COMPUTER GRAPHICS AND IMAGE PROCESSING 8, 1–15 (1978)

KANDIDATS: An Interactive Image Processing System

ROBERT M. HARALICK AND GARY MINDEN

The University of Kansas, Lawrence, Kansas 66044

Received December 6, 1976; accepted April 19, 1977

KANDIDATS is a comprehensive digital image processing system that interacts with the user at a command string level. It includes prompting for parameter input and checks user input for errors. Image analysis operations available in KANDIDATS consist of utility functions, image transforms, spatial clustering, and Bayesian classification. The versatility and capabilities of KANDIDATS arise from a modular programming structure and file structure. These attributes allow processing of images that are a few thousand rows by a few thousand columns in a minicomputer system with only 32K words of main memory.

1. INTRODUCTION

Image data are collected in the course of scientific experiments, medical tests surveillance operations, and satellite and telescopic photography. To an increasing extent, these data must be processed by computer before they can be interpreted by humans. Computer processing has evolved over the last decade from the use of specialized machines to more flexible program packages using sophisticated full-scale computer systems. This move toward computer processing of images has been necessitated by the large volume of image data being produced at the present. One example is NASA's LANDSAT satellite.

Image processing encompasses all the various operations that can be applied to photographic or image data. These include, but are not limited to, image compression, image restoration, image enhancement, preprocessing, quantization, spatial filtering, and other image pattern recognition techniques. Interactive image processing refers to the use of an operator or analyst at a console with a means of assessing, preprocessing, feature extracting, classifying, identifying, and displaying the original imagery or the processed imagery for his subjective evaluation and further interaction.

Image processing software must resolve some of the most difficult problems confronting programmed systems: handling large amounts of data and heavy and varied computational loads. These problems require sophisticated and complex software for even the simplest image processing tasks.

During the past decade a number of multi-image picture processing software packages have been put together. Among them are VICAR at the Jet Propulsion

Laboratory, IDIMS at the Electromagnetic Systems Laboratory, DIMES at the U.S. Army Engineer Topographic Laboratory, LADIES at the Los Alamos Scientific Laboratory, KANDIDATS at the University of Kansas, LARSYS at Purdue University, System 101 at Stanford Technology Corporation, ORSER at Pennsylvania State University, PIXSYS at Oregon State University, AMIDS at the Rome Air Development Center, VL at Scripps Institute of Oceanography, MIDAS at Environmental Research Institute of Michigan, WALDIPS at NASA Wallops Island, XAP at the University of Maryland, and SCIMPL at the University of Southern California, to name a few [1–18]. Yet despite the need which the various different image processing locations have for sharing and adopting similar conventions for image data structure and image input/output interface routines, there has been relatively little intercommunication between the users either on a one-to-one basis through correspondence or on a one-to-many basis through the software journals. In the hope of beginning such an exchange, we discuss the structure and design philosophy of KANDIDATS, the interactive image processing system developed at the University of Kansas.

2. SOFTWARE SYSTEM DESIGN PERSPECTIVE

The software of an image processing system includes processing programs, a monitor that directs the flow of control and information between subsystems, and a data structure for digital images. The processing of an image takes place in a series of discrete steps, each resulting in an image that may be used later in the sequence. The monitor initiates each step by calling it into action and providing the necessary information for it to run to completion.

2.1. Monitor

The monitor is the interface between the operator and the image processing system. The monitor and its subsystems should assume responsibility for as much bookkeeping as possible in order to free the operator from routine tasks. However, it should be flexible enough to allow the operator to control these tasks if necessary. The monitor must also examine all operator input to ensure to the largest extent possible that all his entries are free of error and clearly inform him of any that are not.

2.2. Processing Programs

Processing programs are concerned only with operating on data. No direct FORTRAN binary I/O appears in the processing programs. Rather, these programs access images through a standard set of access I/O subroutines. Therefore, these programs know nothing about the image I/O environment they are in. This makes them more useful in interactive, batch, or special purpose systems and facilitates transportability between different users and computer systems. See [19] for a more detailed discussion of the I/O routines.

