drme . ip . iria . nato

nato advanced study institute

digital image processing and analysis

analyse et traitement d'images digitales

bonas (gers)
1976

june 14th - 25th

NATO ADVANCED STUDY INSTITUTE 1976
DIGITAL IMAGE PROCESSING AND ANALYSIS
ANALYSE ET TRAITEMENT D'IMAGES DIGITALES

IMAGE PROCESSING SOFTWARE AND DATA STRUCTURES

R. M. Haralick
University of Kansas Center for Research
Lawrence (U.S.A.)

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 used by human interpreters. Computer image processing has evolved over the last decade from the use of specialized programs to more flexible program packages using both sophisticated computer and minicomputer 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 of this production is the Earth Resources Technology Satellite which produces 782 images each day.

Image processing encompasses all the various operations which 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. Now image processing is becoming interactive. An operator, user, or analyst sits 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 which can only be accessed a segment at a time with processing tasks that provide heavy and varied loads to the computer system. These problems require sophisticated and complex software for even the simplest image processing tasks. In this paper we discuss the structure and design philosophy for an interactive image processing system. The ideas in this paper are motivated by the image processing systems that are beginning to be used in both the large computer and minicomputer environment.

II. SOFTWARE REQUIREMENTS

In order to be able to meet the general needs of interactive image processing, the software must meet a number of requirements concerning the hardware environment on which it executes, the systems programmer environment which cares for and maintains it, and the user environment which it must ultimately satisfy. All three environments must be expected to change. Technology will introduce new hardware systems. Neophyte users will become experienced users, the experienced users will leave, and new neophyte users will come. Likewise for the system programmers. And while all this is going on users will request new image operations which must be quickly and easily added to the software by the system programmer. If the total software package is not designed to be easily used, easily learned, easily programmed, easily modified, and easily transportable in this changing environment, then a 100,000 source statement image processing system can become a patched up unworkable worthless burden in a matter of a few years.

This paper describes a general approach to the software design problem for image processing systems. The proposed design of the interactive I/O facilititates easy use. The structural modularity of the software facilitates easy modification. The generalized random access I/O, memory allocation, and overlay structure facilitate using the software on old or new, large or small computers with few required changes.

Section III describes how the user should use the system. Section IV describes the software structure.

III. THE USER'S PERSPECTIVE

The user's chief concerns are how easy it is for him to interact with the system to get his job done and how much bookkeeping and remembering can the system do for him in order to lessen his burden. Section III.1 discusses some approaches which facilitate easy interactive I/O and section III.2 discusses the concept of processing history records to automatically keep track of all image parameters and the sequence of processes the image has been through.

III.1 Interactive I/O

The interactive I/O must be functional for the neophyte as well as experienced user. This suggests that a variety of different ways of interacting with the system be allowed. The neophyte user knows only enough about image processing to know the general processing area which he thinks he wants to use. A menu format which at first displays on the console screen four to twelve different broad image processing categories from which the user can select will usually be enough to get the neophyte going. After he selects one of the broad areas, a second level new menu appears on the screen which shows either a list of related image processing categories or a list of actual image processing operations. A third level, if necessary, would show a list of four to twelve image processing operations. The menu concept is illustrated in figure 1.

The more experienced user will find the menu interaction system tiresome. He desires a quick way to initiate his image processing task. Ideally, he would just like to type in one command word or process acronym and be done. This emphasis on brevity has consequences. The experienced user may forget the exact spelling or phrasing of a processing name. In this situation he needs some help like an alphabetically organized list of the command name acronyms that he can request to be put on his console screen. One command word

for short vocabulary should be able to give him this help.

Then there is the user who is not a neophyte, but has not been around long enough to be fully experienced. For such a user, an alphabetically organized list of process acronyms will not be enough; he will guess that there are two or more possible acronyms that might do what he wants. He needs to be able to ask the system what any particular process acronym or command name means. For him a vocabulary command which can explain what any command acronym or abbreviation means needs to be proveded.

Finally, there is the case where an experienced or neophyte user is not able to get satisfactory operation for some process because he does not understand what the process does or because he does not understand how to set up the parameters for the process correctly. Here, a complete explanation defining what the process does, legal ways of calling the process into action, what the parameters of the process do, and legal values for the parameters is needed. An explain command can fulfill this need.

