comunicaciones técnicas

1979 Serie Naranja: Investigaciones No. 165

DESIGN OF A HIGH LEVEL LANGUAGE FOR IMAGE PROCESSING*

T. Radhakrishan**
Renato Barrera
Adolfo Guzmán
Armando Jinich

* Reporte Técnico PR-78-22

** Concordia University, Montreal, Canádá

Recibida: 20 de julio de 1979

INSTITUTO DE INVESTIGACIONES
EN MATEMATICAS APLICADAS
Y EN SISTEMAS
UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

APARTADO POSTAL 20-726
MEXICO 20, D. F.
548-54-65

1. INTRODUCTION:

The language L is an extension to ALGOL and it is intended for programming image processing applications. It was developed with the specific problems of multispectral remote sensing data analysis in mind and is a natural outgrowth of the experience obtained in the design and implementation of the PR System [1]; is introduced with the objective that an individual with the knowledge of both a programming language and the PR problems should be able to solve those problems rather "easily" with L. Furthermore, we hope that such programs in L will be shorter and more "readable" than those in a POL (procedure oriented language). This approach of extending a POL to facilitate programming a specific class of problems is not totally new [2]. In fact, a language for picture processing, PAX [3], already exists. PAX was designed in 1960's as an extension (a set of subroutines) to FORTRAN. But since no new data type was introduced in PAX, a programmer still has to do a lot of book-keeping. It must be kept in mind that all image handling within L is done as if the whole images exist in random access memory; Therefore, it is a problem of implementation to do the transfer to and from disk that will undoubledly be required.

In this report, we specify only those parts of L which are different from ALGOL. Knowledge of ALGOL and the use of syntax diagrams are assumed [4,5].

Nothing particular about ALGOL, other that is used in the present PR implementation, has guided our base-language selection. Most probably other languages like PASCAL, ALGOL-W, or PL-1, with record and reference handling capabilities, would be more suitable for general image processing applications [6].

By way of extension, the language L includes the following:

(i) Six new data types: (four categories): Images, Image Groups; Boolean Images and Boolean Image Groups; Window; Virtual-Image, Virtual Image Groups;

(ii) A set of additional declarations to define, to store, and to process images conveniently.

(iii) A set of unary and binary operators on images and ways of defining predicates with the new data types.

(iv) The concept of "attribute" is introduced to define an integral property of an image. An attribute will have a value as determined by the associated procedure known to the compiler. Provisions are made in L to access, compute, and to store such attributes. This has been found valuable, from the earlier experiences, in solving PR problems. An important property of attributes is it provides in-built documentation and concise input and output.

II. DECLARATIONS:

In this section, we describe all the additional (to ALGOL) declarations. First, we present the motivation for a declaration, then we follow it with its syntax. An example is included for easy understanding and it is under-lined.

In the syntax definitions, the following syntactic units of ALGOL are used: [When it is necessary to distinguish among the different occurences of a syntactic unit, subscripts are used as ID_1, ID_2, etc.]:

a) IC : an integer constant

b) ID : an identifier (a simple name)

c) IV : an integer variable (a name that can take integer value)

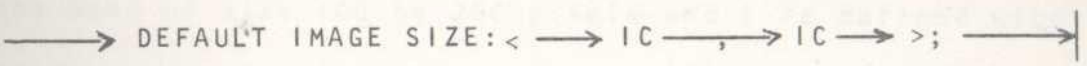
1) "Bits per pixel" is introduced to define the number of bits required to store one pixel of an image. We recommend that it be an integral multiple of the byte length of the computer for ease of implementation:

Syntax:

E.g. BITS PER PIXEL: 8;

2) An image is a collection of one or more rectangular array of pixels. The following declaration is self explanatory.

Syntax:

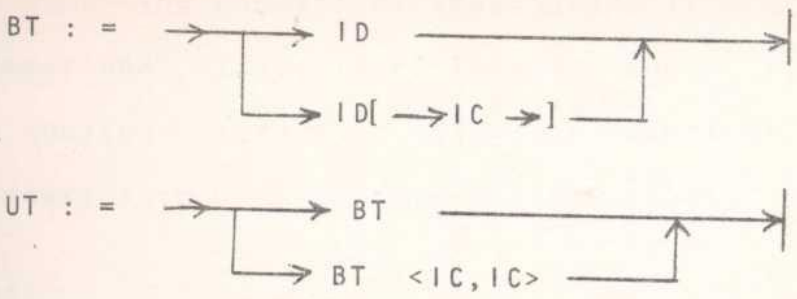


E.g. DEFAULT IMAGE SIZE: <1024,520>;

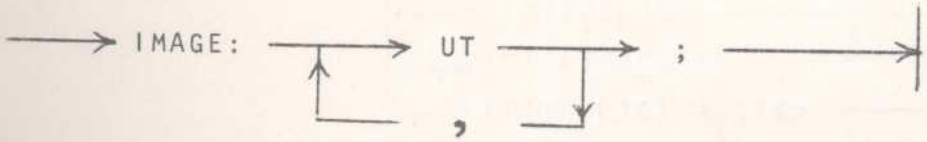
3) The data type image is declared by the following declaration. While programming in L, a programmer may not treat an image as a collection of pixels, if she does not prefer to do so.