Because these programs know nothing about their environment, they do not attempt extensive error handling; some indication is merely returned to the calling program if an error condition occurs. Parameters input to processing programs are checked for errors and if any are found the appropriate error indication is returned.

2.3. Data Structure

The structure for accessing image data on peripheral devices is important to the implementation, efficiency, and responsiveness of the processing programs. The data structure should support the many formats used in digital image processing including multiple images, several data types, and ground truth information. The programmer needs to be able to access one band at a time, all bands, all numeric bands, or all ground truth bands from a multi-image set. Both sequential and random access should be possible. Finally, processes should not be limited to accessing images by row only. Some processes, especially those concerned with context dependent spatial processing, work much better with rectangular or square subimages, e.g., 64 rows by 64 columns.

3. KANDIDATS

3.1. KANDIDATS Monitor

The KANDIDATS monitor is separated into two parts. There is a resident portion that dispatches control to other parts of the monitor or to process devices, and the command string interpreter. The resident monitor calls the command string interpreter to accept and decode a user-input string. The command string interpreter returns the command, input and output image names and option flags. The resident monitor then uses the command name to call the appropriate process driver.

The command interpreter accepts text strings in the following syntax:

```
command  [dest. device]  [dest. image]  (option flags) [←]
[src. device]  [src. image]  [src. image]]  [(option flags)]
```

Depending on the command, some of these fields can be optional. The interpreter operates as a top down parser. That is, initially the parser expects a command name in the text string, then a device name, and then a destination image name. As the parts of the command are found the tokens are placed in a labeled common area in the resident monitor for use by the process drivers.

There are some checks that the command interpreter performs. For example, image names, if given, are checked for correct syntax; device names, if given, must be valid or known to the monitor. However, it is the job of the process drivers to make sure all the necessary parts of the command string were entered. With this arrangement each process determines what is required to complete the request. Thus, tables describing the requirements of each process and the associated scanners are not needed. Required items not given by the user are asked for by the process driver. Extraneous items are ignored.

The linear quantizing process, for example, requires three items from the user: source image, destination image, and number of desired quantized levels. Any of the following text strings would be accepted by the command string interpreter:

```
#:  QUANT  OUT  IMG←IN  IMG
#:  QUANT  DP   OUT   IMG←IN  IMG
#:  QUANT
#:  QUANT  OUT  IMG
```

The command string interpreter does not have a facility for entering process parameters. So, in all cases the QUANT driver would ask for the number of quantized levels. The first two example cases are essentially complete and require no further input. In the third case both source and destination image names are missing and these would be asked for from the user. The fourth case requires the user to supply a source image name.

The organization of command input has several advantages. The inexperienced user inputs what he thinks is necessary. If additional items are required or items were misentered, e.g., misspelled, the system will ask for a new input string.

Users can set up command strings and parameters on standard ASCII source files which are called run files. The monitor can be instructed to input commands from a run file rather than the user console. Thus, a simple batch facility is available. This facility can also provide elementary macro processes for image processing. This is because, in run mode, the command drivers ask the user to correct errors. Hence, if only the initial source image and final destination image are left out of the command string from the run file, the command driver will ask the user for the first image name. The user then can supply the name of the image to be processed and the system will continue from that point performing the steps specified in the run file until the last command, whereupon the command driver will ask for the final destination image name.

The process driver handles one other facility, the user interrupt. Before calling the process program a user interrupt address is supplied to the system. The user can interrupt the process at any time with control going to the last interrupt address specified. Images files are typically closed and control returned to the resident monitor at this time.

3.2. KANDIDATS Processing Programs

There are many different algorithms that can be applied to digital images. KANDIDATS contains over 100 image processing functions applicable in the remote sensing/land use classification area. This is by no means exhaustive of the possible algorithms and new algorithms are constantly being added.

Appendix 1 gives an outline of the image processing capabilities of KANDIDATS.

