~~040172~~
043072

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. Anuta[1], E. M. Rodd[2], R. E. Jensen[2]

and P. R. Tobias[2]

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

---

[1]LARS (Laboratory for Applications of Remote Sensing) Purdue University

[2]IBM 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 algorthm is very time consuming and closed boundaries are not guaranteed. The boundary finding algorthm 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[5]. 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.

$$S_p^2 = \frac{\sum_i x_{i1} + \sum_i x_{i2}}{(n_1-1) + (n_2-1)}$$

Note that the $\sum_i x_{ij}^2$ is computed as usual by

$$\sum_i x_{ij} = \frac{n_j \sum_i x_{ij} - (\sum_i x_{ij})^2}{n_j}$$

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_p^2 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 σ_1 and σ_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_i 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 (1), 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 contantains 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 relaeased 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 influencial 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 desireable 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 usefullness 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.

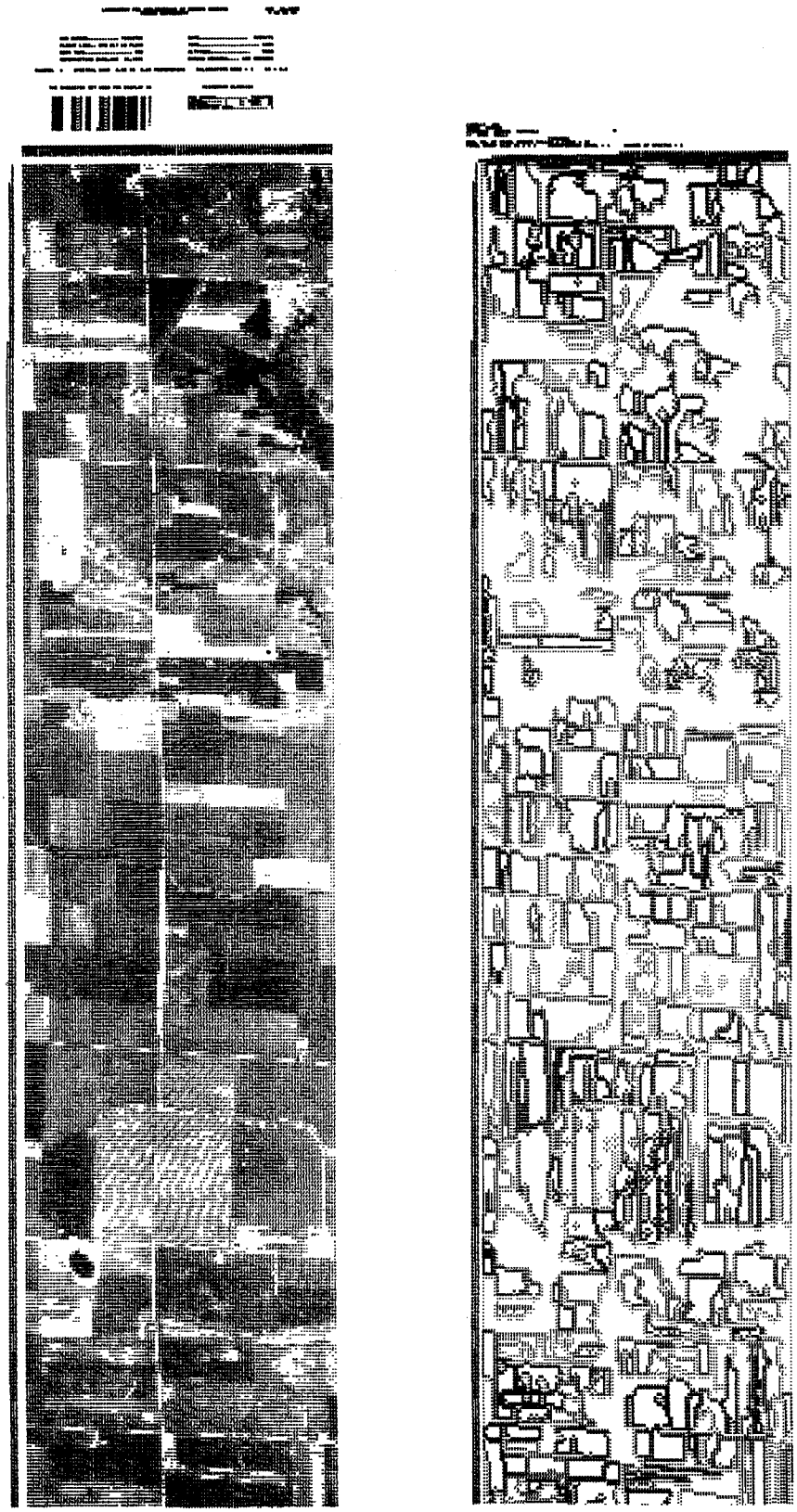
Further work is needed to evaluate and optimize the algorithm. Once satisfactory performance is achieved the output of the CBA must be transferred to a field classification algorithm to achieve the initial intended purpose of the CBA. Classification of the fields could be performed as soon as the fields 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.

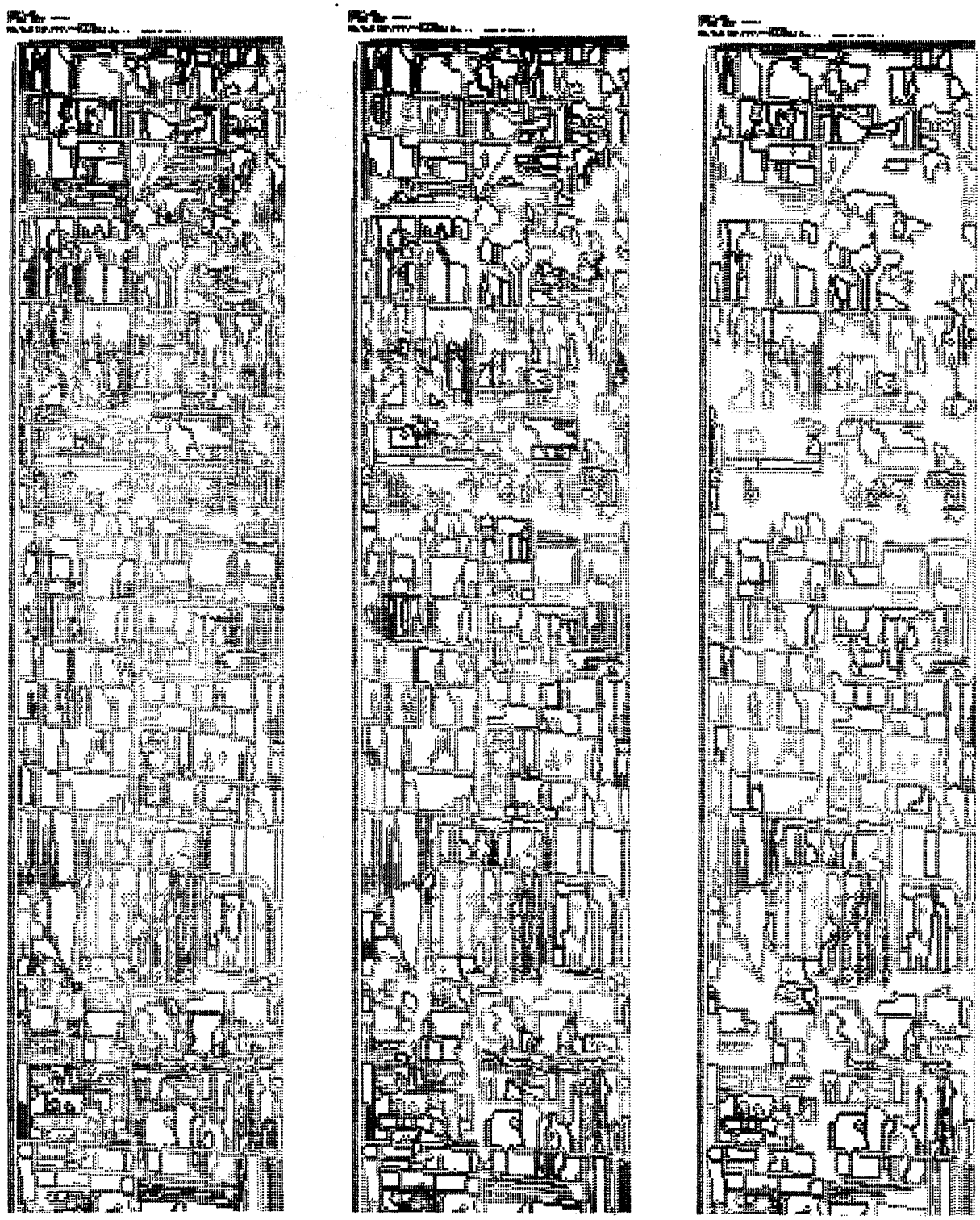
References

1. Rosenfeld, A., "Picture Processing by Computer", Academic Press, New York, 1969.
2. Anuta, P. E., "Spatial Registration of Multispectral and Multitemporal Digital Imagery Using Fast Fourier Transform Techniques", IEEE Transactions on Geoscience Electronics, Vol. GE-8, No. 4, October, 1970.
3. Wacker, A. G., Landgrebe, D. A., "Boundaries in Multispectral Imagery by Clustering", Proceedings of the 9th IEEE Symposium on Adaptive Processes, University of Texas, Austin, December 7 - 9, 1970.
4. Rodd, E. M., "Closed Boundary Field Selection in Multispectral Digital Images", IBM Publication No. 320.2420, IBM Houston Scientific Center, January 14, 1972.
5. Ostle, B., "Statistics in Research", Iowa State University Press, Ames, Iowa, 1963.
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.



a) Gray Scale Printout b) CBA Output

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.