Syntax:

Let the term UT be defined as



Then image is defined as follows:



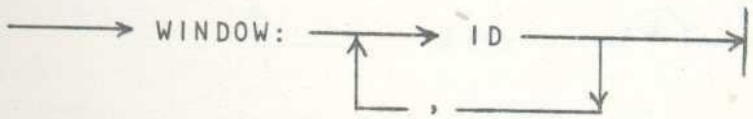
e.g.: IMAGE: A,B <100,200>;

IMAGE: C, D[3], E[5]<10,20>;

Each rectangular array of pixels is called a band. An image has one or more bands. The above example defines the images A and C each with one band and D with 3 bands, all having the default image size. Additionally B is defined with one band of size 100 by 200 pixels and E is defined with five bands, each of which has a size of 10 by 20 pixels. It is clear that all bands of a single image have the same rectangular size.

4) A window is defined to position and view a (rectangular) portion of an image. Some operations on windows and defining its frame size are discussed in section IV.

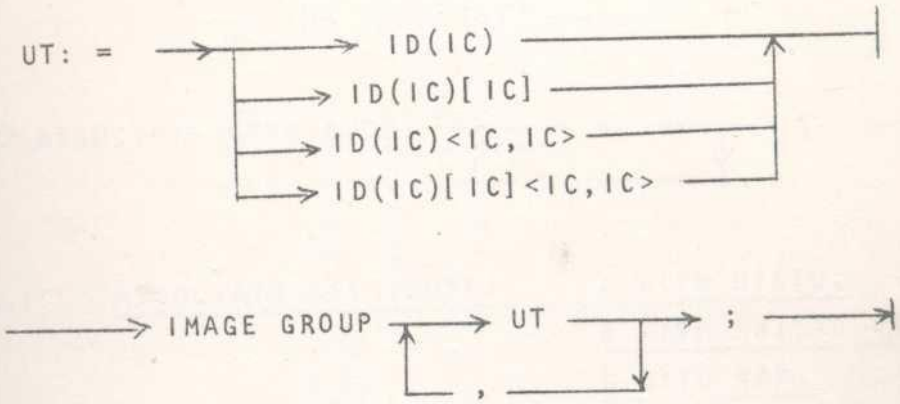
Syntax:



e.g.: WINDOW: P,Q,R;

The concept of image group is similar to the one-dimensional arrays in FORTRAN or ALGOL. It is introduced to consider a group of images of identical dimension and process them with the use of subscripts.

Let:



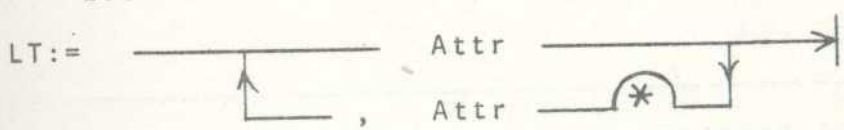
E.g. IMAGE GROUP: A(3), B(2)[3];

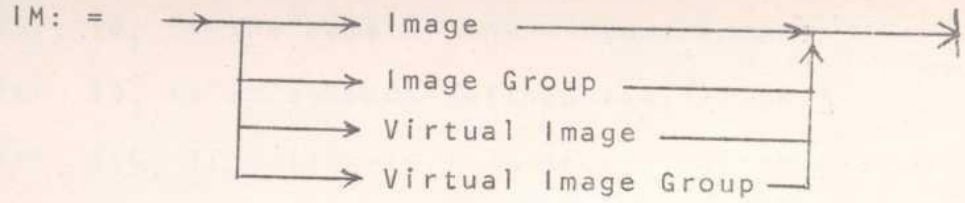
IMAGE GROUP: C(3)<100,200>, D(2)[3]<40,10>;

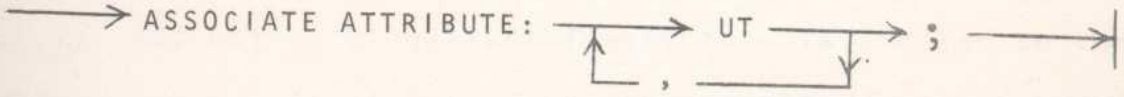
The image group A has three components and each of them has a single band; the image group B has 2 components and each of them has 3 bands. They all have default image size. The image groups C and D are similar to A and B respectively; but their sizes are different as shown in the declaration.

5) As stated in section I, an image may be associated with one or more attributes. Let Ω denote the set of all attributes known to the compiler of L.

Let Attr ε Ω and



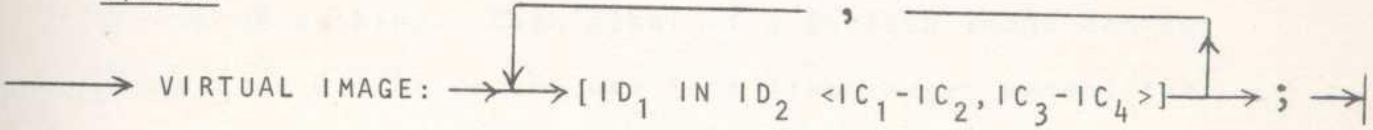




E.g.: ASSOCIATE ATTRIBUTE: A WITH HISTO,
B WITH (HISTO,VAR),
C WITH VAR;

6) Virtual image is a concept quite useful in supervised classification. A virtual image can be associated with attributes, like real images, but no (real) pixels are associated with it. It is defined as a fixed sub section of a real image, and the corresponding pixels are not stored separately other than in the real image. When a virtual image is stored on (or retrieved from) a disk, only its attributes are stored. It is the responsibility of the programmer to keep track of the associated real image*.