The process drivers insure that all items necessary for processing have been input and put this information in suitable form for the processing program. The necessary items are found in a resident labeled common area where they were left by the command string interpreter. Destination and source devices and images are checked for correctness. The driver allows the user to reenter a source image name if it cannot be found in the system file structure. Default devices are assigned for the process if none were given in the command text string. When all items are acceptable the driver proceeds to ask the user for any additional required processing parameters. Prompting messages are output if the system is not set to read parameters from the run file. A standard set of question/answer subroutines is used to interact with the user. This means that questions are standard throughout the system. The question/answer subroutines check all user input for consistency with the current task. When an error is detected the question is reasked at the console device, even when input is from a run file. Thus, the user can correct errors in a run file without aborting the process and restarting. After all parameters have been entered and checked for errors, the processing program is called.

Processing programs are treated somewhat like functions on images. A processing program is given input image(s) and parameters as operands and produces an output image as a result. These programs are implemented in a structured manner, i.e., one function per program module. The one entry/one exit rule has been relaxed somewhat to allow both a normal and an error exit. Processing programs must access the image and check the image and parameters for consistency. Error status is set and the error return is taken when some error is detected by the program. The error return saves the calling program from testing the error indicator after each call. Error status values are standard throughout the system.

All programs, including those in the monitor, push their names on a stack upon entry and pop their names just prior to normal exit. This is an aid for the system programmer or debugger. The stack maintains a record of the routines entered in the order entered. If an error occurs the stack can be dumped to find the trace of program calls.

Processing programs allocate available memory dynamically. Thus the size of images is not fixed but is limited only by available memory. Typically, images up to 1000 elements per subimage can be processed without special effort in a 24K memory of 18 bit words.

3.3. KANDIDATS Digital Images

Digital images of interest may be as large as a thousand rows by a thousand columns by several bands. Even on large computer systems it would be difficult to keep the entire image in main memory; on a minicomputer it is impossible. We must be satisfied with having only a small portion, say N rows by P columns, of the image in main memory at any one time while the remaining portion is on a peripheral device. The image data structure and access methods must enable us to read or write reasonable portions or blocks of the image in a random or sequential fashion.

KANDIDATS maintains digital image information in a form known by the acronym "SIF" (Standard Image Format). This standard form of digital image representation establishes a data base that digital image-processing routines can interact with on an independent basis. SIF allows multidigital-images containing both grey tone and symbol map information as well as processing history information to be maintained via KANDIDATS system routines. The system routines permit the creation of image files in a manner that gives the user easy access to image data and minimizes storage space. All image access by processing programs is through these system routines. Images can be maintained in four data modes: integer, floating point, double integer, and double precision. Images may be manipulated either in sequential or direct access input/output. Sequential image manipulation directs information to and from the image in the sequence in which the information is physically recorded on the storage device. Direct access image manipulation allows access of image information in a random fashion.

The format of a KANDIDATS standard image file has several nice features. The file structure allows easy access to any single band of a multiband image without accessing any other bands. Alternatively several bands can be accessed at one time without much physical movement of the disk access arm. Allowing the image to be blocked in a variety of ways reduces the memory requirements when spatial neighborhoods such as 64 rows by 64 columns need to be in main memory at the same time.

The SIF multi-image file exists on the various storage devices as a binary file with fixed length records. The length of the record is determined by the amount of image data to be stored in a logical record and the length of the variable length byte in which grey tone intensity is encoded. Logical records are separated into three basic categories. They are:

1. *The identification record.* This record contains information with respect to the file size, data mode, and format in which the image data is recorded.
2. *The descriptor records.* These records provide:
   (a) A history of image processing that has led to the creation of the current image file, and
   (b) Information about the image or file that cannot be stored in the identification or descriptor records.
3. *The image data.*

3.3.1. The Identification Record

The identification record contains twenty 18-bit words of pertinent information about the SIF file. This places a lower bound on the logical record size of the file. These twenty words of information are ordinarily stored in an array during the period when a particular SIF file is being manipulated. The contents of the IDENT array include:

(1) Coordinates of the image on the display screen.
(2) Size of the image.
(3) Relative size of a resolution cell.
(4) Number of descriptor records.
(5) Minimum, maximum, number of greytones, number of bits per grey tone.
(6) Total number of bands and number of symbolic bands.
(7) Number of rows and columns in each subimage block or logical record.
(8) Data mode (absolute binary, two's complement binary, floating, etc.).

