ภาคผนวก

ภาคผนวก ก

คำอธิบายโปรแกรม

คำอธิบายโปรแกรม

โปรแกรมข้างล่างต่อไปนี้เป็นโปรแกรมที่ใช้ในการทดสอบการรู้จำลายนิ้วมือซึ่งแบ่งการทำงานเป็น 2 ส่วน ส่วนแรกที่เกี่ยวข้องกับขบวนการที่เกี่ยวกับภาพทั้งหมดจะทำการคำนวนโดยใช้ MATLAB และส่วนที่สองเกี่ยวข้องกับขบวนการรู้จำและการจำแนกลายนิ้วมือจะคำนวนโดยใช้ ภาษาซี

ชื่อโปรแกรม	คำอธิบายโปรแกรม
pcttomat.c	เป็นโปรแกรมแปลี่ยนรูปแบบของภาพจาก .pct เป็น .mat เพื่อใช้ในการแปลงภาพที่รับเข้ามา ให้สามารถแสดงรูปลายนิ้วมือและใช้ในการปรับปรุงภาพและขบวนการแยกคุณลักษณะ
fngrcg.c	เป็นโปรแกรมที่ใช้ในการสอนโครงข่ายประสาทเทียมแบบแพร่กลับให้รู้จำรูปแบบของลายนิ้วมือที่ใช้ในการสอนโครงข่าย
fngidn.c	เป็นโปรแกรมที่ใช้ในการจำแนกลายนิ้วมือโดยการจำค่าน้ำหนักที่ได้จากโปรแกรมการสอนโครงข่าย
variance.m	เป็นโปรแกรมบน MATLAB ที่ใช้ในขบวนการปรับปรุงภาพเบื้องต้นโดยวิธีที่เรียกว่า วิธีการแยกพื้นหลังกับลายนิ้วมือโดยอาศัยคุณสมบัติการกระจายของข้อมูล
lthresh.m	เป็นโปรแกรมบน MATLAB ที่ใช้ในขบวนการปรับปรุงภาพเบื้องต้นโดยวิธีที่เรียกว่า วิธีการทำภาพสองระดับโดยการหาตำแหน่งจุดแยกในบล็อคย่อยๆ
direction.m	เป็นโปรแกรมบน MATLAB ที่ใช้ในขบวนการแยกคุณลักษณะโดยวิธีที่เรียกว่า วิธีการเข้ารหัสทิศทาง
orientation.m	เป็นโปรแกรมบน MATLAB ที่ใช้ในขบวนการแยกคุณลักษณะโดยวิธีที่เรียกว่า วิธีการหาสนามทิศทาง

<u>หมายเหตุ</u> ภาพที่ได้จากโปรแกรม pcttomat.c ต้องทำการบีบข้อมูลจาก 512x512 จุดภาพ เป็น 256x256 จุดภาพ ทั้งนี้เนื่องจากต้องการความรวดเร็วในการคำนวนโดยที่ไม่สูญเสียข้อมูลที่สำคัญๆ ดังนั้นโปรแกรมอื่นๆที่เหลือจะอ่านภาพขนาด 256x256 จุดภาพ

ภาคผนวก ข

รายละเอียดของภาพจากฐานข้อมูล NIST ชุดที่ 4

รายละเอียดของภาพจากฐานข้อมูล NIST ชุดที่ 4

ภาพที่ใช้ในงานวิทยานิพนธ์เป็นภาพที่ได้จาก NIST(National Institute of Standards and Technology) ชุดที่ 4 ซึ่งสามารถค้นได้จาก

ftp://sequoyah.nist.gov/pub/databases/data/prints4.tar.Z มีรายละเอียดของข้อมูลดังต่อไปนี้

Fingerprint Samples for NIST Special Database 4
Fingerprint Image Groups

C.I. Watson
National Institute of Standards and Technology
Advanced Systems Division
Image Recognition Group
November 19, 1992

1.0 INRODUCTION

This file describes the Fingerprint Samples which contain 8-bit gray
scale images of randomly selected fingerprints. The fingerprints are being made available as a sample of what is contained in the NIST fingerprint database (NIST Special Database 4). The samples contain 50 (25 pairs) uncompressed fingerprints stored in NIST's IHead raster data format. Each print is 512 X 512 pixels with 32 rows of white space at the bottom of the print.

