An offprint from the

Social Science Computer Review

Duke University Press
Durham, NC 27708
U.S.A.

Design and Collection of a Handwriting Sample Image Database

M. D. Garris

NIST, under sponsorship from the Bureau of the Census, has collected a database consisting of 2,100 pages of binary image data of handprinted characters including numerals and text. *NIST Special Database 1* contains handwriting samples from 2,100 writers geographically distributed across the United States. NIST is currently using this database to research field-isolation, box detection and removal, character segmentation, and writer-independent neural character recognition. In addition to advancing the design of algorithms, this database and its study can aid the social sciences. Observations can be compiled and used to improve form design and field layout strategies. Region-based studies and comparisons are also possible due to regionally distributed and referenced populations of writers in the database. In creating this large database, obstacles to the effective archiving of images have been dealt with and eliminated. This paper describes the database's content in terms of its collection, organization, and usefulness. The strategies and conventions used to develop *NIST Special Database 1* can be applied to any discipline requiring the use of image archives.

Learning algorithms and artificially intelligent machines show great promise for use in a wide range of exciting applications. Many potential computer applications have proven to be virtually unsolvable by conventional programming techniques. Now, new programming paradigms in conjunction with high-speed parallel hardware implementations have created the technology necessary to unlock many useful applications. Automated character recognition, the recognition of handprinted text, is one of the application domains experiencing great advances from these new technologies. The systems being developed demand more, however, than just sophisticated software and hardware. They require large sets of statistical exemplars in order to effectively model real problems. It was the need for a large collection of handwriting samples that motivated the production of the database of handprinted characters.

Social Science Computer Review 10:2, Summer 1992. CCC 0894-4393/92/$1.50.

Improved recognition algorithms and the increased performance of computers have touched off an imaging revolution. Computer databases containing transcribed text from paper archives are now being replaced by databases generated via automated recognition from image archives. This wave of new imaging technology offers a new way of information archiving, retrieval, and processing for many disciplines, including the social sciences. But several key issues must be addressed to make imaging effective.

First, there is the issue of data storage. A full page of text containing 60 lines with 80 characters per line requires less than 5,000 bytes of computer memory. Image archives place a much greater demand on mass storage requirements. A single binary image of the same information at 300 dots per inch is over 1 megabyte in size. If the gray scale image contained 256 possible shades of gray, as opposed to only 2 possible values (white or black) in a binary image, the same information requiring 5,000 bytes as text now demands over 8,000,000 bytes of storage.

The storage requirements for images leads to a second issue, data compression. The feasibility and usefulness of image technologies are greatly reduced without the use of effective data compression. Standards are required to guarantee interchangeability among imaging devices and image applications. In addition to data compression conformance, there is a need for a uniform image format. A computer image is encoded as a stream of bits, 1s and 0s. This one-dimensional bit stream may be encoded in many ways, depending on various host architectures. Ancillary information is required to interpret correctly the information that has been encoded. But what minimal set of image descriptors is required to guarantee maximum function? Much energy is being invested to standardize image formats and image compression techniques, but much more is still required (Dept. of Defense, 1988; CCITT, 1984).

A third issue facing the effective use of images deals with archival storage media. Today's storage options include both magnetic and optical media. If images are stored as a static archive, then a read-only medium such as CD-ROM or a write-once read-many (WORM) medium is appropriate. However, if an application requires only short term storage of images, then large read-writable magnetic disks may be appropriate. *NIST Special Database 1* provides example solutions to each of these three issues.

Database Content

NIST Special Database 1 contains 2,100 full-page images of handwriting samples printed by 2,100 different writers geographically distributed across the United States with a sampling roughly proportional to population density (Wilson & Garris, 1990). Each page

in this handwritten character database is a 300 pixels-per-inch image of a filled form. These image data were collected for use in training and testing high-speed, high-throughput recognition engines; therefore, the images were digitized as binary in order to reduce communication bandwidth and storage costs.