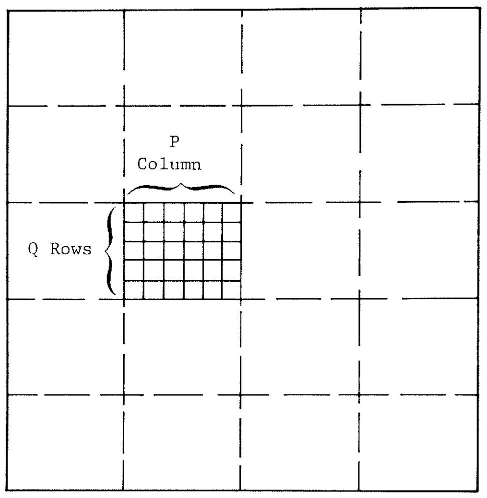

FIG. 1. Illustrates an image and a Q row by P column subimage block. Each logical record is a subimage block. The subimage blocks are mutually exclusive and are all of the same size with the exception of those blocks on the bottom or right-hand side of the image. These blocks may be of smaller size.

3.3.2. The Descriptor Records

Descriptor records, if any, follow the identification record. Descriptor records tell the entire processing history of the image: which commands were executed, when they were executed, what they were executed on, and with what parameters they were executed. The number of descriptor records is given in the identification array. Each descriptor record contains a maximum of twenty words of actual information. This ensures that as descriptor records are copied from file to file, no information is lost as the logical record length changes. Descriptor records provide general information about the image as well as a detailed history of the processing steps performed on the image data. Thus there are two distinct classes of descriptor records:

(a) Processing history descriptor records.
(b) Free format information descriptor records.

Processing history descriptor records. Processing history records provide a detailed account of the action performed by one processing routine on the file. These records are arranged so that they indicate the name of the processing routine, the date of the processing, and the name(s) of the input file(s).

Ancillary data descriptor records. Information pertinent to the image is also stored in descriptor records. Routines record the parameters used when processing an input image to an output image. Data gathered by the routine, e.g., a mean vector and covariance matrix for each category in an image, can also be placed in descriptor records.

Free format records (*optional*). Free format records are ASCII records beginning with the 5-character string "*□□□□." The remaining 45 characters (18 words)

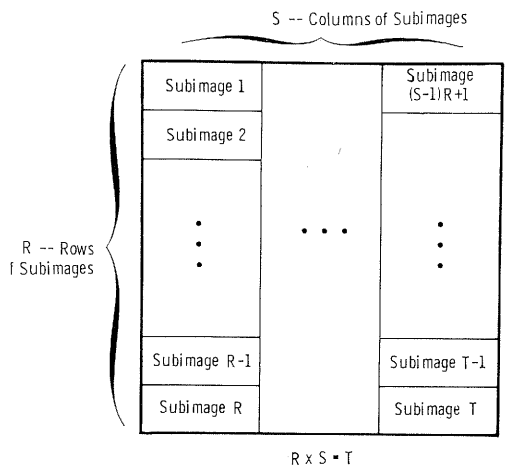

FIG. 2. Illustration of the manner in which an image is covered by subimages.

of the record contain general information about the SIF image file. Free format records can be placed anywhere within the entire descriptor record set as long as they do not interfere with the structure of the processing history descriptor records.