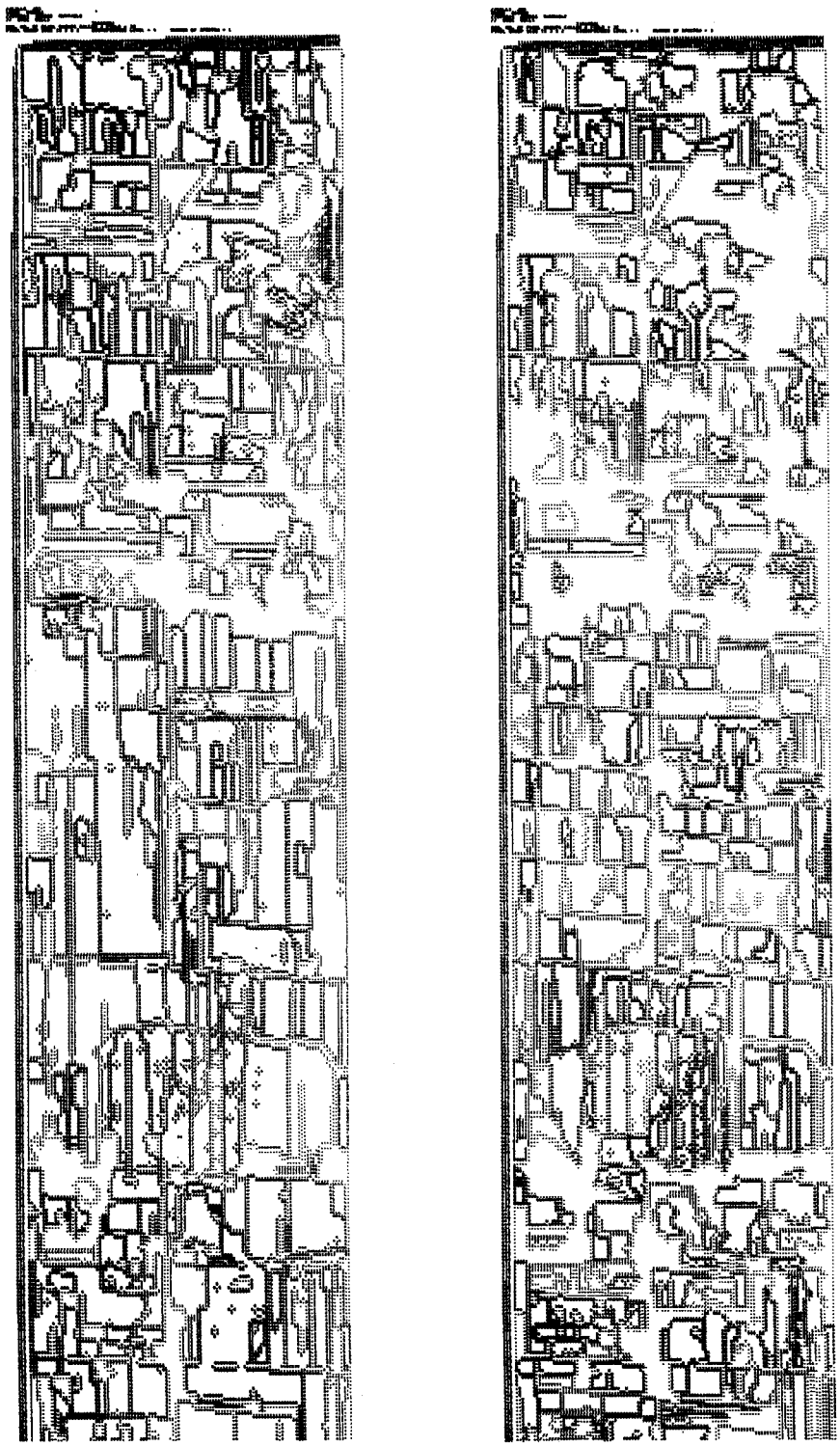


MINFLD = 32 MINFLD = 64 MINFLD = 128

Figure 2. Comparison of the effect of three values of minimum field size on CBA output. Significance Level = 4, Channels 4, 6, 8.

SIGLEV = 2 SIGLEV = 3 SIGLEV = 4

Figure 3. Comparison of CBA output for three values of significance level. Channels 4, 6, 8 and MINFLD = 128.



Channel 6 Channels 4,6,8

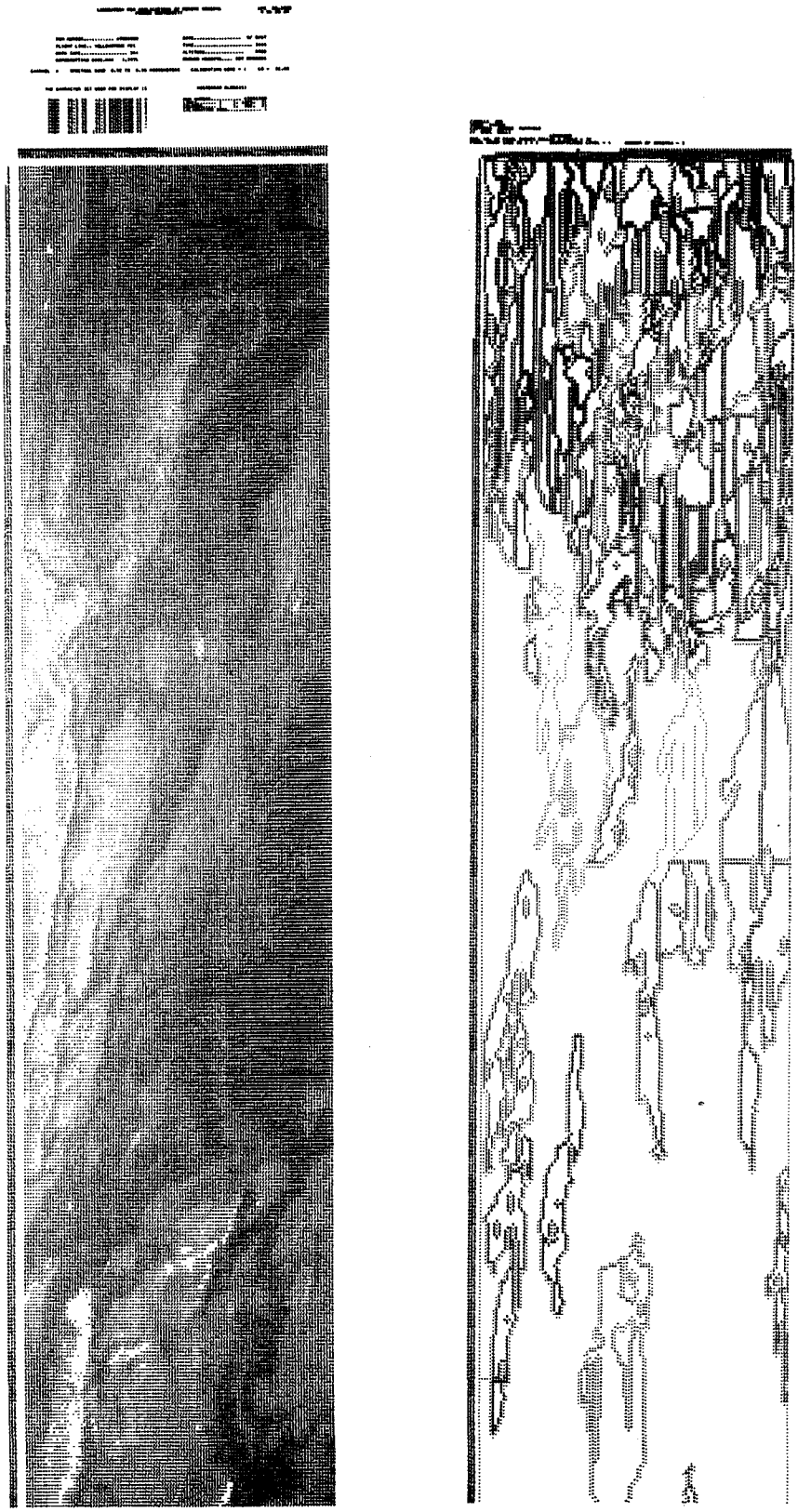
Figure 4. Comparison of CBA output for two sets of channels. Significance level 4 and MINFLD = 128 used.

