USER'S GUIDE AND PROGRAM DOCUMENTATION FOR CLASSY

(ADAPTIVE MAXIMUM-LIKELIHOOD CLUSTERING PROGRAM)

Prepared for

GODDARD SPACE FLIGHT CENTER

By

COMPUTER SCIENCES CORPORATION

Under

Contract NAS 5-27888
Task Assignment 80300

Prepared by:

Kevin J. Ingram for 3/27/85
R. White Date

Approved by:

John W. Wood 3/27/85
J. Wood Date
Technical Supervisor

G. Mahler 3/27/85
G. Mahler Date
Functional Area Manager

ABSTRACT

The Massively Parallel Processor (MPP) implementation of the CLASSY algorithm for adaptive maximum-likelihood clustering is described. The purpose, method, and capabilities of the program are outlined; instructions for using the program, including user parameters, file structures, and other external interfaces are given; and details of program structure, including descriptions of the individual program modules and specifications for internal data structures, are presented. The mathematical formulas used by the program are also discussed, with emphasis on modifications to take advantage of the MPP environment.

TABLE OF CONTENTS

TABLE OF CONTENTS (Cont'd)

TABLE OF CONTENTS (Cont'd)

LIST OF ILLUSTRATIONS

Figure

LIST OF TABLES

Table

SECTION 1 - INTRODUCTION

1.1 PURPOSE

The CLASSY program performs unsupervised clustering of multichannel image data using an adaptive maximum-likelihood method that automatically tries to find the optimum number of clusters. By using the Massively Parallel Processor (MPP) to compile cluster statistics, rapid convergence can be obtained for image subsets as large as 16,384 pixels.

1.2 PROCESSING PROCEDURE OVERVIEW

After reading the user-supplied processing parameters and setting up control tables and flags, CLASSY extracts up to 16,384 multichannel pixels from the input image(s) and loads them into the memory planes of the MPP Array Unit (ARU). If the input image(s) contain more than 16,384 pixels, the pixels to be loaded are selected quasi-randomly.

CLASSY then begins the clustering procedure by computing the mean value vector, the covariance matrix, and the traces of the skew and kurtosis tensors for the full set of pixels in the ARU memory. By comparing the skew and kurtosis values with the values that would be obtained if the data all belonged to a single multivariate normal distribution, CLASSY divides the data into two clusters and estimates the mean vector, covariance matrix, and a priori cluster probability for each.

At this point, CLASSY enters the main program loop. This loop alternates between two phases: a statistics-gathering phase and a decision phase. In the statistics-gathering phase, the estimated means, covariances, and a priori cluster probabilities for the existing clusters are iteratively refined until a (local) maximum of the likelihood function is obtained. In the decision phase, the skew, kurtosis,

cluster probabilities, and various measures of cluster similarity are used to determine which existing clusters should be subdivided, which should be merged into a single cluster, and which contain so few pixels that they can be eliminated.

When no further changes in the clusters occur, or a maximum iteration count is reached, CLASSY generates a pseudoimage containing the most probable cluster assignment for each pixel and then stores both this pseudoimage and a table of the cluster statistics on disk files.

During execution, CLASSY reports the clusters created, merged, or eliminated as the result of each decision phase. As options, the program can report current mean value vectors and covariance matrices for each cluster every time the decision phase is entered. It can also generate and display a pseudocolor class map after each iteration of the statistics-refinement process.

The five main functional modules of CLASSY and the main data flows between them and also to and from disk files and user interfaces are shown in Figure 1-1.

1.3 RESTRICTIONS

The initial MPP implementation of CLASSY supports only 8-bit image data, with nonnegative values between 0 and 255. A maximum of 21 image channels are permitted.

CLASSY supports a maximum of 32 clusters. When this limit is reached, the program performs one more iteration of the statistics-gathering and decision phases and then terminates execution unless one or more clusters are found that can be consolidated or eliminated.

The appendix to this document reports on the status of CLASSY testing and suggests program modifications and enhancements, and code revisions.