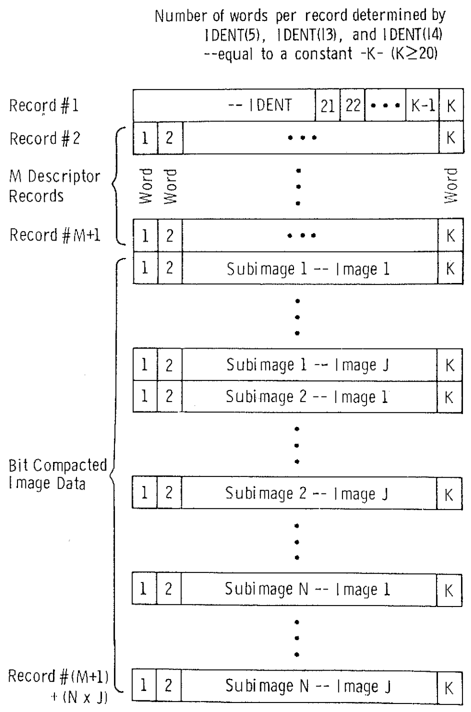

FIG. 3. Illustration of a standard image file on the diskpack.

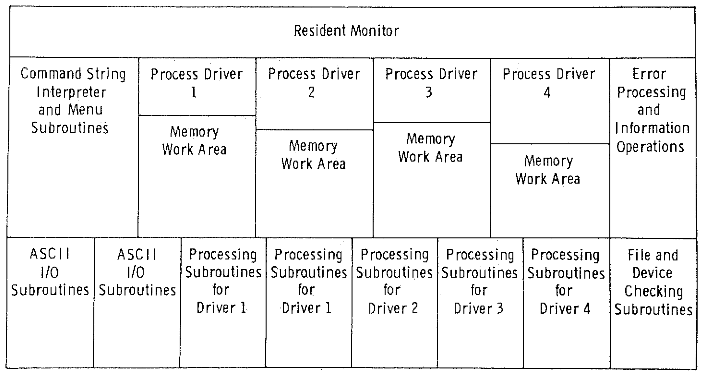
Resident Monitor
Command String Interpreter and Menu Subroutines
Process Driver 1
Memory Work Area
Process Driver 2
Memory Work Area
Process Driver 3
Memory Work Area
Process Driver 4
Memory Work Area
Error Processing and Information Operations
ASCII I/O Subroutines
ASCII I/O Subroutines
Processing Subroutines for Driver 1
Processing Subroutines for Driver 1
Processing Subroutines for Driver 2
Processing Subroutines for Driver 3
Processing Subroutines for Driver 4
File and Device Checking Subroutines

FIG. 4. Illustrates a memory overlay structure.

3.3.3. The Image Data

Image data records make up the third section of the SIF file. The digital multi-image is divided into mutually exclusive rectangular regions called blocks and each logical record contains one block. This allows the flexibility of having the logical records correspond to image rows, image columns, or rectangular blocks of some given dimension. The data that is stored in each subimage block is encoded in one of the five data modes specified in the identification array: absolute binary, two's complement binary, floating point, double integer, and double precision. The byte length for the absolute binary and two's complement binary is also specified in the identification array. The bytes are packed into 36-bit double words.

SIF file organization. (1) Data points within a SIF subimage block are ordered sequentially in a row by row manner as shown in Fig. 1. This array (compacted if possible) is written as one logical record.

(2) These blocks cover the entire image in a columnwise fashion as shown in Fig. 2.

(3) Figure 3 illustrates how the entire SIF multi-image file appears on the storage device. Each record is K words long. K is determined by the block size, the data mode, and the number of bits/point if the data mode is integer. The identification record is first, followed by the descriptor records, and then the image data. If the entire image is viewed as a four-dimensional array (L, M, N, P) where:

L = number of image bands,

M = number of blocks,

N = number of rows per block,

P = number of columns per block,

then the subscripts would vary from fastest to slowest in the following order: columns, rows, bands, blocks.