The fingerprints are classified into one of five categories (L = left loop, W = whirl, R = right loop, T = tented arch, and A = arch) with an equal number of prints from each class (5). All classes are stored in the NIST IHead id field of each file.

2.0 TAR FILE

The fingerprint data and software are stored in a tar file (prints4.tar, These are the same prints that were in the original file prints.tar) which is compressed with the UNIX compress utility. When untarred the file will create the main directory prints4. The fingerprint data will be in the subdirectory "data" and the software will be in the subdirectory "src." The filenames for the fingerprint data range from f01.pct - f25.pct with the matching second rolling in s01.pct - s25.pct.

3.0 FINGERPRINT FILE FORMAT

After digitization, certain attributes of an image are required to correctly interpret the 1-dimensional pixel data as a 2-dimensional image. Examples of such attributes are 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 which manipulates the raster data of an image is able to first read the header and determine the proper interpretation of the data which follows it.

Numerous image formats exist, but most image formats are proprietary. Some are widely supported on small personal computers and others on larger workstations. A header format named IHead has been developed 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. 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, to represent images captured from different scanners and cameras, and to satisfy the image requirements of diversified applications including, but not limited to, image archival/retrieval, character recognition, and fingerprint classification. Figure 1 illustrates the IHead format.

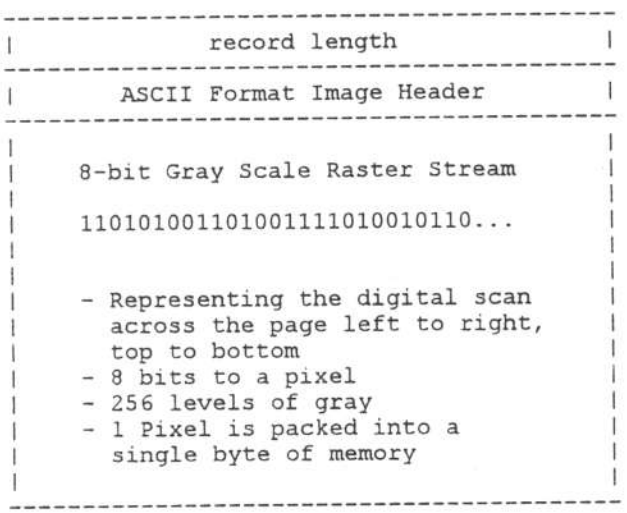

Figure 1: An illustration of the Ihead raster file format.

Since the header is represented by the ASCII character set, IHead has been successfully ported and tested on several systems including UNIX workstations and servers, DOS personal computers, and VMS mainframes. All attribute fields in the IHead structure are of fixed length with all multiple character fields 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. A fixed-length field containing the size in bytes of the header is prefixed to the front of an IHead image file as shown in Figure 1.

```
/*************************************************************/
/*          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 2: The IHead `C' programming language structure definition.

The IHead structure definition written in the `C' programming language is listed in Figure 2. Figure 3 lists the header values from an IHead file corresponding to the structure members listed in Figure 2. This header information belongs to the sample file f01.pct. Referencing the structure members listed in Figure 2, the first attribute field of IHead is the identification field, id. This field uniquely identifies the image file, typically by a file name. The identification field in this example not only contains the image's file name, but also the classification of the fingerprint, any references to another classification, and the sex of the individual. This convention enables an image recognition system's hypothesized classification to be automatically scored against the actual classification printed in the id field.