Syntax:



*This definition of virtual image has not been totally satisfactory to all the group members. We hope to improve it with further experience.

(i) ID_1 is the name of the virtual image

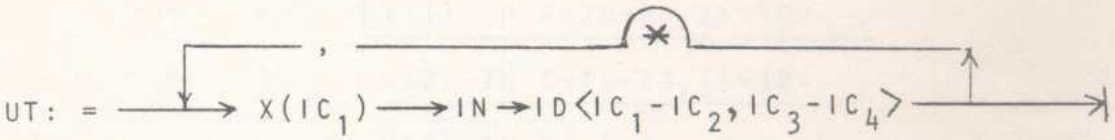
(ii) ID_2 is an already defined real image.

(iii) $(IC_1\text{-}IC_2),(IC_3\text{-}IC_4)$ defines a rectangular window in ID_2.

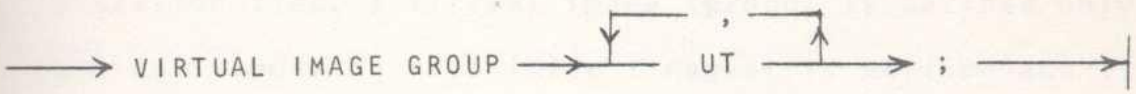
E.g. VIRTUAL IMAGE: [X IN A<10:100,10:200>] ,
[Y IN A<100:100,100:200>] ;

7) The notion of virtual image group is similar to the image group. It is introduced to experiment with several virtual images and each member of a group is accessed with a subscripted variable.

Syntax:



where $X(IC_1)$ is the subscripted name of the virtual images and it must be X(1),X(2),X(3)... in the consecutive order. An image group must have at least 2 members.



Since a window does not satisfy all the requirements of masking, the type BOOLEAN IMAGE is defined for the purpose of masking. Each pixel of a Boolean image can be either 1 or 0. It is defined just like images except the keyword BOOLEAN is prefixed. It is possible to define both Boolean images and Boolean image groups. Both of them are assumed to have only one band.

E.g. BOOLEAN IMAGE: P,Q<100,200>;

BOOLEAN IMAGE GROUP: L(3), M(2)<40,50> ;

The Boolean constants 1 and 0 are referred to as in ALGOL. Now we have the following image types:

	Real	Boolean	Virtual
IMAGE	Yes	Yes	Yes
IMAGE GROUP	Yes	Yes	Yes

E.g. VIRTUAL IMAGE GROUP: [X(1) IN A<10-20,10-30> ,
X(2) IN B< 5-15, 5-25>] ,
[Y(1) IN A<20-40,20-40> ,
Y(2) IN C<15-25,15-30> ,
Y(3) IN B< 0-5, 0-8 >] ;

In the above example two virtual image groups X and Y are defined with 2 and 3 components respectively. At the execution time, a virtual image (group) is defined only if the corresponding real image (images) is defined and it is the responsibility of the programmer.

In the declaration section, the declarations should occur in the following order:
IMAGE, IMAGE GROUP, VIRTUAL IMAGE, VIRTUAL IMAGE GROUP and ASSOCIATE ATTRIBUTE. WINDOW and BOOLEAN IMAGE declarations can be placed anywhere in that section.

III. ADDRESSING:

In this section we describe how the various data types and their components can be addressed while programming. The three groups of parentheses; ( ),[ ], < >, are used as follows:

1. A(2) refers to the second component image of the image group A.
2. B[3] refers to the third band of the multi-band image B.
3. C<4,5> refers to the pixel on the 4-th row and the 5-th column on the band (or single-band image) C.

Some valid combinations:

A(2)[3]<4,5> refers to a pixel

A(2)[3] refers to a band

A(2) refers to an image

A(2)<4,5> refers to a vector consisting of the pixel <4,5> in every band of the image A(2).

4. X.ALL Refers to the image X and all its associated attributes. Only I/O operations can be made with X.ALL.
5. X.ATT Refers to the set of all attributes of the image X. Like (4), it can be used only for I/O operations.
6. X:W Refers to the image X viewed through the window W. Before using this mode of reference, the programmer must POSITION the window W on X. (refer to Section IV).

7. X. Attr Refers to the attribute Attr associated with the image X. Attr ε Ω e.g. X. HISTO

8. CHANGED (X.HISTO) It is a predicate, TRUE or FALSE (HISTO can be replaced in this example by any Attr ε Ω). Each attribute has associated to it a status bit.

   The status bit is tested by this predicate. CHANGED IS TRUE if the image has been altered, since the last computation of the attribute in such a way that the attribute no longer corresponds to the modified image.

9. COMPUTE (X. HISTO) This invokes the computation of the attribute HISTO on the image X and stores the result. Also it resets the status bit. The status bit is set when the image is changed during the execution.