3 Channels Untransformed

Principal Component 1

Principal Components 1,2,3

Figure 5. Comparison of CBA output from unaltered data and principal components data. MINFLD = 128, Significance Level 4.



Gray Scale Printout CBA Output

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

```
FORTRAN IV     MODEL 44  PS      VERSION 3,  LEVEL 4   DATE  72215

                C      MAIN ROUTINE FOR FIELD SELECTION PROGRAM.
                C      MAIN CONTROLS THE  GEOMETRY OF FIELD BUILDING.
 0001                  IMPLICIT  INTEGER*2  (E-Z)
 0002                  INTEGER*4  OUT, IN, JNR, NS
 0003                  LOGICAL*1  SAME, ASIN, ENOUGH, CLOSE, OPEN
 0004                  COMMON  AFTOT(250,4), AFS(250,4), ASUM(256,4), ASQ(256,4), AT(4),
                      1    AFLD(250), APGR1, AST, APGR, OUT, IN, JNR, NS
 0005                  COMMON /INT2/ PIX(1280), RWFLD(256), OLDRW(256), STACK(250),
                      1    FSR(250), LASTR(130), SP(130,100), PXGR, PXGR1, NCHAN ,
                      2    ROWS, W1, W2, WL2, WIDE, MFS, JW, JS, JR, RL1, IFLD, SIGLEV,
                      3    IDF, R1, R2, IR1, IR2, ICLOSE, PCLASS, PWIDE, NEXT, JLINE
 0006                  COMMON /RESULT/ SAME, ASIN(256), ENOUGH(250), CLOSE(250),OPEN(250)
 0007                  INTEGER*4  ITIMER
 0008                  INTEGER*4 ID, RUNNUM
 0009                  COMMON /LARS/ ID(200), RUNNUM, RCSEL(30),
                      1    COV(256,6), AFCOV(250,6)
 0010                  EXTERNAL ITIMER
 0011                1 J = ITIMER(1)
                C      THE CALL TO ITIMER SETS  THE CLOCK.
 0012                  CALL GETPRM
 0013                  IFLD = 1
 0014                  DO 37 I = 1,250
 0015                  DO 30 J = 1,4
 0016                  AFTOT(I,J) = 0.0
 0017               30 AFS(I,J) = 0.0
 0018                  DO 33 J = 1,6
 0019               33 AFCOV(I,J) = 0.0
 0020                  OPEN(I) = .FALSE.
 0021                  CLOSE(I) = .TRUE.
 0022               37 AFLD(I) = 0.0
 0023                  OLDRW(W1) = 1
 0024                  RWFLD(W1) = 1
 0025                  FSR(1) = 1
 0026                  JR = R1+PXGR
                C
                C      THIS CALL READS OVER ROWS OF THE PICTURE PRECEEDING THE FIRST ROW
                C      TO BE ANALYZED.
 0027                    CALL RDOVER
                C
 0028               39 WL2 = W2-PXGR1
 0029                  CALL PRINTO
                CV     INITIALIZATION IS NOW COMPLETE.
                C
                C      STARTING ON FIRST ROW.
 0030                  CALL PGROW
                C      DO FIRST ROW OF PIXEL GROUPS.
                C
 0031                  WL3 = WL2-PXGR
 0032                  JW = W1
                C      SUBROUTINE SET ASSUMES THAT THE COLUMN OF THE NEW FIELD IS IN
                C      JW.
 0033                  CALL SET
 0034                  OLDRW(W1) = 1
 0035                  DO 100 JW = W1,WL3,PXGR
 0036                  JS = JW+PXGR
 0037                  CALL STU
                C      CALL TO STU HERE ACTUALLY COMPARES TO A GROUP BESIDE. THIS
                C      OCCURRS BECAUSE IN THE FIRST ROW OLDRW IS JUSED TO STORE THF
                C      CURRENT ROW.
 0038                  IF (SAME) GO TO 45
                C      PIXEL GROUP TO RIGHT IS DIFFERENT SO START A NEW FIELD.
 0039                  IFLD = IFLD+1
 0040                  NEXT = NEXT +1
 0041                  FSR(IFLD) = 1
 0042               45 CALL ADD(IFLD,JS)
                C      ONE MORE PIXEL GROUP IN FIELD IFLD.
 0043                  RWFLD(JS) = IFLD
                C      BOTH RWFLD AND OLDRW ARE LOADED BECAUSE PRINT WILL USE RWFLD
                C      AND OLDRW IS NEEDED WHEN THE NEXT ROW IS STARTED.
 0044              100 OLDRW(JS) = IFLD
 0045                  CALL PRINT
                C      PLACE RESULTS OF FIRST ROW IN PRINT ARRAY VIA CALL TO PRINT.
                C
 0046                  IR1 = R1+PXGR
 0047                  IR2 = R2-PXGR1
 0048                  JNR = 1
                C      INITIALIZATION OF DO 250 LOOP WHICH WILL PROCESS ALL
                C      REMAINING ROWS.
```

```
FORTRAN IV     MODEL 44  PS      VERSION 3,  LEVEL 4   DATE  72215

 0049                  DO 250 JR = IR1,IR2,PXGR
 0050                  JNR = JNR +1
 0051                  CALL PGROW
                C      READ IN NEXT ROW OF PIXEL GROUPS.
 0052                  DO 160 JW = W1,WL2,PXGR
 0053                  ASIN(JW) = .FALSE.
                C      INITIALIZATION OF PIXEL GROUPS AS ASSIGNED.
 0054                  JS = JW
 0055                  CALL STU
                C      CHEKC AGAINST GROUP ABOVE.
 0056                  IF (.NOT. SAME) GO TO 130
                C
                C      PUT GROUP INTO SAME FIELD AS GROUP ABOVE.
 0057                  ASIN(JW) = .TRUE.
 0058                  NFLD = OLDRW(JW)
 0059                  RWFLD(JW) = NFLD
 0060                  CALL ADD(NFLD,JW)
 0061                  IF (JW .EQ. W1) GO TO 160
                C
                C      CHECK BACK ACROSS FOR UNASSIGNED GROUP.
                C      IF ONE GROUP BACK IS ALREADY ASSIGNED, GO TO NEXT GROUP.
 0062                  JS = JW-PXGR
 0063              120 IF (ASIN(JS)) GO TO 160
 0064                  CALL STR
                C      COMPARE AGAINST ONE GROUP BACK. IF IT IS DIFFERENT,
                C      TRANSFER TO PROCESSING OF NEXT GROUP.
 0065                  IF (.NOT. SAME) GO TO 160
 0066                  ASIN(JS) = .TRUE.
 0067                  NFLD = RWFLD(JW)
 0068                  RWFLD(JS) = NFLD
 0069                  CALL ADD(NFLD,JS)
 0070                  JS = JS -PXGR
                C
                C      CHECK TO SEE IF BACK AT BEGINNING OF ROW.
 0071                  IF (JS .GE. W1) GO TO 120
 0072                  GO TO 160
 0073              130 IF (JW .EQ. W1) GO TO 160
                C      SINCE THIS GROUP DIFFERS FROM THE ONE ABOVE, ASK IF THE ONE
                C      TO ITS LEFT IS ASSIGNED AND IF SO COMPARE TO IT.
                C
 0074                  IF (ASIN(JW-PXGR)) GO TO 135
 0075                  IF (JW .EQ. WL2) GO TO 140
                C      THE ABOVE IF PROVIDES FOR SPECIAL TREATMENT AT THE END OF A ROW.
 0076                  GO TO 160
 0077              135 JS = JW
 0078                  CALL STR1
 0079                  IF (SAME) GO TO 137
 0080                  IF (JW .EQ. WL2) GO TO 140
                C      THE IF ASKS IF END OF ROW.
 0081                  GO TO 160
 0082              137 ASIN(JW) = .TRUE.
 0083                  NFLD = RWFLD(JW-PXGR)
 0084                  RWFLD(JW) = NFLD
 0085                  CALL ADD(NFLD,JW)
 0086                  GO TO 160
 0087              140 CALL SET
 0088                  FSR(IFLD) = JLINE+1
                C      SPECIAL CODE FOR NEW FIELD AT END OF ROW.
 0089                  ASIN(JW) = .TRUE.
 0090                  RWFLD(JW) = IFLD
 0091                  IF (ASIN(JW-PXGR)) GO TO 160
                C      IF ONE BACK IS ASSIGNED NO NEED TO COMPARE IT.
                C
 0092                  JS = JW-PXGR
 0093              150 CALL STR
 0094                  IF (.NOT. SAME) GO TO 160
                C      LOOP TO CHECK BACK ACROSS.
 0095                  ASIN(JS) = .TRUE.
 0096                  RWFLD(JS) = IFLD
 0097                  CALL ADD(IFLD,JS)
 0098                  JS = JS-PXGR
 0099                  IF (ASIN(JS)) GO TO 160
 0100                  IF (JS .GE. W1) GO TO 150
 0101              160 CONTINUE
                C
                C      FOLLOWING IS GOING BACK THROUGH ROW TO PICK UP ANY GROUPS
                C      NOT YET ASSIGNED AND CALLING THEM NEW FIELDS.
 0102                  JW = WL3
```

```
FORTRAN IV     MODEL 44  PS      VERSION 3,  LEVEL 4   DATE  72215

 0103              170 IF (ASIN(JW)) GO TO 190
 0104                  ASIN(JW) = .TRUE.
 0105                  CALL SET
 0106                  FSR(IFLD) = JLINE+1
 0107                  RWFLD(JW) = IFLD
 0108              180 JW = JW-PXGR
 0109                  IF (JW .LT. W1) GO TO 200
                C      IF AT START OF ROW THEN ARE FINISHED.
                C
 0110                  IF (ASIN(JW)) GO TO 190
 0111                  JS = JW
 0112                  CALL STR
                C      CHECK AGAINST ONE BEHIND IF IT IS NOT ASSIGNED.
 0113                  IF (.NOT. SAME) GO TO 170
                C      IF A NEW FIELD GO BACK AND ASSIGN.
                C
 0114                  ASIN(JW) = .TRUE.
 0115                  RWFLD(JW) = IFLD
 0116                  CALL ADD(IFLD,JW)
                C      NOW GO CHECK NEXT GROUP BACK ACROSS.
 0117                  GO TO 180
 0118              190 JW = JW-PXGR
 0119                  IF (JW .GE. W1) GO TO 170
 0120              200 CALL PRINT
 0121                  DO 220 I = W1,WL2,PXGR
 0122                  OPEN(OLDRW(I)) = .TRUE.
 0123                  CLOSE(RWFLD(I)) = .FALSE.
 0124              220 OLDRW(I) = RWFLD(I)
                C      RWFLD IS MOVED TO OLDRW FOR COMPARISONS FROM NEXT ROW.
                C      ALSO OPEN NOW INDICATES WHICH VECTORS OF AFLD ARE OPEN ON
                C      LAST ROW AND CLOSE INDICATES WHICH VECTORS ARE CLOSED
                C      ON CURRENT ROW.
                C
 0125                  DO 230 ICLOSE = 1,250
 0126                  IF (OPEN(ICLOSE) .AND. CLOSE(ICLOSE)) CALL CLSTK
 0127                  OPEN(ICLOSE) = .FALSE.
 0128              230 CLOSE(ICLOSE) = .TRUE.
                C      FOR ALL FIELDS JUST CLOSED, CLASSIFICATION IS PERFORMED
                C      AND THE VECTOR OF AFLD RETURNED TO THE STACK.
                C     THEN OPEN AND CLOSE ARE INITIALIZED FOR THE NEXT ROW.
                C
                C      PRINT1 IS CALLED TO PRINT RESULTS OF 50 ROWS.
                C      PRINT1 IS CALLED AFTER THE FRIST 100 ROWS HAVE BEEN PROCESSED
                C      TO PRINT THE FIRST FIFTY ROWS AND THEN AFTER EACH FIFTY ROWS
                C      HAVE BEEN PROCESSED TO PRINT AN ADDITIONAL 50 ROWS.
 0129                  IF (JNR .GE. 100 .AND. MOD(JNR,50) .EQ. 0) CALL PRINT1
 0130              250 CONTINUE
 0131                  JR = IR2+1
 0132                  CALL PRINT1
                C      THIS CALL TO PRINT1 WILL PRINT RESULTS OF ALL REMAINING
                C      ROWS.
 0133                  ATIME = FLOAT(ITIMER(OUT))/100.0
                C      ATIME WILL CONTAIN TIME (VIRTUAL CPU TIME) SINCE THE
                C      CALL TO ITIMER IN STATEMENT 1.
 0134                  WRITE (OUT,267) ATIME
 0135              267 FORMAT (' TIME USED',F8.2,' SECONDS')
 0136                  STOP
