ENCYCLOPEDIA OF COMPUTER SCIENCE AND TECHNOLOGY

EXECUTIVE EDITORS

Allen Kent *James G. Williams*

UNIVERSITY OF PITTSBURGH
PITTSBURGH, PENNSYLVANIA

ADMINISTRATIVE EDITORS

Carolyn M. Hall *Rosalind Kent*

PITTSBURGH, PENNSYLVANIA

VOLUME 31
SUPPLEMENT 16

MARCEL DEKKER, INC. NEW YORK • BASEL • HONG KONG

DESIGN, COLLECTION, AND ANALYSIS OF HANDWRITING SAMPLE IMAGE DATABASES

INTRODUCTION

A handwriting sample image database is comprised of two major types of data (images and reference classifications). The images represent a specific application where a computer is used to automatically convert the pixel data in the image into ASCII data for further computer processing. In the case of automated character recognition, these images include credit card slips, checks, insurance claims, tax forms, and so on. These types of systems greatly reduce the amount of human labor required to enter information into computer databases.

Handwriting sample image databases also contain reference classifications for use in training and testing recognition systems. In May 1992, the First Census Optical Character Reconsign Systems Conference was hosted by the National Institute of Standards and Technology (NIST) under the sponsorship of the Bureau of the Census (1). At this conference, 47 different character recognition systems were evaluated as to how well they could classify images of individual characters including digits and uppercase and lowercase letters. Of the 47 systems, 23 achieved error rates between 3% and 5% on digits. These performance levels are sufficient for the technology to be economically advantageous and these systems appear to be approaching near-human performance. System developers agree that improving the performance of these systems will require an enormous amount of effort and an enormous increase in the number of images used to train these systems. This is why handwriting sample image databases are so important, and without reference classifications assigned to each image in the database, these types of system evaluations and future system improvements are not possible.

The next section discusses the design and collection of *NIST Special Database 1*. The third section describes the production of segmented character databases *NIST Special Database 3* and *NIST Special Database 7*. A method for machine-assisted labeling of segmented character images is presented. The fourth section introduces a method for measuring the complexity of large collections of handwriting.

DESIGN AND COLLECTION OF *NIST SPECIAL DATABASE 1*

In 1988, the Image Recognition Group at NIST undertook a project sponsored by the Bureau of the Census to design and collect a large database of handprinted characters. The database was designed to be used in training and testing high-speed high-throughput character recognition engines. *NIST Special Database 1* (SD1) (2) contains 2100 full-page images of handwriting samples printed by 2100 different writers geographically distributed across the United States with a sampling roughly proportional to population

density. The writers used in this collection were permanent Census field representatives experienced in filling out forms.

Database Content

Each of the 2100 pages in the database is an image of a structured form filled in by a unique writer. A field template specifying the number of entry fields, their size, and location was used. An image of one of the blank forms used in the database is shown in Figure 1. The form is comprised of 3 identification boxes, 28 numeral boxes, 2 alphabetic boxes, and 1 unconstrained text paragraph box. This structured form layout provides a total character count of over 1,000,000 characters in the database; about 300,000 numerals and 700,000 alphabetic characters. In addition to the primary form images, 33 isolated subimages of the boxes on each primary page, excluding the name field, are included, accounting for 71,400 individual images in the entire database. With an individual form image requiring approximately 1 MB of memory, the total image database, in uncompressed form, occupies approximately 3 GB of mass storage. Therefore, the images are 2-dimensionally compressed in accordance with CCITT Group 4 (3,4), reducing the overall size of the database to under 700 MB.

Handwriting Sample Form Layout

Figure 2 displays an actual form from the database. Each entry field on this form is represented as a box. The writers in the database have been made anonymous by blacking out the name field. The string of machine-printed information above each box instructed the writer what to print in the box. The instructions on the form requested that the writer print within the box the information provided above each box. Assuming the writer followed the directions and correctly completed the form, each box is self-referenced. This method of collection reduces the overall cost incurred by eliminating the need for transcribing the printed samples by hand. The instructions do not specify what writing implement should be used. Therefore, the database contains a random assortment of pencils and pens resulting in handwriting samples varying in width, contrast, and color.

Careful planning went into the design of this form. The form strategy applied was developed to ensure successful data capture based on current forms processing techniques. Every field is consistently defined as a bold box explicitly defining the location and spatial extent of each field. The single-line boxes are 7 mm in height, giving writers ample room to fit entire characters within the box. This served to minimally constrain the writer's print but more importantly aids the automated field isolation within a recognition system. By using a consistent field demarcation such as a rectangular box, a single software or hardware solution can be implemented to locate every field on the form. The boxes on this form are maximally spaced in an attempt to minimize crowding and clutter. The more cluttered a form layout, the more difficult it becomes for a computer to locate and identity fields, thereby increasing the potential for recognition failures. This implies a trade-off between minimizing the amount of paper handled by increasing the amount of data entered on each page versus lower recognition rates due to increased clutter and increased recognition confusion.

Ignoring the first three identification boxes shown in Figure 2, as one scans down the form, there is a progression of increasing recognition difficulty. The first series of boxes are comprised of digits only, followed by boxes comprised of alphabetic characters.

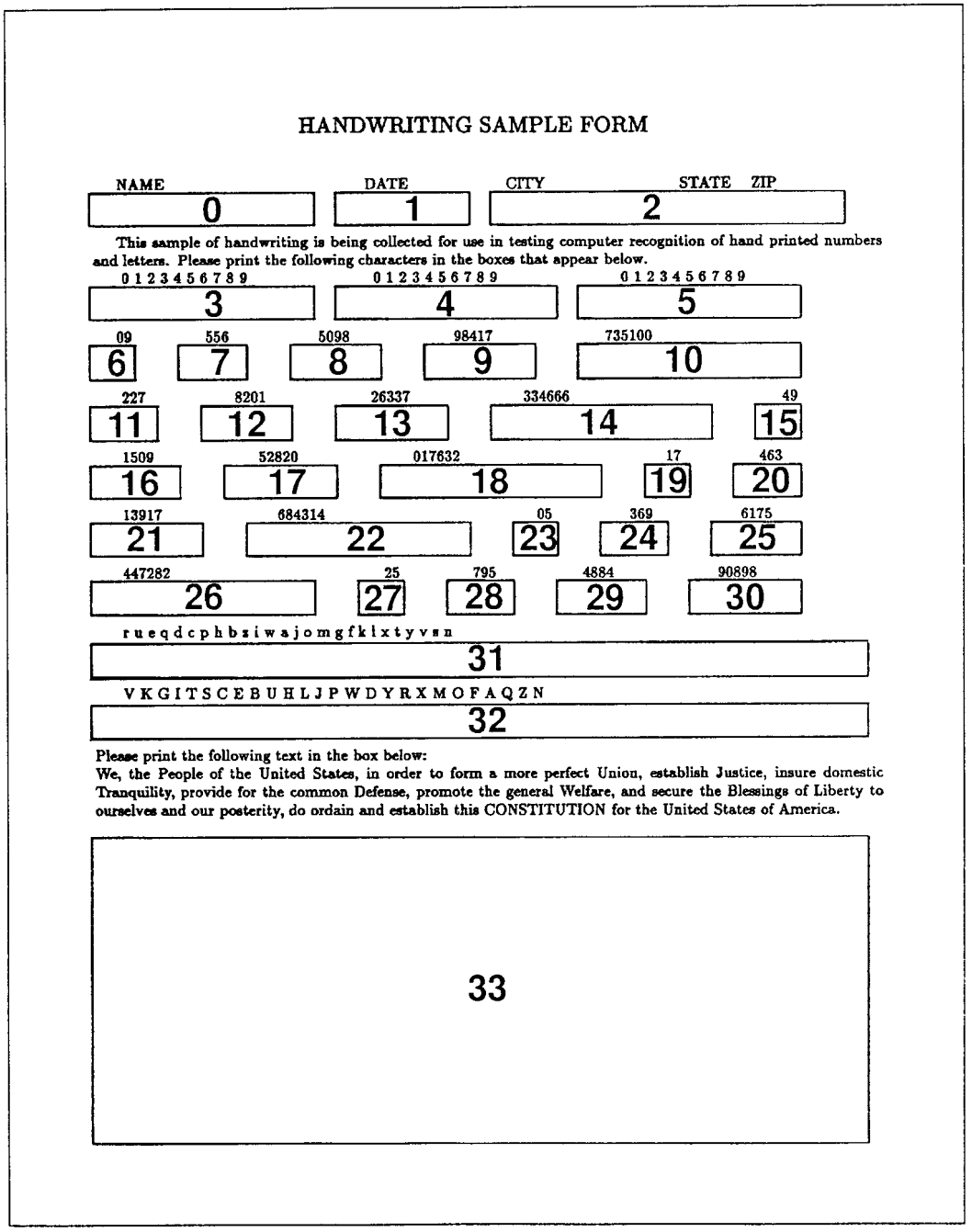
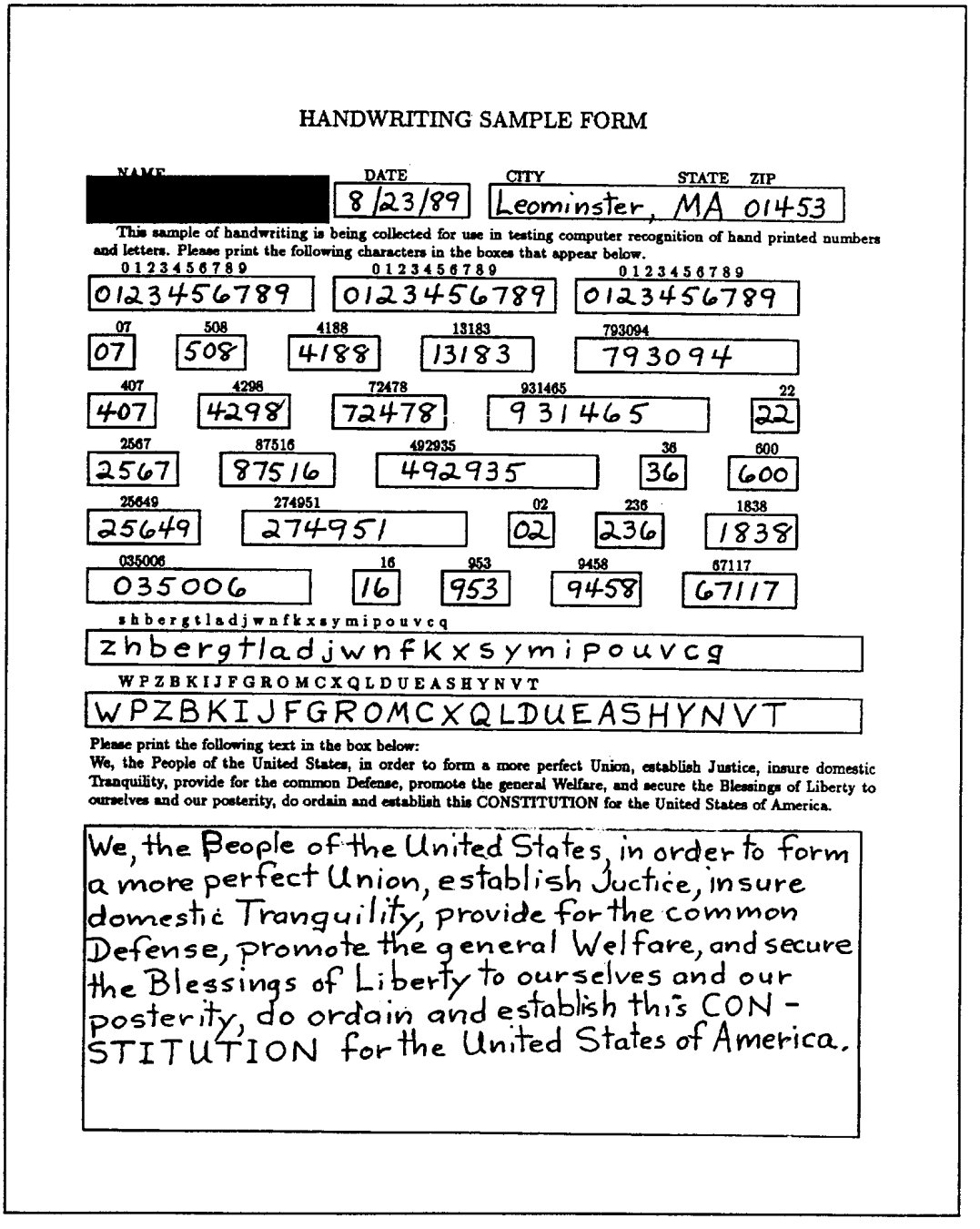