NOTE: We define a null pixel as a pixel with no value and represent it as Λ. It is different from a pixel with 0 value.

IV. OPERATIONS ON DATA TYPES:

The following factors are important:

(i) No operation on image groups is provided, that is, there is no operation similar to the vector addition of APL.

(ii) The addressing modes X.ALL and X.ATT can be used only with I/O operations.

(iii) A virtual image reference can never occur on the left hand side of an assignment statement. This means pixels (images) can not be changed by referring through a virtual image.

Fig. 1a. A:=B

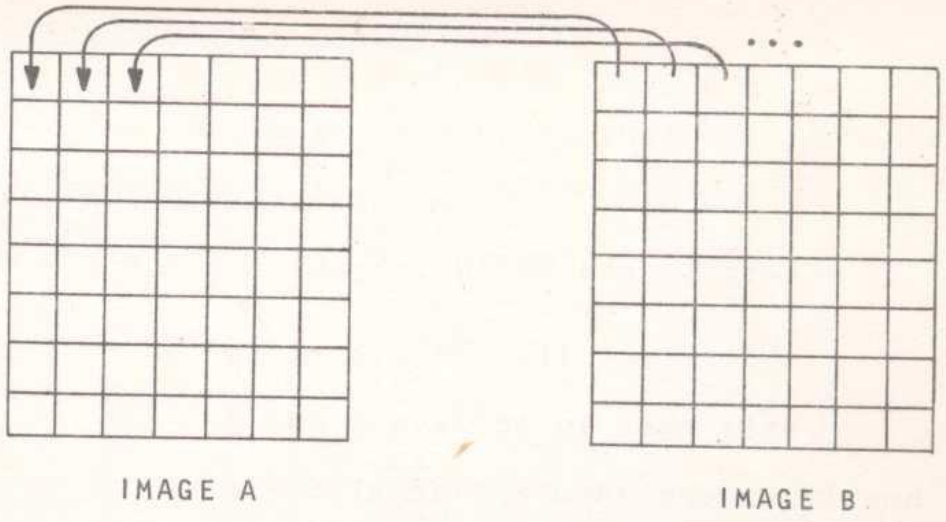


Fig. 2a. A:=B:W

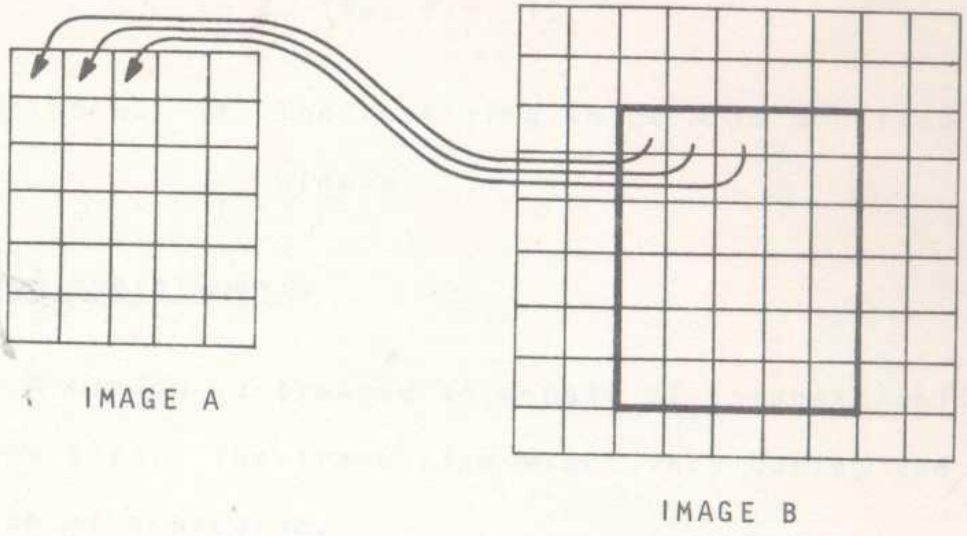


Fig. 3a. A:W1:=B:W2

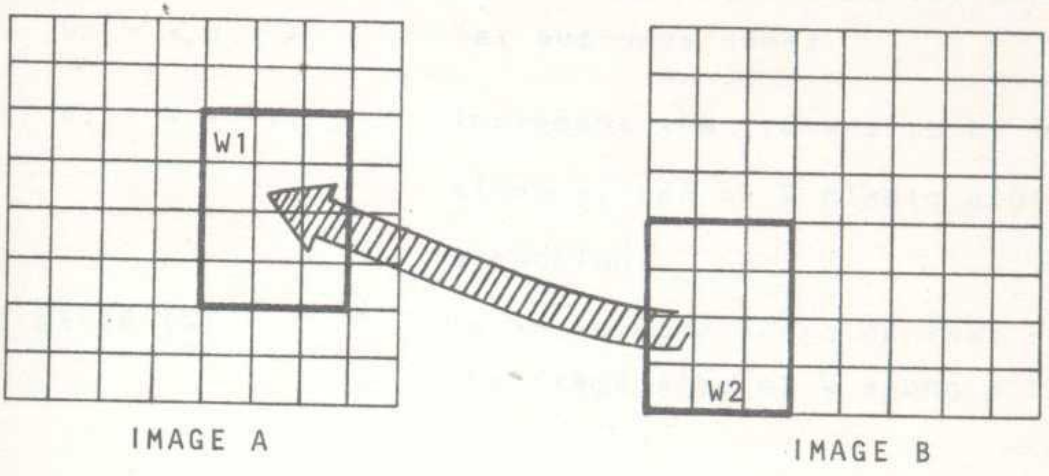


FIGURE 1. IMAGE ASSIGNMENT

1. Assignment operations:

The syntax is similar to that of Algol. Consider the following examples:

A,B,C are images; W1,W2 are windows:

a) A:=B => $a_{ij} = b_{ij}$ for all i,j.
A and B must be of same size.
or A $\geq$ B in both dimensions (x,y) and in bands. (See Fig. 1)

b) B:=C:W1 => The image C viewed through W1 is assigned to B. (See Fig. 1)

c) A:W1:=B:W2 => The receiving image A is modified by a window

2. Window operations:

A window is treated as a pair of integers defining its frame size. The frame size might vary during the execution of a program.

a) W: = <20,10> define the frame size for W same as
W: = <p,v> (a) but uses names

b) W: = W +<1,2> increment the frame-size by 1 pixel along x, and by 2 pixels along y direction.

c) XSIZE (W) is an integer function that returns the frame size of W along X axis

d) YSIZE (W) Similar to (c) but along the Y axis.

e) POSITION (W,A,x,y) will position the window W on the image A at (x,y)

f) INRANGE (W,A) is a predicate, TRUE when the window W is completely in the range of the image A. (See Fig. 2); otherwise it returns a value FALSE.

g) LOCATE (W,A,x,y) It is a procedure that returns the (x,y) present position of the window W on the image A.

3. Operations on Images: There are two ways to provide operations on images: (i) To define operators as the arithmetic operators +,- etc. (ii) To provide built-in functions as in PAX. We adopt both strategies. For some common and well known operations, we use the strategy (i), and for the others we provide built-in functions. Both these aspects are summarized in Table I and Table II respectively.

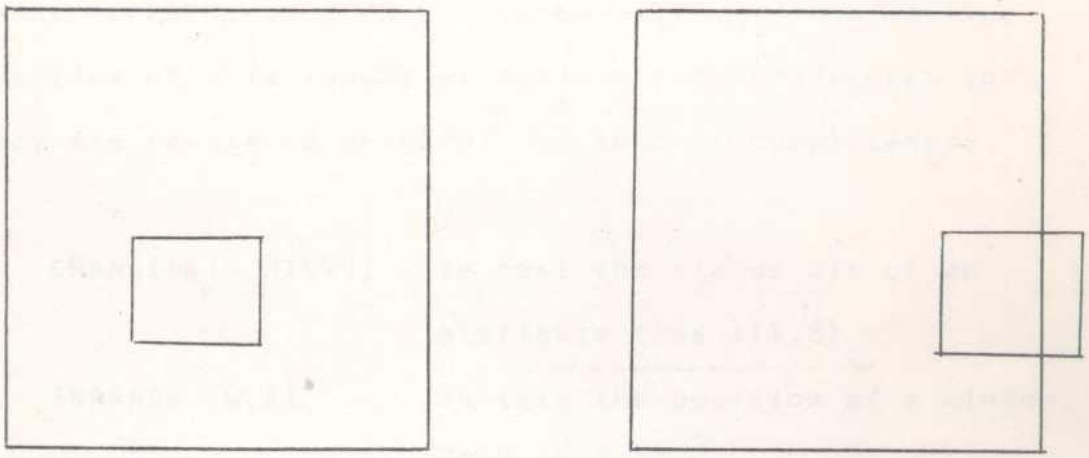


Fig. 2. Window positions

V. PREDICATES FOR FLOW CONTROL:

Predicates in programming languages are useful to control the flow of execution. They are used in decision making (IF...THEN...ELSE....) and in iterative execution (FOR, REPEAT and DO WHILE statements). In the earlier sections of this report we defined some predicates and they are re-stated here for the sake of completeness.

1. CHANGED (X.HISTO) — To test the status bit of an attribute (see III.8)
2. INRANGE (W,A) — To test the position of a window. (See IV.2.f)
3. <i,j> IN A — is TRUE if the pixel <i,j> is inside the image A; FALSE otherwise. The image A may be qualified by a window as A:W. A variant of this predicate is quite useful in a FOR statement as shown below:

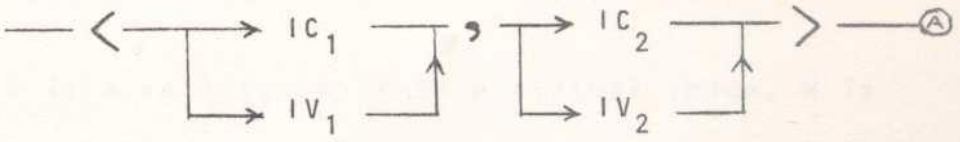
FOR ALL <i,j> IN A:W DO;

The following block is executed for every pixel in the image A viewed through the window W.