Every image operation works on an input image and produces and output image, statistics, or features of the input image. Once the image processing operation is specified by a menu or command name procedure, the input and output files or devices must be specified. One natural way is the question/answer approach. The question "What is the input image name?" appears on the console screen and the user types in the name,whereupon the software makes sure the file name syntax is correct and then asks about the output image name.

The question and answer approach is not the only one. There are a variety of different convenient ways of combining command names with input and output file names into a command string. For example, take the process of quantizing which has the command name QUANT.

Possible command string syntaxes include

QUANT OUTPUT IMG ← INPUT IMG

INPUT IMG.> QUANT > OUTPUT IMG

OUTPUT IMG = QUANT (INPUT IMG)

The first syntax is used in the KANDIDATS system at the University of Kansas and can be interpreted as quantize to an image named OUTPUT IMG the image named INPUT IMG. The second syntax, analogous to the engineering diagram showing input, black box, and output, is used in the IDIMS system at the Electromagnetic Sensing Laboratory. It can be interpreted as take the image INPUT IMG and go through the process QUANT to produce the resulting image called OUTPUT IMG. The third syntax, a mathematical functional form notation, is used in the EIES system at Control Data Corporation and can be interpreted as: the image OUTPUT IMG is a quantized version of the image INPUT IMG.

When the image processing task does not produce another image but produces statistics about the input image, there is the question of whether these statistics should be put on the user console screen for examination, to a line printer for hard copy, or to be made into a source file to be saved on the mass store device. Natural variations of the basic command string can easily permit these distinctions to be made. Take for example the process of getting a histogram of the grey tone on an image. Let the command name be HIST. The following three command strings can direct the histogram to the terminal, to the line printer or to the disk pack as a file called TONE HST:

HIST TT ← INPUT IMG

HIST LP ← INPUT IMG

HIST TONE HST ← INPUT IMG

Once the basic combination of input image, output image, and command name have been specified, the image process itself may require additional parameters. The equal interval quantizing command, for example, will need to know how many quantized levels the user desires for the output image. Other processes such as the histogram command will need to know what portion of the image should be histogramed. In this case the user will have to specify a subimage size and location. Because it often is the case that similar kinds of information will be needed for a variety of different commands, the prompting questions that the software provides should be identical in format and style. Certainly a request for the same information should always appear to a user in the same format.

Because there may be many steps to an image processing task, it will be important for the image processing system to have two modes of operation: an interactive mode and a batch-like mode. The batch-like mode should provide a way for the experienced user to write a sequence of ten or twenty command strings stored as a system standard ASCII file which he can then ask the image processing system to execute or run. For image processing tasks that need to be repeated on a few images this capability will save the user much wasted terminal time. Figure 2 illustrates such a command string program.

III.2 Processing History Records

Interactive image processing usually proceeds on a step by step basis with many intermediate images created and then deleted before a final resulting image is produced. (The intermediate images are deleted to save mass store file space.) If the user has not kept careful notes about how it was that he produced the final image, all the processing may be lost for he could not report what processing steps he had done to his initial image to achieve his final image. Since the best bookkeeper in the world is the computer, the user should expect that the image processing system will keep processing history records for each image.

Processing history records fall in three general categories. The first category consists of some free format information that the user wants associated with the image. Such information could be about the image's origin, associated photography, method of digitization, tape identifiers for the tapes holding copies of the original digital image, associated files of ground truth information, specific information about targets in the image, and perhaps some information about the purpose of the image processing. What the user wants to put in these records, how much he wants to put, and the format in which he puts them should all be left free. These records should be considered like ASCII source information with no structure. They should let the user tell any story he likes.

The second category of descriptor records is more structured and accurately tells what processes the image has been through. The input image(s), the process name, the output image, the date of the processing, and all the parameters required by the process all should appear in these records. Because the kind and number of parameters varies as a function of the image process,

these descriptor records themselves will have to be a data structure to allow easy access to them by program or user.

The third category of descriptor records has statistical information about the image which either processing programs create or ground truth information which the user must input. This information cannot be free format because it is information that other processing oeprations must access in order to do their job. Hence, the order in which the pieces of this kind of information appear must be standardized. Examples of statistical information include histograms, minimum and maximums, and mean and covariance matrices. Examples of ground truth information include tables which relate a land use type or target name to be associated with some numeric symbol on a band of the image and lists giving land use types, numeric symbol, and associated polygonal areas on the image of the named land use type.