FIGURE 1 The Handwriting Sample Form is comprised of 34 indexed entry field boxes.



HANDWRITING SAMPLE FORM

NAME [redacted] DATE 8/23/89 CITY Leominster, STATE MA ZIP 01453

This sample of handwriting is being collected for use in testing computer recognition of hand printed numbers and letters. Please print the following characters in the boxes that appear below.

0 1 2 3 4 5 6 7 8 9

0123456789 | 0123456789 | 0123456789

07 | 508 | 4188 | 13183 | 793094

07 | 508 | 4188 | 13183 | 793094

407 | 4298 | 72478 | 931465 | 22

407 | 4298 | 72478 | 931465 | 22

2567 | 87516 | 492935 | 36 | 600

2567 | 87516 | 492935 | 36 | 600

25649 | 274951 | 02 | 236 | 1838

25649 | 274951 | 02 | 236 | 1838

035006 | 16 | 953 | 9458 | 67117

035006 | 16 | 953 | 9458 | 67117

s h b e r g t l a d j w n f k x s y m i p o u v c q

zhbergtladjwnfkxsymipouvcq

W P Z B K I J F G R O M C X Q L D U E A S H Y N V T

WPZBKIJFGROMCXQLDUEASHYNVT

Please print the following text in the box below:
We, the People of the United States, in order to form a more perfect Union, establish Justice, insure domestic Tranquility, provide for the common Defense, promote the general Welfare, and secure the Blessings of Liberty to ourselves and our posterity, do ordain and establish this CONSTITUTION for the United States of America.

We, the People of the United States, in order to form a more perfect Union, establish Juctice, insure domestic Tranquility, provide for the common Defense, promote the general Welfare, and secure the Blessings of Liberty to ourselves and our posterity, do ordain and establish this CON - STITUTION for the United States of America.

FIGURE 2 A completed form containing very legible and neat handprint.

There are only 10 unique classes of digits, 0 through 9, versus 26 possible classes of the alphabetic characters, A through Z. The reduced size of possible classes makes the recognition of numeric character fields easier than the recognition of alphabetic fields which, in turn, are easier to recognize than alphanumeric fields. There also is a progression down the form of increased character segmentation difficulty. The segmentation of lowercase characters is challenging because extenders on the characters, g, j, p, q, and y, often extend beyond the bottom of the box. The Constitution box, the last box on the form, pushes the outer limits of current segmentation and recognition technology due to the handwriting being unconstrained; no specific line breaks designated, no form lines to guide the writer left to right, no form lines to constrain the height of the characters, and so on.

There are 50 variations of the form layout in the database. As stated above, a single field template was used so that all 2100 forms contain the same number of boxes, each of the same size and relative location. The variations are realized in the information provided above each box. Every form requested that the writer print the sequence of digits, 0 through 9, three times in boxes 3, 4, and 5. Depending on the form variation, the digits in boxes 6 through 30 vary; however, the number of digits in each box remain fixed. The variations in forms provide 50 different random orders of the lowercase alphabet and 50 different random orders of the uppercase alphabet across the 2100 forms.

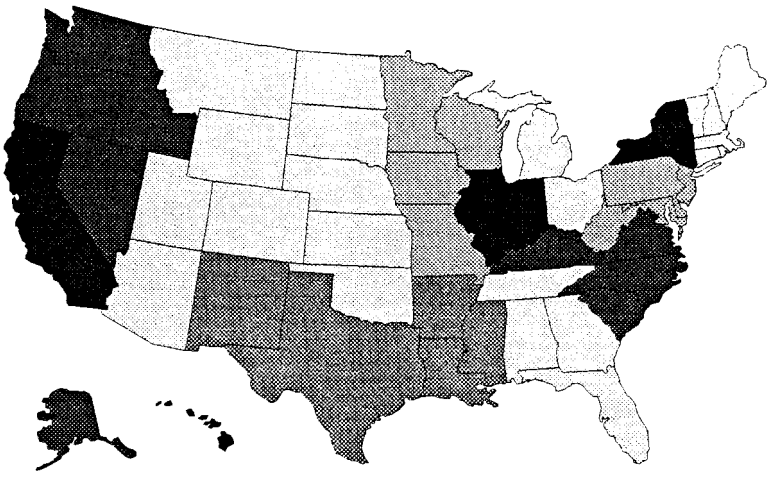
Database Acquisition

The 50 form variations were tightly specified using a typesetting software package and printed on a laser printer. The 50 templates were then massively reproduced with a photocopier. From copies of the original 50 variations, 3520 blank forms were mailed to 12 regional offices within the Bureau of the Census. There, the forms were filled out by Census field representatives and returned via business return envelopes. This process greatly reduced administrative and mailing overhead expenses while providing a sampling roughly proportional to geographic population distributions within the United States. Figure 3 illustrates the 12 different census regions.

Out of 3520 forms mailed to regional offices, 2100 completed forms were returned in time to be included in the database. The number of forms mailed to each regional office and the number of completed forms included in the database from that region are listed in Figure 4. Region 0 represents forms in which the field containing city, state, and zip code was left empty. From August 1989 through October 1989, the forms received at NIST were sequentially indexed, sorted by region, logged, and digitized. Figure 5 lists the information recorded in the Historical Log provided with the database. Included is the form identification index, the form variation type (1 of 50), the date received and processed at NIST, the assumed writing implement used in completing the form, the color of the implement's ink or lead, and a subjective quality rating.

IHead Image File Format

Image file formats and effective data compression and decompression are critical to the usefulness of these types of databases. Each page returned was digitized in binary form at 12 pixels per millimeter [300 dots per inch (dpi)], 2-dimensionally compressed using CCITT Group 4, and temporarily archived onto computer magnetic mass storage. Once all forms were digitized, the images were mastered and replicated onto ISO-9660 formatted CD-ROM discs for permanent archiving and distribution.



Boston, MA	Chicago, IL	Atlanta, GA
New York, NY	Kansas City, KS	Denver, CO
Philadelphia, PA	Seattle, WA	Dallas, TX
Detroit, MI	Charlotte, NC	Los Angeles, CA

FIGURE 3 The Bureau of the Census' geographical regions within the United States.