```
IMAGE FILE HEADER
~~~~~~~~~~~~~~~~
Identity            :  f2001_06.pct A  M
Header Size :  288 (bytes)
Date Created:  TUE NOV 25 10:26:29 1992

Width       :  512 (pixels)
Height            :  512 (pixels)
Bits per Pixel    :  8
Resolution  :  500 (ppi)
Compression :  0 (code)
Compress Length   :  0 (bytes)
Scan Alignment    :  8 (bits)
Image Data Unit   :  8 (bits)
Byte Order  :  High-Low
MSBit       :  First
Column Offset     :  0 (pixels)
White Pixel :  255
Data Units  :  Unsigned
Scan Order  :  Row Major,
                       Top to Bottom,
                       Left to Right
Parent            :  a1736.pct
X Origin          :  0 (pixels)
Y Origin          :  0 (pixels)
```

Figure 3: The IHead values for the fingerprint sample file f01.pct.

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, 19.6850 pixels/mm (500 pixels/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 which signifies which compression technique, if any, has been applied to the raster image data which 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 complen field must be used in addition to the image's pixel dimensions.

The attribute field, align, stores the alignment boundary to which scan lines of pixels are padded. Pixel values of 8-bit gray scale images are stored 1 pixel (or 8 bits) to a byte, so the images will automatically align to an even byte boundary.

The next three attribute fields identify data interchanging issues among heterogeneous computer architectures and displays. The unitsize field specifies how many contiguous bits 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, data incompatibilities across computer hardware and data format assumptions within application software can be identified and effectively dealt with.

The pix_offset attribute defines a pixel displacement from the left edge of the raster image data to where a particular image's significant image information begins. The whitepix attribute defines the value assigned to the color white. For example, the gray scale image described in Figure 3 is gray print on a white background and the value of the white pixel is 255. 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 images used in this database were mixed and renamed from their original filenames and the `link' to the original filename was stored in the parent field. 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 successfully manage binary and gray scale images.

4.0 INCLUDED SOFTWARE

Included with the fingerprint images is software written in the `C' programming language. Three programs are included in the src directory: dumpihdr, ihdr2sun, and ihdr2raw. These routines are provided as an example to software developers of how IHead images can be manipulated and used. Descriptions of these programs and their subroutines are given below.

4.1 Compilation

After coping the files in the src directory, executable binaries can be produced by invoking the UNIX utility make to execute the included makefile. An example of this command follows.

```
# make -f makefile.mak
```

4.2 Dumpihdr <Ihead file>

Dumpihdr is a program which 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 report shown in Figure 3 was generated using this utility. The main routine for dumpihdr is found in the file dumpihdr.c and calls the external function readihdr().

Readihdr() is a function 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 is listed in Figure 2 and is found in the file ihead.h

4.3 Ihdr2sun <Ihead file>

Ihdr2sun converts an image from NIST IHead format to Sun rasterfile format. Ihdr2sun 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 this program is found in the file ihdr2sun.c and calls the external function ReadIheadRaster() which is found in the file rasterio.c.

ReadIheadRaster() is the procedure responsible for loading an IHead image from a file into main memory. 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. ReadIheadRaster() returns an initialized IHead structure, the uncompressed raster data, the image's width and height in pixels, and pixel depth.

Writeihdrfile() is a routine that writes an IHead image into a file. This routine opens the passed filename and writes the given IHead structure and corresponding data to the file. Writeihdrfile() is found in the src file rasterio.c.

4.4 Ihdr2raw <Ihead file> <out file>

Ihdr2raw converts an image from NIST Ihead format to "raw" image data. This routines loads an Ihead formatted image from a file and writes the data to the specified output file with no header information. REMEMBER THE CLASSES ARE LOCATED IN THE HEADER OF EACH FILE AND WILL NOT EXIST IN THE "RAW" FILE.

Any questions send to:

Craig Watson
phone => (301) 975-4402
e-mail => craig@magi.nist.gov