The users should be able to examine these descriptor records by some commands very much in the same manner that he tells the image processing system to do a job. Descriptor record operations should include the capability to list all descriptor records, list only the names of the processes the image has undergone in order of their occurrence, list only free format records, and list only the statistical records. In addition, the user will need to have a way to add unstructured information to the descriptor file. A descriptor record deletion operation is needed so that the user can correct any misinformation he added to the descriptor record history. However, the user should not be allowed to delete processing history records created by processing programs.

IV. The Software Structure

In order to satisfy the requirements of easy addition of new processing operations, high portability to new computer hardware environments, and easily understandable software by the system's programmers, the image processing system software must be structured and highly modular.

As shown in Figure 3, one way of structuring the software is to have a monitor to take care of the flow of control. A typical operation sequence would be for the monitor to call the command string interpreter or menu prompting routine to accept a user processing assignment. The menu mode of operation could be initiated by a command through the command string interpreter as shown. Once a processing option has been selected, input and output file names given, and source or destination devices assumed or given, control can be returned to the monitor.

The monitor's next job is to call the process driver subroutine and pass it all the information it has received from the menu or command string interpreter. The process drivers first call device and file checking routines to make sure the input image files exist and to make sure the specified devices are legal for the process. Depending on the task, various parameters may need to be obtained. The process driver can call ASC11 I/O prompting subroutines, each one asking a specific question, to obtain the required information. When all parameters are specified consistently the process driver can allocate the available memory in the best way it knows how and then call the processing subroutine specifying all the information it has in the calling argument list.

The processing subroutine's first job is to initialize proper input and output image files and copy the descriptor records associated with the input image(s) to the descriptor records associated with the output image. Then image processing can begin by accessing the subimage blocks or segments in an appropriate order, passing these subimages to a number crunching routine which does the actual processing. On return from the number crunching routine, the image processing routine writes the processed subimage to the output image file and accesses the next group of subimage blocks on the input image. When all the subimage blocks have been processed, the files are closed and control is passed back to the driver which returns control to the monitor.

IV.1 Errors and User Generated Interrupt Handling

During each stage of processing, error should be prevented from occurring. For example, in a quantizing operation, the command string interpreter must make sure that the input image file name is syntactically a correct one. The process driver makes sure that the input image file specified really exists. The ASCII I/O routine must make sure that the specified number of quantization levels is a positive integer greater than 1. In the processing subroutine, if access to an image file cannot be done because of bad records or because something happens related to image file handling or because something happens relating to a numerical calculation, then an error status word should be set and control passed back to the monitor. The monitor's action should be to let the user know what the error was and prompt him for a new processing assignment.

Each different computer hardware and operating system is likely to handle default errors and trace of calls information differently. To maintain portability and control this should be done internally by the image processing system. One possible way is to have an array which operates as a stack. Each time an entry is made into a subroutine, the subroutine calls a routine to push its name on the stack. Just before normal exist from a subroutine, the subroutine can call a routine to pop its name from the stack. If an error occurs during processing, a status variable can be set and an alternate return taken which does not pop the subroutine name from the stack. Thus, if it is conceivable for an unexpected error to occur in a subroutine , then the subroutine must have two additional arguments: a status word and an alternate error return.

Pushing and popping subroutine names for internal trace of calls information has an added advantage to the system programmer and debugger, for it makes available an interactive trace capability. Upon setting a trace flag by inputting a command TRACE, each time a subroutine pushes or pops its name from the stack, the push and pop routine can output to the console the same names. In this manner a real time trace becomes available and the exact moment and sequence in which an error occurs can be located.

There is one other kind of special condition the subroutines must be able to handle: user generated interrupts. Sometimes after a user initiates a processing task assignment, he realizes that he himself entered a legal but incorrect parameter. Rather than wait the required time for the task to finish, he would like to interrupt the process, return to parameter input interaction again or perhaps even to the monitor level in order to specify a different task. To facilitate this, at each appropriate place in the processing sequence where control can be returned to and where processing can begin again, a call can be made to a subroutine which sets a return address (or statement number) to which control is returned upon a user generated interrupt.

