NIST Special Database 12

Census Miniform Training Database 2
Binary Images from Paper and Microfilm

Stan Janet

Computer Systems Laboratory
Technology Building Room A-216
National Institute of Standards and Technology
Gaithersburg, MD 20899 USA
Email: stan@magi.ncsl.nist.gov

1. Introduction

This document describes the NIST Special Database 12 (SD-12), a database of Census Miniform images. A Miniform is a non-sensitive portion of the Industry and Occupation section of an actual 1990 Census Long Form. This database is designed for the evaluation of optical character recognition (OCR) systems in a difficult but realistic form-based task on binary images from paper and microfilm.

Each miniform image contains three fields with handwritten answers to the following questions (Long Form Questions 28b, 29a and 29b respectively):

```
Describe the activity performed at location where employed.
What kind of work was this person doing?
What were this person's most important activities or duties?
```

A possible set of responses would therefore be:

```
hospital
registered nurse
patient care
```

Question 28a was not included in the test in order to comply with privacy provisions required by Title 13 of the U.S. Code.

The forms were scanned from paper and microfilm, the latter yielding images of far lesser quality. The images are 624 by 744 pixels sampled at 78.74 pixels/cm (200 pixels/inch). They are packed five to a file and are CCITT Group 4 *[1]* compressed. Source code for image manipulation, including programs to uncompress and unpack the images, is present on the CD-ROM. The code is written in the C programming language and was developed on Sun workstations running SunOS 4.1.1*. Figure 1 shows a better-than-average quality image scanned from microfilm. Figure 2 shows a typical image scanned from paper.

SD-12 was produced in conjunction with The Second Census Optical Character Recognition Systems Conference *[2]*, sponsored by NIST and the Bureau of the Census, in which the participants sought to determine the state of the art of the OCR industry on a challenging, realistic task. The results

* Specific hardware and software products identified in this paper were used in order to adequately support the development of the technology described in this document. In no case does such identification imply recommendation or endorsement by the National Institute of Standards and Technology, nor does it imply that the equipment identified is necessarily the best available for the purpose.

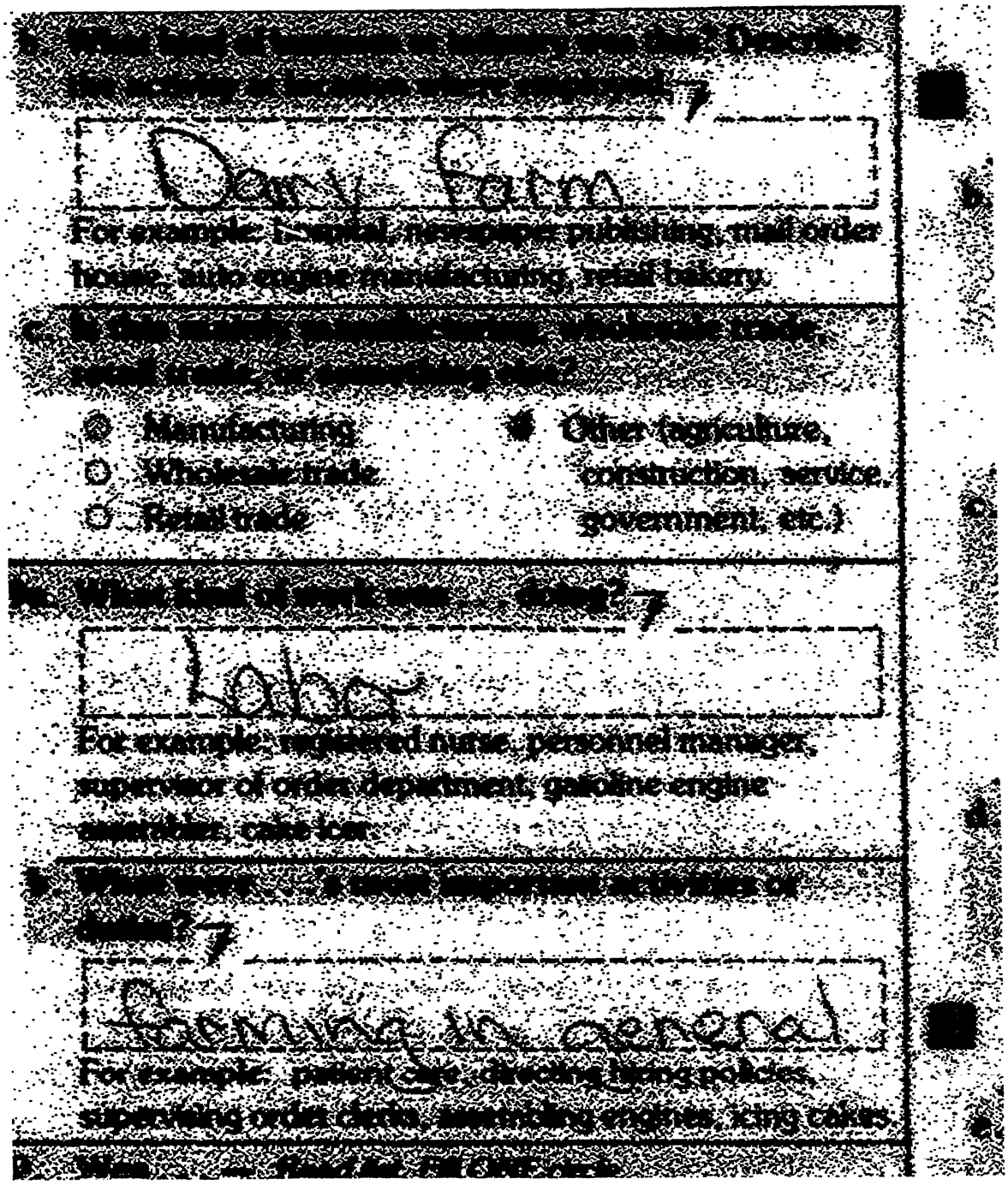

Figure 1. Example form scanned from microfilm.