To conserve memory in the resident portion of KANDIDATS, parameters are passed to command drivers via labeled common areas. The use of labeled common

Portion of resident KANDIDATS monitor:

```
                          •
                          •
                          •
      COMMON / COMAND / IFC, OTDEVN,   OTDEV,   OTTYPE,
     *     OTFILE(2),  INDEVN, INDEV,   INTYP , INFILE(2),
     *     INFIL2(2),     INFIL3(2),   LFLAG(26)
                          •
                          •
100   CALL COMIN2(IN, PROMPT, FNAME, DNAME, IFC, OTDEVN,
     *                 OTFILE, INDEVN, INFILE, LFLAG, $9010)
                          •
                          •
      GO TO ( . . ., 2300, . . . ), IFC
                          •
                          •
2300  CALL DQUANT ( $9010 )
      GO TO 100
                          •
                          •
9010  CALL ERRPRC
      GO TO 100
```

Portion of process driver:

```
      SUBROUTINE DQUANT ( * )
                          •
                          •
                          •
     COMMON / COMAND /IFC, OTDEVN, OTDEV, OTTYPE,
    *                OTFILE(2), INDEVN, INDEV, INTYP, INFILE(2),
    *                INFIL2(2), INFIL3(2), LFLAG(26)
     COMMON / WORKSP / ISIZE, IWORK(4000)
                          •
                          •
     CALL KDPUSH ( "DQUAN", "T" )
     CALL DEVCHK ( DEVMSK, 1, DEVCMB, $9010 )
     ISIZE = 4000
     CALL NUMLVL ( NLEVL )
     CALL QUANT ( INDEV, INFILE, OTDEV, OTFILE, NLEVL,
                  IEV, $9010 )
                          •
                          •
     CALL KDPOP
     RETURN
                          •
                          •
     9010      CALL CLOSE (INDEV )
               CALL CLOSE ( OTDEV )
               RETURN 1
```

FIG. 5. Skeleton of KANDIDATS resident monitor and process driver. Information is passed from the command string interpreter through the labeled common COMAND to the process drives. The contents of this common are: internal command index, output device name, output logical unit, output device type, output file name, input device name, input logical unit, input device type, input file names, and option flags.

```
# :TERTS TSITE IMG ←MT2 (RC)
ENTER FIRST, LAST ROWS ( 1 -2340 ) -- 1, 1000
ENTER FIRST, LAST COLUMNS ( 1 - 824 ) -- 1, 800
# :QUANT TSITE QNT ←TSITE IMG
ENTER NUMBER OF QUANTIZED LEVELS ( 0 ) -- 32
# :RCNV TSITE RCN ← TSITE QNT
ENTER WINDOW SIZE -- 3, 3
# :XPCMP TSITE CMP ←TSITE QNT
ENTER VERTICAL MODE ⟨C⟩ COMPRESS, ⟨E⟩ XPAND, ⟨N⟩ ONE -- C
ENTER RATIO FOR VERTICAL OPERATION -- 1000,100
ENTER HORIZONTAL MODE ⟨C⟩ COMPRESS, ⟨E⟩ XPAND, ⟨N⟩ ONE -- C
ENTER RATIO FOR HORIZONTAL OPERATION -- 140, 824
# :TSIF ID1 ←TSITE CMP (F)
ENTER BAND TO DISPLAY ( 1 - 4 ) -- 2
# :GDTI TSITE RCN ←TSITE CMP
(Note: The RC flags cause the row, column questions to be asked)
```

FIG. 6. An interactive processing example where rows 1,1000 and columns 1800 are transferred from ERTS tape to a standard image format on the disk. The image TSITE RCN is then equal interval quantized to 32 levels, rectangular convolved with a 3 × 3 window, compressed 10 to 1 vertically and 824 to 140 horizontally, and has band 2 displayed on the video display device.

eliminates long lists of arguments for each call to a command driver. In the case of a driver calling a processing program the argument list consists of input and output file codes, input and output file names, processing parameters, error status, and alternate error return. An outline of the communication between the resident monitor, process driver, and process program is given in Fig. 5. Because the process programs do not rely on the environment of the KANDIDATS monitor they can be used in other software systems without modification.