 0137                  END
```

```
FORTRAN IV     MODEL 44  PS      VERSION 3,  LEVEL 4   DATE  72215

 0001                  SUBROUTINE PGROW
                C      READS IN A NEW ROW OF PIXEL GROUPS AND COMPUTES THE SUMS AND SUMS
                C      OF SQUARES OF VALUES OF EACH PIXEL GROUP.
 0002                  IMPLICIT  INTEGER*2  (E-Z)
 0003                  INTEGER*4  OUT, IN, JNR, NS
 0004                  LOGICAL*1  SAME, ASIN, ENOUGH, CLOSE, OPEN
 0005                  COMMON  AFTOT(250,4), AFS(250,4), ASUM(256,4), ASQ(256,4), AT(4),
                      1    AFLD(250), APGR1, AST, APGR, OUT, IN, JNR, NS
 0006                  COMMON /INT2/ PIX(1280), RWFLD(256), OLDRW(256), STACK(250),
                      1    FSR(250), LASTR(130), SP(130,100), PXGR, PXGR1, NCHAN ,
                      2    ROWS, W1, W2, WL2, WIDE, MFS, JW, JS, JR, RL1, IFLD, SIGLEV,
                      3    IDF, R1, R2, IR1, IR2, ICLOSE, PCLASS, PWIDE, NEXT
 0007                  COMMON /RESULT/ SAME, ASIN(256), ENOUGH(250), CLOSE(250),OPEN(250)
 0008                  LOGICAL*1 DATA(1280,4), DUML
 0009                  LOGICAL*1 IDATA(2740)
 0010                  INTEGER*4 ID,LNID,ERRRV,RUNNUM
 0011                  COMMON /LARS/ ID(200),RUNNUM,RCSEL(30),COV(256,6),
                      1    AFCOV(250,6)
 0012                  DIMENSION PIXEL(768,4)
 0013                  L1 = 1
 0014                  L2 = WIDE
 0015                  WO = W1-1
 0016                  DO 30 I = 1,PXGR
 0017                  CALL TOPRV(I1,ID(6),ERRRV,IDATA,2,ID(5),RCSEL,LNID)
 0018                  IF (ERRRV .EQ. 1) STOP 10
 0019                  IF (ERRRV .NE. 0) WRITE (15,15) ERRRV
 0020               15 FORMAT (' ERROR ',I8)
 0021                  DO 20 NS =1,NCHAN
 0022                  K = (NS-1)*ID(6)
 0023                  KK = L1
 0024                  DO 20 J = W1,W2
 0025                  DATA(KK,NS) =IDATA(K+J)
 0026               20 KK = KK+1
 0027                  L1 = L2+1
 0028               30 L2 = L2+WIDE
 0029                  WORDS =WIDE*PXGR
 0030                  DO 100 NS = 1,NCHAN
 0031                  CALL SHIFT (PIXEL(1,NS),DATA(1,NS),WORDS)
 0032                  L = -PXGR
 0033                  DO 50 JW = W1,WL2,PXGR
 0034                  L = L+PXGR
 0035                  ASUM(JW,NS) = 0.0
 0036                  ASQ(JW,NS) = 0.0
 0037                  KK = L
 0038                  DO 50 I = 1,PXGR
 0039                  DO 40 J = 1,PXGR
 0040                  ASUM(JW,NS) = ASUM(JW,NS)+PIXEL(KK+J,NS)
 0041               40 ASQ(JW,NS) = ASQ(JW,NS) + PIXEL(KK+J,NS)**2
 0042               50 KK = KK+WIDE
 0043              100 CONTINUE
 0044                  NS1 = NSPTRA-1
 0045                  L1 = 0
 0046                  DO 110 NS = 1,NS1
 0047                  L2 = NS+1
 0048                  DO 110 NSI = L2,NCHAN
 0049                  L1 = L1+1
 0050                  L = -PXGR
 0051                  DO 110 JW = W1,WL2,PXGR
 0052                  L = L+PXGR
 0053                  COV(JW,L1) = 0.0
 0054                  KK = L
 0055                  DO 110 I = 1,PXGR
 0056                  DO 105 J = 1,PXGR
 0057              105 COV(JW,L1) = COV(JW,L1) + PIXEL(KK+J,NS)*PIXEL(KK+J,NSI)
 0058              110 KK = KK+WIDE
                C      THE DO 110 LOOPS COMPUTE THE COVARIANCE MATRIX FOR EACH PIXEL GROUP
                C      ON THIS ROW
 0059                  RETURN
 0060                  ENTRY RDOVER
                C       ENTRY RDOVER READS OVER ROWS OF THE PICTURE WHICH ARE NOT
                C      TO BE PROCESSED.
 0061                  IF (R1 .EQ. 1) RETURN
                C      IF R1 IS 1, THEN NO ROWS NEED TO BE READ OVER.
 0062                  I1 = R1-1
 0063                  DO 120 I = 1,I1
 0064              120 CALL TOPRV(I1,ID(6),ERRRV,IDATA,2,ID(5),RCSEL,LNID)
 0065                  RETURN
 0066                  END
```

```
FORTRAN IV    MODEL 44  PS       VERSION 3,  LEVEL 4   DATE  72215

0001            SUBROUTINE ADD (INTO,FROM)                                      ADD00010
                C     ADD ADDS A NEW PIXEL GROUP INTO AFTOT AND AFS.            ADD00020
                C     INTO IS THE ROW OF AFTOT AND AFS CONTAINING               ADD00030
                C     THE FIELD TO BE ADDED INTO. FROM CONTAINS THE             ADD00040
                C     ELEMENT OF ASUM AND ASQ CONTAINING THE CURRENT PIXEL      ADD00050
                C     GROUP WHICH IS BEING ADDED.                               ADD00060
0002            IMPLICIT  INTEGER*2  (E-Z)                                      ADD00070
0003            INTEGER*4  OUT, IN, JNR, NS                                     ADD00080
0004            LOGICAL*1  SAME, ASIN, ENOUGH, CLOSE, OPEN                      ADD00090
0005            COMMON  AFTOT(250,4), AFS(250,4), ASUM(256,4), ASQ(256,4), AT(4),ADD00100
                1     AFLD(250), APGR(), AST, APGR, OUT, IN, JNR, NS            ADD00110
0006            COMMON /INT2/ PIX(1280), RWFLD(256), OLDRW(256), STACK(250),    ADD00120
                1     FSR(250), LASTR(130), SPI(30,100), PXGR, PXGRI, NCHAN     ADD00130
                2     ROWS, W1, W2, WL2, WIDE, MFS, JW, JS, JR, RLI, IFLD, SIGLEV,ADD00140
                3     IDF, R1, R2, IR1, IR2, ICLOSE, PCLASS, PWIDE, NEXT        ADD00150
0007            COMMON /RESULT/ SAME, ASIN(256), ENOUGH(250), CLOSE(250),OPEN(250)ADD00160
0008            INTEGER*4  JD, RUNNUM                                           ADD00170
0009            COMMON /LARS/ ID(200), RUNNUM, RCSEL(30), COV(256,6),           ADD00180
                1     AFCOV(250,6)                                              ADD00190
0010            NC = (NSQTAA**2-NCHAN )/2                                       ADD00195
0011            AFLD(INTO) = AFLD(INTO) + 1.0                                   ADD00200
0012            DO 50 NS = 1,NCHAN                                              ADD00210
0013            AFTOT(INTO,NS) = AFTOT(INTO,NS) + ASUM(FROM,NS)                 ADD00220
0014         50 AFS(INTO,NS) = AFS(INTO,NS)+ASQ(FROM,NS)                        ADD00230
0015            DO 60 LI = 1,NC                                                 ADD00240
0016         60 AFCOV(INTO,LI) = AFCOV(INTO,LI) + COV(FROM,LI)                  ADD00300
                C     THE DO 60 LOOPS ADD THE COVEARIANCE MATRIX FOR            ADD00310
                C     THE GROUP INTO THAT OF THE FIELD                          ADD00320
0017            RETURN                                                          ADD00330
0018            ENTRY SEE                                                       ADD00340
                C     WHEN A NEW FIELD IS STARTED, A NUMBER IS PULLED FROM      ADD00350
                C     THE STACK OF AVAILABLE VECTORS OF AFTOT AND AFS           ADD00360
                C     (AND ELEMENT OF AFLD). THIS VECTOR IS USED FOR THE NEW FIELD.ADD00370
0019            IFLD = STACK(NEXT)                                              ADD00380
0020            NEXT = NEXT +1                                                  ADD00390
0021            NC = (NCHAN **2-NCHAN )/2                                       ADD00395
0022            AFLD(IFLD) = 1.0                                                ADD00400
0023            DO 100 NS = 1,NCHAN                                             ADD00410
0024            AFTOT(IFLD,NS) = ASUM(JW,NS)                                    ADD00420
0025        100 AFS(IFLD,NS) = ASQ(JW,NS)                                       ADD00430
0026            DO 110 LI = 1,NC                                                ADD00435
0027        110 AFCOV(IFLD,LI) = COV(JW,LI)                                     ADD00440
0028            RETURN                                                          ADD00450
0029            ENTRY CLSTK                                                     ADD00460
                C     ENTRY CLSTK CLOSES OUT A CLEMENT OF AFLD BY RETURNING     ADD00470
                C     IT TO STACK, CLASSIFYING THE FIELD AND ZEROING            ADD00480
                C     AFLD AND THE ROW OF AFTOT AND AFS USED FOR THE CLOSED FIELD.ADD00490
0030            NEXT = NEXT -1                                                  ADD00495
0031            STACK(NEXT) = ICLOSE                                            ADD00500
0032            CALL CLASS                                                      ADD00510
0033            DO 200 J = 1,4                                                  ADD00520
0034            AFTOT(ICLOSE,J) = 0.0                                           ADD00530
0035        200 AFS(ICLOSE,J) = 0.0                                             ADD00540
0036            DO 210 J = 1,6                                                  ADD00550
0037        210 AFCOV(ICLOSE,J) = 0.0                                           ADD00560
0038            AFLD(ICLOSE) = 0.0                                              ADD00640
0039            RETURN                                                          ADD00650
0040            END                                                             ADD00660
```

```
FORTRAN IV    MODEL 44  PS       VERSION 3,  LEVEL 4   DATE  72215

0001            FUNCTION ITIMER (J)                                             ITI00010
0002            ITIMER = 0                                                      ITI00020
0003            RETURN                                                          ITI00030
0004            END                                                             ITI00040
```

```
FORTRAN IV    MODEL 44  PS       VERSION 3,  LEVEL 4   DATE  72215

0001            SUBROUTINE CLASS                                                CLA00010
                C     CLASS CLASSIFIES A CLOSED FIELD. THE ELEMENT OF AFLD WITH THE CLA00020
                C     RELAVENT DATA IS ICLOSE. CLASS ALSO ALTERS THE SP PRINT OUT CLA00030
                C     ARRAY TO REFLECT THE CLASSIFICATION.                      CLA00040
                C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * CLA00050
                C     THE DUMMY PER FIELD CLASSIFIER BEGINS WITH THE FIRST EXECUTABLE CLA00060
                C     STATEMENT AND CONTINUES TO BUT NOT INCLUDING STATEMENT 150.  CLA00070
                C     THE CODE FROM STATEMENT 150 ON PROPAGATES THE CLASSIFICATION CLA00080
                C     INDEX THROUGHOUT THAT PART OF THE SP PRINTOUT ARRAY WHICH    CLA00090
                C     REPRESENTS THIS FIELD.                                       CLA00100
                C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * CLA00110