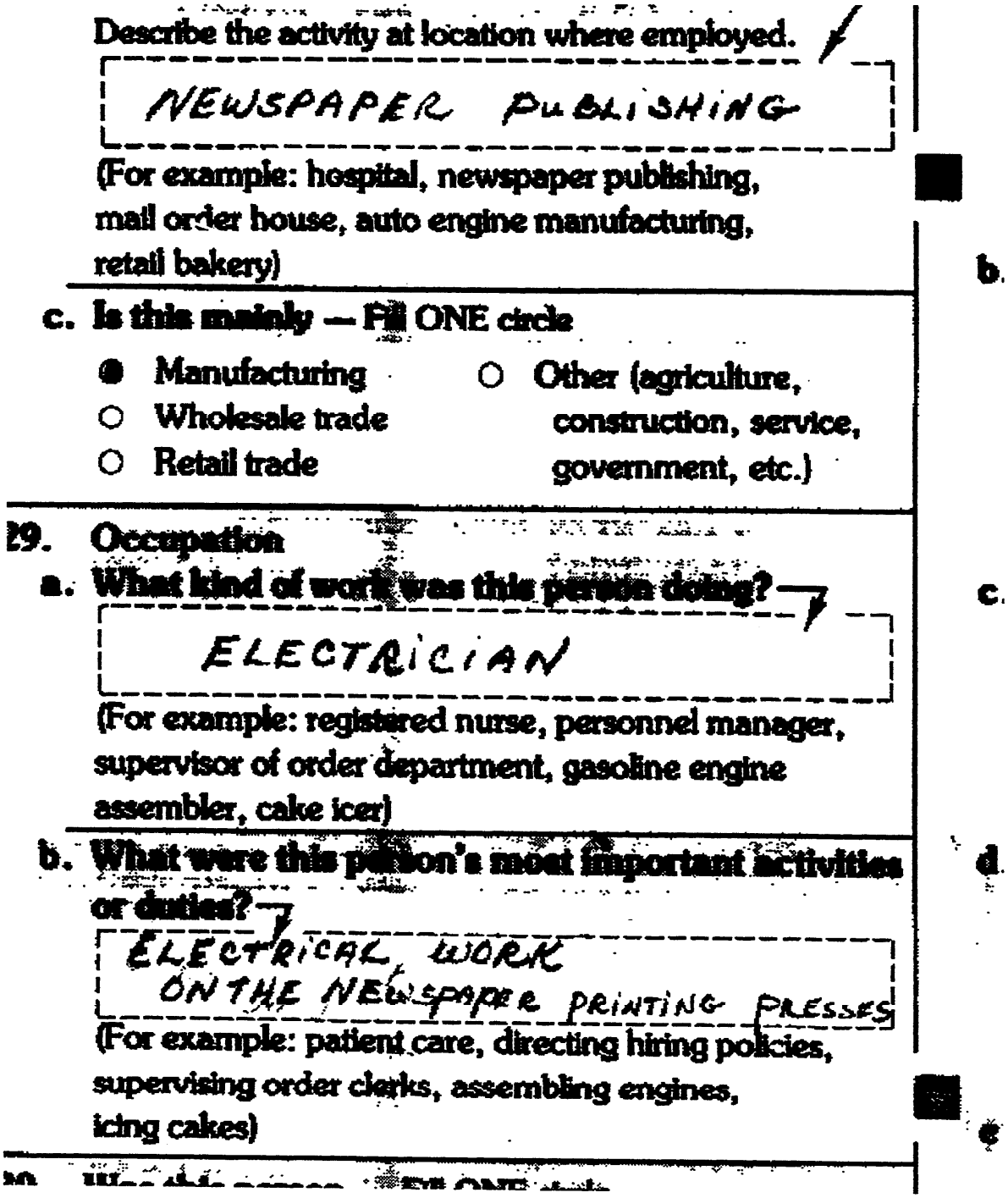

Figure 2. Example form scanned from paper.

of the Conference were published in NIST Internal Report (IR) 5452. That report is available on the Internet in PostScript form via anonymous FTP from the server *sequoyah.ncsl.nist.gov*, maintained by NIST's Visual Image Processing Group. It is also available on request in hardcopy form.

The SD-12 CD-ROM is the second of three produced for the Conference and was intended for system training. The first CD-ROM, SD-11, also contained training data. The third, SD-13, was used for the actual system testing. Both training databases contain reference files, files with ASCII transcriptions of what was written in each field. The test disc was produced without reference files. Those files are distributed with the CD-ROM on a DOS-formatted 3.5" floppy disk and have been publicly distributed from the FTP server since the conclusion of the Conference.

The main focus of this report is to describe the contents of the SD-12 database (Section 1) and to document the file formats (Section 2). Section 3 then describes the provided software in greater detail. Section 4 contains a brief summary of the results from the Second Census Conference. Section 5 contains descriptions of the files on the FTP server that are related to this database.

2. CD-ROM Contents

2.1. Mounting Procedure

In order for the files to be accessed in typical UNIX* environments, the CD-ROM must first be mounted, which requires super-user privileges. For example, under SunOS 4.1.1, the user *root* would execute the following command once the caddy containing the CD-ROM has been inserted into the drive:

```
# mount -t hsfs -o ro /dev/sr0 /cd
```

The command assumes that */dev/sr0* is configured in the UNIX kernel as the CD-ROM device, and that the directory hierarchy present on the disk is to appear under the directory */cd*, which must already exist.

The mounting process can be simplified by putting the following line in the file */etc/fstab*:

```
/dev/sr0 /cd hsfs ro 0 0
```

By storing the device name, filesystem type and mount options (*ro*, for read-only) in */etc/fstab*, the CD-ROM can be mounted by simply specifying the desired mount point:

```
# mount /cd
```

2.2. Disk Hierarchy

Figure 3 illustrates the directory tree structure of SD-12. Images are packed in MIS files (the MIS file format is defined in Section 3.3) in groups of five and the MIS files are grouped in subdirectories of 100 files each. The files of images scanned from paper are organized under the top-level directory *data2* while the files of images scanned from microfilm are under *data1*. For each MIS file, there is a *.ref* file containing ASCII transcriptions of the fields on the five miniforms.

2.3. Image Files

The SD-12 database contains 6000 miniform images from paper (18,000 fields) under the *data2* hierarchy and 12,500 miniform images from microfilm (37,500 fields) under the *data1* hierarchy. All miniforms were scanned at 78.74 binary pixels/cm (200 pixels/inch), and the dimensions for each were 624 pixels wide and 744 pixels high. Paper forms were scanned using a Fujitsu 3096 scanner. Microfilm was scanned on a Kodak Imagelink Digital Workstation.

The miniforms represent an almost random sampling of 1990 Census Long Forms, with only a small percentage of them having been discarded. Criteria for discarding a form included empty fields, scanning errors, and hopelessly poor handwriting. A manual check was also made to ensure that no

* UNIX is a registered trademark of AT&T.