IV.2 Memory Management

In order for the image processing software to execute in both the large and minicomputer environment, careful attention must be paid to the memory management constraints found on minicomputers. Care must be given to the overlay structure so that the different process drivers can overlay one another and the different processing routines can overlay one another. This implies that BCD I/O must be separated from any file to file image processing operations so that the memory consumming BCD I/O can be in separate overlay links.

Since the total amount of memory available is fixed, and the amount of code required by the different processing routines can vary, memory available for working arrays will vary. Since the process driver knows which subroutine it calls, it in effect knows how much code the processing routines will take. Therefore, it can allocate all remaining memory for working arrays.

Another constraint that the minicomputer operating environment has is the restricted nature of the overlaying structure. Instead of being able to overlay routines in a general tree structured way with a level (n + 1) overlay being loaded in memory immediately below its associated level n overlay, some minicomputer operating systems require that level (n + 1) overlays be loaded in memory immediately below the lowest level n overlay. To achieve efficient memory management the various overlay links should be about the same size for each level. Figure 4 shows a general way of overlaying which is consistent with minicomputer restrictions and which minimizes wasted memory space.

IV.3 Image Access Protocol

In order to isolate the computer environment in which the software executes and the image processing software subroutines, no image file handling should be done directly by any of the image processing routines. Rather, a call to a generalized image access routine should be made. In this manner, the processing routines never need to take into account physical record buffering or be concerned about whether the image is coming from a random disk device or a super large external random access memory. In this manner, all operations of actual image access which are dependent on the available hardware of computer peripherals can be concentrated in a few subroutines.

Because the image access subroutines are so important in designing a set of standardized image processing subroutines, we define in section IV.3.1 the multi-image, which is what is accessed, we describe in section IV.3.2 what the access routines must do, and we suggest in section IV.3.3 a file structure for the multi-image.

IV.3.1 The Multi-Image

We will consider the spatial domain for a digital multi-image to be a rectangular area which is divided up into small mutually exclusive rectangular regions called resolution cells. On a one band image, each resolution cell has a single value or number specifying the average grey tone intensity for that resolution cell. On an N-band multi-image, each resolution cell has an N-tuple of numbers. The n^{th} component of the N-tuple specifies the average grey tone intensity for that resolution cell on band n. In essence, then, each band of a digital multi-image is like a matrix of numbers. The multi-image itself is like a set of N matrices stacked one on top of the other.

Because the number of rows and columns of a multi-image can easily be a few thousand, the entire image cannot be stored in menory at once and to save file space, the grey tone intensity values are often packed as bytes, many to a computer word. This problem with space has two consequences. The first is that the entire image cannot be accessed at once. Rather it must be accessed segment by segment. We will consider the segments to be logical records and we will assume that a logical record corresponds to a subimage block of S_R rows by S_C columns for one band. The most frequently used segment or block is one complete image row.

The second consequence is due to the possibility of packing bytes. It leads to the variety of data modes an image can have. For example, when the grey tone intensity values lie between 0 and 63, it would be most space saving to store the intensity values as six bit absolute binary bytes. Of course other interpretations of the byte are possible: a two's complement form, for instance.

To be consistent with the way most computers do arithmetic, an absolute binary byte can be as large as one bit less than the number of bits in an integer word. No such constraint is necessary for the two's complement form byte. The two forms of integer bytes plus the other data modes supported by the language in which the software is written lead to seven multi-image data modes: (1) absolute binary bytes, (2) two's complement binary bytes, (3) double integer, (4) real, (5) double precision, (6) complex, and (7) double precision complex. Because there are seven data modes possible we will suggest the constraint that all bands of a multi-image file must be in the same data mode.

Sometimes in thematic maps and classified image data files, image bands are created in which the value of the n^{th} band for any resolution cell does not have the grey tone intensity interpretation. Rather, the value stands as a symbol or an index to some category name. For example, in a map which shows areas of wheat, corn, and bare ground, the value 1 could be the symbol for wheat, the value 2 could be the symbol for corn, and the value 3 could be the symbol for bare ground. To distinguish these kinds of bands from the bands having grey tone interpretations, we name those bands having grey tone interpretations as numeric bands and we name those bands whose values are really symbols for category names as symbolic bands. To make matters easy, we will arrange the bands so that the symbolic bands are always the last ones on an image. The operational significance of this numeric/ symbolic distinction for bands is that those image operations which involve arithmetic manipulations must be only done on numeric bands and they must leave the symbolic bands alone.