In this application, a raster image is a digital encoding of light reflected from discrete points on a scanned form. The 2-dimensional area of the form is divided into discrete locations according to the resolution of a specific grid. Each cell of this grid is represented by a single bit value 0 or 1 called a pixel; 0 represents a cell predominantly white, 1 represents a cell predominately black. This 2-dimensional sampling grid is then stored as a 1-dimensional vector of pixel values in raster order; left to right, top to bottom.

After digitization, certain attributes of an image are required to correctly interpret the 1-dimensional pixel data as a 2-dimensional image. Examples of such attributes are

Region	Office	# Mailed	# In SD1
0	Anonymous	0	12
1	Boston	310	152
2	New York	250	51
3	Philadelphia	350	196
4	Detroit	250	158
5	Chicago	220	126
6	Kansas City	240	152
7	Seattle	270	121
8	Charlotte	350	188
9	Atlanta	300	232
10	Dallas	350	241
11	Denver	400	250
12	Los Angeles	230	221
		3520	2100

FIGURE 4 Forms mailed to each region and included in *NIST Special Database 1*.

REGIONAL OFFICE 1
NAME: BOSTON, MASS
OFFICE CODE NO.: 2100
MAILED: 310 pieces

	INDEX	TMPLT	DATE RECEIVED	WRITING TOOLS	COLOR	QUALITY RATING
1)	0817	40	08-29-1989	PENCIL	BLACK	MEDIUM
2)	0818	03	08-29-1989	BALL POINT	BLACK	LIGHT
3)	0819	13	08-29-1989	PENCIL	BLACK	LIGHT
4)	0852	40	08-29-1989	FELT TIP PEN	BLACK	MEDIUM
5)	0855	46	08-29-1989	PENCIL	BLACK	LIGHT
6)	0869	20	08-29-1989	PENCIL	BLACK	LIGHT
7)	0872	30	08-29-1989	BALL POINT	BLACK	MEDIUM
8)	0874	26	08-29-1989	BALL POINT	BLUE	LIGHT
9)	0889	48	08-29-1989	BALL POINT	BLACK	LIGHT
10)	0891	44	08-29-1989	PENCIL	BLACK	DARK
11)	0892	23	08-29-1989	PENCIL	BLACK	MEDIUM
12)	0895	48	08-29-1989	PENCIL	BLACK	LIGHT
13)	0896	44	08-29-1989	PENCIL	BLACK	MEDIUM
14)	0897	19	08-29-1989	FELT TIP PEN	BLACK	DARK

FIGURE 5 A portion of the Historical Log provided in *NIST Special Database 1*.

the pixel width and pixel height of the image. These attributes are stored in a machine-readable header prefixed to the raster bit stream. A program that manipulates the raster data of an image is able to first read the header containing these attributes and determine the proper interpretation of the data that follows it.

NIST has designed, implemented, and distributed images based on this paradigm. A header format named IHead has been developed for use as an image interchange format. Numerous image formats exist; some are widely supported on small personal computers, others supported on larger workstations; most are proprietary formats, few are public domain. IHead is an attempt to design an open image format which can be universally implemented across heterogeneous computer architectures and environments. IHead has been successfully ported and tested on several systems including UNIX workstations and servers, DOS personal computers, and VMS mainframes. Both documentation and source code for the IHead format are publicly available. IHead has been designed with an extensive set of attributes in order to (1) adequately represent both binary and gray level images, (2) represent images captured from different scanners and cameras, (3) and satisfy the image requirements of diversified applications, including but not limited to image archival/retrieval, character recognition, and fingerprint classification.

The IHead structure definition written in the C programming language is listed in Figure 6; Figure 7 lists the header values from an IHead file corresponding to these structure members. This header information belongs to the isolated box image displayed in Figure 8. Referencing the structure members listed in Figure 6, the first attribute field of IHead is the identification field, **id**. This field uniquely identifies the image file, typically by a file name. The identification field in this example not only contains the image's file name but also the reference string the writer was instructed to print in the box. The reference string is delimited by double quotes. This convention enables an image

```
/*************************************************************
      File Name: IHead.h
      Package:  NIST Internal Image Header
      Author:   Michael D. Garris
      Date:     2/08/90
*************************************************************/
/* Defines used by the ihead structure */
#define IHDR_SIZE        288     /* len of hdr record (always even bytes) */
#define SHORT_CHARS      8       /* # of ASCII chars to represent a short */
#define BUFSIZE          80      /* default buffer size */
#define DATELEN          26      /* character length of data string */

typedef struct ihead{
  char id[BUFSIZE];               /* identification/comment field */
  char created[DATELEN];          /* date created */
  char width[SHORT_CHARS];        /* pixel width of image */
  char height[SHORT_CHARS];       /* pixel height of image */
  char depth[SHORT_CHARS];        /* bits per pixel */
  char density[SHORT_CHARS];      /* pixels per inch */
  char compress[SHORT_CHARS];     /* compression code */
  char complen[SHORT_CHARS];      /* compressed data length */
  char align[SHORT_CHARS];        /* scanline multiple: 8|16|32 */
  char unitsize[SHORT_CHARS];     /* bit size of image memory units */
  char sigbit;                    /* 0->sigbit first | 1->sigbit last */
  char byte_order;                /* 0->highlow | 1->lowhigh*/
  char pix_offset[SHORT_CHARS];   /* pixel column offset */
  char whitepix[SHORT_CHARS];     /* intensity of white pixel */
  char issigned;                  /* 0->unsigned data | 1->signed data */
  char rm_cm;                     /* 0->row maj | 1->column maj */
  char tb_bt;                     /* 0->top2bottom | 1->bottom2top */
  char lr_rl;                     /* 0->left2right | 1->right2left */
  char parent[BUFSIZE];           /* parent image file */
  char par_x[SHORT_CHARS];        /* from x pixel in parent */
  char par_y[SHORT_CHARS];        /* from y pixel in parent */
}IHEAD;
```

FIGURE 6 The IHead C programming language structure definition.

recognition system's hypothesized answers to be automatically scored against the actual characters printed in the box.

The attribute field, **created**, is the date on which the image was captured or digitized. The next three fields hold the image's pixel **width**, **height**, and **depth**. A binary image has a pixel depth of 1, whereas a gray-scale image containing 256 possible shades of gray has a pixel depth of 8. The attribute field, **density**, contains the scan resolution of the image; in this case, 12 pixels per millimeter (300 dpi). The next two fields deal with compression.

In the IHead format, images may be compressed with virtually any algorithm. Whether the image is compressed or not, the IHead is always uncompressed. This enables header interpretation and manipulation without the overhead of decompression. The **compress** field is an integer flag that signifies which compression technique, if any, has been applied to the raster image data following the header. If the compression code is zero, then the image data is not compressed, and the data dimensions, width, height, an depth, are sufficient to load the image into main memory. However, if the compression code is nonzero, then the **complen** field must be used in addition to the image's pixel

dimensions. For example, the image described in Figure 7 has a compression code of 2. By convention, this signifies that CCITT Group 4 compression has been applied to the image data prior to file creation. To load the compressed image data into main memory, the value in **complen** is used to load the compressed block of data into main memory. Once the compressed image data has been loaded into memory, CCITT Group 4 decompression can be used to produce an image which has the pixel dimensions consistent with those stored in its header. A compression ratio of 20 to 1 was achieved using CCITT Group 4 compression on the full-page images in SD1.

The attribute field, **align**, stores the alignment boundary to which scan lines of pixels are padded. Pixel values of binary images are stored 8 pixels (or bits) to a byte. Most images, however, are not an even multiple of 8 pixels in width. To minimize the overhead of ending a previous scan line and beginning the next scan line within the same byte, a number of padded pixels are provided to extend the previous scan line to an even-byte boundary. Some digitizers extend this padding of pixels out to an even multiple of 8 pixels; other digitizers extend this padding of pixels out to an even multiple of 16 pixels. This field stores the image's pixel alignment value used in padding out the ends of raster scan lines.

The next three attribute fields identify binary interchanging issues among heterogeneous computer architectures and displays. The **unitsize** field specifies how many contiguous pixel values are bundled into a single unit by the digitizer. The **sigbit** field specifies the order in which bits of significance are stored within each unit; most significant bit first or least significant bit first. The last of these three fields is the **byte_order** field. If **unitsize** is a multiple of bytes, then this field specifies the order in which bytes occur within the unit. Given these three attributes, binary incompatibilities across computer hardware and binary format assumptions within application software can be identified and effectively dealt with.

The **pix_offset** attribute defines a pixel displacement from the left edge of the raster image data to where a particular image's significant image information begins. The **whitepix** attribute defines the value assigned to the color white. For example, the binary image described in Figure 7 is black text on a white background and the value of the white pixels is 0. This field is particularly useful to image display routines. The **issigned** field is required to specify whether the units of an image are signed or unsigned. This attribute determines whether an image with a pixel depth of 8 should have pixels values interpreted in the range of -128 to $+127$, or 0 to 255. The orientation of the raster scan may also vary among different digitizers. The attribute field, **rm_cm**, specifies whether the digitizer captured the image in row-major order or column-major order. Whether the scan lines of an image were accumulated from top to bottom or bottom to top is specified by the field **tb_bt**, and whether left to right, or right to left, is specified by the field **rl_lr**.

The final attributes in IHead provide a single historical link from the current image to its parent image, the one from which the current image was derived or extracted. In Figure 7, the **parent** field contains the full path name to the image from which the image displayed in Figure 8 was extracted. The **par_x** and **par_y** fields contain the origin, upper-left-hand corner pixel coordinate, from where the extraction took place from the parent image. These fields provide an historical thread through successive generations of images and subimages. We believe that the IHead image format contains the minimal amount of ancillary information required to successfully manage binary and gray-scale images.