0002            IMPLICIT  INTEGER*2  (E-Z)                                      CLA00120
0003            INTEGER*4  OUT, IN, JNR, NS                                     CLA00130
0004            LOGICAL*1  SAME, ASIN, ENOUGH, CLOSE, OPEN                      CLA00140
0005            COMMON  AFTOT(250,4), AFS(250,4), ASUM(256,4), ASQ(256,4), AT(4),CLA00150
                1     AFLD(250), APGR(), AST, APGR, OUT, IN, JNR, NS            CLA00160
0006            COMMON /INT2/ PIX(1280), RWFLD(256), OLDRW(256), STACK(250),    CLA00170
                1     FSR(250), LASTR(130), SPI(30,100), PXGR, PXGRI, NCHAN     CLA00180
                2     ROWS, W1, W2, WL2, WIDE, MFS, JW, JS, JR, RLI, IFLD, SIGLEV,CLA00190
                3     IDF, R1, R2, IR1, IR2, ICLOSE, PCLASS, PWIDE, NEXT, JLINECLA00200
0007            COMMON /RESULT/ SAME, ASIN(256), ENOUGH(250), CLOSE(250),OPEN(250)CLA00210
0008            LOGICAL*1 NXTRW                                                 CLA00220
0009            DATA NCLASS /256/                                               CLA00230
0010            INTEGER*4 JD,RUNNUM                                             CLA00240
0011            COMMON /LARS/ ID(200), RUNNUM, RCSEL(30), COV(256,6),           CLA00250
                1     AFCOV(250,6)                                              CLA00260
                C                                                               CLA00270
                C                                                               CLA00280
0012            AMFS = MFS                                                      CLA00290
0013            IF (AFLD(ICLOSE) .GE. AMFS) GO TO 120                           CLA00300
                C     FIELDS WITH LESS THAN AMFS (AMFS IS THE MINIMUM FIELD SIZE CLA00310
                C     INPUT INTO SUBROUTINE GETPRM) PIXEL GROUPS ARE NOT CLASSIFIED.CLA00320
0014            PCLASS = 304                                                    CLA00330
0015            GO TO 150                                                       CLA00340
0016        120 NCLASS = NCLASS + 1                                             CLA00350
0017            IF (NCLASS .EQ. 304) NCLASS = 257                               CLA00360
0018            PCLASS = NCLASS                                                 CLA00370
0019        150 J = FSR(ICLOSE)                                                 CLA00380
0020        270 NXTRW = .FALSE.                                                 CLA00390
0021            DO 280 I = 1,PWIDE                                              CLA00400
0022            IF (SPI(I,J) .NE. ICLOSE) GO TO 280                             CLA00410
0023            SPI(I,J) = PCLASS                                               CLA00420
0024            NXTRW = .TRUE.                                                  CLA00430
0025        280 CONTINUE                                                        CLA00440
0026            J = J+1                                                         CLA00450
0027            IF (NXTRW .AND. J .LE. JLINE) GO TO 270                         CLA00460
0028        350 CONTINUE                                                        CLA00470
                C     THE DO 350 LOOP PROPAGUATES THE CLASSIFICATION INDEX      CLA00480
                C     PLUS 256 FOR THIS FIELD THROUGHOUT ALL ELEMENTS OF        CLA00490
                C     SP REPRESENTING THE FIELD.                                CLA00500
0029            RETURN                                                          CLA00510
0030            END                                                             CLA00520
```

```
FORTRAN IV    MODEL 44  PS       VERSION 3,  LEVEL 4   DATE  72215

0001            SUBROUTINE STU                                                  STU00010
                C     THE STATISTICAL SUBROUTINE FOR T-TEST. RETURNS SAME = TRUE IF STU00020
                C     TWO GROUPS, BEING COMPARED ARE THE SAME.                  STU00030
                C     ENTRY STU COMPARES A PIXEL GROUP WITH THE FIELD CONTAINING STU00040
                C     THE PIXEL GROUP IMMEDIATELY ABOVE  THE CURRENT PIXEL GROUP.. STU00050
                C                                                               STU00060
                C     J CONTAINS THE ELEMENT OF AFLD REPRESENTING SAMPLE 2 IN THE STU00070
                C     PAPER. JS IS THE INDEX OF THE CURRENT PIXEL GROUP. NOTE   STU00080
                C     THAT JS IS PASSED TO THIS SUBROUTINE VIA COMMON WHEREAS   STU00090
                C     J IS COMPUTED DIFFERENTLY ACCORDING TO WHICH              STU00100
                C     ENTRY POINT IS USED.                                      STU00110
                C                                                               STU00120
0002            IMPLICIT  INTEGER*2  (E-Z)                                      STU00130
0003            INTEGER*4  OUT, IN, JNR, NS                                     STU00140
0004            LOGICAL*1  SAME, ASIN, ENOUGH, CLOSE, OPEN                      STU00150
0005            COMMON  AFTOT(250,4), AFS(250,4), ASUM(256,4), ASQ(256,4), AT(4),STU00160
                1     AFLD(250), APGR(), AST, APGR, OUT, IN, JNR, NS            STU00170
0006            COMMON /INT2/ PIX(1280), RWFLD(256), OLDRW(256), STACK(250),    STU00180
                1     FSR(250), LASTR(130), SPI(130,100), PXGR, PXGRI, NCHAN    STU00190
                2     ROWS, W1, W2, WL2, WIDE, MFS, JW, JS, JR, RLI, IFLD, SIGLEV,STU00200
                3     IDF, R1, R2, IR1, IR2, ICLOSE, PCLASS, PWIDE, NEXT        STU00210
0007            COMMON /RESULT/ SAME, ASIN(256), ENOUGH(250), CLOSE(250),OPEN(250)STU00220
0008            DIMENSION ATSTAT(34,4)                                          STU00230
0009            DATA ATSTAT /12.706, 4.303, 3.182, 2.776, 2.571, 2.447, 2.365,  STU00240
                1     2.306, 2.262, 2.228, 2.201, 2.179, 2.160, 2.145, 2.131, 2.120,STU00250
                2     2.110, 2.101, 2.093, 2.086, 2.080, 2.074, 2.069, 2.064, 2.060,STU00260
                3     2.056, 2.052, 2.048, 2.045, 2.042, 2.021, 2.000, 1.980, 1.960,STU00270
                4     31.821, 6.965, 4.541, 3.747, 3.365, 3.143, 2.998, 2.896,  STU00280
                5     2.821, 2.764, 2.718, 2.681, 2.650, 2.624, 2.602, 2.583,  STU00290
                6     2.567, 2.552, 2.539, 2.528, 2.518, 2.508, 2.500, 2.492,  STU00300
                7     2.485, 2.479, 2.473, 2.467, 2.462, 2.457, 2.423, 2.390,  STU00310
                8     2.358, 2.326,                                             STU00320
                9     63.657, 9.925, 5.841, 4.604, 4.032, 3.707, 3.499, 3.355,  STU00330
                A     3.250, 3.169, 3.106, 3.055, 3.012, 2.977, 2.947, 2.921,  STU00340
                B     2.898, 2.878, 2.861, 2.845, 2.831, 2.819, 2.807, 2.797,  STU00350
                C     2.787, 2.779, 2.771, 2.763, 2.756, 2.750, 2.704, 2.660,  STU00360
                D     2.617,  2.576                                             STU00370
                E     636.619, 31.598, 12.941, 8.610, 6.859, 5.959, 5.405, 5.041,STU00380
                F     4.781, 4.587, 4.437, 4.318, 4.221, 4.140, 4.073, 4.015,  STU00390
                G     3.965, 3.922, 3.883, 3.850, 3.819, 3.792, 3.767, 3.745,  STU00400
                H     3.707, 3.690, 3.674, 3.659, 3.646, 3.551, 3.460, 3.373, 3.291/STU00410
0010            J = OLDRW(JW)                                                   STU00420
0011            GO TO 10                                                        STU00430
0012            ENTRY STR                                                       STU00440
                C     ENTRY STR COMPARES A GROUP WITH THE FIELD CONTAINING      STU00450
                C     THE PIXEL GROUP TO THE RIGHT OF THE CURRENT GROUP.        STU00460
0013            J = RWFLD(JS+PXGR)                                              STU00470
0014            GO TO 10                                                        STU00480
0015            ENTRY STRI                                                      STU00490
                C     ENTRY STRI COMPARES A FILED CONTAINING A GIVEN PIXEL GROUP STU00500
                C     TO THE GOURP IMMEDIATELY TO ITS RIGHT.                    STU00510
0016            J = RWFLD(JS-PXGR)                                              STU00520
0017        10  ANI = AFLD(J)*APGR                                              STU00530
                C     ANI IS NUMBER OF PIXELS IN FIELD CURRENT PIXEL GROUP IS BEING STU00540
                C     COMPARED TO.                                              STU00550
0018            BNUMB = ANI*APGR-2.0                                            STU00560
0019            DO 25 NS = 1,NCHAN                                              STU00570
0020            AV1 = ASUM(JS,NS)/APGR                                          STU00580
0021            AV2 = AFTOT(J,NS)/ANI                                           STU00590
                C     AV1 AND AV2 ARE THE MEANS OF THE SAMPLES.                 STU00600
0022            D1 = ASQ(JS,NS)-ASUM(JS,NS)**2/APGR                             STU00610
0023            D2 = AFS(J,NS)-AFTOT(J,NS)**2/ANI                               STU00620
                C     D1 AND D2 ARE THE VARIANCES OF THE TWO SAMPLES. IN THE    STU00630
                C     FOLLOWING LOGIC THE STANDARD DEVIATIONS OF THE SAMPLES    STU00640
                C     ARE COMPARED TO THE MEANS. IF EITHER STANDARD DEVIATION   STU00650
                C     IS LARGER THAN 15% OF THE MEAN, SAME = .FALSE. IS RETURNED.STU00660
0024            IF (D1 .EQ. 0.0) GO TO 12                                       STU00670
0025            IF (SQRT(D1* APGR)) .GT. .15*AV1) GO TO 15                      STU00680
0026        12  IF (D2 .EQ. 0.0) GO TO 18                                       STU00690
0027            IF (SQRT(D2/ANI) .LE. .15*AV2) GO TO 18                         STU00700
0028        15  SAME  = .FALSE.                                                STU00710
0029            RETURN                                                          STU00720
0030        18  ASP = (D1+D2)/BNUMB                                            STU00730
0031            IF (ASP .NE. 0.0) GO TO 20                                      STU00740
                C     ASP IS THE SP**2 IN THE PAPER                             STU00750
0032            AT(NS) = 6.0                                                    STU00760
0033            GO TO 25                                                        STU00770
0034        20  AT(NS) = (AV1-AV2)/SQRT(ASP*(1.0/ANI+APGR1))                   STU00780
```