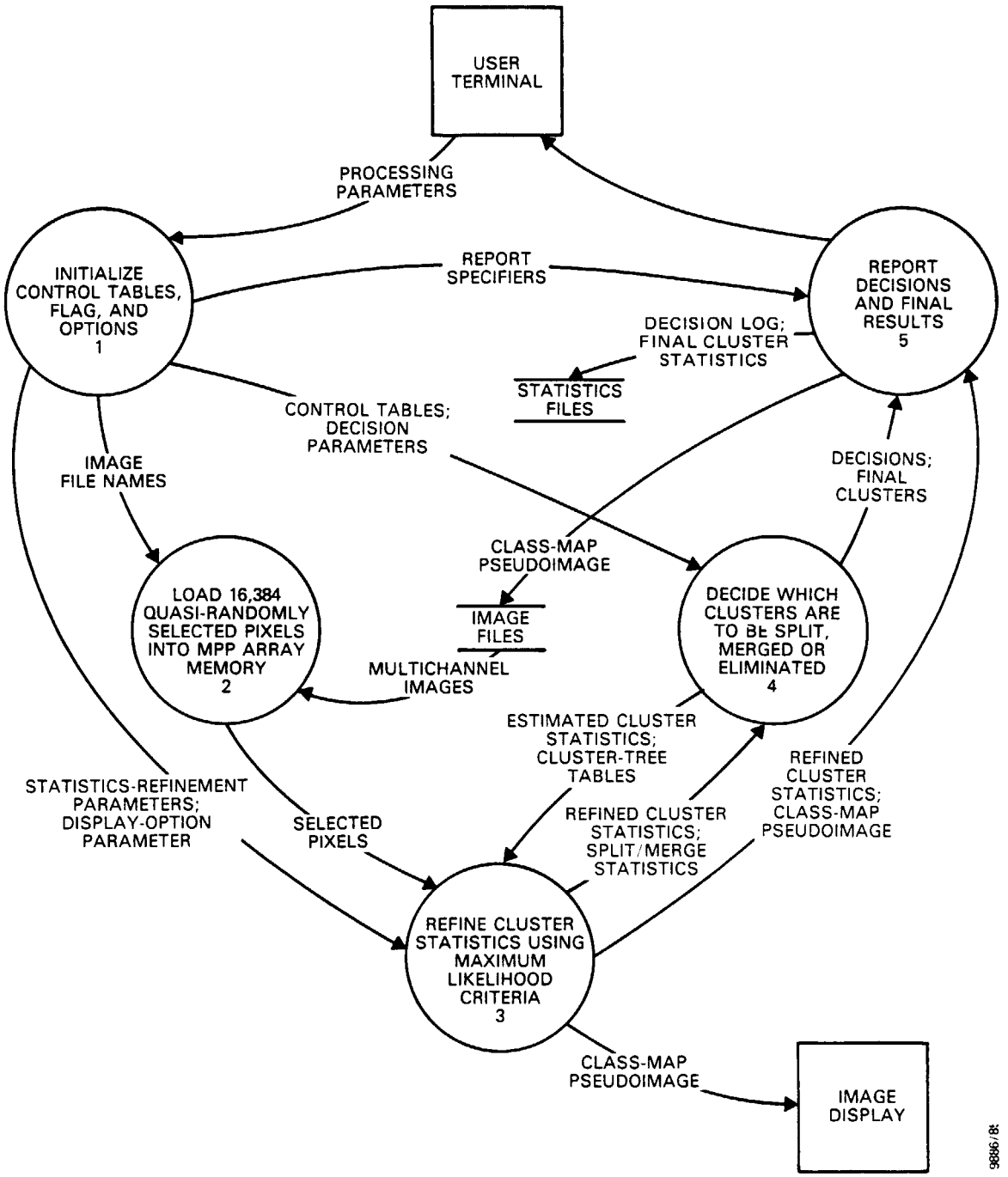

Figure 1-1. Principal Function and Data Flows for CLASSY

SECTION 2 - USER'S GUIDE

CLASSY is designed to run under the Transportable Applications Executive (TAE) with or without the Catalog Manager. The program may be invoked using TAE tutor mode or by a single TAE command line. In the latter case, an existing parameter file must normally be referenced.

2.1 PROCESSING PARAMETERS

CLASSY requires the user to specify the number and names of the input image channels and the name of the file to receive the final classification statistics. The user has the option of requesting a log of the decision process in four different levels of detail; an output pseudoimage giving, for each pixel, the number of the cluster to which it most likely belongs; and a pseudocolor display of the current most likely cluster assignments after each iteration of the clustering algorithm.

The default values for a number of parameters used in the various tests for cluster splitting, merging, and elimination may also be replaced by user-specified values. The parameters are described in the following subsections. During a processing session, the user may also obtain full details about each parameter by invoking the TAE "Help" facility.

2.1.1 MANDATORY PARAMETERS

Name	Description
NCHAN	Integer. Number of image channels; must be between 2 and 21.
INIMAGES	Set of NCHAN file names. Names of files containing input image channels; one name must be supplied for each channel.
STATFILE	File name. File to receive statistics for final set of clusters.