IMAGE FILE HEADER

Identity	: box_03.pct "0123456789"
Header Size	: 288 (bytes)
Date Created	: Thu Jan 4 17:34:21 1990
Width	: 656 (pixels)
Height	: 135 (pixels)
Bits per Pixel	: 1
Resolution	: 300 (ppi)
Compression	: 2 (code)
Compress Length	: 874 (bytes)
Scan Alignment	: 16 (bits)
Image Data Unit	: 16 (bits)
Byte Order	: High-Low
MSBit	: First
Column Offset	: 0 (pixels)
White Pixel	: 0
Data Units	: Unsigned
Scan Order	: Row Major, Top to Bottom, Left to Right
Parent	: hsf_0/f0000_14/f0000_14.pct
X Origin	: 192 (pixels)
Y Origin	: 732 (pixels)

FIGURE 7 The IHead values for the isolated box image displayed in Figure 8.

Database Examples

SD1 embodies a wide range of handwriting styles. The completed forms in this database illustrate the difficulty in recognizing handprinted characters with a computer. Frequently, even humans cannot positively identify characters without confirming their best guesses against the font information printed on the forms above each box. A quick scan of these handwriting samples shows great variation in size, slant, contrast, spacing, shape, the random interchanging of uppercase and lowercase, and the random switching between print and cursive script. In this section, a select set of handwriting samples from the database are shown in an attempt to illustrate to the reader the extreme variation in handwriting existing between different writers.

Compare the handprint in Figure 2 to the sample shown in Figure 9. If all handprint were of the style and quality shown in Figure 2, the challenge of recognizing handprint would no longer exist. The quality of handprint in Figure 9 is dramatically lower. Especially notice how the quality of the writing degrades from left to right, top to bottom, within the Constitution box. The characters in the top left corner of this box are well

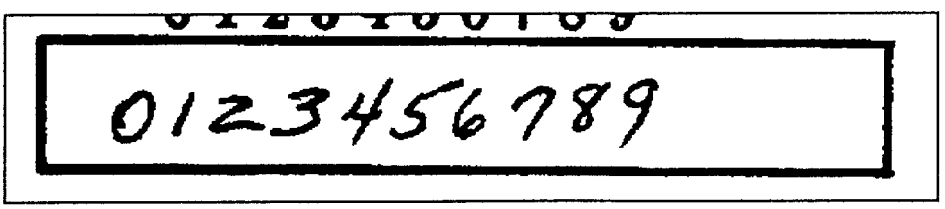


FIGURE 8 An IHead image of an isolated box from *NIST Special Database 1*.

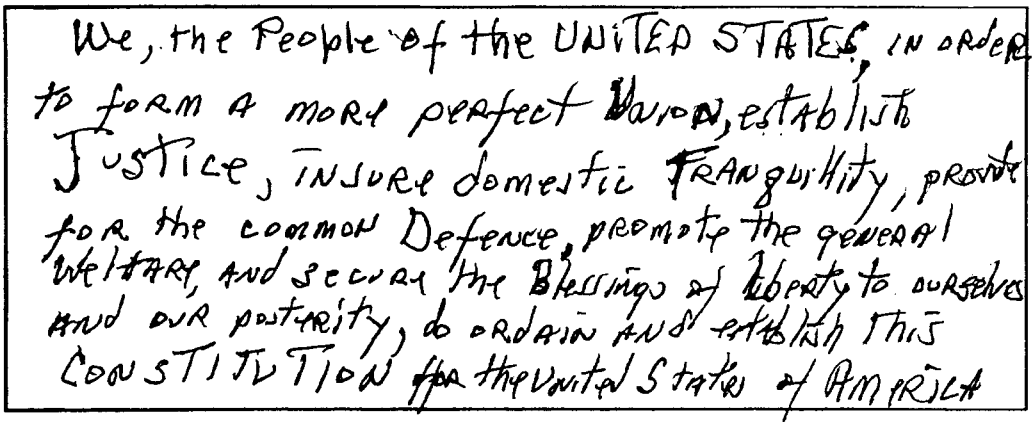


FIGURE 9 Handprint which is not very legible.

spaced both horizontally and vertically and appear reasonably legible. As the writer became cramped for space at the end of lines and toward the bottom of the box, the restriction of space visibly impacted the neatness and readability of the person's writing.

Figure 10 shows an example of a person's handprint written with a pronounced slant. It is interesting to note how the slant of characters from this writer varies. The characters printed in the digit boxes contain substantial slant, but when the person printed within the unconstrained Constitution box, the slant is even more pronounced. It is curious how the slant is almost completely missing from the characters printed in the two alphabetic boxes.

The average size of characters between writers also greatly varies. Figure 11 contains a sample of handprint which is relatively tall. If this person wrote any taller, the characters would not remain within the boxes. Note how the writer's lowercase extenders on the g, y, q, p, and j all extend well below the bottom of the lowercase alphabet box. In contrast to tall print, Figure 12 shows a portion of a form containing extremely small handprinted characters. Here the writer's handprint is almost the same size as the machine-printed information on the form.

In this database, the writers were not told what writing implement should be used to fill in the form. Therefore, the forms in this database represent different hardness of pencils, different colored ink pens, and different pen tips. A static scanner setting was used to digitize all the forms in the database regardless of the contrast between a form's background and the handprinted information it contains. The result is a database of images varying greatly in image quality or contrast. An example of a box completed using a hard-lead pencil is shown in Figure 13. The characters in this image are barely readable, some are not. Note how the individual characters are breaking up. Most character recognition systems would have significant problems reading this image. On the other hand, Figure 14 shows a section of a form which was completed using a broad felt-tipped pen. In this image, the pen strokes are extremely wide, causing most interior holes in characters to be closed. Note how difficult it is to distinguish 3's from 8's.

It has been observed that writers frequently make no distinction when printing lowercase and uppercase letters. Also, writers tend to randomly mix handprint with cursive script. Figure 15 shows a section of a form completed by a writer who printed nearly

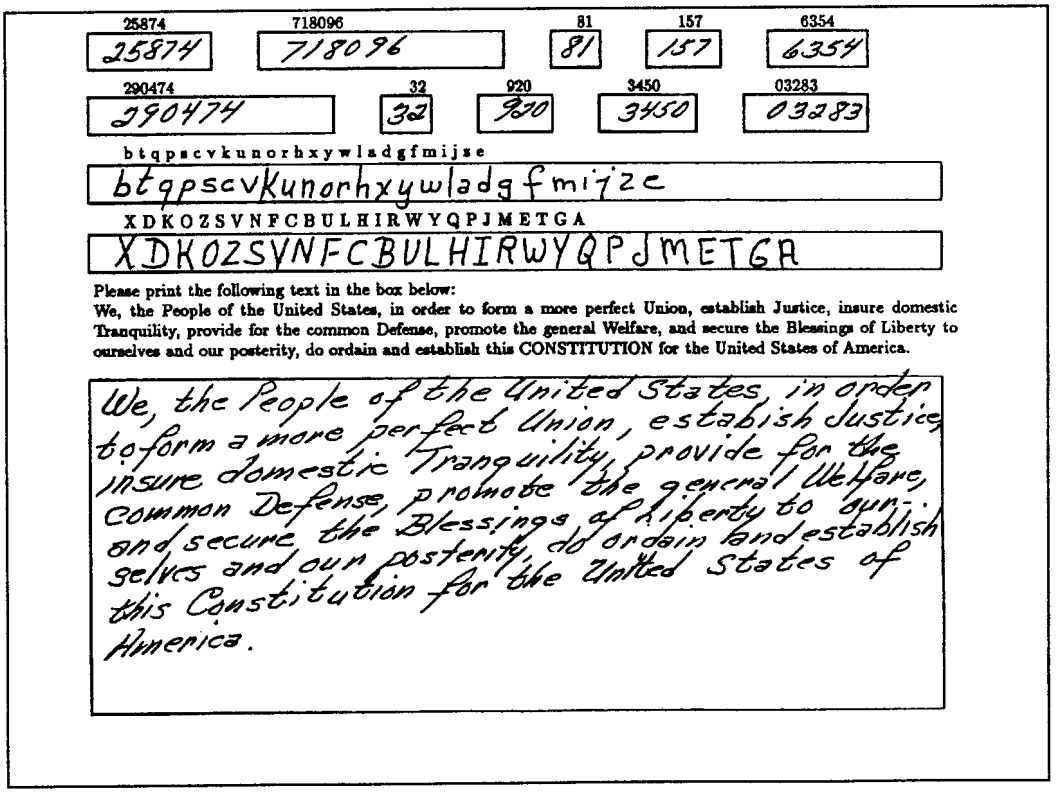


FIGURE 10 Handprint written with a pronounced slant.