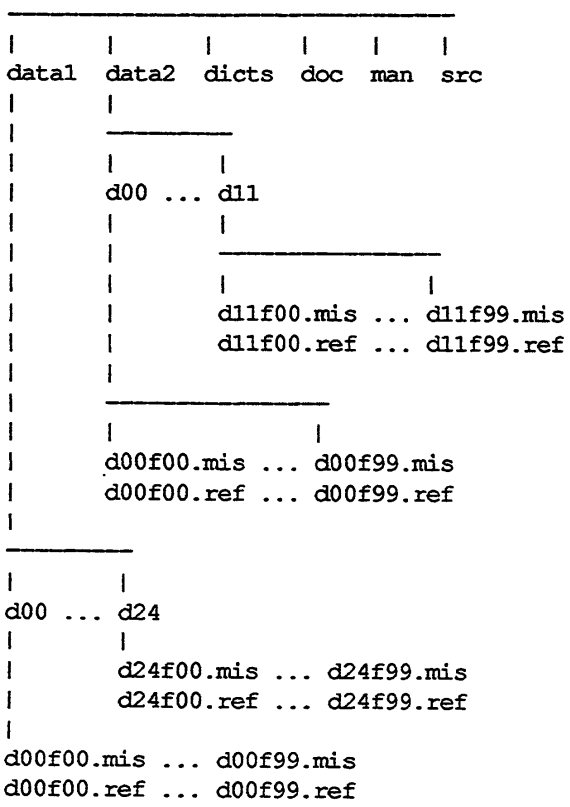

Figure 3. CD-ROM directory tree structure.

miniform fields contained sensitive information that would identify an individual, such as proper names and names of small companies.

SD-11 contains 13,500 miniform images scanned from microfilm (40,500 fields). SD-13 contains 3000 more miniform images from paper (9000 fields) and 3000 more miniform images from microfilm (9000 fields).

2.4. Reference Files

The reference files in SD-12 contain the target strings that Conference participants' systems were required to match exactly in order to be considered correct at the field level. To avoid hypotheses being considered incorrect due to largely irrelevant differences from the reference strings, both strings were to be normalized in a manner announced to the participants. Specifically, the normalization process entailed these five operations:

1. mapping lower-case characters to upper-case
2. mapping punctuation characters to spaces
3. removing leading or trailing spaces
4. mapping multiple adjacent spaces to one
5. truncating the strings at 36 characters

Reference files for fields on all paper forms were entered independently by two keyers, and by a third keyer if the transcriptions were different. The strings keyed during the 1990 Census were used to create the reference files for all microfilm forms.

2.5. Dictionaries

Each disc also contains three types of dictionary files for each of the three questions on the mini-form which are useful in the correction of raw recognition system output and in steering hypotheses toward answers that would be expected for occupation-related questions. One type of dictionary contained words, while the others contained phrases (complete field responses consisting of one or more words).

The word dictionary files contain all of the words that appeared in responses to the corresponding three questions in a sample of 132,247 1980 Census Long Forms:

Filename	#Words
dicts/word_1.lng	14545
dicts/word_2.lng	14503
dicts/word_3.lng	17239

The long phrase dictionary files contain all of the full responses to the three corresponding questions:

Filename	#Phrases
dicts/phrase_1.lng	53005
dicts/phrase_2.lng	52484
dicts/phrase_3.lng	68939

The short phrase dictionary files contain the full responses that appear at least twice in the master phrase list for the corresponding field:

Filename	#Phrases
dicts/phrase_1.sht	9310
dicts/phrase_2.sht	9575
dicts/phrase_3.sht	8757

The short phrase dictionaries are approximately 15% the size of the long ones, but contain approximately 60-70% of the unique reference strings. There is a fundamental trade-off associated with dictionary size. Adding words or phrases to a dictionary may improve the coverage of the dictionary over an application. Any such increase in coverage will contribute to improved OCR accuracy. On the other hand, every additional word or phrase in the dictionary increases the confusion among the different words or phrases in the dictionary. This increase in confusion may contribute to reduced OCR accuracy. Conference and NIST tests suggest that larger dictionaries will not improve results in this application *[2]*.

The dictionaries in SD-11 were created from the words and phrases that were given to the questions in a sample of 132,247 1980 Census Long Forms. The dictionaries in SD-12 were created by removing misspelled words from the 1980 Census files, then adding in reference strings from SD-11 that weren't already present in the files. The dictionaries in SD-13 were created from those in SD-12 by adding in the new reference strings from SD-12. The complete lists are available from the Census Bureau's anonymous FTP server, *ftp.census.gov*, in the directory *pub/ocr/1980i+o*.

NIST IR 5180 *[3]* contains a more complete discussion of the dictionary production process as it was originally undertaken. With the production of a second training database and the test database, the process was ultimately expanded and redesigned as described above. The final process is discussed in slightly greater depth in the Conference Report *[2]*.

2.6. Supplied Source Code

NIST Special Database 12 contains source code for a program to check a directory tree of hypothesis files and eight programs that manipulate IHead and MIS image files, the two image formats relevant to the benchmark. The programs are written in the C language and are included under the top level database directory *src*. These programs are described briefly in Section 4.

3. File Formats

3.1. Image Files

Image file formats and effective data compression and decompression are critical to the usefulness of image archives. 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 specified grid. Each cell of this grid, which is called a pixel, is represented by a single bit value 0 or 1. The former represents a predominately white pixel; the latter represents a predominately black pixel. 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. Successive scan lines (top to bottom) contain the values of a single row of pixels from the grid.

Certain attributes of a raster image are required to interpret the 1-dimensional pixel data as a 2-dimensional image. Examples of such attributes are 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 is used to manipulate the raster data must first read the header to determine the proper interpretation of the data which follows it.

3.2. IHead Header Format

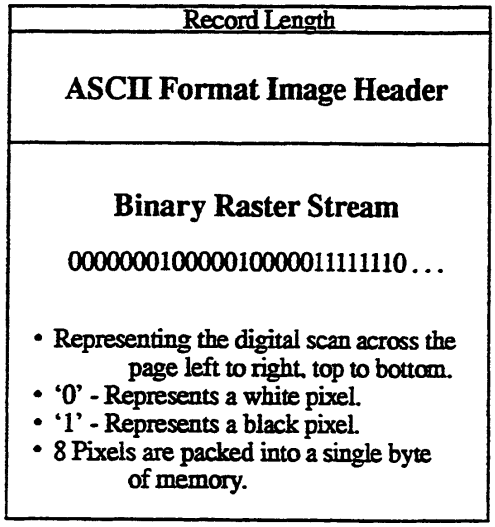

Figure 4. IHead file layout.

Numerous image formats exist, but most image formats are proprietary and, furthermore, many are much more predominant on some platforms, such as personal computers, than on others. A header format named IHead has been developed by NIST for use as a general purpose image interchange format. The IHead header is an open image format which can be universally implemented across heterogeneous computer architectures and environments. Source code for the IHead format is publicly available and included with this database. IHead has been designed with an extensive set of attributes in order to adequately represent both binary and gray level images, to represent images captured from different scanners and cameras, to support various image compression algorithms, and to satisfy the image requirements of diversified applications including, but not limited to, image archival/retrieval, character recognition, and fingerprint classification. Figure 4 illustrates the IHead format.

Since the header is represented by the ASCII character set, IHead has been successfully ported and tested on several platforms including UNIX workstations and DOS personal computers. All

attribute fields in the IHead structure are of a fixed length. All multiple character fields are null-terminated, allowing the fields to be loaded into main memory in two distinct ways. The IHead attribute fields can be parsed as individual characters and null-terminated strings, an input/output format common in the C programming language, or the header can be read into main memory using record-oriented input/output.

