Final Report for the LARS/Purdue-IBM Houston Scientific Center Joint Study Program

by

P.E. Anuta, E. M. Rodd, R. E. Jensen and P. R. Tobias

The Laboratory for Applications of Remote Sensing

Purdue University
Lafayette, Indiana
Final Report
for the
LARS/Purdue - IBM Houston Scientific Center
Joint Study Program
by
P. E. Anuta¹, E. M. Rodd², R. E. Jensen²
and P. R. Tobias²

I. Summary

This report describes work performed during the second part of the LARS/IBM HSC Joint Study program which began April 1, 1971 and terminated April 1, 1972. Work during the first part of the study ending November 30, 1971 concentrated on analysis of the HSC vidicon film scanning system. Included was a comparison of data derived from this system and the LARS/U. of Michigan airborne multispectral scanner system. Comparison of vidicon digitized film and film digitized on a rotating drum microdensitometer was also included. This work is reported in IBM Publication No. 320.2421 "First Interim Progress Report for IBM Houston Scientific Center/LARS - Purdue Joint Study Program." The second part of the study consisted of development of a closed boundary finding algorithm by IBM and its evaluation by LARS. The purpose of the algorithm is to enable automatic determination of boundaries, such as agricultural

¹LARS (Laboratory for Applications of Remote Sensing) Purdue University
²IBM Scientific Center, Houston, Texas
field boundaries, in digitized aerial or satellite imagery. The algorithm is described and a pictorial evaluation is presented in this report.

II. Field Selection in Digital Images

One of the major problems in the development of automatic pattern recognition systems for earth resources image data is the delineation of closed boundaries around homogeneous areas containing a material of interest. Identification of these closed areas allows categorization of all the enclosed image elements by group pattern recognition techniques. The reliability and speed of the pattern recognition process is markedly increased when groups, i.e., fields, of samples are used. If the fields are not known each image element in the data must be classified separately which is very time consuming and less accurate. Thus, closed boundary finding methods are being widely studied as part of earth resources data analysis research.

Recent work reported by Rosenfield\(^1\) and Anuta\(^2\) approached the boundary finding problem using the picture gradient which spatially differentiates the data and produces an edge picture. The problem with this approach is that the gradient is inherently "noisy" and produces borders which are discontinuous, of varying width, and the output also consists of many isolated spurious border points. A more stable and lower noise approach using clustering is reported by Wacker.\(^3\) The quality of his boundary output is high; however, the algorithm is very time consuming and closed boundaries are not guaranteed. The boundary finding algorithm reported here guarantees closure and is relatively fast. The algorithm description which follows is extracted from an IBM report by Rodd\(^4\) who developed the program.
III Closed Boundary Finding Algorithm (CBA)

In the following discussion each point of a digital picture is called a pixel (picture element) and a group of points is called a pixel group. In the implementation, pixel groups are squares of \( g \) pixels on a side and thus have \( g^2 \) pixels in each group. A field is a collection of pixel groups which have been found to come from the same statistical distribution. The algorithm proceeds by computing statistics for a pixel group and comparing them to the statistics of existing sets of pixel groups using the \( t \)-test. The \( t \)-test is used to determine whether two adjacent groups came from the same distribution and thus from the same field.

1. Statistical Method

Before describing the geometry of building fields using more than one spectral range we describe the statistical method used when comparing two samples. The \( t \)-test is used to compare a pixel group which has not been classified as part of a field with some other pixel group or set of pixel groups which form a defined field. The formulas used make two assumptions about the population formed by the pixels: (1) the populations of all fields are normal and (2) the populations of all fields have the same variance.

Call the unclassified pixel group sample 1 and the collection of one or more pixel groups which are classified sample 2. The goal is to compute a value of \( t \) to compare against a critical value to determine if the two samples are from a single population (i.e. determine if the pixel group is part of the same field as sample 2).
Let:

\( X_{ij} \) be pixel \( i \) of sample \( j \),
\( \bar{X}_j \) be the mean of sample \( j \),
\( x_{ij} \) be \( (X_{ij} - \bar{X}_j) \),
\( n_j \) be the number of pixels in sample \( j \).