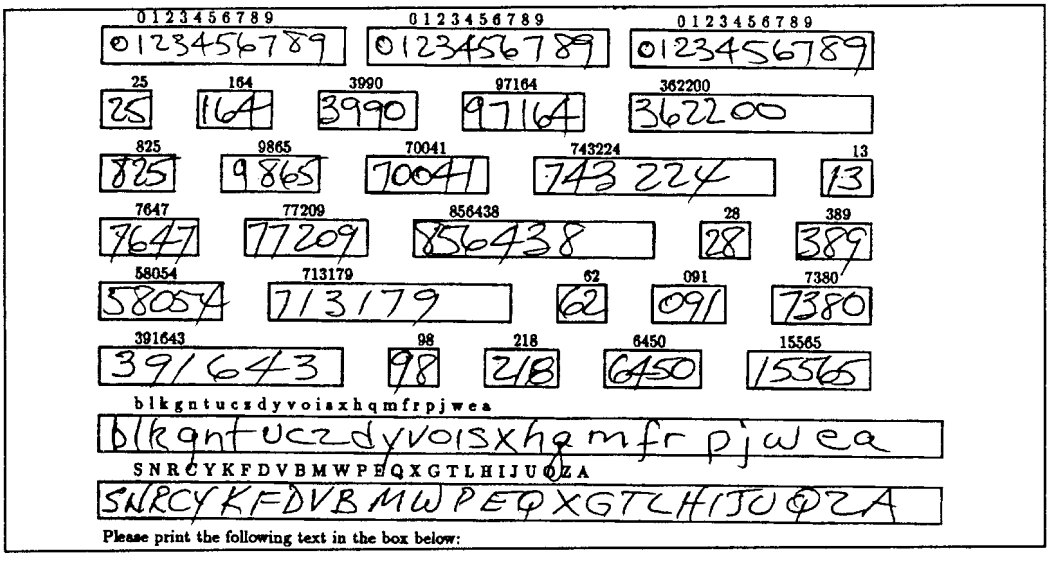


FIGURE 11 Handprint which is very tall.

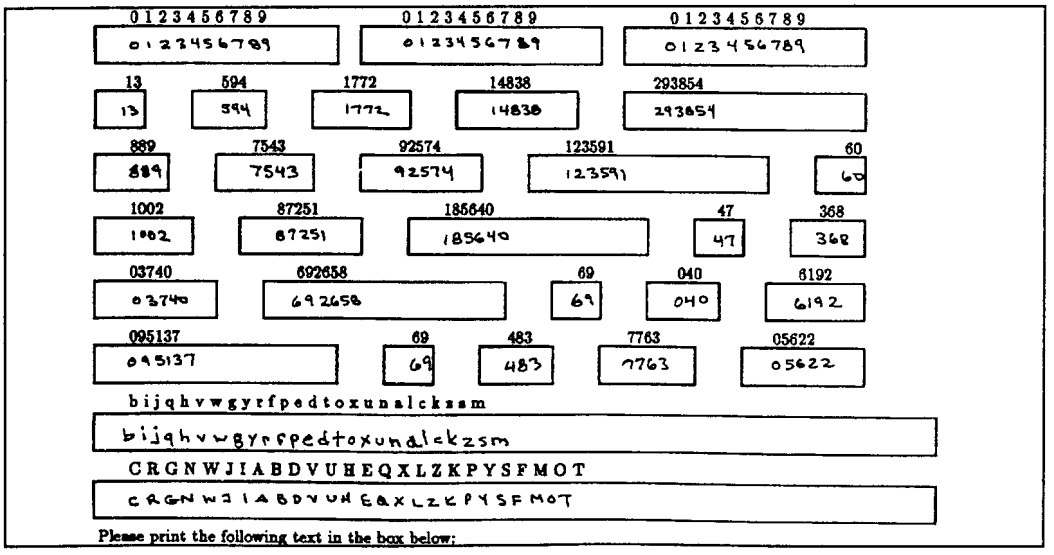


FIGURE 12 Handprint which is very small.

every lowercase letter in the lowercase alphabetic box the same way as the uppercase letters were printed. Note how the writer of this form printed within the alphabetic boxes but switched completely to cursive script when filling in the Constitution box. In Figure 16, the two boxes illustrate a writing style in which the writer printed all characters in the lowercase alphabetic box as cursive and filled in the upper case alphabetic box with printed letters. A robust recognition system must account for these inconsistencies.

Measurements Acquired During Processing

Robust document recognition systems detect and account for form rotation within an image. NIST has developed a model recognition system based on the forms in SD1 (5). This hybrid system combines traditional image processing, biologically motivated image

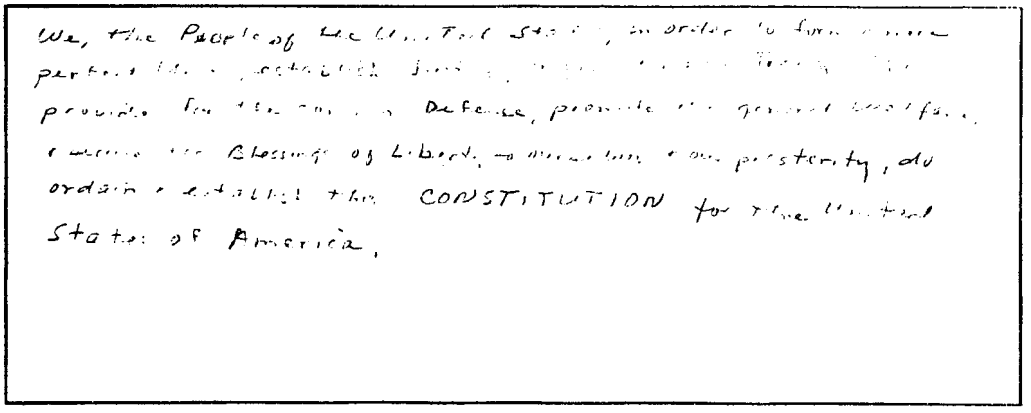


FIGURE 13 Digitized handprint which was printed very lightly with a hard-lead pencil.

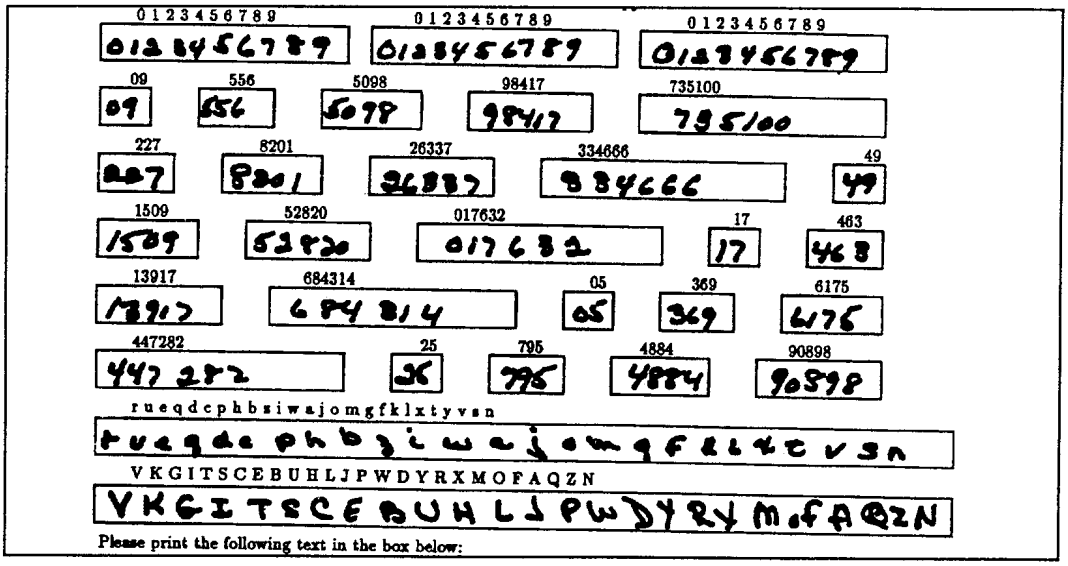


FIGURE 14 Digitized handprint which was printed with a broad felt-tipped pen.

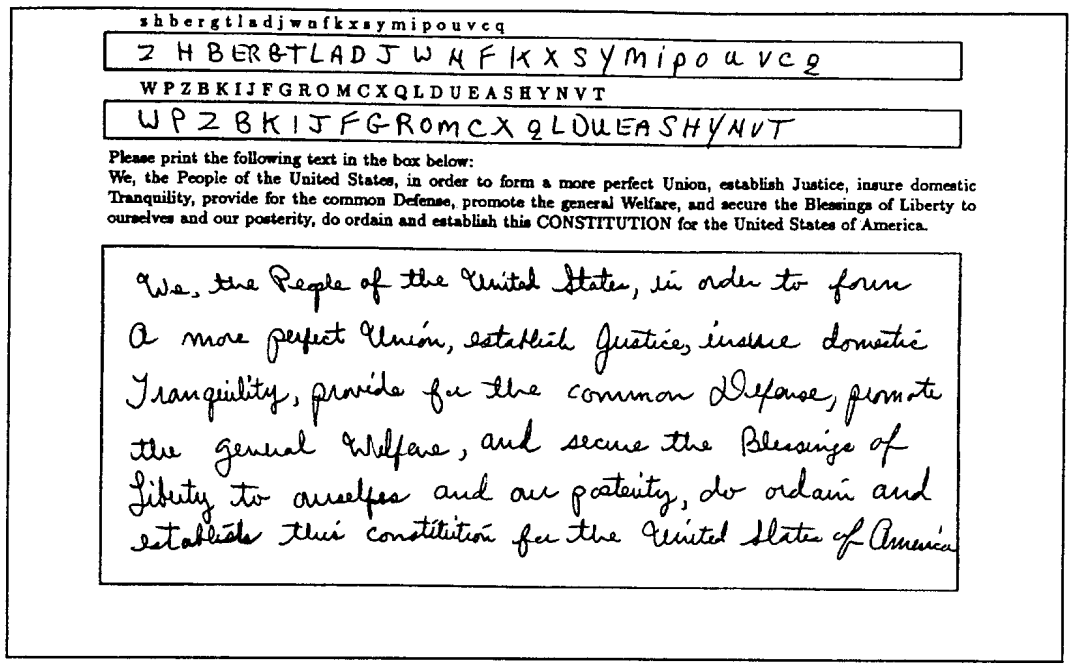


FIGURE 15 Example of printing lowercase letters as uppercase and switching to cursive script.