The *Record Length* in Figure 4 represents an 8-byte field at the beginning of each IHead file containing the null-terminated ASCII string "288", which defines the size in bytes of the image header. Figure 5 contains the IHead structure definition written for the C programming language that the image header should be read into.

Referencing the structure members listed in Figure 5, the first attribute field of IHead is the identification field, *id*. This field uniquely identifies the image file, typically by a file name. The attribute field *created* is the date on which the image was 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 grayscale 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 pixels per inch.

The next two fields determine the image data compression algorithm used, if any. In the IHead format, images may be compressed with virtually any algorithm. The IHead header is always uncompressed, even if the image data is compressed. This enables header interpretation and manipulation without the overhead of decompression. The field *compress* is an integer flag which signifies if a compression algorithm has been applied to the raster image data that follows the header. If the compression code is zero, then the image data is not compressed, and the data dimensions: width, height, and depth, are sufficient to load the image into main memory. However, if the compression code is nonzero, then the field *complen* must be used in addition to the image's pixel dimensions in order to load the compressed image data into main memory. Once the compressed image data has been loaded into memory, the appropriate decompression algorithm can be used to produce an image which has the pixel dimensions consistent with those stored in its header.

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) per byte. General images, however, do not have a width that is an an even multiple of 8 pixels. In order to minimize the overhead of ending a previous scan line and beginning the next scan line within a single byte, digitizers typically use padding pixels to extend the previous scan line to a multiple of 8 or 16 pixels (a one or two-byte boundary). The *align* 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 interchange 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 *byte_order*. If *unitsize* is a multiple of 8 greater than or equal to 16, this field specifies the order in which bytes occur within the unit — in the high bits of the unit first, or vice-versa. Given these three attributes, binary incompatibilities across computer hardware and binary format assumptions within application software can be identified and effectively dealt with.

```
#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 date 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 5. The C language *struct* definition for IHead images.

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 images in this database are 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 specifies whether the pixels of an image are signed or unsigned. For example, 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. The field *tb_bt* whether the scan lines of an image were accumulated from top to bottom, or bottom to top, and the field *rl_lr* specifies whether left to right, or right to left.

The final attributes in IHead provide a single historical link from the given image to a parent image, one from which it was derived or extracted. The *parent* field typically contains the full pathname to the image from which the current image was extracted. The *par_x* and *par_y* fields contain the origin point (upper left-hand corner pixel coordinate) in the parent image from where the extraction took place. These fields provide a historical thread through successive generations of image and subimage processing. If the image has no parent, the three fields contain an empty string. The IHead image format contains the minimal amount of ancillary information required to successfully manage binary and grayscale images.

3.3. MIS File Format

In order to facilitate operations on images of homogeneous dimensions and simplify image grouping, 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. As can be seen in Figure 7, a pointer to the IHead structure is a member in the MIS structure.

An MIS file contains one or more individual images stacked vertically within the same contiguous raster memory, with the first scanline of each image following the last scanline of the previous image. The individual images are referred to as MIS entries. The resulting contiguous raster memory is referred to as the MIS memory. An MIS memory containing five entries of uniform width, height, and depth is illustrated in Figure 6. 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 possibly compressed.

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

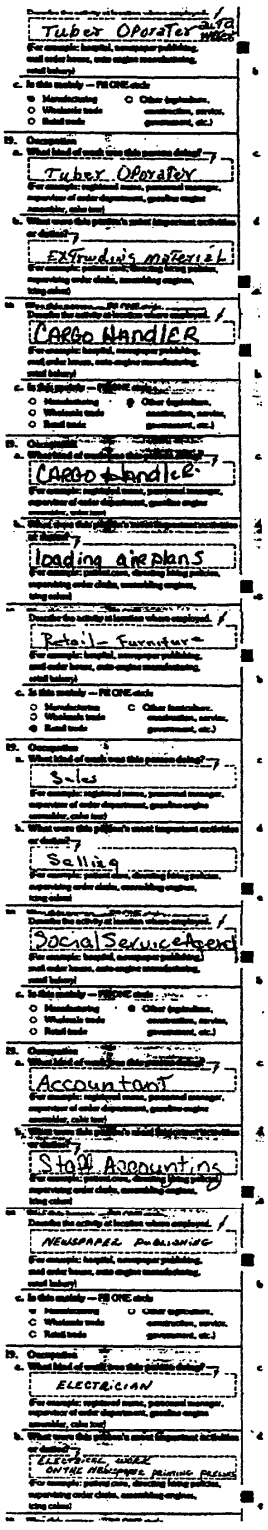

Figure 6. Layout of an MIS file containing five miniforms.

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, while *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.

```
typedef struct misstruct {
    IHEAD * head;
    unsigned char * data;
    int misw;
    int mish;
    int misd;
    int entw;
    int enth;
    int ent_num;
    int ent_alloc;
} MIS;
```

Figure 7. The C language *struct* definition for MIS images.

Figure 7 contains the MIS structure definition for the C programming language. The structure contains an IHead structure *head*, and an MIS memory *data*. In addition, there are 6 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 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 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.

All image files in NIST Special Databases 11 through 13 are MIS files. The supplied program *dumpihdr* prints the attributes of either MIS or IHead files. The following is that program's output on a example file from SD-13:

```
$ dumpihdr /cd/sd13/data3/d00/d00f00.mis
IMAGE FILE HEADER
~~~~~~~~~~~~~~~~~

Identity        :  d00/d00f00
Header Size     :  288 (bytes)
Date Created    :  Thu Oct 14 09:39:57 1993
Width           :  624 (pixels)
Height          :  3720 (pixels)
Bits per Pixel  :  1
Resolution      :  200 (ppi)
Compression     :  2 (code)
Compress Length :  252100 (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,
```

The following command sequence illustrates how an MIS file can be copied to a magnetic disk, decompressed, and then split using the supplied program *fragmis* into the five images that comprise it:

```
$ cd /tmp
$ cp /cd/sd13/data3/ d00/d00f00.mis .
$ chmod 644 d00f00.mis
$ ls -l d00f00.mis
-rw-r--r--  1 stan     252396 Apr  6 23:55 d00f00.mis
$ decomp d00f00.mis
$ ls -l d00f00.mis
-r-xr--r--  1 stan     290456 Apr  6 23:56 d00f00.mis
$ fragmis d00f00.mis f
$ ls -l f_???.pct
-rw-r--r--  1 stan      58328 Apr  6 23:56 f_000.pct
-rw-r--r--  1 stan      58328 Apr  6 23:56 f_001.pct
-rw-r--r--  1 stan      58328 Apr  6 23:56 f_002.pct
-rw-r--r--  1 stan      58328 Apr  6 23:56 f_003.pct
-rw-r--r--  1 stan      58328 Apr  6 23:56 f_004.pct
```