The order of consideration of pixels is not known to the user. If the programmer needs a specific scan sequence (row major or column major) he should not choose this

construct; instead he can construct a (two level) nested loop. Because the scan-order is insignificant in this construct, asynchronous parallel processor may be used to perform the functions specified [7].

Syntax:



e.g.:

FOR ALL <i,j> IN A:W DO;
IF <1024,2047> IN B THEN PRINT, LARGE;

A number of other predicates can be formed using the functions in Table II and the relational operators ($> \geq < \leq = \neq$) of ALGOL.

VI. INPUT/OUTPUT OPERATIONS:

We conform to the syntax of ALGOL, except the I/O list may include images. The I/O list may include any of the following:

X, X·ALL, X·ATT, X·Attr, X:W, Y·ATT, Y·Attr

where X is a real image, Y is a virtual image, W is a window and Attr ε Ω . Recollect that when a virtual image is written (or read) only its attributes are written.

As in any I/O operation, it will be the programmer's responsibility to keep track of the record lenght and the number of items written, etc. For the sake of I/O, the bands of an image will be written in the ordinal sequence 1,2,3....etc.; each band is written (read) just like a two dimensional array.

As a convention, the attributes of an image will be written (read) before its pixels. The order in which the various attributes are written will correspond to their order in the declaration.

VII. ATTRIBUTES OF IMAGES AND BANDS

As stated in the earlier sections, the set of all attributes known to the compiler of L is denoted by Ω. There is only an operational difference between functions and attributes. Certain functions, in PR applications, are intimately associated and used with images. It is worthwhile that such functions may be stored, retrieved, and addressed as an integral part of the image. As a result we have introduced the concept "attributes". An attribute can be associated to one or more images but this association is static. Similarly an image may be associated with one or more attributes.

An attribute is characterized by the following:

(i) Each attribute has a corresponding procedure (function) that computes certain values. The function has only one image as its argument. The COMPUTE (A'HISTO) statement (refer III.9) invokes a call to this function.

(ii) The data type returned by the above function determines the storage requirement and a knowledge of this requirement is necessary at compile-time for storage allocation.

(iii) Each association of an attribute with an image has a status bit assigned. In sections III.8 and III.9,

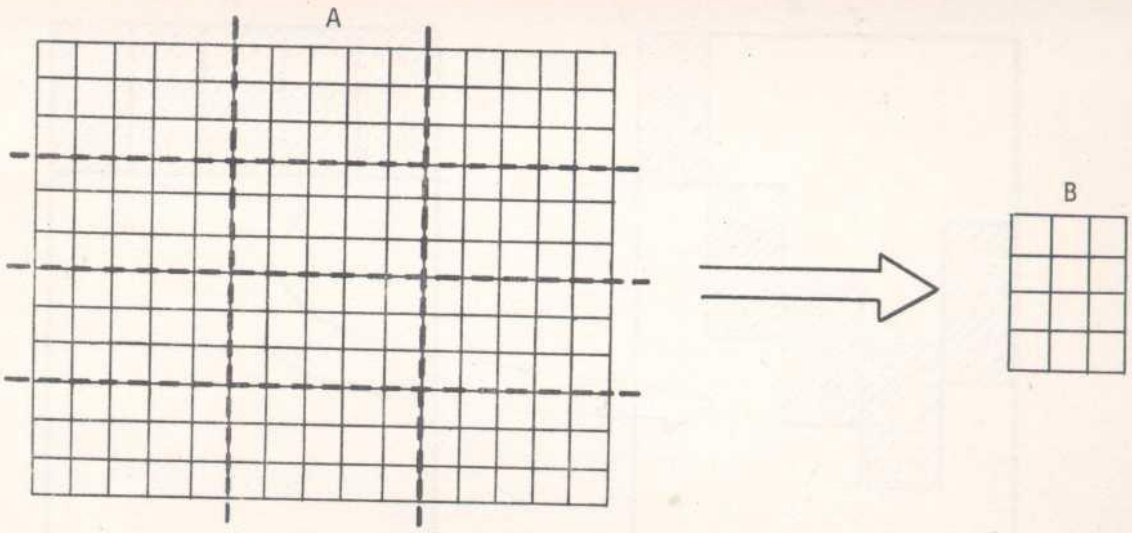
we described the access and control of this status bit. When there is a change in the image, some of the status bits (not necessarily all of them) associated with that image will be set. A status bit is reset by the associated procedure. Thus, the logic of resetting is embedded into the procedure and the logic of setting is integrated with the semantics of the assignment statement.

The concept of attributes is expected to improve the readability of PR-programs and the in-built documentation. This premise has to be verified in the "language-evaluation-stage". However, the programmer should remember that this facility is not without its a price.

In Table III we present a set of attributes (set Ω) as construed at the time of the design of L. After the evaluation stage, this set may change.
Eventually, when the system programmer wants to add a new member to Ω, she should remember the above requirements.