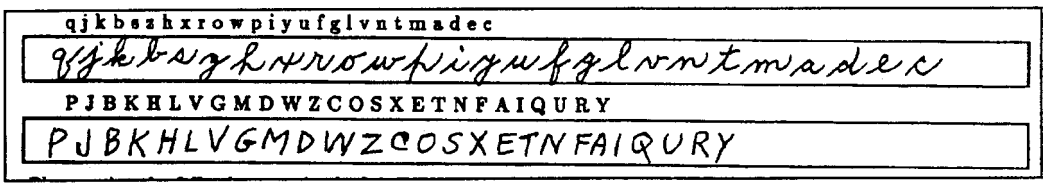


FIGURE 16 Example of writing all lowercase letters in cursive.

filtering, and neural networks on a massively parallel machine. One component in this system identifies form rotation and normalizes the image appropriately. The database contains images of forms rotated between +1.45° and −2.23° with an average of 0.3°.

This rotation was introduced at two different points in time. First, the original 50 form variants were reproduced using a photocopier. This introduced small rotational variations in the pages produced by the photocopier. The second source of rotational noise was introduced during the scanning of the completed forms. Note that despite the tight controls NIST placed on the printing, reproducing, and scanning of these forms, significant rotational noise exists in the database. Figure 17 shows an image from the database of a form with substantial rotational noise.

PRODUCTION OF SEGMENTED CHARACTER IMAGE DATABASES

As stated earlier, SD1 was collected to be used in training and testing high-speed high-throughput character recognition engines. Character classifiers typically recognize one individual character at a time, so in order to train and test these classifiers a subset of the more than 1,000,000 characters contained in SD1 had to be segmented into individual images and assigned reference classifications. This section discusses the creation of two segmented character image databases, *NIST Special Database 3* (SD3) (6) and *NIST Special Database 7* (SD7) (7).

Creation of *NIST Special Database 3*

The 2100 form images contained in SD1 are included in SD3. The digit and uppercase and lowercase fields on each form were segmented into isolated character images with each resulting image automatically assigned a character classification. Each segmented image from a field was assigned a consecutive character from the field's associated reference string. Using this technique, each image is assigned its correct reference classification, assuming there are no errors incurred during segmentation.

Unfortunately, the segmenter used was only about 80% accurate, frequently merging and splitting characters. Therefore, the classification assignments from the reference string often became unsynchronized with the actual images. This caused many images to be labeled incorrectly. Each referenced image was manually checked by a human, discarding the image when segmentation errors were detected and correcting the character image's classification when recognition errors were detected. Every referenced character image in this database has been manually checked and verified at two independent times, each time by a different person. This checking process took approximately 6 months to complete. As a result, SD3 contains 313,389 referenced character images.

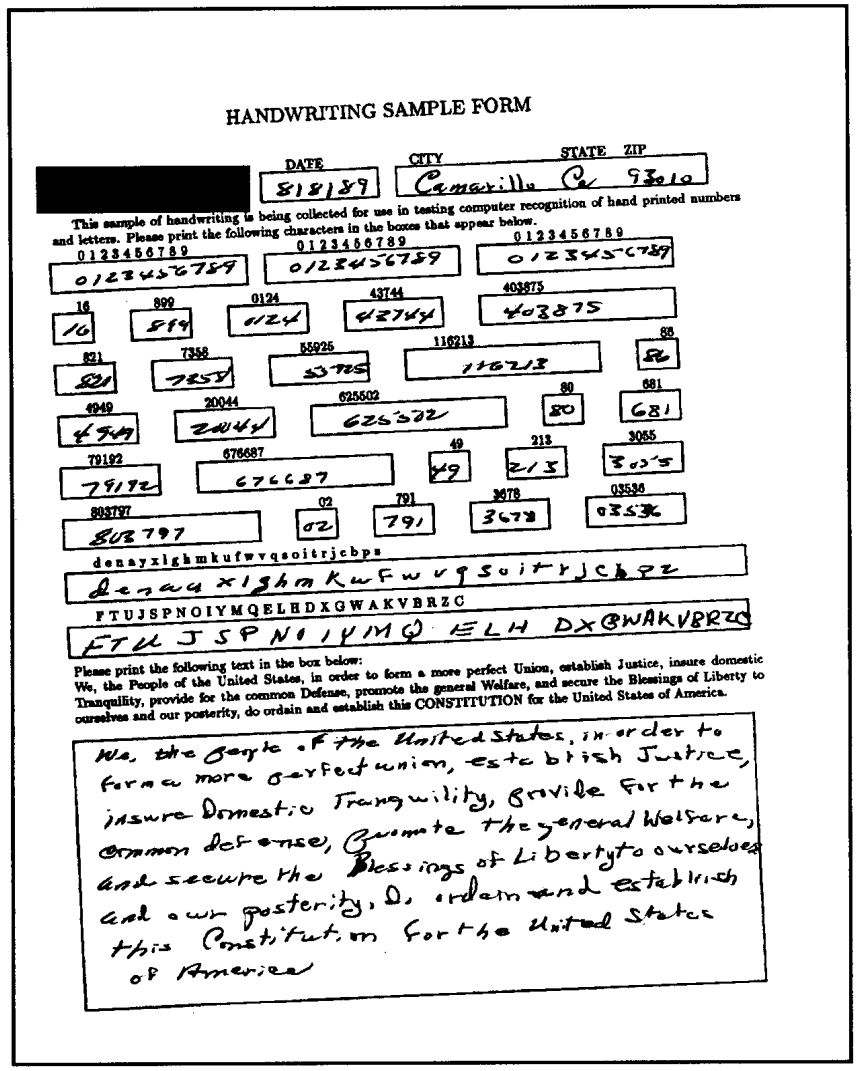


FIGURE 17 An image of a form from the database containing substantial rotational noise.

Multiple Image Set (MIS) File Format

Based on experience gained from creating and manipulating large on-line image databases, NIST has developed a number of diversified file formats. One such file format has been developed to manage large volumes of segmented character images. Storing character images in individual files has proven to be very inefficient, especially when manipulating databases containing hundreds of thousands of characters. Devoting a separate file node for each character image creates enormous file system overhead, and unreasonably large directory tables must be allocated. Rarely are experiments conducted on only a single-character image in isolation. Rather, most experiments require a large sample of characters. Experience has shown that the gathering of a large sample of characters from a file system where the images have been stored in individual files greatly burdens the computer's disk controller. This results in slow experiment loading times as well as limiting the access of other applications to data stored on the same storage device.

In addition to creating large directory tables, storing segmented character images in individual files results in sparse usage of the storage device. This sparseness is even more exaggerated when the images are compressed. For example, segmented character images in SD3 have been centered within a 128 × 128 binary pixel image. The resulting image size is 2344 bytes, 296 bytes for the IHead header and 2048 bytes of image data. These files when CCITT Group 4 compressed average 360 bytes in size, 296 bytes for the IHead header and only 64 bytes of compressed image data. Storing these compressed image files onto CD-ROM for example, which uses a 2048 block size, would be extremely wasteful. Only 18% of each block containing image data would be used.

In light of these observations, NIST has developed a Multiple Image Set (MIS) file format. The MIS format allows multiple images of homogeneous dimensions and depth to be stored in one file. MIS is a simple extension or encapsulation of the IHead format described previously. It can be seen in Figure 18 that the IHead structure is included as a member within the MIS definition.

An MIS file contains one or more individual images stacked vertically within the same contiguous raster memory, the last scanline of the previous image concatenated to the first scanline of the next image. The individual images that are concatenated together are referred to as MIS *entries*. The resulting contiguous raster memory is referred to as the MIS *memory*. The MIS memory containing one or more entries of uniform width, height, and depth is stored using the IHead header format. The IHead attribute fields are sufficient to describe the MIS memory. The IHead structure's **width** attribute specifies the width of the MIS memory, and likewise the IHead structure's **height** attribute specifies the height of the MIS memory. In this way, the MIS memory can be stored just like any normal raster image, including possible compression.

Due to the uniform dimensions of MIS entries, the IHead structure's **width** attribute also specifies the width of the entries in the MIS memory. What is lacking from the original IHead definition is the uniform height of the MIS entries and the number of MIS entries in the MIS memory. Realize that given the uniform height of the MIS entries, the number of entries in the MIS memory can be computed by dividing the entry height into the total MIS memory height. The interpretation of two of the IHead attribute fields, **par_x** and **par_y**, changes when the IHead header is being used to describe an MIS memory. The **par_x** field is used to hold the uniform width of the MIS entries, and the **par_y** field is used to hold the uniform height of the MIS entries. In other words, **width** and **height** represent MIS memory width and MIS memory height, respectively, whereas **par_x** and **par_y** represent MIS entry width and MIS entry height, respectively. Using this convention, an MIS file is treated like an IHead file.