The language in which an image processing system is implemented must be portable and available on small systems. FORTRAN IV was chosen for these reasons. Over 97% of the code in KANDIDATS is in FORTRAN. The remaining 3% is in assembly language because of execution time considerations.

4.2. Processing Example

Figure 6 shows an example of interactive image processing with KANDIDATS. The "# :" symbol indicates that the command string interpreter is ready to receive command input. After the user has entered the command and carriage return the process driver is brought into memory. The prompting shown for additional parameters is from the process drivers via the question/answer subroutines. Processing takes place between the last answer for a question and the next "# :" symbol for each command.

The last command in Fig. 6 allows the user to enter ground truth information via a display device. Ground truth is placed in the descriptor records of image TSITE RCN.

4. CONCLUSION

We have discussed the implementation of KANDIDATS, the interactive digital image processing system at the University of Kansas. KANDIDATS is divided into three parts: monitor, processing programs, and image access. Each part is constructed with the user in mind. The system handles bookkeeping and prompting for the user. Programs are written in a modularized manner for each of modification and enhancements. The system is structured such that only a small number of modules need to be in memory at any one time and this allows the processing of large image data sets in a minicomputer environment.

APPENDIX 1: OUTLINE OF KANDIDATS OPERATIONS

A. Utility

1. Information operations
   - (a) Examine image (image editor subsystems)
   - (b) Explain
   - (c) Short vocabulary
   - (d) Vocabulary
   - (e) Add descriptor record(s)
   - (f) Delete descriptor record(s)
   - (g) Transfer identification block
   - (h) List descriptor records
2. Transfer operations
   - (a) Transfer image to/form display
   - (b) Transfer symbolic image to line printer
   - (c) Transfer LANDSAT tape to disk
   - (d) Transfer Skylab (Universal) tape to disk
   - (e) Transfer LARS tape to disk
3. Spatial domain operations
   - (a) Combine images
   - (b) Select subimage
   - (c) Expand/compress image
   - (d) Flip image
   - (e) Rotate image 180°
   - (f) Transpose image
   - (g) Mosaic images
   - (h) Rubber sheet image
   - (i) Reblock image
4. Greytone operations
   - (a) Equal interval quantize
   - (b) Equal probability quantize
   - (c) Rectangular convolution
   - (d) Algebraic operations
   - (e) Linearly combine
   - (f) Contour map
   - (g) Mode change
   - (h) Principal components projection
   - (i) Mask
5. Operations to compute statistics

   (a) Histogram by band
   (b) Scattergram by band pairs and category
   (c) Histogram and range plots by band and category
   (d) Count second-order marginals
   (e) RMS error between bands of two images
   (f) Generate statistical descriptor records
   (g) Compute mean and covariance by category
   (h) Compute mean and covariance of entire image

6. Control operations
   (a) Brief output messages
   (b) Long output messages
   (c) Call external routine
   (d) Stop execution
   (e) Run command file
   (f) Build command
7. Ground truth operations
   (a) Input ground truth data
   (b) Create symbolic image from descriptor records
   (c) Create symbolic image from binary images
   (d) Count categories in image
   (e) Count categories per block
8. Magnetic tape operations
   (a) Card transfer to/from tapes for main computer facility
   (b) Tape dump
   (c) Write end of file
   (d) Forward space record
   (e) Forward space file
   (f) Backspace record
   (g) Backspace file
   (h) Rewind
9. Image generation
   (a) Make bar image
   (b) Make checkerboard image
   (c) Simulate image from statistics

B. Clustering

1. Edge and texture extraction
   (a) Roberts gradient
   (b) Laplacian
   (c) Replace by local minimum or maximum
   (d) Generate textural image transform
2. Spatial clustering operations
   (a) Image thresholding operations
   (b) Spatially connect homogeneous regions