Each page in the database is an image of a structured form filled in by a unique writer. A single 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 composed 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 more than 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 megabyte of memory, the total image database, in uncompressed form, occupies approximately 3 gigabytes of mass storage. Therefore, the images are two-dimensionally

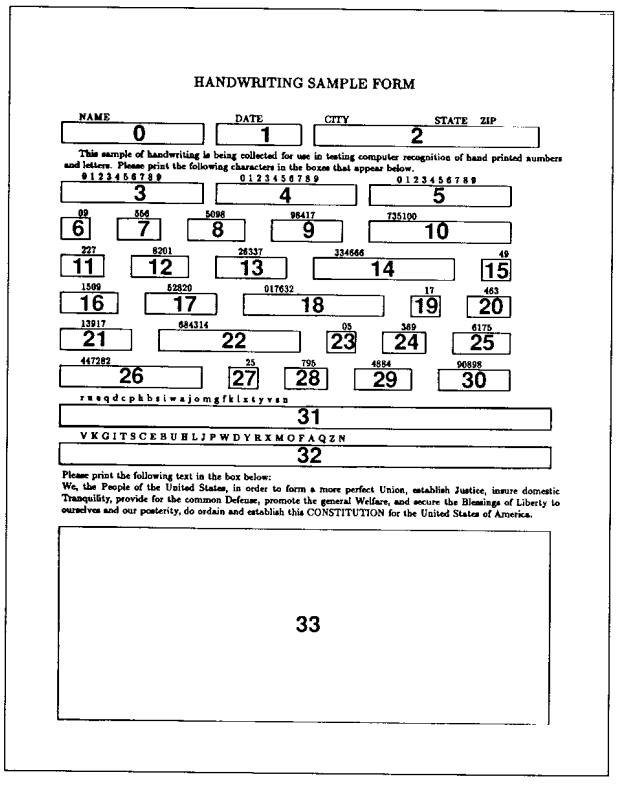


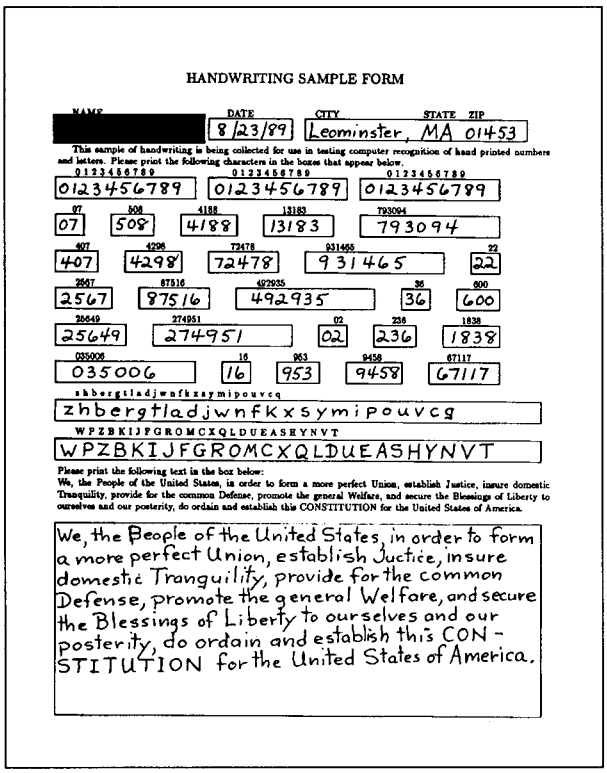
Figure 1 The Handwriting Sample Form is composed of 34 indexed entry field boxes

compressed in accordance with CCITT Group 4 (1984), reducing the overall size of the database to under 700 megabytes.

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 name field has been blacked out to make the writers in the database anonymous. The string of machine printed information above each box instructs the writer what to print in the box. The instructions on the form request that the writer print the information provided above each box within the box. Assuming the writer follows the directions and correctly completes 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 consis-


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.

0123456789 0123456789 0123456789

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

abbergtladjwnfkxsymipouvcq

zhbergtladjwnfkxsymipouvcq

WPZBKIJFGROMCXQLDUEASHYNVT

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 legible and neat handprint

tently defined as a bold box explicitly defining the location and spatial extent of each field. The single line boxes are 7mm in height, giving writers ample room to fit entire characters within the box. This size serves to constrain minimally the writer's print and aids the automated field isolation within the 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 identify 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 and lower recognition rates due to increased clutter and increased recognition confusion.

Ignoring the first 3 identification boxes shown in Figure 2, as one scans down the form one sees a progression of increasing recognition difficulty. The first series of boxes contain digits only, followed by boxes containing alphabetic characters. There are only 10 unique classes of digits, 0 through 9, versus 26 possible classes of the alphabetic characters, *A* through *Z*. The smaller number 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 because the handwriting is unconstrained: No specified line breaks are designated, no form lines guide the writer left to right, no form lines constrain the height of the characters, and so forth.

There are 50 variations of the form layout in the database. As stated above, a single field template was used so that all 2,100 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 remains fixed. The variations in forms provide 50 random orders of the lowercase alphabet and 50 random orders of the uppercase alphabet across the 2,100 forms.