Figure 18 lists the MIS structure definition written in the C programming language. The structure contains an IHead structure, **head**, and an MIS memory, *data*. In addition, there are six other attribute fields which hide the details of the IHead interpretation from application programs that manipulate MIS memories. The MIS attributes **misw** and **mish** specify the width and height, respectively, of the MIS memory. These values are the same as the **width** and **height** attributes contained in the IHead structure pointed to by **head**. The MIS attributes **entw** and **enth** specify the uniform width and height, respectively, of the MIS entries. These values are the same as the **par_x** and **par_y** attributes contained in the IHead structure pointed to by **head**. The MIS attribute **ent_alloc** specifies how many MIS entries of dimension **entw** and **enth** have been allocated to the MIS memory **data**. The MIS attribute **ent_num** specifies how many entries out of the possible number allocated are currently and contiguously contained in the MIS memory **data**.

```
/*************************************************************
        Filename: Mis.h
        Author: Michael D. Garris
        Date: 7/18/90
*************************************************************/
typedef struct misstruct{
    IHEAD *head;
    unsigned char *data;
    int misw;
    int mish;
    int entw;
    int enth;
    int ent_num;
    int ent_alloc;
} MIS;
```

FIGURE 18 The MIS C programming language structure definition.

Checking the results of segmenting the digit, uppercase, and lowercase fields from the 2100 full-page form images produced 313,389 referenced character images in SD3. Each segmented character was centered and stored into a separate 128 × 128 binary pixel memory preserving the original size and density of the character. The characters segmented from each form image were then organized into three different MIS files containing character images segmented from the digit fields, uppercase fields, and lowercase fields. An example of an MIS memory is displayed in Figure 19.

There is a potential for 6300 MIS files in the database, 2100 forms times the 3 character groups. Due to segmentation errors, the database actually contains 6166 MIS files including 23,125 digits, 44,951 uppercase letters, and 45,313 lowercase letters. A statistical log is provided with SD3 which lists all those forms missing one or more of the three MIS file groups. Every MIS file in the database has been compressed using CCITT Group 4, and the database is approximately 135 MB in size. Uncompressed, the database would require approximately 2.75 GB of storage.

Character Classification (CLS) File Format

For each segmented character image in SD3 there is an associated character classification which has been assigned and manually checked. Punctuation characters are often interpreted by disk operating systems and shells as special characters; so a character-naming convention has been developed at NIST which avoids these ambiguities. The classification value stored for each character image is the hexadecimal representation of the character's ASCII value. For example, an image of a handprinted 2 is assigned the class 32, a lowercase z is assigned the class 7a, and a handprinted semicolon (;) is assigned the class 3b.

These classifications are stored in Character Classification (CLS) files and correspond one-to-one to the entries in an associated MIS file in the database. Figure 19 displays an MIS memory along with its associated CLS file. The CLS file is an ASCII file with the first line containing the number of classifications values listed in the file, one value per successive line. Each ASCII text line is terminated with the linefeed character,

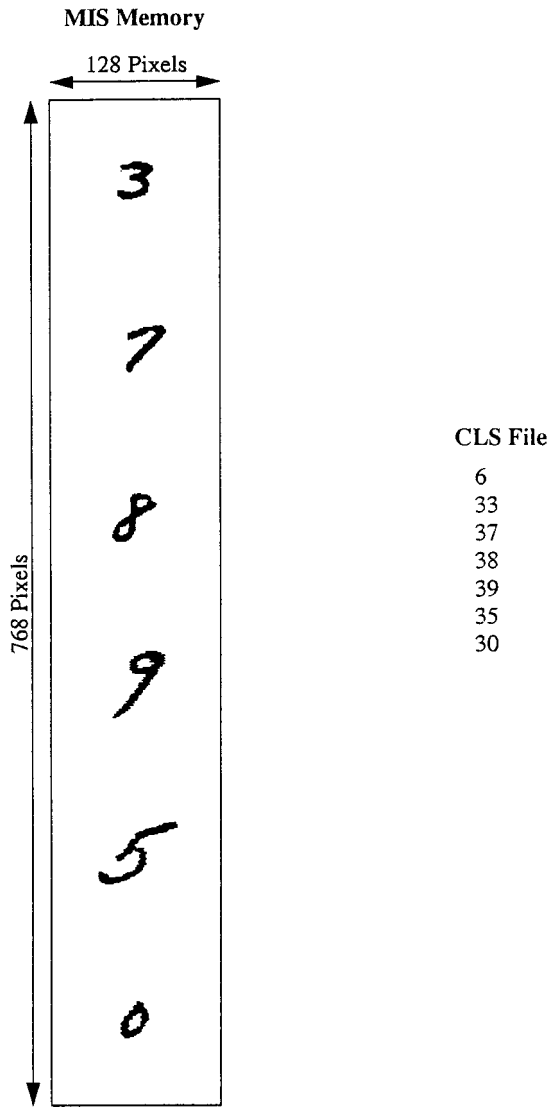


FIGURE 19 MIS memory and CLS file examples.

0x0A. Every MIS file in SD3 has an associated CLS file, and for each entry in a MIS file, there is a corresponding classification value in the CLS file. Figure 20 lists the frequency with which each digit and letter occurs within SD3.

Creation of *NIST Special Database 7*

Upon the creation of SD3, a conference for comparing the performance of various handprint character recognition systems was proposed. It was determined that SD3 would be distributed to conference participants for use in training and initial testing of their character classifiers. Details of the conference are recorded in Reference 1.

Digits		Uppers		Lowers	
0	22,971	A	1,649	a	1,738
1	24,772	B	1,691	b	1,766
2	22,131	C	1,732	c	1,794
3	23,172	D	1,712	d	1,748
4	21,549	E	1,759	e	1,769
5	19,545	F	1,756	f	1,608
6	22,128	G	1,707	g	1,740
7	23,209	H	1,677	h	1,791
8	22,029	I	2,102	i	1,838
9	21,619	J	1,633	j	1,446
		K	1,662	k	1,740
		L	1,830	l	2,021
		M	1,716	m	1,731
		N	1,732	n	1,777
		O	1,774	o	1,800
		P	1,723	p	1,589
		Q	1,664	q	1,554
		R	1,707	r	1,738
		S	1,755	s	1,776
		T	1,559	t	1,692
		U	1,763	u	1,827
		V	1,749	v	1,777
		W	1,705	w	1,765
		X	1,785	x	1,861
		Y	1,631	y	1,614
		Z	1,778	z	1,814

FIGURE 20 Frequency distribution of digits and letters in SD3.

Because all of SD3 was used as training data, a new segmented character database was required to test the systems entered in the conference. Blank handwriting sample forms like the ones used in SD1 were distributed to high school math and science students. From the set of filled out forms, 500 were selected for use in creating SD7. In association with the conference, this database is also referred to as *NIST Test Data 1* (TD1).

Machine-Assisted Reference Classifications

To test the character classifiers submitted to the conference, referenced character images had to be produced from the 500 forms. The 6 months required to produce SD3 was too long and demanded too much manual effort to meet the proposed schedule for the conference. Therefore, a new method for creating reference character images was proposed.

Within this same period of time, the Image Recognition Group as NIST completed the development of a massively parallel model recognition system (5). The system was designed to automatically process the Handwriting Sample Forms in SD1, reading the handprinted information contained in the digit and alphabetic fields. The system locates the entry field boxes on the form, segments the handprint within each field, and uses a neural network classifier to assign character classifications to each of the segmented images. Segmented character images along with their recognized classifications are stored as output from the model recognition system. At that time, the system was 94% accurate across all of SD1 with no rejections when reading digits, and less than 80% accurate when reading uppercase and lowercase letters. These performance statistics were com-

puted by using the NIST Scoring Package, *NIST Special Software 1* (SS1) (8,9) which applies the methods described in Reference 10.

A database production method was proposed that uses the results of the model recognition system to initially assign classifications to segmented character images. The details of this machine-assisted method are documented in Reference 11. The recognition system computes a confidence value for each character classification made. Based on the confidence value, a classification is labeled by the recognition system as accepted if the confidence is sufficiently high, otherwise the classification is labeled as rejected.

The segmented images whose classifications were accepted by the system are sorted by class and visually checked by a human. This phase, known as the *checking pass*, is very efficient, as up to 1024 characters images allegedly belonging to the same class are displayed simultaneously on a computer screen. In this way, any images containing segmentation errors or not belonging to the class being displayed are quickly located and flagged. The remaining images not flagged, along with their verified classifications, are stored as part of the database. The checking pass quickly incorporates the correctly segmented and correctly classified character images into the database.

A second phase, known as the *correcting pass*, is more labor intense. Here, images containing segmentation errors are separated from images that are correctly segmented but have been assigned an incorrect classification. The images containing segmentation errors are discarded, whereas the remaining images are reclassified by the human. The correcting pass is initiated by collecting all the images initially rejected by the recognition system and any images flagged during the checking pass and sorting them all by class. During the correcting pass, the images are displayed one at a time along with the image's currently assigned classification. The human is asked to verify that the current image is properly segmented. If the image is deemed to contain segmentation errors, it is discarded. If the image is properly segmented, the human is asked to verify that the current classification is correct. If the classification is deemed incorrect, the human is asked to enter the correct classification. Nondiscarded images from the correcting pass are then collected into a new checking pass.

The segmented images along with their classifications oscillate between the checking and correcting passes with image/classification pairs being incorporated into the database within the checking pass and image/classification pairs being reclassified or discarded within the correcting pass. Upon several complete oscillations, the database production reaches a steady state and the process is terminated. Figure 21 illustrates the two-pass process. Using this machine-assisted labeling approach, SD7 was ready for CD-ROM mastering in 2 weeks.

