
13. H. Takahashi, "A Simple Recognition Method for Handwritten Kanji Characters by Using Primitive Connective Directions of Thinning," *Trans. Inst. Electron. Info. & Commun. Eng.* **PRL82-8,** 57–62 (1982) (in Japanese).
14. N. Itoh and H. Maruyama, "A Method of Detecting and Correcting Errors in the Results of Japanese OCR," *Trans. Info. Proc. Soc.* **33,** No. 5, 664–670 (1992) (in Japanese).
15. T. Amano, K. Toyokawa, T. Mano, and S. Toriyama, "A Document Image Analysis and Recognition System for Japanese Family Registration," *Proceedings of the Asian Conference on Computer Vision*, IEEE, Singapore, 1995, Vol. III, pp. 373–377.

Received August 24, 1994; accepted for publication November 8, 1995

Akio Yamashita *IBM Research Division, Tokyo Research Laboratory, 1623-14 Shimotsuruma, Yamato-shi, Kanagawa-ken 242, Japan (YAMASITA at TRL, yamasita@trl.ibm.co.jp).* Mr. Yamashita received the B.E. and M.E. degrees in electrical engineering from the University of Tokyo in 1983 and 1985, respectively. In 1985 he joined IBM Japan Ltd., where he is currently a member of a pattern recognition group at the Tokyo Research Laboratory. His research interests include pattern recognition, image processing, and document understanding. Since joining IBM, he has conducted research on alphanumeric OCR, model-based layout analysis, and printed Kanji OCR (DRS).

Tomio Amano *IBM Research Division, Tokyo Research Laboratory, 1623-14 Shimotsuruma, Yamato-shi, Kanagawa-ken 242, Japan (AMANO at TRL, amano@trl.ibm.co.jp).* Mr. Amano received the B.E. and M.E. degrees from Keio University in 1982 and 1984, respectively. In 1984 he joined IBM Japan Ltd., where he is a research staff member at the Tokyo Research Laboratory. He is a member of a pattern recognition group, where his research interests include pattern recognition, document image processing, and user interface. Since joining IBM, he has conducted research on OCR and block segmentation of document images.

Yuki Hirayama *IBM Research Division, Tokyo Research Laboratory, 1623-14 Shimotsuruma, Yamato-shi, Kanagawa-ken 242, Japan (HIRAYAMA at TRL, hirayama@trl.ibm.com).* Mr. Hirayama received the B.E. degree in mechanical engineering and the M.E. degree in information engineering from Tokyo University in 1989 and 1991, respectively. In 1991 he joined IBM Japan Ltd., where he is a member of a pattern recognition group at the Tokyo Research Laboratory, working on layout analysis for OCRs.

Nobuyasu Itoh *IBM Research Division, Tokyo Research Laboratory, 1623-14 Shimotsuruma, Yamato-shi, Kanagawa-ken 242, Japan (ITON at TRL, iton@trl.ibm.com).* Mr. Itoh received the B.E. and M.E. degrees in biological engineering from Osaka University in 1982 and 1984, respectively. In 1984 he joined IBM Japan Ltd., where he was a member of a pattern recognition group at the Tokyo Research Laboratory, working on linguistic postprocessing for OCRs and on-line handwriting recognition. He is currently on the staff of a user interface group, and his primary research interest is in language modeling for Japanese speech recognition.

Shin Katoh *RIOS Systems Co., Ltd., 90-6 Kawakami-cho, Totsuka-ku, Yokohama-shi, Kanagawa-ken 244, Japan (Kato_Shin@notes.rios-systems.co.jp).* Mr. Katoh received the B.E. and M.E. degrees in chemical engineering from the Tokyo Institute of Technology in 1978 and 1980, respectively. In 1982 he joined IBM Japan Ltd., where he was a member of a pattern recognition group at the Tokyo Research Laboratory until 1995, working on Kanji OCR systems and on-line handwriting recognition systems. He is now on temporary assignment with RIOS Systems, a subsidiary firm of IBM Japan, where he works on the planning of OCR and image products for the consumer market.

Takashi Mano *Software Development, Yamato Laboratory, 1623-14 Shimotsuruma, Yamato-shi, Kanagawa-ken 242, Japan (JL05269 at YMTVM4).* Dr. Mano received the B.S., M.S., and Ph.D. degrees in mathematics from Sophia University, Tokyo. In 1984, he joined IBM Japan Ltd., where he has worked in areas related to image processing and pattern recognition. In 1992, he worked with the Tokyo Research Laboratory research staff to develop the system for family registration data entry.

Kazuharu Toyokawa *IBM Research Division, Tokyo Research Laboratory, 1623-14 Shimotsuruma, Yamato-shi, Kanagawa-ken 242, Japan (TOYOKAWA at TRL, toyokawa@trl.ibm.co.jp).* Dr. Toyokawa received the B.S., M.S., and Ph.D. degrees in physics from Osaka University, Osaka, Japan, in 1969, 1971, and 1974, respectively. From 1974 to 1975, he was a research fellow at the Yukawa Foundation at Osaka University; in 1976, he was a research fellow at the Broadcast Science Research Laboratory at the Japan Broadcast Corporation, NHK. In 1977 he joined IBM Japan Ltd., where he has worked in areas related to thin films, display technology, image processing, and pattern recognition. From 1982 to 1984, he was an international assignee at the IBM Thomas J. Watson Research Center in Yorktown Heights, New York, where he conducted research related to data compression. He is a research staff member at the IBM Tokyo Research Laboratory and currently manages a group doing research in areas related to image processing and pattern recognition.