Database Acquisition

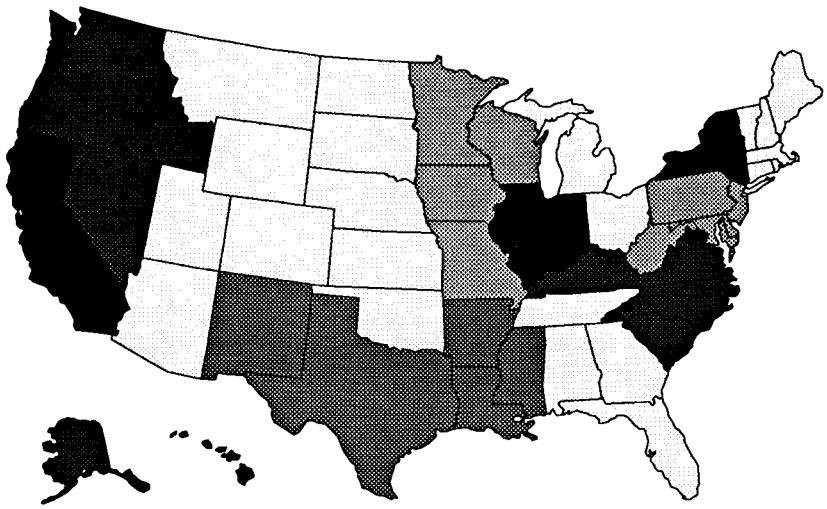
The 50 form-layout variations were tightly specified using a typesetting software package and printed on a laser printer. The 50 tem-

plates were then massively reproduced with a photocopier. From copies of the original 50 variations, 3,400 blank forms were mailed to 12 regional offices within the Bureau of the Census. There, field data takers filled out the forms and returned them 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 census regions.

Of 3,400 forms mailed to the regional offices, 2,100 completed forms were returned. From August 1989 through October 1989, the forms received at NIST were sequentially indexed, sorted according to region, logged, and digitized. Figure 4 lists the information recorded in the historical log provided with the database, including the form identification index, the form variation type (1 of 50) listed under the column TMPLT in the figure, 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.

Image File Format

As stated in the introduction, image file formats and effective data compression and decompression are critical to the usefulness of image archives. Each page returned was digitized in binary form at 300 dots per inch, two-dimensionally compressed using CCITT Group 4,



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

REGIONAL OFFICE 1
NAME: BOSTON, MA
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 4 A portion of the Historical Log provided in *NIST Special Database 1*

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 disks for permanent archiving and distribution.

In this application, a raster image is a digital encoding of light reflected from discrete points on a scanned form. The two-dimensional area of the form is divided into discrete locations according to the resolution of a specified 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 predominantly black. This two-dimensional sampling grid is then stored as a one-dimensional vector of pixel values in raster order, left to right, top to bottom. Successive scan lines (top to bottom) contain the values of a single row of pixels from the grid concatenated together.

After digitization, certain attributes of an image are required to interpret correctly the one-dimensional pixel data as a two-dimensional image. Examples of such attributes are the pixel width and pixel height of the image. These attributes can be stored in a machine-readable header prefixed to the raster bit stream. A program that is used to manipulate the raster data of an image is able to first read the header and determine the proper interpretation of the data that follow it. Figure 5 illustrates this file format.

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 ex-

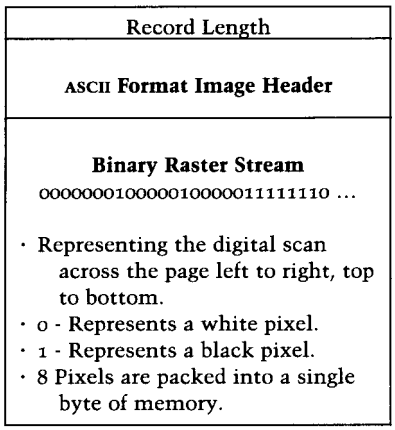


Figure 5 An illustration of the IHead raster file format

ist; 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 that can be implemented universally across heterogeneous computer architectures and environments. 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 adequately represent both binary and gray level images; represent images captured from different scanners and cameras; and satisfy the image requirements of diversified applications, including but not limited to image archival/retrieval, character recognition, and fingerprint classification.