Database Content

The above labeling method produced approximately 83,000 images of handprinted digits and letters, and assigned to each image a correct reference classification for testing character recognition systems. The images from each form were organized into MIS files according to groups similar to SD3. In all, about 59,000 digits and 24,000 uppercase and lowercase letters are contained in SD7. The reference classifications were organized into corresponding CLS files and are provided on a separate floppy disk. This way the images can be distributed as testing material separate from the reference classifications which are the answers to the test. In May 1992, the First Census Optical Character Recognition Systems conference was hosted at NIST and the success of this event was heavily dependent on the production of SD3 and SD7.

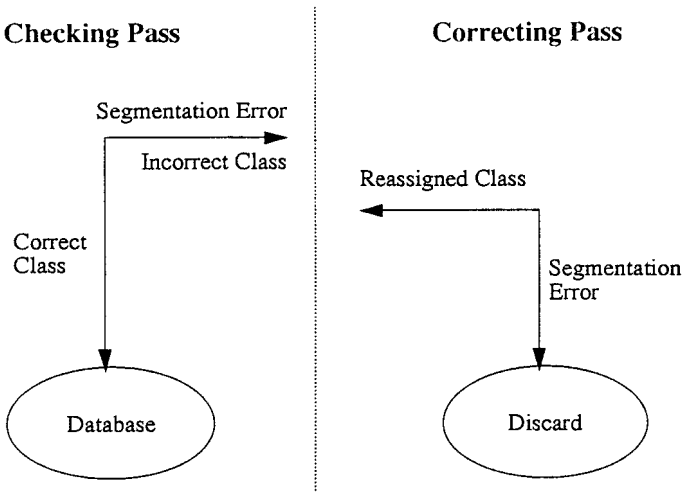


FIGURE 21 Machine-assisted reference classification labeling.

ANALYSIS OF HANDWRITING SAMPLE COMPLEXITY

As stated earlier, SD3 contains handwriting samples from regional data-takers employed by the Bureau of the Census, whereas SD7 contains handwriting samples from high school math and science students. Based on these samples, one might ask, "Are the handwriting styles within these two databases similar or are they quite diverse? Does this diversity make one database more difficult to recognize than the other? If given a choice, what combination of writers would make up the ideal training set for maximizing recognition performance across the remaining writers from both databases?" Questions like these and many more face researchers and developers of optical character recognition systems. The goal of these databases is to provide sufficiently numerous and varied training and testing images that the performance of a character classifier can be reliably measured. It is widely acknowledged within the neural network and pattern recognition communities that the robustness of a classifier depends on the generality contained within the training set. A method of cross-validation has been applied to these databases in an attempt to unlock some of the answers related to handwriting complexity (12).

In standard v-fold cross-validation, a population of segmented and referenced character images is divided into v equally sized partitions. A character classifier is tested, reserving one partition as testing data and using the remaining partitions as training data. Recognition results are accumulated as each partition is used in turn for testing, and the remaining partitions are used for training. Statistics are then computed and analyzed to assess the amount of diversity contained within a given population.

To compare diversity across two database populations, a cross-cross-validation must be conducted. Once again, each database is divided into equally sized partitions. In this case, a large number of partitions from the first database are selected as a fixed training set, whereas each partition in the second database is used, one at a time, for testing. Statistics are computed on the results from testing with each partition from the second database. Then the roles of testing and training are reversed: A large number of partitions

from the second database are selected as a fixed training set, whereas each partition from the first database is used for testing. The statistics computed are used to analyze the diversity between the two databases.

A study comparing SD3 to SD7 was conducted in Reference 12. In this study, the first 500 writers from SD3 were compared against the 500 writers in SD7. The two collections of 500 writers were each divided into 10 partitions with handprinted digits from 50 different writers in each partition. A standard 10-fold cross-validation was computed for each set. Each partition of 50 SD3 writers were selected in turn and used for testing the character classifier trained with the remaining 450 SD3 writers; 10 results were accumulated. The same was done for each partition of 50 SD7 writers, testing each partition in turn against the character classifier trained with the remaining 450 SD7 writers. The results from both standard 10-fold cross-validations are entered along the main diagonal of the validation comparison matrix shown in Figure 22. The results are shown as a mean percent error with associated standard deviation.

The cross-cross-validation is shown in the off-diagonal elements of the matrix in Figure 22. Four hundred fifty SD3 writers were used to train the character classifier, and then the partitions of 50 SD7 writers, one partition at a time, were used in testing. The resulting statistics are shown in the top-right element of the matrix. In reverse, 450 SD7 writers were used to train the character classifier, and then the partitions of 50 SD3 writers, one partition at a time, were used in testing. The results of training on SD7 writers and testing with SD3 writers is shown in the bottom-left element of the matrix.

The actual percentages of error shown in the matrix are not relevant to the comparison. The same character classifier was used throughout in an attempt to gain information about the data, not to gain information about the performance of the classifier. What is important are the differences in relative magnitudes between the various cross-validation statistics. The on-diagonal elements of the validation comparison matrix show that SD7 is a more diverse digit set than is SD3. When trained and tested on partitions of itself, the SD3 tests achieved a mean error rate of only 1.7%, whereas the SD7 tests achieved a mean error rate of 3.8%. In essence, SD7 is a more complex and difficult database to recognize than is SD3. The off-diagonal elements show that SD3 used as training prototypes for SD7 is markedly inferior to SD7 used as a training set for SD3. The classifier

Mean Error $\pm\sigma$	Test SD3 50 writers	Test SD7 50 writers
Train SD3 450 writers	$1.7\times10^{-2} \pm 0.3$	$6.8\times10^{-2} \pm 0.4$
Train SD7 450 writers	$3.5\times10^{-2} \pm 0.3$	$3.8\times10^{-2} \pm 0.5$

FIGURE 22 Cross-validation results comparing SD3 and SD7.

trained on SD7 and testing with SD3 achieved a mean percent error of 3.5%, whereas the classifier trained on SD3 and testing with SD7 achieved a mean percent error of 6.8%. This suggests that handprinted digits in SD7 represent a superset of the variation found in the samples contained in SD3. In other words, SD7 is a more general dataset.

CONCLUSION

Databases like *NIST Special Databases 1, 3,* and *7* are critical to advancing automated character recognition technologies. The Image Recognition Group at NIST is currently using these databases to research areas including field isolation, box detection and removal, character segmentation, and writer-independent character classification. Through the production of these databases, the First Census Optical Character Recognition Systems Conference was made possible, and practical lessons learned include issues of file formats, compression, and organization. As a result, techniques such as machine-assisted labeling of reference classifications and cross-validation studies for analyzing database complexity have been developed. These general techniques have broad application beyond character recognition. Together, these databases represent the largest publicly available collection of handprinted character images available for recognition system training and testing with copies being distributed and used around the world.

REFERENCES

1. R. A. Wilkinson, J. Geist, S. A. Janet, P. J. Grother, C. J. C. Burges, R. Creecy, B. Hammond, J. J. Hull, N. J. Larsen, T. P. Vogl, and C. L. Wilson, "The First Census Optical Character Recognition System Conference," Technical Report NISTIR 4912, National Institute of Standards and Technology, July 1992.
2. C. L. Wilson and M. D. Garris, "Handprinted Character Database," *NIST Special Database 1*, **HWDB**, April 1990.
3. Department of Defense, "Military Specification—Raster Graphics Representation in Binary Format, Requirements for, MIL-R-28002," December 1988.
4. CCITT, *Facsimile Coding Schemes and Coding Control Functions for Group 4 Facsimile Apparatus*, 1984, ITU, Geneva, Fascicle VII.3-Rec. T.6.
5. M. D. Garris, C. L. Wilson, J. L. Blue, G. T. Candela, P. J. Grother, S. A. Janet, and R. A. Wilkinson, "Massively Parallel Implementation of Character Recognition Systems," in *Conference on Character Recognition and Digitizer Technologies*, SPIE, San Jose, CA, 1992, Vol. 1661, pp. 269–280.
6. M. D. Garris and R. A. Wilkinson, "Handwritten Segmented Characters Database," Technical Report Special Database 3, **HWSC**, National Institute of Standards and Technology, February 1992.
7. R. A. Wilkinson, "Handprinted Segmented Characters Database," Technical Report Test Database 1, **TST1**, National Institute of Standards and Technology, April 1992.
8. M. D. Garris and S. A. Janet, "Scoring Package Release 1.0," Technical Report Special Software 1, **SP**, National Institute of Standards and Technology, October 1992.
9. M. D. Garris and S. A. Janet, "NIST Scoring Package User's Guide, Release 1.0," Technical Report NISTIR 4950, National Institute of Standards and Technology, October 1992.
10. M. D. Garris, "Methods for Evaluating the Performance of Systems Intended to Recognize Characters from Image Data Scanned from Forms," Technical Report NISTIR 5129, National Institute of Standards and Technology, February 1993.

11. R. A. Wilkinson, M. D. Garris, and J. Geist, "Machine-Assisted Human Classification of Segmented Characters for OCR Testing and Training," in *Conference on Character Recognition Technologies*, D. P. D'Amato (ed.), SPIE, San Jose, CA, 1993, Vol. 1906, pp. 208–217.
12. P. J. Grother, "Cross Validation Comparison of NIST OCR Databases," in *Conference on Character Recognition Technologies*, D. P. D'Amato (ed.), SPIE, San Jose, CA, Vol. 1906, pp. 296–307.

MICHAEL D. GARRIS