Note that \( N_j = g^2 \) if only one pixel group is considered.

First compute the pooled estimate of variance.

\[
S_p^2 = \frac{\sum_{i=1}^{n_1} x_{i1}^2 + \sum_{i=1}^{n_2} x_{i2}^2}{(n_1-1) + (n_2-1)}
\]

Note that the \( \sum_{i} x_{ij}^2 \) is computed as usual by

\[
\sum_{i} x_{ij} = n_j \sum_{i} x_{ij} - \left( \frac{\sum_{i} x_{ij}}{n_j} \right)^2
\]

Now the value of \( t \) is:

\[
t = \frac{\bar{X}_1 - \bar{X}_2}{\sqrt{S_p^2 \left( \frac{1}{n_1} + \frac{1}{n_2} \right)}}
\]

A two-tailed test is used with the number of degrees of freedom equal to \( (n_1 - 1) + (n_2 - 1) \).

Occasionally a single pixel group will contain pixels from two areas with radically different values of \( \bar{X}_j \) or will contain mostly pixels from say a darker area and one or more pixels from an extremely bright area such as a road or a house with a white roof. This will make \( S_p^2 \) very large and consequently make \( t \) very small thus confirming the null hypothesis that the two samples are
from the same field. Actually this is not the case for sample 1 does not belong with any other pixel groups. In fact, if sample 1 is included in the field with sample 2, any further computations of t using the field with samples 1 and 2 now combined into a single field will produce a large $s^2_p$ and small t and an acceptance of the null hypothesis. What has really happened is that sample 1 violates assumption 2. To avoid the problem it is necessary to compute the standard deviations $\sigma_1$ and $\sigma_2$ and if either is greater than some empirical constant times the appropriate $\bar{x}_j$, the null hypothesis is rejected.

2. Building of Fields

The picture is processed one row of pixel groups at a time. First the field building algorithm will be described in terms of processing a picture of one spectral band. Before processing starts on a row of pixel groups, the sums of the elements and sums of the squares of the elements of each pixel group in the row are computed. The pixel groups are numbered left to right so that $s_1$ is the sum of elements of the leftmost pixel group in the row and $q_1$ is the sum of the squares of the elements of the leftmost pixel group. There are W pixel groups in a row.

The k'th group of the row of pixel groups being processed will be designated as group k. The first group in the current row is thus group 1. The field to which group k is assigned is given by $f_k$. The k'th pixel group in the previous row of pixel groups is designated as group (k). The first such group is group (1). The field to which group (k) is assigned is given by $(f)_k$.

The first row is easiest to process. Group 1 is arbitrarily assigned to field 1 and field 1 is noted as containing one pixel
group. Letting $S_n$ be equal to the sum of the pixels in field $n$ and $Q_n$ be equal to the sum of squares of pixels in field $n$, then $S_1$ is equal to $s_1$ and $Q_1$ is equal to $q_1$.

Now group 2 is compared to group 1 (group 1 is sample 2 and group 2 is sample 1). If the null hypothesis is accepted, group 2 is added to field 1. Adding to field 1 means noting that field 1 now has 2 pixel groups and

$$S_1 = S_1 + s_2$$
$$Q_1 = Q_1 + q_2.$$  

If the null hypothesis is rejected, then group 2 is assigned to field 2 and field 2 is noted as containing one pixel group and

$$S_2 = s_2$$
$$Q_2 = q_2.$$  

Next group 3 is compared to group 2 and in general group $k$ is compared to group $k-1$ until the end of the row is reached.

The second row and all succeeding rows are processed by the following steps.