IHead has been successfully ported and tested on several systems, including UNIX workstations and servers, DOS personal computers, and VMS mainframes. The attribute fields in IHead can be loaded into main memory in two distinct ways. Since the attributes are represented by the ASCII character set, the attribute fields may be parsed as null-terminated strings, an input/output format common in the C programming language. IHead can also be read into main memory using record-oriented input/output. The fixed length of the header is prefixed to the front of the header as shown in Figure 5. 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 the structure members listed in Figure 6. 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 contains not only the image's field name, but also the reference string the writer was instructed to print in the box. The reference string is delimited by double quotes. This con-

```
/*****************************************************************
      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

vention enables an image 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, 300 dots per inch. 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 that follow

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

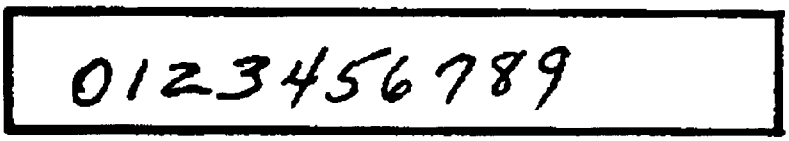


Figure 8 An IHead image of an isolated box from *NIST Special Database 1*

the header. If the compression code is zero, then the image data are not compressed, and the data dimensions—width, height, and depth—are sufficient to load the image into main memory. If the compression code is nonzero, however, 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. This signifies that CCITT Group 4 compression has been applied to the image data prior to file creation. In order 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 have been loaded into memory, CCITT Group 4 decompression can be used to produce an image that has the pixel dimensions consistent with those stored in its header. Using CCITT Group 4 compression and this compression scheme on the images in this database, a compression ratio of 20 to 1 was achieved.

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. In order 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 in order 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 digitzers 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 dealt with effectively.