VIII. PROGRAMMING EXAMPLES

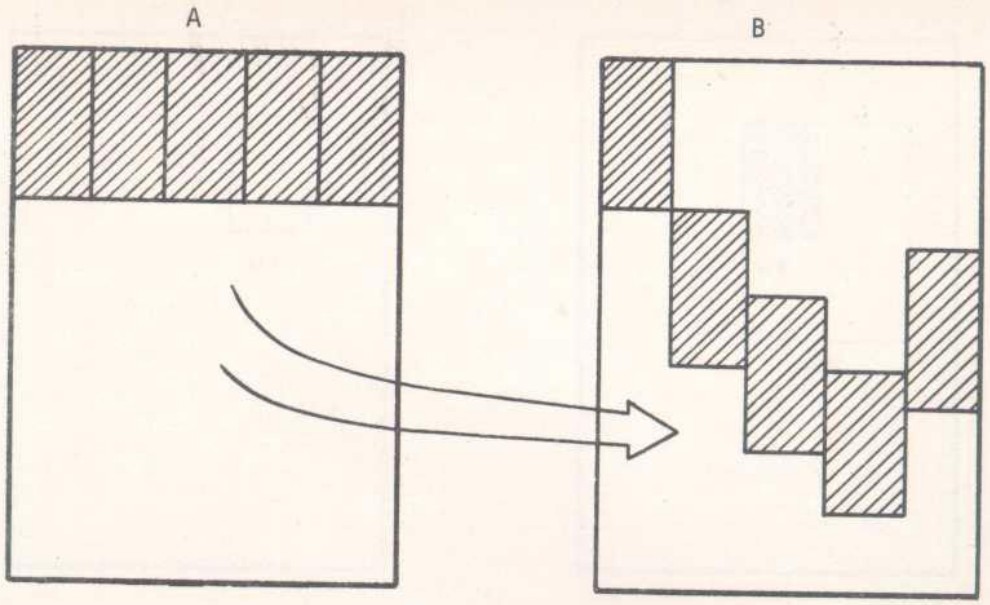
A few examples follow which give an adequate feeling the expressive power of the language. The simple programs presented implement algorithms which in, say, plain ALGOL, are complicated by the need for explicity controlling image data input/output and non-existence of the simplifying syntax constructs.



In this example image B is formed by taking each pixel as a 5x3 mean of pixels in image A.

```
IMAGE A<15,12>,B<3,4>
WINDOW V;
V:=<5,3>
FORALL <I,J> IN B DO
   BEGIN
      POSITION(V,A,I*5,J*3);
      B<I,J>:= MEAN(A:V);
END;
```

EXAMPLE 1 - MEAN IMAGE

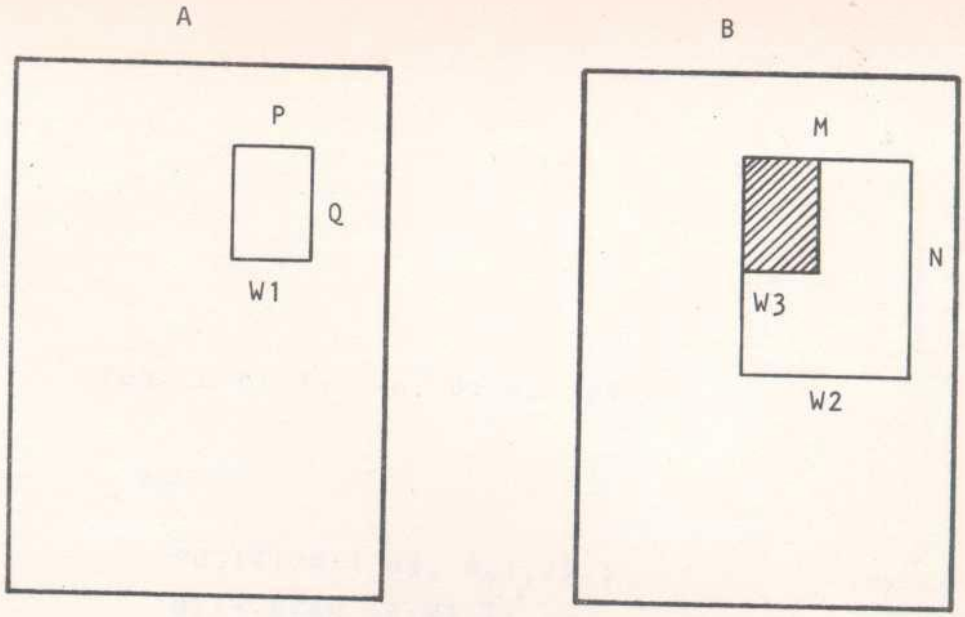


This example shows an arbitrary piece-wise transformation from image A to image B which could be useful in geometric correction problems.

```
IMAGE A,B<n,m>;
WINDOW V,W;
V:=W:=<p,q>;
FORALL <i,j> IN A DO
   BEGIN
     POSITION(V,A,I,J);
     POSITION(W,B,F1(I),F2(J));
     B:W:=A:V;
END;
```

F1 and F2 are the X and Y transformation functions.