(a) Group 1 is compared to the field containing group (1). Two possibilities exist. Either the null hypothesis is accepted and group 1 is considered part of field $(f)_1$ or the null hypothesis is rejected and group 1 is different.

If they are the same (i.e. group 1 is part of field $(f)_1$), group 1 is added to field $(f)_1$. Adding means noting that field $(f)_1$ has one more pixel group and

$$S(f)_1 = S(f)_1 + s_1$$
$$Q(f)_1 = Q(f)_1 + q_1.$$
Note that \( S(f)_1 \) means the \((f)_1\)'th element of \( S \) where \((f)_1\) is the field to which group \((l)\), the first group on the previous row, was assigned.

If they are different processing continues to group 2.

(b) Group 2 through \( W-1 \) are processed in the same manner. Group \( k \) is compared to the field which contains group \((k)\). If they are the same, group \( k \) is added to field \((f)_k\). Then a check is made to see if group \( k-1 \) has been assigned to a field. If so, processing continues to group \( k+1 \). If not, group \( k-1 \) is compared to field \( f_k \) (which is the same as \((f)_k\)). If they are different, processing continues to group \( k+1 \). If they are the same, group \( k-1 \) is added to field \( f_k \) and then a check is made to see if group \( k-2 \) has been assigned to a field and so on.

If they (group \( k \) and field \((f)_k\)) are different a check is made to see if group \( k-1 \) has been assigned to a field. If not, processing moves to group \( k+1 \). If so, group \( k \) is compared to field \( f_{k-1} \). If they are the same group \( k \) is added to field \( f_{k-1} \). If not, processing continues to group \( k+1 \).

(c) Group \( W \) is processed just like groups 2 through \( W-1 \) except that upon termination of processing of group \( W \), there is no group \( k+1 \) to begin processing.

(d) After group \( W \) is processed, a second pass of the row is made, this one backwards across the row. The pass starts at group \( W \). When a group \( k \) is encountered which has not been assigned to a field, group \( k \) is assigned to a new field. Then
if group k-1 is unassigned, it is compared to group k. If they are the same, group k-1 is added to field f_k and group k-2 is checked to see if it is unassigned and so forth. Whenever group k-j is unassigned but different from the field which contains groups k through k-j+1, then group k-j is assigned to a new field and a check is made to see if group k-j-1 is assigned and so forth. Whenever group k-j is already assigned, scanning continues for unassigned groups.

To handle multi-spectral data, a vector S, a vector Q, a vector s and a vector q must be maintained for each spectral band. When two samples are compared, a t is computed for each spectral band. If the null hypothesis is rejected on the basis of the value of t for any spectral band, then the samples are different.

IV. Implementation

The algorithm has been implemented on System/360 under CP-67/CMS. The program is written in FORTRAN with the exception of assembly language subroutines for input data conversion and for timing. The data for each channel are on a separate file on secondary storage. The data files are sequential with each record containing one row of pixels. It is simple to adapt to some other format of raw data because all input of pixel data is done from a single subroutine.

The following parameters are input to the program:
1. The value of g, the number of pixels on one side of a pixel group.
2. Those rows and columns of the picture which are to be processed.
3. The significance level to be used for the t-test. Four values are available.

4. The number of channels to be processed.

The program can process any number of rows of pixels on a single run. The number of columns is limited by the dimensions of certain variables. Thus, the program is designed to process pictures which are long strips.

Output is to a line printer. Results for the first 50 rows of pixel groups are printed after the first 100 rows are processed and results for the next 50 rows printed after another 50 rows have been processed and so forth. On the output, each field is assigned a character. Only the boundaries of each field are printed. The program is intended to be used with a per field classification program so that when a field is closed (closed means that no pixel groups in the current row of pixel groups belong to that field), the classification program is invoked to classify the closed field as corn etc. Also before printing, any fields in the last row to be printed which are still open need to be classified.