The supplied program *xtrctmis* could have also been used. It allows IHead files to be unpacked from MIS files individually:

```
$ xtrctmis d00f00.mis f0.pct 0
$ xtrctmis d00f00.mis f1.pct 1
$ xtrctmis d00f00.mis f2.pct 2
$ xtrctmis d00f00.mis f3.pct 3
$ xtrctmis d00f00.mis f4.pct 4
$ ls -l f?.pct
-rw-r--r--  1 stan      58328 Apr  6 23:56 f0.pct
-rw-r--r--  1 stan      58328 Apr  6 23:56 f1.pct
-rw-r--r--  1 stan      58328 Apr  6 23:56 f2.pct
-rw-r--r--  1 stan      58328 Apr  6 23:56 f3.pct
-rw-r--r--  1 stan      58328 Apr  6 23:56 f4.pct
```

The *dumpihdr* program can in turn be run on any of the resulting IHead files, e.g.:

```
$ dumpihdr f0.pct
IMAGE FILE HEADER
~~~~~~~~~~~~~~~~~
Identity        :  f0.pct
Header Size     :  288 (bytes)
Date Created    :  Fri Apr  7 00:03:26 1995
Width           :  624 (pixels)
Height          :  744 (pixels)
Bits per Pixel  :  1
Resolution      :  200 (ppi)
Compression     :  0 (code)
Compress Length :  0 (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
```

3.4. Reference Files

Reference files in Databases SD-11 and SD-12 are distributed in the data subdirectories on the CD-ROM's, while those for SD-13 are distributed on floppy disk in *zip* format *[4]* with a directory hierarchy mirroring the MIS files on the CD-ROM. The file correspondence is evident through their identical basenames; the reference filenames have the extension *.ref*. Each reference file contains fifteen lines, with the first three lines containing ASCII transcriptions of the three questions in the MIS file's first miniform, etc.

The first eight characters of every line in a reference file contain a 7-character field string called a field identifier which is unique inside the file, followed by a space character. The characters that follow that space comprise the reference string that the recognition system output for that field is to be compared against.

As stated previously, reference files for fields on all paper forms were entered independently by two keyers, and by a third keyer if the transcriptions were different. The strings keyed during the 1990 Census were used to create the reference files for all microfilm forms. While no amount of care can resolve truly ambiguous handwriting, any remaining differences between the reference strings from what was actually written by Census respondents should not pose a significant problem to the recognition task.

The reference files are directly readable by the NIST Image Recognition System Scoring Package *[5]*. The software was used to produce results for the two Census OCR Systems Conferences *[2,6]*. Version 1.0 of the Scoring Package is available on CD-ROM from NIST Standard Reference Data just as SD-12 and the other two CD-ROM's are. Version 2.0 will be released soon, and information will be available on the FTP site or from the author of this document via email.

The files of recognition system hypotheses were required to have the same general format as reference files, but with hypotheses following field identifier and space character. The directory hierarchy was to mirror that of the test images, with *.hyp* extensions in place of *.mis* files. Corresponding files of field-level confidence values were also to be returned with *.con* extensions. In those, a floating point number between 0.0 and 1.0 (inclusive) was to follow the field identifier and space character. Appendix A contains an example form reference file and hypothetical recognition system output files.

4. Program Descriptions

NIST Special Database 12 contains source code for a program to check a directory tree of hypothesis files and eight programs that manipulate IHead and MIS image files, the two formats used in the benchmark. The programs are written in the C programming language and are included under the top-level database directory *src*. These programs, their primary supporting subroutines, and associated filenames are described below. Manual pages are included in Appendix B and are located on the CD-ROM in the top-level directory *man*.

4.1. Program Compilation

CD-ROM is a read-only storage medium. This fact requires that the files located in the directory *src* be copied to a writable disk partition prior to being compiled into executable form. Once the files have been copied, binaries can be produced by invoking the UNIX utility *make*. An example command sequence that copies the files to */usr/local/ocr2/src* from under the CD-ROM mount point */cd* is:

```
$ cp -pr /cd/src /usr/local/ocr2/src
$ cd /usr/local/ocr2/src
$ chmod 755 .
$ chmod 644 *
$ make -f makefile.mak
```

4.2. chk_hyps <root-directory> <subdirectory-count>

The program *chk_hyps* is supplied to check the format of Conference hypothesis files and their directory structure. Submissions that did not meet the Conference specifications were rejected. This program performs eight checks, such as confirming the presence of expected hypothesis files and lines and confirming correct line syntax.

4.3. decomp <IHead file in> <IHead file out>

The program *decomp* decompresses an image in IHead format. The output file specified will be an image in IHead format with its image data uncompressed. The main routine for the program is found in *decomp.c* and calls the external functions readihdrfile() and writeihdrfile().

The procedure readihdrfile() is responsible for loading an IHead image from a file into main memory and is found in the file *loadihdr.c*. This routine reads the image's header data returning an initialized IHead structure by calling readihdr(). In addition, the image's raster data is returned to the caller uncompressed. The images in this database have been 2-dimensionally compressed using CCITT Group 4 *[1]*, therefore readihdrfile() invokes the external procedure grp4decomp() which decompresses the raster data. Upon completion, readihdrfile() returns an initialized IHead structure, the uncompressed raster data, and the image's width and height in pixels. The Group 4 compression and decompression software was developed by the CALS Test Network and adapted by NIST for use with this and other databases. It is found in the file *g4decomp.c*.

The function readihdr() is responsible for loading an image's IHead data from a file into main memory. This routine allocates, reads, and returns the header information from an open image file in an initialized IHead structure. This function is found in the file *ihead.c*. The IHead structure definition listed in Figure 5 is found in the include file *ihead.h*.

4.4. dumpihdr <IHead file>

The program *dumpihdr* reads an image's IHead data from the given file and formats the header data into a report which is printed to standard output. The main routine of the program is found in the file *dumpihdr.c* and calls the external function readihdr().

4.5. fragmis <misfile> <rootname>

The program *fragmis* takes the concatenated MIS entries contained in a single MIS file and writes each entry to a separate IHead image file. The program is given the MIS file to be fragmented and the root name to be used in creating the resulting IHead image files. A sequential index and an extension *.pct* will be added to the specified root name in order to create unique file names. The main routine for the program is found in *fragmis.c* and calls the external function fragmis(). The MIS structure definition listed in Figure 7 is found in the include file *mis.h*.

4.6. xtrctmis <misfile> <outfile> <index>

The program *xtrctmis* copies a specified MIS entry into a separate IHead image file. The inputs are the MIS file to be used, the file in which the MIS entry is to be stored, and the index of the MIS entry to be copied. The index is zero-oriented. The main routine for the program is in *xtrctmis.c* and calls the external routine xtrctmis().

4.7. ihdr2sun <IHead file>

The program *ihdr2sun* converts an image from NIST IHead format to Sun rasterfile format. It loads an IHead formatted image from a file into main memory and writes the raster data to a new file appending the data to a Sun rasterfile header. The main routine for the program is found in the file *ihdr2sun.c* and calls the external function readihdrfile(). The include file *rasterfile.h*, which is needed to compile *ihdr2sun*, is part of the SunOS distribution and is not included with SD-12.

4.8. sunalign <Sun rasterfile>

The program *sunalign* is a program which ensures the Sun rasterfile input has scanlines of length equal to an even multiple of 16 bits. It has been found that some Sun rasterfile applications assume scanlines which end on an even word boundary. IHead images may contain scanlines which do not conform to this assumption. Therefore, it may be necessary to run *sunalign* on an image which has been

converted using *ihdr2sun*. The main routine for the program is found in the file *sunalign.c*. Compilation of this program will also require the Sun include file *rasterfile.h*.

4.9. Other Programs

The software distribution also contains source code for two programs that are not relevant to this database. One of the programs, *xtrctcls*, operates on a file type (CLS) which is not part of SD-12. The other, *htoc*, is a simple utility program for hexadecimal-to-ASCII character conversion.

Manual pages for all 9 of the programs included on the SD-12 CD-ROM are included in Appendix B.

5. Conference

5.1. Introduction

The goals of the First *[6]* and Second *[2]* Census Optical Character Recognition (OCR) Systems Conferences were scientific in nature. The first goal was to gauge the state of the art of OCR of handprinted characters with respect to the particular problems associated with entering census data into a computer database. The second was to learn what is currently limiting the state of the art. The third goal was to determine whether new databases of handprinted characters for use either in training or in testing could be expected to help improve the state of the art of OCR for applications such as the census, and if so, what types of new databases are needed.

Neither Conference was designed to produce results that could be used as the basis for purchasing an OCR system. Decisions regarding the application of an OCR system to any specific task should be based on the results of proctored tests with test materials that are representative of that task.

The tests of the First Conference consisted of classifying approximately 85,000 binary images of properly segmented, isolated characters (roughly 60,000 digits, 12,000 upper case, and 12,000 lower case letters). These activities were carried out from February through May of 1992. The results of the Conference are available on the anonymous FTP server. The test disk (SD-7) and a previous training disk of characters (SD-3) are no longer available. Those data sets and new data are collected on SD-19, available from NIST's Standard Reference Data Group*.

The activities of the Second Conference started in January of 1993, with twenty five organizations agreed to participate. The first and second sets of training materials were shipped to the participants at the end of August and the beginning of October, 1993, respectively. The test materials were shipped to the participants by express carrier to arrive on December 1, 1993. The OCR results returned to NIST for scoring were to be received by the participant's express carrier by December 15, 1993 in order to be considered on-time. However, late results were accepted provided that an express carrier received them by January 31, 1994.

The test was very hard, and many participating organizations either withdrew from participation or did not submit results for scoring before the Conference deadlines. Some participants submitted results only for the test with forms scanned from paper, while others submitted results for both paper and microfilm. The participants returned hypothesis files and field-level confidence files to NIST for scoring on floppy disks, using the same directory structure used on the test CD-ROM.

Each participating organization was asked to fill out a questionnaire about the algorithms used in their OCR system. At the Conference meeting, many participants presented quite detailed descriptions of the systems they used. The presentations were considered more illuminating, so the viewgraphs from the meeting are reproduced in the Conference Report.

5.2. Measures

Two different measures of classification error were calculated for this Conference: the field error rate and the field distance rate *[2,7]*. The field rejection rate is defined as the ratio of the number of fields rejected by the rejection process (based on their field-level confidence values) to the total number of fields presented for classification. The field error rate $R_e(r)$ as a function of field rejection rate r was defined as:

---