FORTRAN IV MODEL 44 PS VERSION 3, LEVEL 4 DATE 72215

```
      25 CONTINUE
      30 IDT = BNUMB
         IF (IDT .GT. 30) IDT = 30
         IF (IDF .LE. 30) GO TO 40
         IF (IDF .LE. 40) GO TO 39
         IF (IDF .LE. 60) GO TO 38
         IF (IDF .LE. 120) GO TO 37
         IDT = IDT+1
      37 IDT = IDT+1
      38 IDT = IDT+1
      39 IDT = IDT+1
C        THE ABOVE LOGIC GETS THE CORRECT INDEX FOR THE NUMBER OF DEGREES
C        OF FREEDOM.
      40 SAME = .TRUE.
         DO 60 NS = 1,NCHAN
      60 IF (ABS(AT(NS)) .GT. ATSTAT(IDT,SIGLEV)) SAME = .FALSE.
C        IF THE VALUE OF T (AT(NS)) IS GREATER THAN THE CRITERION VALUE
C        FOR THE T FOR ANY SPECRA, THEN THE NULL HYPOTHESIS THAT
C        THE TWO SAMPLES ARE THE SAME IS REJECTED.
         RETURN
         END
```

FORTRAN IV MODEL 44 PS VERSION 3, LEVEL 4 DATE 72215

```
         SUBROUTINE PRINT
         IMPLICIT  INTEGER*2  (E-Z)
         INTEGER*4  OUT, IN, JNR, NS
         LOGICAL*1  SAME, ASIN, ENOUGH, CLOSE, OPEN
         COMMON  AFTOT(250,4), AFS(250,4), ASUM(256,4), ASQ(256,4), AT(4),
        1     AFLD(250), APGRI, AST, APGR, OUT, IN, JNR, NS
         COMMON /INT2/ PIX(1280), RWFLD(256), OLDRW(256), STACK(250),
        1     FSR(250), LASTR(130), SP(130,100), PXGR, PXGRI, NCHAN ,
        2     ROWS, W1, W2, WL2, WIDE, MFS, JW, JS, JR, RL1, IFLD, SIGLEV,
        3     IDF, R1, R2, IR1, IR2, ICLOSE, PCLASS, PWIDE, NEXT, JLINE
         COMMON /RESULT/ SAME, ASIN(256), ENOUGH(250), CLOSE(250),OPEN(250)
         DIMENSION NUMBER(48), PRNT(130), HEAD(10), P1(130), P2(130)
         EQUIVALENCE (ASUM(1),P1(1)), (ASQ(1),P2(1))
         LOGICAL*1 FIRST
         DATA HEAD / '0', '1', '2', '3', '4', '5', '6',
        1     '7', '8', '9' /
         DATA EMPTY / ' ' /
         DATA NUMBER  /
        .             'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J',
        .     'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V',
        .     'W', 'X', 'Y', 'Z', '1', '2', '3', '4', '5', '6', '7', '8',
        .     '9', '0', '=',                     1H', '*', '(', ')',
        .     '-',          '&', '+',                '$',         '/',
        .          ',',          '.' /
C
C        ENTRY TO PRINT MERELY STORES RESULTS OF THE CURRENT ROW INTO
C        THE PRINT ARRAY, SP.
         JLINE = JLINE+1
         I = 0
         DO 100 J = W1,WL2,PXGR
         I = I+1
     100 SP(I,JLINE) = RWFLD(J)
         RETURN
         ENTRY PRINT1
C        ENTRY PRINT1 ACTUALLY PRINTS RESULTS OF A NUMBER OF ROWS.
C        BEFORE PRINTING, ANY FIELDS WHICH ARE OPEN ON THE LAST
C        ROW TO BE PRINTED ARE CLASSIFIED (NOT CLOSED).
         JLAST = MOD(JNR,50)+50
         IF (JNR .LT. 50) JLAST = JNR
         J = 0
         DO 110 JW = W1,WL2,PXGR
         J = J+1
         IF (SP(J,JLAST) .GE. 256) GO TO 110
C        WHEN A FIELD IS CLOSED, SP IS SET TO THE FIELD IDENTIFICATION
C        INDEX PLUS 256. THUS ONLY THOSE FIELDS WITH ENTRIES IN SP
C        OF LESS THAN 256 ARE STILL OPEN.
         ICLOSE = RWFLD(JW)
C        SUBROUTINE CLASS ASSUMES THAT ICLOSE CONTAINS THE ELEMENT OF
C        AFLD TO BE CLASSIFIED.
         CALL CLASS
         FSR(ICLOSE) = 101
     110 CONTINUE
C        FSR SHOULD BE EQUAL TO THE FIRST ROW OF A FIELD WHICH
C        NEEDS TO HAVE ITS ENTRIES IN THE SP ARRAY PROCESSED.
C        SINCE THOSE ROWS OF THIS FIELD DOWN TO ROW JR
C        ARE ABOUT TO BE PRINTED, THE NEXT ROW OF THIS FIELD
C        WHICH WILL HAVE ACTIVE ENTRIES IN SP IS ROW JR+PXGR.
C
C        ALL FIELDS UNCLASSIFIED IN THE AREA TO BE PRINTED ARE
C        CLASSIFIED.
C
         DO 200 PLINE = 1,JLAST
         JRLINE = JRLINE + PXGR
         DO 160 J = 1,PWIDE
         SPK = SP(J,PLINE)
         IF (J .EQ. 1 .OR. J .EQ. PWIDE .OR. JR .EQ. IR2+1 .AND.
        1     PLINE .EQ. JLAST .OR. FIRST .AND. PLINE .EQ. 1) GO TO 150
         PRNT(J) = EMPTY
         IF (SPK .EQ. SP(J,PLINE+1) .AND. SPK .EQ. SP(J,PLINE-1)
        1     .AND. SPK .EQ. SP(J+1,PLINE) .AND. SPK .EQ. SP(J-1,PLINE))
        2     GO TO 160
     150 PRNT(J) = NUMBER(SPK-256)
     160 CONTINUE
C        THE DO 160 LOOP SETS UP A LINE TO BE PRINTED.
C        THE VECTOR PRNT CONTAINS THE TEXT OF THE LINE TO BE PRINTED.
C        THE LINE CONTAINS BLANKS EXCEPT FOR BOUNDARIES OF FIELDS.
C        ON BOUNDARIES THE FIELD IDENTIFICATION CHARACTER IS PRINTED.
     200 WRITE (OUT,210) JRLINE, (PRNT(J),J=1,PWIDE)
```