Note that the maximum number of fields which can be open at one time is twice the number of pixel groups in a row. Thus the vectors S and Q need have only 2W elements since each time a field is closed, those elements of S and Q which contained the information concerning the field just closed can be released for use by a new field.
A source listing for the program modified to work on the IBM System 360 44PS operating system is included in Appendix I. A discussion of the input data and the program is supplied in Appendix II.

V. Evaluation of the CBA

In order to analyze the performance of the closed boundary finding algorithm the program was applied to two remote sensor data sets. The primary application was to aircraft multispectral scanner data from flights over agricultural areas in Indiana. Regions such as this contain a uniform checkerboard pattern of rectangular fields oriented generally in a north-south direction. Imagery gathered over this area using north-south flight paths will contain a field boundary structure which is colinear with the X-Y axes of the imagery. These conditions make data of this type ideal for testing boundary finding algorithms due to the predictability of the scene border structure. A 1 mile by 8 mile strip of this agricultural scanner data was used to test the CBA using a variety of parameter values for the algorithm.

The CBA was also applied to a second data set to observe the behavior of the algorithm on irregular and randomly shaped features in the imagery. The data set is from a flight over Yellowstone National Park, Wyoming, and contains meadow, forest, rock outcrops and other irregular features.

These test cases are presented as examples to show the typical performance to be expected from the algorithm for these image context categories. The evaluation method is strictly subjective since no reference standard or qualitative evaluation method exists for testing
the boundary results. Field by field visual analysis enables a reasonable judgement to be made as to the quality and consistency of the closed fields the algorithm is finding. Thus, gray scale pictorial printouts and the corresponding closed boundary finding algorithm printouts having the same scale are presented in figure form for documentation and evaluation purposes in this report.

A typical CBA result is presented in Figure 1. A gray scale pictorial computer printout is presented in Figure 1a for a strip of data in the .58 - .65 micrometer band of typical Indiana farm land. This image was generated using a set of 10 print chain characters which reproduce an approximation of a gray scale. The CBA output in Figure 1b was generated using the following parameter values: Minimum points per field = 128; Significance level = 4; Channels = 4,6,8; Pixel group size = 2 by 2. It is difficult to visually evaluate the results on a large scale. Individual fields must be selected in the gray scale printout and the corresponding area located in the CBA output. Inspection of many fields in this manner reveals that the basic border structure of the scene is being captured and in general each field in the gray scale tends to be broken up into two or more subfields. A fixed sequence of symbols is used to represent the closed boundaries and the sequence is repeated as the set of available symbols is exhausted. It is assumed to be desireable to have too many fields defined rather than too few since then there will be less chance of a CBA field covering more than one real field. Also it is entirely reasonable to expect that real fields will contain variabilities which translate to separate sub-fields when observed spectrally.
In order to evaluate the sensitivity of the CBA to variations in the program parameters several figures were generated from CBA output for visual comparison. The minimum number of points allowable for a field is moderately influential in determining the nature of the border output. In Figure 2 three values for MINFLD are illustrated. The difference between MINFLD of 32 and 64 is not great; however, sharper and more linear borders are obtained for MINFLD = 128. The minimum field of 128 picture elements corresponds to 2.4 acres for the scale of this data. Larger minimums may be desirable if the average size of the sub-fields in the scene is larger than this figure. Next, the significance level was varied and Figure 3 contains output for SIGLEV of 2, 3 and 4. It is apparent that this parameter has the strongest effect on the CBA output. For values of 2 and 3 a great deal of border structure is lost and it is assumed that for this scene and three channel operation significance level 4 is the more desirable value to use.

The CBA was originally written to use one channel only to derive borders. The program was expanded to utilize multiple channels and the three channels used (4, 6, 8) tended to give the best results. These channels cover the .52 - .57 (green), .58 - .65 (red) and .72 - .92 (infrared) portions of the spectrum, respectively. These channels tend to be less cross correlated than other combinations of three thus contain more picture information than would channels demonstrating more correlation. Use of more than three channels tended to produce an excess of borders and the output was judged to be degraded. The comparison presented in Figure 4 is for a one-channel versus a three-channel run. It is apparent that a certain amount
of the horizontal border is lost while the vertical border structure appears to be captured relatively well.