(c) Clean homogenous regions
(d) Split homogenous regions

3. Measurement space clustering operation
   (a) Find mean of regions
   (b) Cluster homogenous regions using measurement space statistics
4. Symbolic image operation
   (a) Shrink symbolic regions maintaining connectivity
   (b) Fill symbolic regions
   (c) Shrink symbolic regions
   (d) Symbolic image composition

C. Pattern Discrimination

1. Feature selection
   (a) Nonparametric feature selection for table look-up rule
   (b) Parametric feature selection using Bhattacharyya coefficient
2. Decision rule creation
   (a) Count second-order marginals
   (b) Smooth marginal distributions
   (c) Normalize marginal distributions
   (d) Determine nonparametric decision rule
   (e) Compute contingency table
3. Classify image
   (a) Classify image by existing table look-up rule
   (b) Compute table look-up rule and classify image

D. Bandwidth Compression

1. Transforms
   (a) Hadamard
   (b) Fast fourier
   (c) Discrete linear basis
   (d) Discrete cosine
   (e) Slant
   (f) Karhunen–Loève
   (g) Fast Karhunen–Loève
2. Compressions
   (a) Band pass
   (b) Energy
   (c) Optimal bit allocation
3. Pre/post processing
   (a) Add/subtract mean
   (b) Log/antilog

REFERENCES

1. H. Frieden, Image Processing System, VICAR, Guide to System Use, Jet Propulsion Lab, Pasadena, California, July, 1971.
2. S. T. Alexander, LADIES Software Description, University of California, Los Alamos Scientific Lab.

3. R. M. Haralick, G. J. Minden, D. R. Johnson, A. Singh, W. F. Bryant, and C. A. Paul, KANDIDATS Image Processing System, Third Symposium on Machine Processing of Remotely Sensed Data, Purdue University, June, 1976.
4. J. Adams and G. Peterson, Stanford Technology Corporation, I/O Subroutine Documentation.
5. R. C. Rathja and J. H. Herzog, PIXSYS, A Users Manual, Pictorial Information Extraction and Enhancement Laboratory, Oregon State University, Pixel Report 080174.
6. Advanced Multispectral Image Descriptor System Report (AMIDS), Rome Air Development Center, Griffiss Air Force Base, New York, January, 1975.
7. F. J. Kriegler, MIDAS, Prototype Multivariate Interactive Digital Analysis System-Phase I, Environmental Research Institute of Michigan, August, 1974.
8. G. Swanland, Experimental Image Exploitation System, Control Data Corporation.
9. Interactive Picture Analysis and Display Laboratory Brochure, Aeronutronic Ford, Palo Alto, California, February, 1976.
10. Interactive Multispectral Image Analysis System Brochure (Image 100), General Electric Company, September, 1973.
11. J. W. Snively and Butt, E. B., The Pax II Picture Processing System, University of Maryland, TR 68-67, May, 1968.
12. J. F. O'Callaghan, DISIMP: Device Independent Software for Image Processing, Dec., 1975.
13. R. M. Hoffer, "ADP of Multispectral Scanner Data for Land Use Mapping," LARS Information Note 080372, The Laboratory for Applications of Remote Sensing, Purdue University, Lafayette, Ind., 1973.
14. L. A. Gambino and M. A. Crombie, Digital mapping and digital image processing, *Photogrammetric Engineering* **40**, 1974, 1295–1302.
15. J. D. Turinetti and R. J. Hoffman, Pattern analysis equipment and techniques, *Photogrammetric Engineering* **40**, 1974, 1323–1330.
16. D. L. Milgram and K. C. Hayes, Image Processing Software Design and Issues, Computer Science Center, University of Maryland.
17. *PECOS II Users Manual*, Technical Memorandum ESL-TM503, Electromagnetic Systems Laboratories Incorporated, Sunnyvale, California, Sept. 1974.
18. K. C. Hayes, *XAP Users Manual*, Computer Science Center, University of Maryland, TR 348, January, 1975, College Park, Maryland.
19. R. M. Haralick, Image Access Protocol for Image Processing Software, *IEEE Transition on Software Engineering*, March 1977, p. 190–192.