Special Correspondence

Image Access Protocol for Image Processing Software

ROBERT M. HARALICK

Abstract—During the past decade a number of multiimage picture processing software packages have been put together. However, only a few of the references to picture processing systems discuss image data structure or input/output routines. This correspondence is a first step in a direction toward getting a communication process started by suggesting some specifications for a multiimage data format and standard input/output interface routines to access the image data.

Index Terms—Digital image processing, image processing, software.

I. Introduction

During the past decade a number of multiimage picture processing software packages have been put together. Among them are VICAR at the Jet Propulsion Laboratory, IDIMS at the Electromagnetic Sensing 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, OSER 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]–[15]. 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 by a one-to-many basis through the software journals.

Only a few of the references to picture processing systems discuss image data structure or input/output routines. This correspondence is a first step in a direction toward getting a communication process started by suggesting some specifications for a multiimage data format and standard input/output interface routines to access the image data. Such a specification would help move the various processing systems toward each other as well as help out those people designing their own new ones.

We will proceed by defining a multiimage, specifying the image input/output access routines, and then suggest a format for the multiimage data on a random disk device.

II. The Multiimage

We will consider the spatial domain for a digital multiimage 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 multiimage, each resolution cell has an N-tuple of numbers. The nth 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 multiimage is like a matrix of numbers. The multiimage itself is like a set of N matrices stacked one on top of the other.

Manuscript received May 11, 1976; revised November 3, 1976.

The author is with the Remote Sensing Laboratory, University of Kansas, Lawrence, KS 66044.

Because the number of rows and columns of a multiimage can easily be a few thousand, the entire image cannot be stored in memory 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 6-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 1 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 multiimage 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 multiimage 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 nth 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 always 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 only be done on numeric bands and should leave the symbolic bands alone.

III. 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 100 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 [16].

Milgram and Hayes [16] suggest two alternate data access methods: simulate random access on sequential files, or restrict use to sequential access. The first alternative is time consuming and does not work well for output to logical records at random. The second alternative is too restrictive for some algorithms, rubber sheeting, for example. A system providing true random access will support both of the alternatives. We assume, therefore, that the image access routines are for a random disk device.

There are four image input/output procedures of major consequence [3], [4]. 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.

These four operations on a random disk device allow, of course, updating of any logical records. Insertion or deletion of records would have to be done by copying appropriate records from the old file to a newly defined random file and then inserting the required information. Any call to one of the I/O procedures' will involve specifying or retrieving basic bookkeeping information about the image. For example, to open or initialize an old file, the procedure must retrieve from some kind of header record for the basic image parameters which include [3], [4], [16], [17] as in the KANDIDATS, System 101, and IDIMS System

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 for new file
IEV	is an event variable indicating status upon completion of initialize action
IALTRT	is the alternate return statement number taken on a bad status.

In addition to these parameters, Adams and Peterson [4] allow the user to specify how he wants the data organized by indicating who goes faster: samples, lines, or bands.

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 array in 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 ambiguities. 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 size can be different from the image size, a decision has to be made regarding whether the 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 [12], [16]. 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
     RWRITE (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 Ith band is to be accessed = 0 if Ith band is not to be involved
IBLKNO	is subimage block number to begin access 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.

An alternative version for the arguments of a READ/WRITE subroutine would omit the IPAR array and include it in the arguments for the initialize or open routine.

IV. The Multiimage File

The multiimage 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 multiimage file. Hence, the only area 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 [3]. 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, except for a possible time factor for disk head positioning. This is not so, however, for sequential accessing. Given the constraint that all bands for subimage block *i* should be able to read sequentially without a rewind after reading all bands for subimage block *i*-1, we see that the file must be organized having the image bands run the fastest. This should also decrease the seek time when processing multiple bands from a random access device.

Therefore, the multiimage 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.

References

[1] H. Frieden, "Image Processing System, VICAR, Guide to System Use," Jet Propulsion Lab., Pasadena, CA, July 1971.

[2] S. T. Alexander, "LADIES Software Description," Univ. 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," in *Proc. 3rd Symp. Machine Process. of Remotely Sensed Data,* Purdue Univ., Lafayette, IN, June 1976, pp. 1A8-1A17.

[4] J. Adams and G. Peterson, Stanford Technol. Corp., I/O Subroutine Documentation.

[5] R. C. Rathja and J. H. Herzog, "PIXSYS, A Users Manual," Pictorial Information Extraction and Enhancement Lab., Oregon State University, Pixel Rep. 080174.

[6] "Advanced Multispectral Image Descriptor System Report (AMIDS)," Rome Air Development Center, Griffiss Air Force Base, NY, Jan. 1975.

[7] F. J. Kriegler, "MIDAS, Prototype Multivariate Interactive Digital Analysis System–Phase I," Environmental Res. Inst. Michigan, Aug. 1974.

[8] G. Swanland, "Experimental Image Exploritation System," Control Data Corp.

[9] "Interactive Picture Analysis and Display Laboratory Brochure," Aeronutronic Ford, Palo Alto, CA, Feb. 1976.

[10] "Interactive Multispectral Image Analysis System Brochure (Image 100)," General Electric Co., Sept. 1973.

[11] J. W. Snively and E. B. Butt, "The Pax II Picture Processing System," Univ. 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, Lab. for Applications of Remote Sensing, Purdue Univ., Lafayette, IN, 1973.

[14] L. A. Gambino and M. A. Crombie, "Digital mapping and digital image processing," *Photogrammetric Eng.,* vol. XL, pp. 1295-1302, Nov. 1974.

[15] J. D. Turinetti and R. J. Hoffman, "Pattern analysis equipment and techniques," *Photogrammetric Eng.,* vol. XL, pp. 1323-1330, Nov. 1974.

[16] D. L. Milgram and K. C. Hayes, "Image Processing Software Design and Issues," Computer Science Cen., Univ. Maryland, College Park, 1976.