These results tended to indicate that it is desireable to use multiple channels for border finding. This approach requires additional computation and the problem of selecting which channels to use is compounded. These considerations led to the decision to test the usefulness of the principal components transformations [6] in the border finding process. This transformation is linear and its effect is to concentrate the variance in the multiple image set in a minimum number of channels. The first principal component is an image channel formed as a linear combination of all original channels such that in this case approximately 75% of the variance in the image set is contained in it. The second principal component contains 12% of the variance, the third 7% and the rest the remaining variance on a decreasing scale. The possible advantages to using the principal components are that fewer channels may suffice and it eliminates the uncertainty of which channels to use.

The principal component transform was performed on the data set which was used to produce the previous output. The CBA was run on the transformed data using the first principal component in one case and the first three in a second case. The results of the runs are displayed in Figure 5 along with the three channel results from Figure 1. Again it is evident that the single channel result tends to miss horizontal boundaries as was observed for the single channel unaltered data. The three component results tend to agree with the
reference case. From this it was concluded that the principal component approach would simplify channel selection but more than one channel would still be required for processing.

The CBA was applied to one other data set obtained from an aircraft scanner flight over Yellowstone National Park, Wyoming, in 1967. A gray scale printout of the .52 - .55 micrometer band of the scanner data is presented in Figure 6. The same parameter values used for previous runs were used for the CBA and the output is pictured on the right in Figure 6. The elongated areas of meadow and rocky material are bordered reasonably well in the center and lower part of the area. The upper area contains forest stands and it is extremely difficult to evaluate what the CBA is doing here. Detailed evaluation of the performance for this scene is beyond the scope of this report and this evidence is presented as an example only.

VI Summary

A closed boundary finding algorithm is described which attempts to extract border information from digitized multispectral imagery. The algorithm was applied to multispectral aircraft scanner data from agricultural and wildland scenes to evaluate its performance. Visual evaluation of results indicate reasonably good performance in the agricultural scene; however, precise evaluation was not possible within the scope of this study. Subjective evaluation of performance for the wildlands case was more difficult than for the agricultural case.
Further work is needed to evaluate and optimize the algorithm. Once satisfactory performance is achieved the output of the CBA must be transferred to a field classification algorithm to achieve the initial intended purpose of the CBA. Classification of the fields could be performed as soon as the fields are closed (completed) or a field output definition could be made and classification performed later. One way this could be accomplished is for the CBA to write a border tape with the fields written in a format easily readable by a per field classifier program.

The processing rate for the present program on an IBM System 360 Model 67 is approximately .25 sec of CPU time per image line of 222 samples for 1 channel processing and .44 sec per line for three channel processing. The flightlines illustrated averaged 770 lines thus the three channel CBA processing time averaged 339 seconds CPU time. Processing time is not of major concern at this stage since quantitative evaluation of the algorithm is not yet achieved.

The close cooperation of the IBM Houston Scientific Center and the LARS Purdue organizations was instrumental in achieving the results obtained from this joint study. It is hoped that this document will serve as stimulus to further work on the problems discussed in this report and in the interim report.
References