IV.3.2 The Image Access Routines

The image access routines get and put image segments from the array the user works with to the buffer area in which the I/O system has the data packed. This buffer area is transparent to the user. It can be as small as enough memory for one logical record or as large as enough memory for one hundred logical records. A call to the image access routines does not necessarily imply a physical disk I/O transfer. From the point of view of the user, he does not know or care about when the disk transfers actually take place. Hence, the level at which the image access routines operate is at a level which is more general than disk I/O transfers.

We assume that the image access routines do random accesses. There are four image input/output procedures of major consequence. They are:

(1) open or initialize an old file;

(2) open or initialize a new file;

(3) read a logical record;

(4) write a logical record.

Any call to one of the I/O procedures will involve specifying or retrieving basic booking information about the image. For example, to open or initialize an old file, the procedure must retrieve from some kind of header record the basic image parameters which includes:

number of rows in the image
number of columns in the image
number of bands in the image
image data mode
number of bits per value
number of rows in a subimage block (logical record)
number of columns in a subimage block (logical record)
minimum value over all bands in the image
maximum value over all bands in the image
number of symbolic bands

These parameters, as suggested above, can be placed in an image identification array. This leads to a CALL sequence like

```
CALL RDKINL  (LU, FILNM, IDENT, NO, IEV, IALTRT)

    LU       is logical unit number
    FILNM    is file number
    IDENT    is identification array
    NO       is 1 for old file
                2 is for new file
    IEV      is an event variable idicating status upon completion
                of initialize action
    IALTRT   is the alternate return statement number taken
                on a bad status
```

To read or write logical records, some of the information in the header record might be useful to the procedure. Hence, we assume that one of the arguments which must be in the read and write entry points is the identification of the header record.

Since it is frequently the case that only a part of the image is of concern throughout an entire processing operation, the read/write procedure argument list must contain information specifying the total subimage size and position which is to undergo processing. This entails specifying the first row, first column, last row, and last column as well as the spatial sampling rate which indicates things like taking every other point along the rows and tripling every point down a column.

Since there are no guarantees that a user must specify a subimage size and position which locates the subimage at block boundaries, there can be some accessing ambiguties. On a read operation we will define all resolution cells to have the value zero which are outside the specified subimage but within a block having some resolution cells in the specified subimage. On a write operation involving any problem blocks we will first read the block, replace all data values inside the subimage region using the information to be written, leave alone any of the read values lying outside the specified subimage, and then write the changed block back on the disk.

Because the subimage designated for processing can be different from the image, a decision has to be made regarding whether any specified block number is relative to the subimage area or whether it is the absolute block number in the original image. It is probably more convenient to have the block number be relative to the subimage area to be processed since that would make identical the block indexing for both input and output images for a particular image manipulation routine.

The read/write procedures must, of course, pack and unpack bytes if the data mode of the image is single integer. In addition it would be useful to have the read/write procedures do a simple grey tone intensity rescaling. The simplest rescaling or renormalizing which can be considered for numeric bands is multiplying or dividing by 2^k for some integer k.

Thus we see that the read/write procedure must have argument lists which specify:

(1) image identification array

(2) subimage size and position

(3) spatial sampling rate

(4) normalizing or rescaling factor for numeric bands

(5) the image bands which are to be read or written

(6) how many subimage blocks are to be read or written at a time

(7) the index for the first subimage block to be read or written

This suggests a CALL sequence like

```
CALL   RREAD
       REWRITE (LU, IMGARY, IBAND, IBLKNO, NSBIMG, IPAR, IDENT, IEV, IALTRT)
```

where

LU is logical unit number

IMGARY is array to store subimage in

IBAND is an array to tell which bands to access

IBAND(I) = 1 if I^{th} band is to be accessed

= 0 if I^{th} band is not to be involved

IBLKNO is subimage block number to begin accesses with

NSBIMG is number of subimage blocks to access

IPAR is the processing parameter array having information on subimage window size and position that all processing will take place in, spatial sampling rate, and grey tone rescaling factor for numeric bands.