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 pixel 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 of 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 a 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 manage binary and gray scale images successfully.

NIST Special Database 1 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 upper- and lowercase, and the random switching between print and cursive script.

Examples of Handwriting Extremes

In this section, a select set of handwriting samples from the database is shown in an attempt to illustrate to the reader the extreme variation in handwriting existing between writers. The first form shown in Figure 2 illustrates neat and legible print. If all handprint were of this style and quality, the challenge of recognizing handprint would no longer exist.

Compare the handprint in Figure 2 with the sample shown in Figure 9. The quality of handprint in the second figure 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 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 affected 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 that the slant of characters from this writer varies. The characters printed in the digit boxes contain substantial slant; the printing within the unconstrained Constitution box has an even more pronounced slant. It is

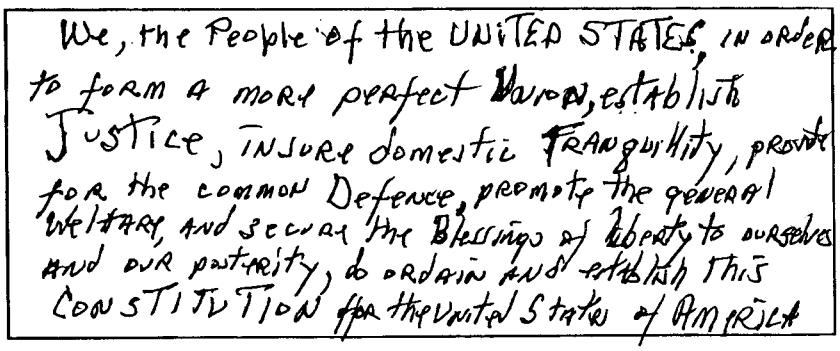


Figure 9 Handprint that is not very legible

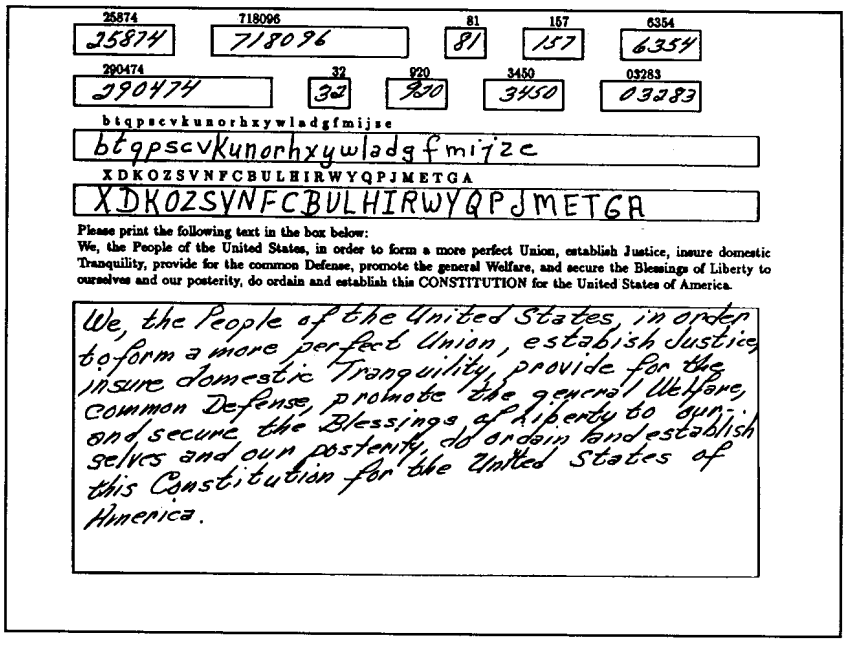


Figure 10 Handprint written with a pronounced slant

curious that the slant is almost completely missing from the characters printed in the two alphabetic boxes.

The average size of printed characters greatly varies between writers also. Figure 11 contains a sample of handprint that is relatively tall. If this person wrote any taller, the characters would not remain within the boxes. Notice how the writer's lowercase extenders on the *g*, *y*, *q*, *p*, and *j* extend well below the bottom of the lowercase alphabet box. 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 contained. The result is a database of images varying greatly in image quality or contrast. Figure 13 shows an example of a box completed with hard lead pencil. The characters in this image are barely readable, some are not readable. Notice that 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 that was completed using a broad felt-tipped pen. In this image, the pen strokes are extremely

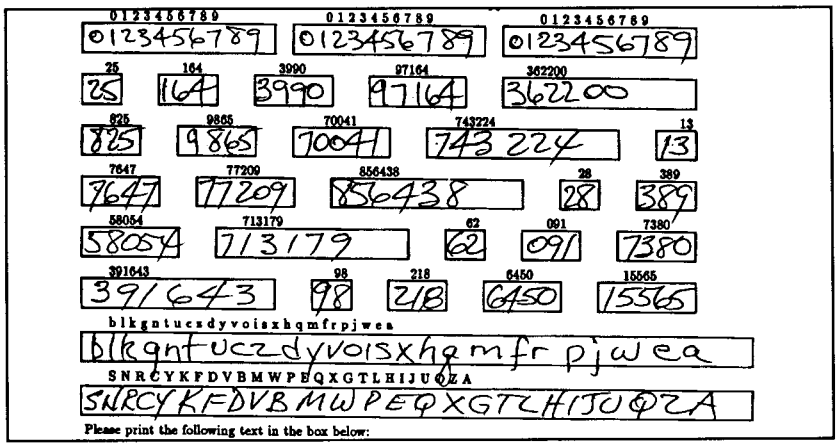


Figure 11 Handprint that is very tall

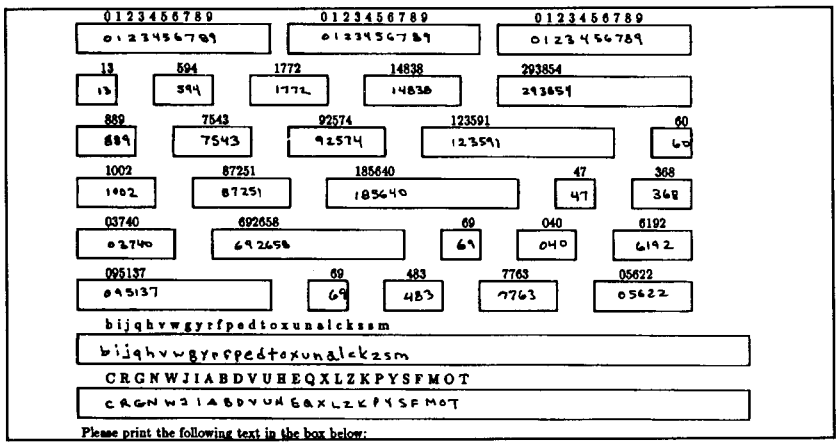


Figure 12 Handprint that is very small