FORTRAN IV MODEL 44 PS VERSION 3, LEVEL 4 DATE 72215

```
     210 FORMAT (' ',I4,1X, 114A1)
         IF (JR .EQ. IR2+1) RETURN
C        THIS IF WILL CAUSE A RETURN WITHN THE LAST CALL TO
C        IS BEING EXECUTED. (JR WAS SET TO IR2+1 IN MAIN JUST
C        BEFORE THE LAST CALL TO PRINT1.)
         DO 300 I = 1,PWIDE
         LASTR(I) = SP(I,50)
C        LASTR IS THE LAST ROW FROM THIS PRINT. IT IS ARRANGED
C        IN COMMON AS THE ZEROE'TH VECTOR OF SP SO THAT
C        IT IS AVAILABLE FOR COMPARISONS THE NEXT TIME
C        PRINT1 IS CALLED.
C
         DO 300 J = 1,50
     300 SP(I,J) = SP(I,J+50)
C        UNPRINTED ROWS OF RESULTS ARE  MOVED TO THE FIRST 50
C        ROWS OF THE SP PRINT ARRAY.
         DO 310 J = 1,250
     310 FSR(J) = FSR(J) -50
C        ELEMENTS OF FSR MUST CORRESPOND TO THE NEW PLACEMENT OF ROWS
C        IN SP. THUS THE DO 310 LOOP ALTERS FSR.
         FIRST = .FALSE.
C        FIRST = .FLASE. INDICATES TO FUTURE CALLS TO PRINT1
C        THAT PRINT11 HAS BEEN CALLED PREVIOUSLY. THE DIFFERENCE
C        IN THE FIRST CALL IS THAT THE FIRST LINE IS A BOUNDARY.
         JLINE = 50
         RETURN
         ENTRY PRINTO
C         ENTRY PRINTO PRINTS THE HEADING AND INITIALIZES JLINE
C        AND FIRST.
         WRITE (OUT,420) MFS,PXGR, PXGR, SIGLEV, NCHAN
     420 FORMAT (' MIN. NO. OF PIXEL GROUPS CALLED A FIELD ='
        1     ,I3/' PIXEL GROUP SIZE IS ',I1,' X ',I1,
        2     '      SIGNIFICANCE LEVEL = ',I1,
        3     '       NUMBER OF SPECTRA = ',I1/)
         J = 0
         DO 430 JW = W1,WL2,PXGR
         J = J+1
         PRNT(J) = EMPTY
         P1(J) = EMPTY
         IF (JW .GT. 99) PRNT(J) = HEAD(MOD(JW/100,100)+1)
         IF (JW .GT. 9) P1(J) = HEAD(MOD(JW/10,10)+1)
     430 P2(J) = HEAD(MOD(JW+0,10)+1)
         WRITE (OUT,440) (PRNT(J),J=1,PWIDE)
         WRITE (OUT,440) (P1(J),J=1,PWIDE)
         WRITE (OUT,440) (P2(J),J=1,PWIDE)
     440 FORMAT (' ',4X,114A1)
         JRLINE = -PXGR+1
         FIRST = .TRUE.
         RETURN
         END
```