IDENT is the image identification array

IEV is the status event variable

IALTRT is the alternate return taken on bad status

IV.3.3 The Multi-Image File

The multi-image file should be able to be accessed randomly as well as sequentially. This implies that the logical records must all be of the same length because some operating systems do not support variable length random files. To simplify the job of buffering, the subimage block requested in a read or write must be one of the logical records in the multi-image file. Hence, the only areas of the image which can be accessed by a read or write is a subimage block of the same size and positioned exactly in the same place as one of the logical records in the file.

The question about the sequential order in which the logical records are placed on the file is governed by whether to have band number or subimage block number go the fastest. If subimage block number goes the fastest, then the file is organized as all subimage blocks from band 1 followed by all subimage blocks from band 2, etc. If band number goes the fastest, the file is organized as all bands for subimage block number 1 followed by all bands for subimage block number 2, etc.

From the point of view of random accessing, the order makes no difference. This is not so, however, for sequential accessing. Given the constraint that all bands for subimage block i should be able to be read sequentially without a rewind after reading all bands for subimage block i-e, we see that the file must be organized having the image bands run the fastest.

Therefore, the multi-image file takes the following form. All records are equal length. The first record must be an identification parameter record. The remaining records must be organized having the image bands run the fastest and the subimage blocks run the slowest.

V. Conclusion

We have discussed the interactive image processing system from the point of view of the user and from the point of view of the hardware. A monitor governing flow of control, command string interpreter and menu prompting routines, separate ASCII I/O routines for each special prompting question, process drivers which call file and device checking subroutines, ASCII I/O routines, allocate memory, and call processing programs all lead to the modularity of the structure. Internal keeping track of trace of calls and error checking keeps the image processing software in control rather than the operating system. The image access protocol isolates the image access

problem from the buffering problem and the independence of the different processing functions from one another lead to an ease of updating and adding new routines, a high portability from one computer to another, and a highly overlayed structure so that execution can take place in a minicomputer environment.

IMAGE PROCESSING CATEGORIES

A. Utility Operations
B. Clustering Operations
C. Pattern Discrimination Operations
D. Transform and Compression Operations
(Figure 1.a)

UTILITY OPERATIONS

1. Information Operations
2. Transfer Operations
3. Spatial Domain Operations
4. Greytone Operations
5. Operations to Compute Statistics
6. Control Operations
7. Ground Truth Operations
8. Magnetic Tape Operations
9. Image Generation Operations
(Figure 1.b)

SPATIAL DOMAIN OPERATIONS

a. Register Image
b. Extract Subimager
c. Expand/Compress Image
d. Flip Image
e. Rotate Image
f. Transpose Image
g. Mosaic Image
h. Rubber Sheet Images
i. Reblock Images
(Figure 1.c)

Figure 1 shows three levels of a menu prompting system. At the first level (figure 1.a) the user selects Utility Operations. At the second level (figure 1.b) the user selects Spatial Domain Operations. At the third level (figure 1.c) the user selects an actual command and would then be prompted by the input and output image name questions.

```
#:TERTS TSITE IMG ← MT2 (RC)                    (NOTE:The RC flags cause
ENTER FIRST, LAST ROWS ( 1 -2340 ) -- 1,1000     the row, column questions
ENTER FIRST, LAST COLUMNS ( 1 - 824 ) -- 1,800   to be asked)
#: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> OMPRESS, <E>  XPAND,  <N> ONE --C
ENTER RATIO FOR VERTICAL OPERATION -- 1000,100
ENTER HORIZONTAL MODE <C> OMPRESS, <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
```

Figure 2.a shows an interactive processing example where rows 1,1000 and column 1,800 are transferred from an ERTS tape to a standard image format on the disk. The image TSITE RCN is then equal interval quantized to 32 levels, rectangular convoluted with a 3x3 window, compressed 10 to 1 vertically and 824 to 140 horizontally, and has band 2 displayed on the video display device. Computer prompting is underlined.