wide, causing most interior holes in characters to be closed. Notice how difficult it is to distinguish 3s from 8s on this form.

Other Sources of Handprint Variations

Two other types of handprint variations are included in this database. It has been observed that writers frequently make no distinction when printing lower- and uppercase letters. Also, writers tend to mix handprint randomly with cursive script. Figure 15 shows a section of a form completed by a writer who printed nearly every lowercase letter in the lowercase alphabetic box the same way as the uppercase letters. Notice that the writer of this form printed within the alphabetic boxes but switched 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 low-

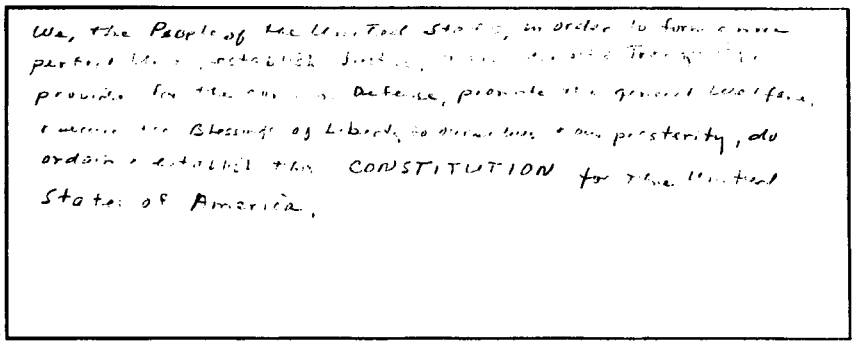


Figure 13 Digitized handprint that was printed very lightly with a hard lead pencil

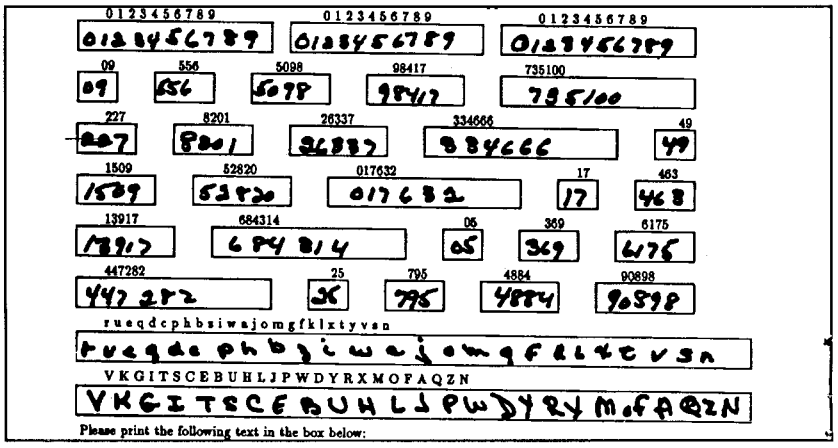


Figure 14 Digitized handprint that was printed with a broad felt-tipped pen

ercase alphabetic box as cursive and filled in the uppercase 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 an automated data-capture system based on the forms in *NIST Special Database 1*. This hybrid system combines traditional image processing, biologically motivated image 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 degrees and −2.23 degrees with an average of 0.3 degrees.

This rotation was introduced at two different points. First, the

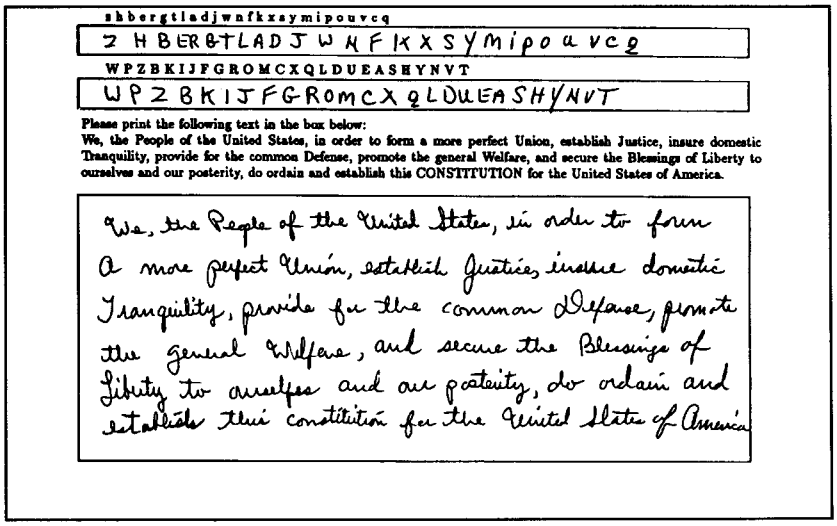


Figure 15 Example of lower-case letters printed as upper case and a switch to cursive script

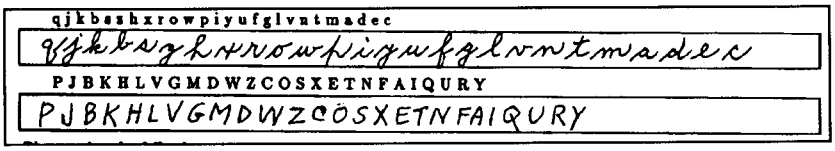


Figure 16 Example of all lower-case letters written in cursive

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.

Analysis of Boxes Left Empty

In addition to advancing the design of recognition algorithms, *NIST Special Database 1* can be used to aid the social sciences as well. For example, of 69,300 boxes across the 2,100 filled forms, 151 boxes were skipped and left empty with nearly 40% of these boxes corresponding to a single 2-character box found at the end of a line in the same location on all forms. This box is referenced as box 15, according to the labels assigned in Figure 1. An example is shown in Figure 18. Apparently in a writer's haste to finish the form, he or she overlooked this small context box. This box was skipped 58 times in this database. The next 2 most frequently skipped boxes were the 2

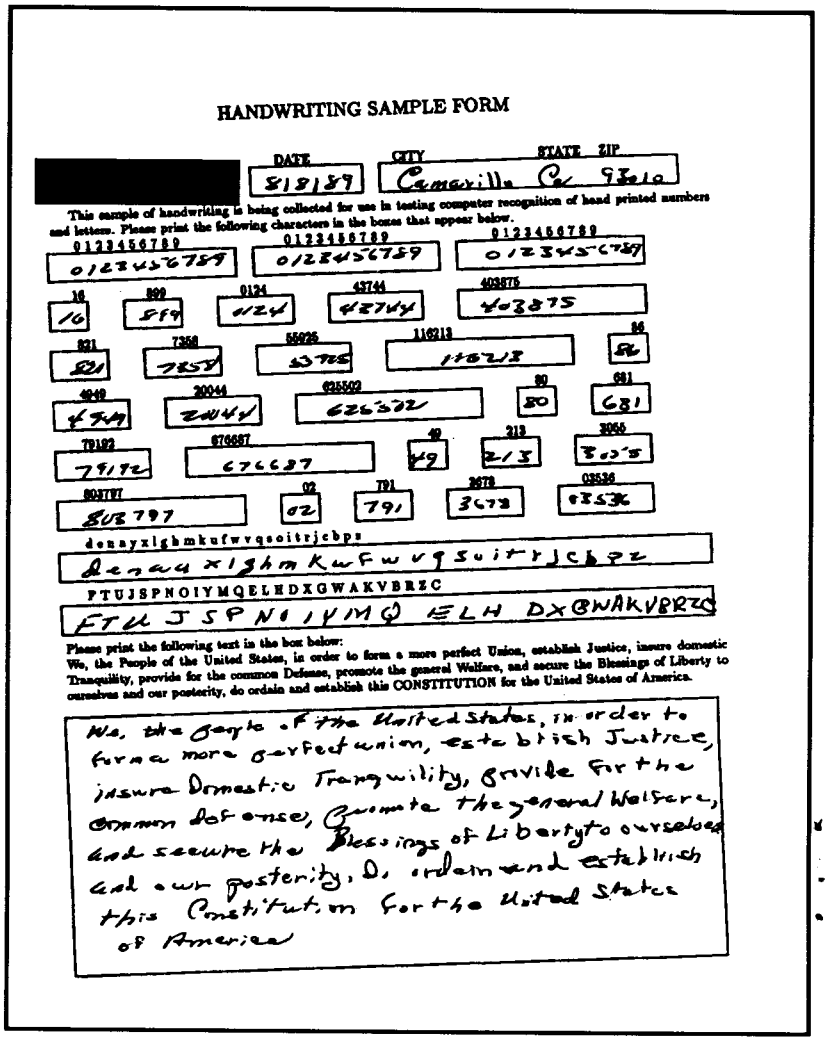


Figure 17 An image of a database form containing substantial rotational noise