6. Ready, P. J. Wintz, P. A., Multispectral Data Compression Through Transform Coding and Block Quantization, "PhD Thesis, School of Electrical Engineering, Purdue University, TR-EE 72-2; and LARS Information Note 050572, May 1972.
Figure 1. (a) Gray scale line printer image of .58 - .65 micrometer band aircraft scanner data. (b) Closed boundary algorithm output for the scanner data. Parameters: MINFLD = 128, SIGLEV = 4, CHANNELS 4, 6, 8. LARS Run No. 71062701.
Figure 2. Comparison of the effect of three values of minimum field size on CBA output. Significance Level = 4, Channels 4, 6, 8.
Figure 3. Comparison of CBA output for three values of significance level. Channels 4, 6, 8 and MINFLD = 128.
Figure 4. Comparison of CBA output for two sets of channels. Significance level 4 and MINFLD = 128 used.
Figure 5. Comparison of CBA output from unaltered data and principal components data. MINFLD = 128, Significance Level 4.
Figure 6. Gray scale printout and CBA output for forest and meadow topography in Yellowstone Park, Wyoming. Parameters: MINFLD = 128, Channels 4, 6, 8; SIGLEV = 4.
Appendix I

Fortran Listing of Closed Boundary Finding Algorithm
Appendix II

Input - Output and Program Variable Information

The CBA program listed in Appendix I operates under the IBM System 360 Model 44 programming system 44PS. The first section of this appendix describes the necessary control information, the second describes the format of the picture data used by the program and the third section describes the program output. The fourth section describes the program variables.

I.

All of the control parameters are read from cards by the subroutine GETPRM. Below is a list of (1) the format of the cards, (2) the program variable into which the data is read, and (3) an explanation of the parameter.

1. 'RUN IDENTIFIER' Format is 4A4. AFS

   This defines an alphabetic identifier for the run. This identifier will be printed at the head of the output.

2. 'GROUP SIZE' Format is II. PXGR

   This requests the value of g, the number of pixels on the side of a pixel group. Thus a pixel group has g**2 pixels.

3. 'ROWS & COLS 414' Format is 4I4. RI R2 W1 W2

   This requests which rows and columns of the picture are to be processed. Thus it is not necessary to process starting at the top of the picture or at the left of the picture. The data is of the form: first row, last row, first column, last column. If the specified rows and columns are such that they do not come out on a pixel group boundary, the program will delete sufficient rows
and/or columns to come out even. The maximum number of columns which can be processed is 125/(the group size parameter). There is no restriction on the number of rows.

4. 'SIGLEV' Format is I1. SIGLEV
   This is the significance level to be used in the t-test.
   The values of '1', '2', '3', or '4'. A value of '1' will cause the null hypothesis to be rejected most readily and will thus give the least resolution (and cleanest results).

5. 'MIN SIZE' Format is I2. MFS
   This requests the number of pixel groups a field must have in order to engage the per field classifier. If you wish to have per field classification performed on all fields, no matter how small, simply punch '01'. Currently this parameter is used only in the dummy per field classifier.

6. 'NUMBER OF CHANNELS' Format is I1. NCHAN
   This requests the number of spectral bands to be used in the picture analysis.

7. 'OUTPUT' Format is I1. OUT
   This requests the FORTRAN logical data set number to which results are to be written.

8. 'RUN NUMBER' Format I8.
   This is the run number of the data set to be processed.

9. 'CHANNELS' Format NI2
   Defines the NCHAN channels to be used.
II.

All use of raw digital picture data is in subroutine PGROW. Thus, to use a different format of picture data, only this subroutine must be altered. Subroutine PGROW is called to read in a row of pixel groups. All that PGROW returns are the values of the sums of the pixels in each group (in each channel) and the sums of the squares of the pixels in each group (in each channel). The actual pixel values are not retained. The sum of the elements of the JS'th pixel group in the row for the NS'th channel is placed in ASO(JW,NS). PGROW has available in the named common /INT2/ the number of channel being processed (in NCHAN), the columns of pixels being considered (in W1 and W2) and the size of a side of a pixel group (in PXGR).

III.

The output consists of a heading giving the title, minimum field size, pixel group size, significance level and number of channels processed; a printout of column numbers; and the body of results. In the body of results, the row number is printed at the left. This is the pixel row, not the pixel group row so that if each pixel group is 2 pixels on a side, these numbers on the left will be 1, 3, 5 etc. The rest of the body of results consists of field identification characters. Only the boundaries of each field are printed. Field identification is assigned by a dummy per field classifier in subroutine CLASS. As each field is
closed, this dummy per field classifier gives a new identifier to it. After all the available characters have been used, it starts over at the beginning of the list of characters.