After the user enters ground truth information via a display device, the ground truth is placed on the descriptor records of the image TSITE RCN and is used in the run mode pattern discriminating processing involked by the command:

```
#:RUN PATDIS BTC
```

```
BAYES TSITE BO1 ←TSITE RCN (D)          (number of categories)
5                                       (ground truth set no.)
1                                       (no. 2nd order marginals)
2                                       (first band pair)
1,2                                     (second band pair)
3,4                                     (probability thresholds)
.03
.21
BAYES TSITE BO2 ← TSITE RCN (D)
5
1
2
1,3
2,4
.03
.21
BAYES TSITE BO3 ← TSITE RCN (D)
5
1
2
1,4
2,3
.03
.21
CNTNG LP ← TSITE RCN,TSITE BO1
Y                                       (descriptor records?)
2                                       (ground truth set no.)
CNTNG LP ← TSITE RCN,TSITE BO2
Y
2
CNTNG LP ← TSITE RCN,TSITE BO3
Y
1
EXIT
```

Figure 2.b shows the command string statements on the run file which applies a table look-up Bayes rule to each of three sets of band pairs from the image TSITE RCN. A contigency table comparing the resulting classified image with ground truth descriptor records is then computed and outputed to a line printer.

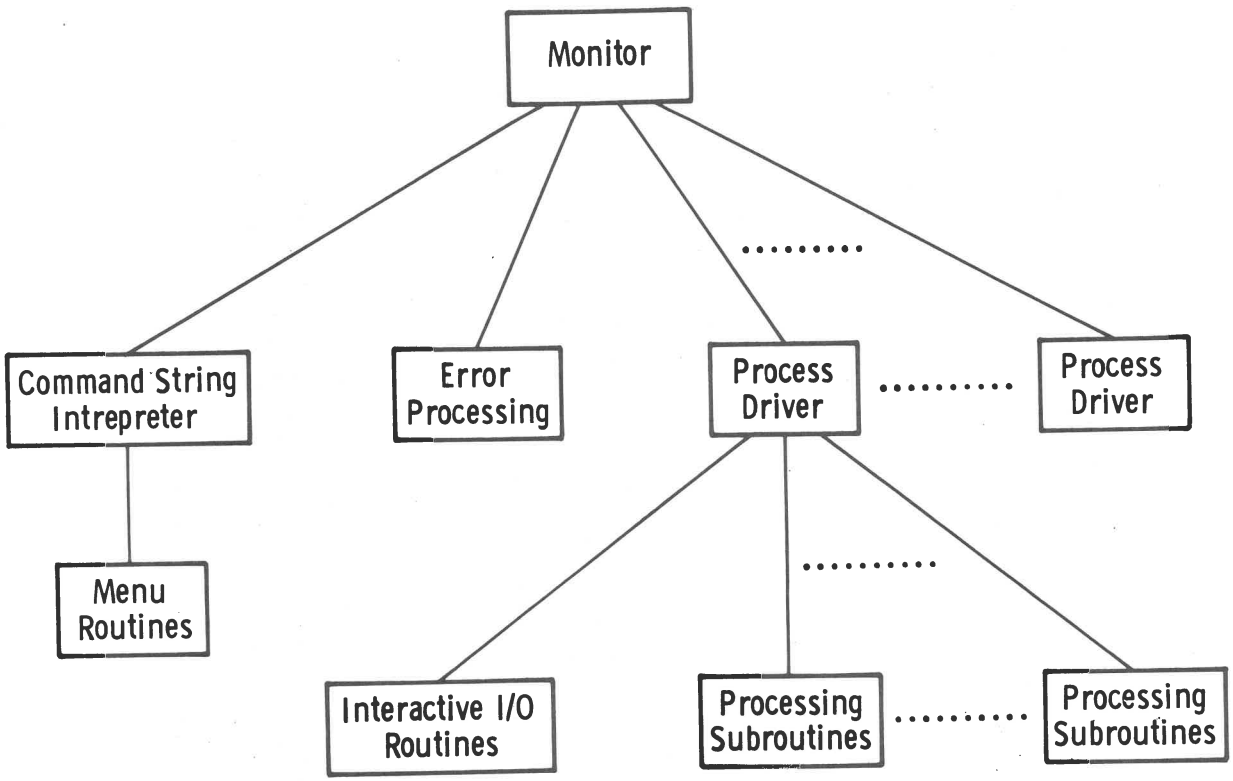

Figure 3. Illustrates the overall softwave flow of control.