* To order: call 301/975-2208, fax 301/926-0416, or send email to srdata@enh.nist.gov

$$R_e(r) = \frac{F_e(r)}{F_c(r) + F_e(r)}$$

where, of the fields not rejected at rate r, $F_e(r)$ is the number recognized incorrectly (even just one character wrong) and $F_c(r)$ is the number recognized correctly.

The field distance $R_d(r)$ as a function of the field rejection rate was defined as:

$$R_d(r) = \frac{C_e(r)}{C_c(r) + C_e(r)}$$

where $C_e(r)$ is the number of characters in the field that are correctly classified when the hypothesis and reference fields are aligned and where:

$$C_e(r) = C_d(r) + C_i(r) + C_s(r)$$

with $C_d(r)$, $C_i(r)$ and $C_s(r)$ representing the respective number of character deletion, insertion, and substitution transformations needed to convert each reference into the associated hypothesis.

The string alignment algorithm adopted was the generalized Levenshtein distance algorithm *[8]*, where the total edit cost is minimized based on edit costs for the operations deletion, substitution and insertion. The costs used were 5, 3 and 1, respectively.

5.3. Summary of Results

Tables 1 and 2 below list the field error and field distance rates at rejection rates of 60, 50, 40, and 0 for both the paper and the microfilm tests for all of the results returned on-time. Table A also gives the field distance and error rates at 0% rejection rate for the 1990 Census human key entry operation (KEY_90) for comparison with the machine results for the paper test.

None of the systems using no segmentation were among the most accurate. This suggests that segmentation is an important subtask for this type of test, at least at the current state of the art. All systems that attempted segmentation except NIST's used intentional oversegmentation as a means of avoiding undersegmentation, and the best performing systems had sophisticated means for recombining segments prior to dictionary-based correction. This suggests that segmentation is the most challenging subtask for this type of test, and that intentional oversegmentation followed by sophisticated recombination methods, possibly in more than one of the downstream subtasks, is the best solution to the segmentation problem at the current state of the art.

Miniforms Scanned from Paper Field Error and Field Distance Rates at Field Rejection Rates								
On-Time System	60%		50%		40%		0%	
CEDAR_0	37.6	20.5	38.7	20.4	41.5	21.3	58.7	37.3
CGK_0	13.8	4.7	19.6	6.2	26.3	9.0	50.5	24.6
ERIM_0	3.6	0.8	6.3	1.6	12.2	4.3	39.7	18.7
ERIM_1	3.9	1.0	7.5	2.4	14.1	5.4	41.9	20.8
HUGHES_0	61.6	26.0	69.2	38.7	74.3	47.5	84.6	63.4
IBM_0	49.6	24.3	56.9	28.8	62.4	32.9	75.0	44.8
IBM_1	53.6	24.7	60.3	29.4	65.0	33.0	76.8	44.8
IBM_2	86.6	58.6	88.4	58.8	89.9	59.4	93.1	63.4
IDIAP_0	10.7	2.6	16.8	5.2	25.8	10.5	52.6	33.4
NIST_0	46.6	16.2	53.8	21.7	60.1	27.2	75.3	46.2
UBOL_0	57.1	36.0	61.8	38.6	64.9	39.9	71.7	43.1
KEY_90	N/A	N/A	N/A	N/A	N/A	N/A	8.5	1.6

Table 1. System Performance on Forms Scanned from Paper.

The complete results, including graphs of performance versus field rejection rate, and an in-depth discussion are in the Conference Report. In addition, analysis of a hybrid system's (created using ERIM_0 and 1990 Census hypotheses) and a voting system's (created using ERIM_0, IDIAP_2 and CGK_2) hypotheses are included.

Miniforms Scanned from Microfilm Field Error and Field Distance Rates at Field Rejection Rates								
On-Time System	60%		50%		40%		0%	
CGK_0	23.4	7.1	31.6	11.3	38.7	15.5	60.7	32.6
ERIM_0	9.7	5.0	16.2	7.5	24.6	11.8	50.0	25.7
ERIM_1	10.1	5.2	16.7	8.0	25.4	12.4	50.9	26.7
IBM_0	66.8	36.6	71.6	40.6	75.1	43.7	83.7	53.3
IBM_1	69.5	36.7	73.8	40.4	77.2	43.9	85.1	53.4
IBM_2	91.3	65.1	92.6	65.3	93.3	64.9	95.6	66.9
NIST_0	77.3	42.3	81.4	48.2	84.3	52.4	90.4	62.6
UBOL_0	70.8	47.6	73.6	48.5	77.1	52.4	82.0	55.4

Table 2. System Performance on Forms Scanned from Microfilm.

6. FTP Server

The Visual Image Processing Group of NIST's Computer Systems Laboratory makes the reports from the Second Census Conference and some files associated with this set of databases available to the public via an anonymous FTP server, *sequoyah.ncsl.nist.gov*. The files listed below are located in the directory *pub/ocr_conf_2*.

In addition, the server also holds files associated with the First Census Conference (*pub/ocr_conf_1*), most of the Internal Reports produced by the Group (*pub/nist_internal_reports*), preprints of other papers (*pub/preprints*), a listing of standard image databases produced by the Group and available on CD-ROM (*pub/databases/catalog.txt*), samples of some standard image databases (*pub/databases/data*), and some source code as well.

6.1. announce.tar.Z

Compressed tar file containing text files that describe the Second OCR Systems Conference task, file formats, etc. It is a copy of the subdirectory *doc/announce* from the Special Database 11 CD-ROM.

6.2. samples_1.tar

Tar file containing a directory with a sampling of Industry and Occupation miniform images. It has the same directory structure that was used for Special Databases 11 and 12, the Second OCR Systems Conference training databases. The subdirectory *data* contains images from microfilm (100 files, 500 miniforms, 1500 total fields) and no reference files, while the subdirectory *data3* contains images from paper (60 files, 300 miniforms, 900 total fields) and the corresponding reference files.

6.3. samples_2.tar

Tar file containing a directory with a sampling of Industry and Occupation miniform images. It is a copy of the subdirectory *data* from the Special Database 11 CD-ROM. The directory contains images from microfilm only (200 files, 1000 miniforms, 3000 total fields) and the corresponding reference files.

6.4. refs_paper.tar.Z

Compressed tar file containing reference files for the paper Conference test images (9000 total fields) on SD-13.

6.5. refs_microfilm.tar.Z

Compressed tar file containing reference files for the microfilm Conference test images (9000 total fields) on SD-13.

6.6. ir_5452.pt1.ps.Z

Compressed PostScript document containing the first 103 pages of the Second Census OCR Systems Conference report.

6.7. ir_5452.pt2.tar.Z

Compressed tar file containing the 158 PostScript files that contain pages 104-261 from the Second Census OCR Systems Conference report. Those pages are mainly viewgraphs from Conference participants and plots of their system performance curves at many rejection rates. Many of the files are several megabytes in size because they are composed of several digitized viewgraphs. So that printer spool directory capacity is not exceeded on UNIX systems, it is recommended that these pages be printed one at a time in a loop that either waits for the queue to empty or sleeps for the typical printing time for a large file (a few minutes on a SparcPrinter, but dozens of minutes on an older printer such a LaserWriter II).

6.8. ir_4912.ps.Z

Compressed PostScript document containing the First Census OCR Systems Conference report.

References

[1] CCITT, "Facsimile Coding Schemes and Coding Control Functions for Group 4 Facsimile Apparatus", 1984.

[2] J. Geist *et al*, The Second Census Optical Character Recognition Systems Conference, NIST Internal Report 5452, National Institute of Standards and Technology, Gaithersburg, MD 20899, May 1994.

[3] R. A. Wilkinson, Dictionary Production for Census Form Conference, NIST Internal Report 5180, National Institute of Standards and Technology, Gaithersburg, MD 20899, April 1993

[4] PKWARE, Inc., General Format of a ZIP File, File *appnote.txt*, included in PKZIP 1.10 distribution.

[5] M. D. Garris and S. A. Janet, NIST Scoring Package User's Guide, Release 1.0, NIST Internal Report 4950, National Institute of Standards and Technology, Gaithersburg, MD 20899, October 1992.

[6] R. A. Wilkinson *et al*, The First Census Optical Character Recognition Systems Conference, NIST Internal Report 4912, National Institute of Standards and Technology, Gaithersburg, MD 20899, July 1992.

[7] M. D. Garris, Methods for Evaluating the Performance of Systems Intended to Recognize Characters from Image Data Scanned from Forms, NIST Internal Report 5129, National Institute of Standards and Technology, Gaithersburg, MD 20899, February 1993.

[8] D. Sankoff and J. B. Kruskal (Editors), *Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison*, Reading, MA: Addison-Wesley, 1983.

Appendix A:
Example Form Reference File
SD-11 data0/d00/d00f01.ref