alphabetic boxes. The uppercase alphabetic box was left empty 15 times, and the lowercase alphabetic box was left empty only 9 times. These 2 boxes are the most difficult to copy on the entire form. The number of times the alphabetic boxes were left empty even when combined is much less than the number of times box 15 was skipped. Observations like this one can be compiled and used to improve form design and field layout strategies.

Conclusion

NIST Special Database 1 proves to be a practical example of how issues such as image format, image data compression, and archival

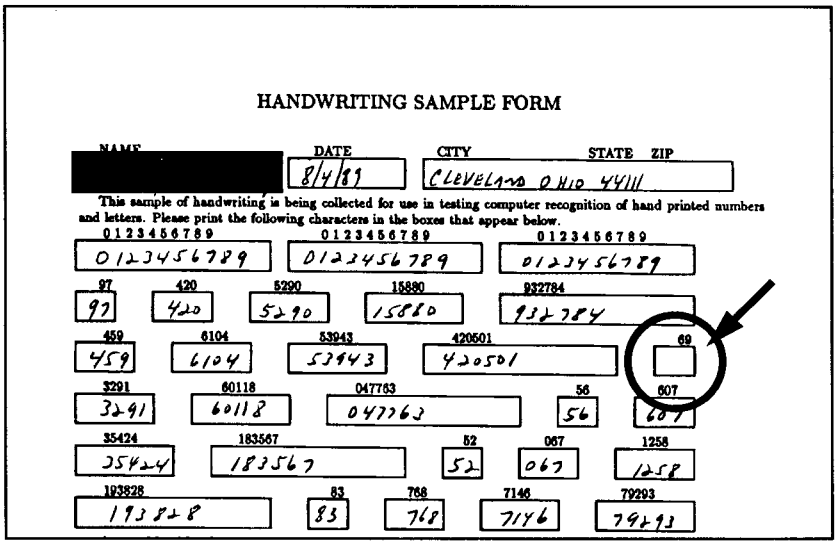


Figure 18 The top portion of a completed form with box 15 left empty

media need to be addressed. NIST is currently using this database to research field-isolation, box detection and removal, character segmentation, and writer-independent biologically motivated neural character recognition. Through the use of this database resource, handwritten character recognition with greater than 96% accuracy has been achieved in NIST laboratories on test sets of more than 1,000 arbitrarily chosen character digits with a classification rate of 10 ms per character (Wilson, 1992; Garris, Wilkinson, & Wilson, 1991). Based on this database, NIST is currently developing a draft standard on methods for evaluating the performance of systems intended to recognize handprinted characters from image data scanned from forms. To date, more than 50 copies of the database have been distributed to universities, private companies, and government agencies. This includes 10 universities and 40 computer companies and recognition laboratories within eight countries. *NIST Special Database 1* is the largest collection of handprinted characters publicly available for recognition system testing.

Note

Michael D. Garris received a B.S. degree in Computer Science from Clarion University of Pennsylvania in 1986. He received an M.S. degree in Computer Science from Johns Hopkins University in 1991. Upon completion of his undergraduate studies, he was employed by the Computer Systems Laboratory at the National Institute of Standards of Technology, where he has worked for more than five years. He has been a member of the Speech Recognition Group and currently is a senior member of the Image Recognition Group, where he serves as a technical project leader and researcher. His latest accomplishments include the design and integration of a neural-based prototype document processing system developed on a massively parallel ma-