EXAMPLE 2 - IMAGE TRANSFORMATION



It is desired to find the position of highest correlation of window W1 within window W2. Therefore W3 (same size as W2) will move within W2 calculating a certain distance function.

```
IMAGE A,B<S,T>;
WINDOW W1,W2,W3;
W1:=<P,Q>;
POSITION(W1,A,X1,Y1);
W2:=<M-P,N-Q>;
POSITION(W2,B,X2,Y2);
W3:=<P,Q>;
M1:= MEAN(A:W1);
```

(continued next page)

EXAMPLE 3. CORRELATION

```
FORALL <I,J>  in  B: W2  DO

  BEGIN

    POSITION ( W3, B,I,J) ;
    M2:= MEAN (B:W3 );
    FORALL <K,L>  IN B: W3 DO

       DIST:= DIST + A:WI <K,L> -M1      % DISTANCE
              -B: W3 <K,L> + M2;         % MEASURE


    IF DIST  < MINDIST THEN
       BEGIN

          MINDIST:= DIST;
          IMIN:=I;
          JMIN:=J;

       END;

END:
```

EXAMPLE 3. CORRELATION (Continued)

REFERENCES

1. Guzmán, A. Heterarchical Architectures for Parallel Processing of Digital Images. Comunicaciones Técnicas, National University of Mexico, IIMAS, AHR-79-3 (PR-79-23).

2. Crespi, S.R.; Morpurgo, R. A Language for Treating Graphs. Comm. of ACM, Vol. 13 (5): 319-323, 1970.

3. _____ PAX II Picture Processing System, in Picture Processing and Psychopictorics (Ed) B.S. Lipkin.

4. _____ Burroughs B6700/B7700 ALGOL Language Reference Manual.

5. _____ Burroughs B6700/B7700 Command and Edit Language (CANDE) Manual.

6. Zahn, C.T. Jr. Data Structures for Pattern Recognition Algorithms, in A. Klinger, K.S. Fu, T. L. Kunii, Data Structures, Computer Graphics and Pattern Recognition, Academic Press, 1977.

7. Guzmán, A.; Segovia, R. A Parallel Configurable Lisp Machine. Comunicaciones Técnicas, National University of Mexico, IIMAS, Vol. 7, No. 133.

TABLE I

Operators on Images

No.	Operator Symbol	An Example	Semantics	Remarks
1	+	A = B + C	$a_{ij} = b_{ij} + c_{ij}$ for all i,j	A,B,C must refer to bands.
2	-	A = B - C	$a_{ij} = b_{ij} - c_{ij}$ for all i,j	The result is limited between
3	*	A = B * C	$a_{ij} = b_{ij} * c_{ij}$ "	0 and 2^n-1
4	/	A = B / C	$a_{ij} = b_{ij} / c_{ij}$ "	Division is integer division
5	∩	A = B ∩ C	$a_{ij} = b_{ij}$ iff $b_{ij} = c_{ij}$ = otherwise	
6	∧	A = B ∧ C	$a_{ij} = b_{ij}$ iff $c_{ij} = 1$ = otherwise	C is a Boolean image
7	¬	A = ¬ B	$a_{ij} = max - b_{ij}$	max = 2^n-1

TABLE II

Built-in Functions

No.	Function Name	List of arguments	An example usage	Semantics and Remarks
1	MAX	a band	Y=MAX(A[1])	returns the maximum value of pixels in the band A[1]
2	MIN	"	Y=MIN(A[1])	similar, but returns the minimum
3	MEAN	"	Y=MEAN(A[1])	
4	VAR	"	Y=VAR(A[1])	returns the variance
5	SHIFT	a band; an integer; a code	SHIFT (A,X,5)	Shift the band A along X by 5 positions. X is replaced by R for right, L for left, U for up, D for down. Vaccated pixels are replaced by Λ
6	ROTATE	a band; an integer; a code	ROTATE(A,X,3)	Similar to shift but no pixel is lost
7	NUMZ	a band	Y=NUMZ(A[1])	Returns the number of zero pixels in A[1].
8	NUML	a band	Y=NUML(A[1])	returns the number of null pixels
9	BOOL	a band, a threshold	A=BOOL(B,τ)	A must be a Boolean image $a_{ij} = 1$ iff $b_{ij} \geq \tau$ $= 0$ otherwise

TABLE II (cont.)

No.	Function Name	List of arguments	An example usage	Semantics and Remarks
10	CONECT	a band, a logical variable, two integer variables, an array, a code	CONECT(A,X,p,q, B,L)	In A, a closed area is searched; if found L is set to TRUE; otherwise L is FALSE. The array B contains the Freeman chain of the boundary and (p,q) contains the starting address of the chain. Set X=1 for 4 connectedness Set X=2 for 8 connectedness
11	CONERA	same as CONECT	CONERA (A,X,p,q, B,L)	Finds, the closed region and sets all the interior and boundary points to Λ.
12	CONBON	"	CONBON(A,X,p,q, B,L)	same as CONERA but the boundary pixels are not replaced by Λ.

TABLE III

Attribute Set

No.	Attribute name	Data Type returned	Semantics	Remarks
1	HISTO	a vector for each band	Computes the histogram of pixel values of a band	The vector has 2^n elements where n is the number of bits/ pixel
2	MEAN	a scalar for each band	self explanatory	-
3	Variance	a scalar for each band	"	-
4	Covariance	a matrix	"	-
		Following are Non-computed. Id Info.		
5	Name	Alphanumeric		name of the image
6	Date	Numeric	DD/MM/YY	
7	Creator	Alphanumeric		
8	Father-Image	Alphanumeric		of which this is a sub-image.