```
r00_f01 SODA POP MANUFACTURING
r00_f02 PROJECT MANAGER
r00_f03 CONSOLIDATION PROJECT MGR
r01_f01 FOOD MANUFACTURING
r01_f02 MARKETING BRAND MANAGER
r01_f03 MANAGING 3 FOOD BRANDS
r02_f01 ADOLEC TREATMENT CENTER
r02_f02 CHILD CARE COUNSLER
r02_f03 PATIENT CARE
r03_f01 HOSPITAL
r03_f02 REG NURSE
r03_f03 PATIENT CARE
r04_f01 BANK SECURITY PRODUCTS
r04_f02 SERVICE WORK
r04_f03 SERVICE BANK EQUIPMENT
```

Hypothetical Corresponding Hypothesis File
data0/d00/d00f01.hyp

```
r00_f01 SODAPOP MANUFACTURING
r00_f02 PROJECT MANAGER
r00_f03 CONSULTATION PROJECT MR
r01_f01 FOOD MANUFACTURING
r01_f02 MARKETING MANAGER
r01_f03 MANAGING 8 FOOD BRANDS
r02_f01 ALCOHOL TREATMENT CENTER
r02_f02 CHILDCARE COUNSELOR
r02_f03 PATIENT CARE
r03_f01 HOSPITAL
r03_f02 REG NURSE
r03_f03 PATIENT CARE
r04_f01 BANK SECURITY PRODUCTS
r04_f02 SERVICE WORK
r04_f03 SERVICE BANK
```

Hypothetical Corresponding Field-Level Confidence File
data0/d00/d00f01.con

```
r00_f01 0.80
r00_f02 0.45
r00_f03 0.33
r01_f01 0.67
r01_f02 0.70
r01_f03 0.93
r02_f01 0.60
r02_f02 0.88
r02_f03 0.77
r03_f01 0.86
r03_f02 0.82
r03_f03 0.55
r04_f01 0.73
r04_f02 0.94
r04_f03 0.89
```

Appendix B:
Manual Pages for Supplied Software

NAME

chk_hyps - is a program that checks the form of the hypothesis subdirectories and files for the 2nd Census OCR Systems Conference.

SYNOPSIS

chk_hyps *<path> <number>*

<path> is the complete path to the directories containing the hypothesis files

<number> is the number of directories present at the path specified above

DESCRIPTION

Chk_hyps checks

1) for the existence of each expected file in each expected subdirectory;

2) for the existence of each expected line of each file;

3) that the LINE_ID is correct for each expected line of each file;

4) that each LINE_ID is followed by one, and only one, space character;

5) that each line (except the last) of each file ends in a NEW_LINE code;

6) that all characters between the LINE_ID and the NEW_LINE code are from the set consisting of the ASCII digits, the ASCII upper case English letters, and the ASCII space character;

7) that the last character before each NEW_LINE code is not an ASCII space;

8) that each file has no extra lines.

EXAMPLES

chk_hyps /pub/ind_occ3/data0 25

BUGS

None known.

NAME

decomp - general decompression of NIST Ihead Raster images

SYNOPSIS

decomp *<IHead file in> <IHead file out>*

DESCRIPTION

Decomp takes a compressed ihead image file, examines the compression field of the header, and applies the correct decompression algorithm storing the results as a decompressed ihead image in the out file.

OPTIONS

IHead file in

Any NIST IHead raster image *IHead file out* Any NIST IHead raster image

EXAMPLES

decomp foo.pct fool.pct

FILES

/usr/share/include/ihead.h

NIST's raster header include file

SEE ALSO

dumpihdr(1)

DIAGNOSTICS

Decomp exits with a status of -1 if opening **ihdrfile** fails.

BUGS

NAME

dumpihdr - takes a NIST IHead image file and prints its header content to stdout

SYNOPSIS

dumpihdr *ihdrfile*

DESCRIPTION

Dumpihdr opens a NIST IHead rasterfile, formats the header contents and prints the header information to stdout.

OPTIONS

ihdrfile any NIST IHead image file

EXAMPLES

dumpihdr foo.pct

FILES

ihead.h NIST's raster header include file

SEE ALSO

ihdr2sun(1)

DIAGNOSTICS

Dumpihdr exits with a status of -1 if opening **ihdrfile** fails.

BUGS

NAME

fragmis - fragments a NIST IHead multiple image set into unique NIST IHead files.

SYNOPSIS

fragmis *<IHead mis file> <rootname>*

DESCRIPTION

Fragmis fragments a NIST IHead multiple image set into NIST IHead files containing only one image per file and giving each file a unique name. The name is generates using the rootname and the images index in the multiple image set.

OPTIONS

ihdrfile Any ihead raster image

EXAMPLES

decomp foo.pct foo1.pct

FILES

/usr/share/include/ihead.h
NIST's raster header include file

SEE ALSO

dumpihdr(1)

DIAGNOSTICS

Decomp exits with a status of -1 if opening **ihdrfile** fails.

BUGS

NAME

htoc - takes a hexadecimal value and returns to standard output the ASCII equivalent value.

SYNOPSIS

htoc *<hex value>*

DESCRIPTION

Htoc is a hexadecimal to character converter. **Htoc** takes the hex value as input and converts it to the ASCII equivalent value then returns this value to standard output.

OPTIONS

hex value

Any two digit hexadecimal string

EXAMPLES

htoc 30

FILES

DIAGNOSTICS

BUGS

NAME

ihdr2sun - takes an NIST IHead binary raster file and converts it to a Sun rasterfile

SYNOPSIS

ihdr2sun *ihdrfile*

DESCRIPTION

Ihdr2sun converts an NIST IHead binary raster file to a Sun rasterfile. The Sun image file created will have the root name of **ihdrfile** with the extension **.ras** appended.

OPTIONS

ihdrfile any IHead binary image

EXAMPLES

ihdr2sun r0000_00.pct

FILES

rasterfile.h	Sun's raster header include file
ihead.h	NIST's raster header include file

SEE ALSO

rasterfile(5)

DIAGNOSTICS

Ihdr2sun exits with a status of -1 if opening **ihdrfile** fails.

BUGS

NAME

sunalign - takes a sun rasterfile and word aligns its scanlines

SYNOPSIS

sunalign *sunrasterfile*

DESCRIPTION

Sunalign takes the file **sunrasterfile** and determines if the stored scan lines in the file require word alignment. If so, the command overwrites the image data making scan lines word aligned. This command is useful when taking clipped images from the HP Scan Jet and importing them into Frame Maker.

OPTIONS

sunrasterfile

any sun rasterfile image

EXAMPLES

sunalign foo.ras

FILES

/usr/include/rasterfile.h

sun's raster header include file

SEE ALSO

rasterfile(5)

DIAGNOSTICS

Sunalign exits with a status of -1 if opening **sunrasterfile** fails.

BUGS

NAME

xtrctcls - takes a cls file and an index into that file and prints out the value for that index in the cls file.

SYNOPSIS

xtrctcls -{*c,h*} <*clsfile*> <*index*>

DESCRIPTION

Xtrctcls takes the **cls file** and an index into the file and prints out the value associated with that index in the cls file. One of the two options must be specific, either -c or -h. The -c option prints the value in ASCII character form. The -h option prints the value in hexadecimal form.

OPTIONS

xtrctcls -h foo.cls 3

FILES

DIAGNOSTICS

Xtrctcls exits with a status of -1 if opening **cls file** fails.

BUGS

NAME

xtrctmis - takes a NIST IHead multiple image set file and an index into that file and copies the image at the indexed position into the outfile.

SYNOPSIS

xtrctmis *<IHead MIS file> <outfile> <index>*

DESCRIPTION

Xtrctmis takes the **NIST IHead MIS file** and an index into that file and copies in NIST IHead format the image at the indexed position into the outfile.

OPTIONS

xtrctmis foo.mis foo1.pct 3

FILES

DIAGNOSTICS

Xtrctmis exits with a status of -1 if opening **mis file** fails.

BUGS