chine for reading handprinted text. Phone: 301-975-2928; FAX 301-590-0932; Internet MDG@MAGI.NCSL.NIST.GOV.

References

CCITT (1984). Facsimile coding schemes and coding control functions for group 4 facsimile apparatus, fascicle VII.3-Rec. T.6.

Department of Defense (1988, 20 December). Military specification-raster graphics representation in binary format, requirements for, MIL-R-28002.

Garris, M. D., Wilkinson, R. A., & Wilson, C. L. (1991, July). Methods for enhancing neural network handwritten character recognition. In *Proc. of the IJCNN*, vol. 1, 695–700.

Wilson, C. L. Forthcoming. A new self-organizing neural network architecture for parallel multi-map pattern recognition—FAUST. *Progress in Neural Networks* 4.

Wilson, C. L., & Garris, M. D. (1990, April 18). Handprinted character database. *NIST Special Database 1*. HWDB.

The *Social Science Computer Review* (SSCORE) appears in Spring, Summer, Fall, and Winter. It is produced by the Social Science Computing Laboratory of North Carolina State University and is published by Duke University Press. SSCORE incorporates the journals *Social Science Microcomputer Review* (SSMR) and *Computers and the Social Sciences* (CASS).

Subscriptions The annual subscription rate for SSCORE is $36 for individuals and $72 for libraries; non-U.S. subscribers should add $8 for postage. Single copies are $18 each. Make payment in U.S. funds to "Duke University Press," and mail to Journals Fulfillment, Duke University Press, 6697 College Station, Durham, NC 27708 (919-684-2173).

Back Issues The following back volumes of SSCORE and its forerunners are available from Duke University Press at $72 each: volumes 1 and 2 of CASS, volumes 3–5 of SSMR, and volume 6 of SSCORE. A complete set of volumes 1 and 2 of SSMR, in loose-leaf binder format, is available from the editorial office for $30 (add $2 for shipping and make payment to "North Carolina State University").

Manuscripts Manuscripts should be submitted double-spaced and in triplicate, preferably limited to twenty pages. Authors are encouraged to request the journal's style guide. Reviewers are encouraged to request the reviewer volunteer form and the software or book review forms.

Editorial Address *Social Science Computer Review*, NCSU Box 8101, North Carolina State University, Raleigh, NC 27695 (919-515-2468, Fax: 919-515-7856).

Book Reviews Books for review should be sent to the Book Review Editor, Carl Grafton, Department of Government, Auburn University at Montgomery, Montgomery, AL 36193 (205-271-9590).

 US ISSN 0894-4393.