FORTRAN IV MODEL 44 PS VERSION 3, LEVEL 4 DATE 72215

```
         SUBROUTINE GETPRM
C        GETPRM READS IN VARIADUS PARAMETERS. THE FIRST IS THE INPUT
C        UNIT NUMBER FROM WHICH THE REMAINING PARAMETERS ARE
C        TO BE READ.
C        THEN COMES THE TITLE, PIXEL GROUP SIZE, ROWS AND
C        COLUMNS TO BE PROCESSED, SIGNIFICANCE LEVEL, MINIMUM FIELD
C        SIZE, NUMBER OF SPECTRA AND THE UNIT NUMBER
C        TO WHICH THE OUTPUT IS TO BE WRITTEN. OTHER INITIALAZATION
C        FUNCTIONS ARE ALSO PERFORMED.
         IMPLICIT  INTEGER*2  (E-Z)
         INTEGER*4  OUT, IN, JNR, NS
         LOGICAL*1  SAME, ASIN, ENOUGH, CLOSE, OPEN
         COMMON  AFTOT(250,4), AFS(250,4), ASUM(256,4), ASQ(256,4), AT(4),
        1     AFLD(250), APGRI, AST, APGR, OUT, IN, JNR, NS
         COMMON /INT2/ PIX(1280), RWFLD(256), OLDRW(256), STACK(250),
        1     FSR(250), LASTR(130), SP(130,100), PXGR, PXGRI, NCHAN ,
        2     ROWS, W1, W2, WL2, WIDE, MFS, JW, JS, JR, RL1, IFLD, SIGLEV,
        3     IDF, R1, R2, IR1, IR2, ICLOSE, PCLASS, PWIDE, NEXT, JLINE
         COMMON /RESULT/ SAME, ASIN(256), ENOUGH(250), CLOSE(250),OPEN(250)
         INTEGER*4 ID,RUNNUM,ERRGRN
         COMMON /LARS/ ID(200),RUNNUM,RCSEL(30)
         DATA ATT /'T' /
C        WRITE (15,5)
C        READ (15,65) IN
       5 FORMAT (' INPUT ')
         IN=5
         IF (IN .EQ. 15) WRITE (15,6)
       6 FORMAT (' RUN IDENTIFIER ')
         READ (IN,7) (AFS(I,1),I=1,4)
       7 FORMAT (4A4)
         IF (IN .EQ. 15) WRITE (15,10)
      10 FORMAT (' GROUP SIZE ')
         READ (IN,15) PXGR
      15 FORMAT (I1)
         IF (IN .EQ. 15) WRITE (15,30)
      30 FORMAT (' ROWS & COLS 4I4')
         READ (IN,35) R1,R2,W1,W2
      35 FORMAT (4I4)
         IF (IN .EQ. 15) WRITE (15,40)
      40 FORMAT (' SIGLEV ')
         READ (IN,120) SIGLEV
         AST = ATT
C
         IF (IN .EQ. 15) WRITE (15,60)
      60 FORMAT (' MIN SIZE ')
         READ (IN,65) MFS
      65 FORMAT (I2)
         R2 = R2-MOD(R2-R1+1,PXGR+0)
         W2 = W2-MOD(W2-W1+1,PXGR+0)
C        THE USE OF PXGR+0  IN THE ABOVE STATEMENTS AVOIDS
C        AN ERROR FOR ILLEGAL TYPE.
         WIDE = W2-W1+1
         ROWS = R2-R1+1
         PXGRI = PXGR-1
         RL1 = MFS-PXGRI
         IDF = 2*PXGR**2-2
         APGR = PXGR**2
         APGRI = 1.0/APGR
         IF (IN .EQ. 15) WRITE (15,80)
      80 FORMAT (' NUMBER OF SPECTRA ')
         READ (IN,120) NCHAN
         IF (IN .EQ. 15) WRITE (15,110)
     110 FORMAT(' OUTPUT ')
         READ (IN,120) OUT
     120 FORMAT (I1)
         WRITE (OUT,130) (AFS(I,1),I=1,4)
     130 FORMAT (30X4A4)
         DO 300 I = 1,250
     300 STACK(I) = 1
         NEXT = 1
         PWIDE = WIDE/PXGR
         DO 310 I = 1,130
     310 LASTR(I) = 0
         JLINE = 0
         IF (IN .EQ. 15) WRITE (15,320)
     320 FORMAT (' RUN NUMBER     (I8)')
         READ (IN,330) RUNNUM
     330 FORMAT (I8)
```

```
FORTRAN IV    MODEL 44  PS       VERSION 3,  LEVEL 4   DATE  72215

0062                  CALL GADRUN (RUNNUM,11,ID,ERRGRN)                          GET00780
0063                  IF(ERRGRN.EQ.0) GO TO 335                                  *
0064                  WRITE(15,5010) ERRGRN
0065            5010  FORMAT(' GADRUN ERROR ',I2)
0066                  PAUSE ' EL BOMBO '
0067            335   CONTINUE                                                   GET00790
0068                  DO 340 I = 1,30                                            GET00800
0069              340 RCSEL(I) = 0
0070                  IF (IN .EQ. 15) WRITE (15,350)                             GET00820
0071              350 FORMAT (' WHICH CHANNELS ',4I2)
0072                  READ (IN,360)(PIX(I),I=1,NCHAN )                           GET00840
0073              360 FORMAT (4I2)
0074                  DO 370 I = 1,NCHAN                                         GET00850
0075              370 RCSEL(PIX(I)) = 1                                          GET00860
0076                  RETURN
0077                  END
```

```
FLGS  L.CTR   OBJCODE   ADDR   STMT      SOURCE STATEMENT

                                    1   *       SHIFT IS A FORTRAN CALLED ROUTINE TO CONVERT PIXEL DATA TO      XX 00030
                                    2   *       PROPER INTEGER*2 FORMAT. THE CALLING SEQUENCE IS                XX 00040
                                    3   *       CALL SHIFT (PIX,DATA,WORDS) WHERE PIX IS AN I*2 VECTOR WHERE    XX 00050
                                    4   *       THE CONVERTED DATA IS TO BE PLACED. DATA IS A LOGICAL VECTOR    XX 00060
                                    5   *       WHERE THE DATA COMES IN ONE NUMBER PER BYTE, AND WORDS IS        XX 00070
                                    6   *       THE NUMBER OF NUMBERS TO BE CONVERTED. WORDS IS AN I*2           XX 00080
                                    7   *       VARIABLE.                                                       XX 00090
000000                              8   SHIFT   CSECT                                                           XX 00100
000000  90EC D00C   0000C           9           STM   14,12,12(13)                                              XX 00110
000004  05C0                       10           BALR  12,0                                                      XX 00120
                    00006          11           USING *,12                                                      XX 00130
000006  5821 0004   00004          12           L     2,4(1)   ADDRESS OF DATA                                 XX 00140
00000A  5831 0008   00008          13           L     3,8(1)   ADDRESS OF WORDS                                XX 00150
00000E  5811 0000   00000          14           L     1,0(1)   ADDRESS OF PIX                                  XX 00160
                                   15   *                                                                       XX 00170
000012  1855                      16           SR    5,5   ZERO BXLE COUNTER                                   XX 00180
000014  184A                      17           SR    10,10  ZERO WORK REGISTER                                 XX 00190
000016  5860 C03A   00040          18           L     6,=F'1'   INCREMENT FOR BXLE                              XX 00200
00001A  4873 000D   00000          19           LH    7,0(3)   LOAD WORDS                                      XX 00210
00001E  5870 C03A   00040          20           S     7,=F'1'   GET END OF LOOP CONDITION                       XX 00220
                                   21   *                                                                       XX 00230
000022  43A2 0000   00000          22   LOOP    IC    10,0(2)  LUAD DATA BYTE                                   XX 00240
000026  40A1 0000   00000          23           STH   10,0(1)  STORE INTO OUTPUT (PIX)                          XX 00250
00002A  4A20 C03E   00044          24           AH    2,=H'1'  ADD TO DATA ADDRESS                              XX 00260
00002E  4A10 C040   00046          25           AH    1,=H'2'  ADD TO STORAGE ADDRESS                           XX 00270
                                   26   *                                                                       XX 00280
000032  8756 C01C   00022          27           BXLE  5,6,LOOP                                                  XX 00290
                                   28   *                                                                       XX 00300
000036  982C D01C   0001C          29           LM    2,12,28(13)                                               XX 00310
00003A  92FF D00C   0000C          30           MVI   12(13),X'FF'                                              XX 00320
00003E  07FE                       31           BR    14                                                        XX 00330
                                   32   LTORG
000040  00000001                  33   DC    F'1'
000044  0001                      34   DC    H'1'
000046  0002                      35   DC    H'2'
                    00000          36           END   SHIFT                                                     XX 00340
```

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 I1. 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 4I4' Format is 4I4. R1 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 1'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.