IV.

Meaning of FORTRAN Variables in the Closed Field Selection Program

This section gives the meaning of the important FORTRAN variables in the Closed Field Selection Program. All the variables discussed below except ASP are in a COMMON block. The program uses COMMON to pass virtually all information from subroutine to subroutine.

Control information

OUT contains the FORTRAN logical unit number to which results are written.

IN contains the FORTRAN logical unit number from which control information is read.

NCHAN contains the number of channels being processed.

PXGR is the value g in the paper. PXGR1 is PXGR-1.

R1 and R2 are the first and last rows of the picture to be processed.

W1 and W2 are the first and last columns of the picture to be processed.

MFS is the minimum field size. This is used only in the dummy per field classifier.
Field statistics

Each vector of ASUM is the vector $s$ in the report for one channel. Each vector of ASQ is the vector $q$ in the report for one channel. Each vector of AFTOT is the vector $S$ in the report for one channel. Each vector of AFS in the vector $Q$ is the report for one channel.

AFLD(1) contains the number of pixel groups in the field whose statistics are contained in the 1'th row of AFTOT and AFS.

AT(1) is computed only in the statistical subroutine (subroutine STU) and contains the value of $t$ for the 1'th channel.

ASP is a local variable in subroutine STU and is the value of Sp**2.

Utilization of field statistics variables

STACK is a stack vector. It is initialized so that STACK(1) = 1. NEXT says which element of the stack to be used next. Each time a field is started, a number is pulled from the stack and this number is the number of the element of AFLD and row of AFTOT and AFS to be used for this field. Also when a new field is started, NEXT is increased by one. Each time a field is closed, a number is returned to the stack. Thus, if a field is closed which was described by the 10'th element of AFLD and the 10'th row of AFTOT and AFS, the number 10 is put into the stack and the value of NEXT is increased by one.

Indication of current processing

JR contains the number of row currently being processed. JR is pixel row, not pixel group row. JR is the DO index on the overall loop over rows. If PXGR is 2, then JR is incremented by two each
time through the loop.

JW contains the number of the column currently being processed. JW is the DO index on the loop passing through the row from left to right.

RWFLD(1) contains the element number of AFLD containing information concerning the 1' th pixel group in the current row. RWFLD is actually the vector f in the report.

OLDRW is the RWFLD from the previous row. It is actually the vector (f) in the report.

Printing of results

SP is the array of results. SP is dimensioned (130,100) so that each vector of SP is a line of printed result. After a row is processed, SP is loaded from the vector RWFLD. When a field is closed, and the field is classified, all entries in SP referring to that field are altered to reflect the classification. To tell whether a field has been classified, the program puts the classification index +256 into SP for each element of SP representing the field just classified. After 100 lines have been processed, 50 lines of results are printed and then after another 50 lines are processed lines 51-100 are printed and so forth. Anytime the print routine is called, results are printed from the first 50 vectors of SP. Then the second fifty vectors of SP are moved to become the first 50. As processing continues, further results are placed in vectors 51-100.

LASTR is a vector of dimension 130. It is positioned in COMMON immediately before SP and is thus logically the zero' th vector of SP. This is done to preserve the last row of results just printed.
This information is needed by the print routine to know whether a boundary was crossed.

FSR(1) contains the number of the first vector of SP which represents the field described by the l'th element of AFLD. FSR is used after per field classification is done to assist in propagating the field classification index (+256) throughout that part of SP representing the field.

JNR contains the total number of rows of pixel groups processed so far. Each time JNR becomes a multiple of 50 (and is at least 100), 50 lines of results are printed.

JLINE is the counter used to index SP.