2.1.2 OUTPUT OPTIONS

Name	Description
OUTIMAGE	File name. File to receive 128-by-128 pseudo-image giving final cluster assignment for each pixel used to develop statistics. If no file name is specified, pseudoimage will not be saved.
CRTLOG	Character string. Type of report to be displayed on terminal during each iteration of the decision phase. Options are as follows: "NONE" --No report "SHORT"--Report only split/merge decisions "MEANS"--Report also means and weights for active clusters each time decision phase is entered "FULL" --Report also convergence behavior after each MPP statistics phase and test values used for split/merge decisions "COVAR"--Report also covariance arrays on entry to decision phase Default is "SHORT."
DISKLOG	Character string. Type of processing report to be written to log file on disk; options are same as for CRTLOG. Default is "NONE."
LOGFILE	File name. Name of file to receive processing reports. Default is "CLASSY.LOG."
DISPLAY	"YES" or "NO". Specifies whether pseudocolor class map should be sent to image display after each iteration of statistics-refinement phase. Default is "NO".

2.1.3 STATISTICS-REFINEMENT PARAMETERS

Name	Description
MAXMITER	Integer. Maximum number of iterations of mean value refinement loop before reentering decision phase. Default is 10.
SPREAD	Real. Constant to be added to all diagonal elements of covariance matrices to discourage single-point clusters (see Section 5.1). Default is 0.25.

2.1.4 DECISION PARAMETERS

Name	Description
MAXDITER	Integer. Maximum number of iterations of decision phase; i.e., of main processing loop. Default is 20.
SPLITER	Integer. Number of iterations for refining estimated statistics for newly created subclusters during split procedure using skew and kurtosis data; zero indicates initial estimates are to be used without steepest-descent refinement. Default is 100.
ELIMTHR	Real. Threshold value of a priori probability at or below which a cluster will be eliminated; value of 0.0 will prevent elimination unless cluster is completely empty. Default is 0.001.
CONLEVEL	Real. Confidence level, in terms of standard deviations for the normal distribution. Used to compute thresholds for skew and kurtosis, above which cluster is tentatively split, and for likelihood ratio, above which tentative split is confirmed or tentative merge rejected. Default is 2.33, corresponding to 99-percent confidence level.
LMULT	Real. Multiplier for likelihood function before it is compared with threshold. Default is 2.0.
LBIAS	Real. Bias term in normalization constant for likelihood function (see Section 5.2). Default is 1.0.
REMRGTHR	Real. Value of logarithm of likelihood ratio of subclusters to parent below which tentative separation is rejected or tentative merge accepted if probability-difference function is below PDIFFTHR. Default is 1.0.
PDIFFTHR	Real. Value of probability-difference function between a cluster and its tentative subclusters below which the subclusters will be eliminated if the logarithm of likelihood ratios is below REMRGTHR. Default is 0.0025.
MERGETHR	Real. Threshold value of cluster similarity function (see Section 5.3) below which two clusters will be tentatively merged. Default is 0.25.

Name	Description
ACOEFF	Real. Coefficient A, which weights the difference of diagonal covariance elements, in cluster similarity function. Default is 0.3.
BCOEFF	Real. Coefficient B, which weights the difference in a priori probabilities, in cluster similarity function. Default is 0.18.

2.2 INPUT IMAGE FILES

The image to be processed by CLASSY may be in either of two standard formats:

- TAE--First disk sector contains label specifying image dimensions; imagery starts in second sector, with each line beginning at a sector boundary
- LAS--Label record describing image size is in a separate file, generated by Land Analysis System (LAS) Image I/O routines and maintained by Catalog Manager; imagery starts in first sector of image file, with each line beginning at a sector boundary

CLASSY automatically determines the file format; if neither a TAE nor an LAS standard label is found, processing is aborted.

Each channel must be in a separate image file. All image channels must contain the same number of lines and of pixels per lines and must be in registration. Each pixel component must be in an 8-bit byte and will be interpreted as a non-negative number between 0 and 255.

If the input image contains more than 16,384 pixels, a sample of exactly 16,384 pixels will be extracted from the entire image by dividing it into a 128-by-128 array of cells and selecting one pixel quasi-randomly from each cell. If the input image contains fewer than 16,384 pixels, pixels containing zero in every channel will be generated to make

up the difference. This will result in an extra, spurious cluster having an all-zero mean vector. If either dimension of the image is less than 128, the array of cells will not be square, and class-map pseudoimages optionally written to an output disk file or displayed on the image terminal will not be meaningful.

2.3 OUTPUT PSEUDOIMAGE FILE

If the user requests generation of an output pseudoimage, CLASSY will write a single image file, containing 128 lines of 128 pixels each, in standard TAE image format. This file consists of a header record, containing the image dimensions, followed by one record for each line, with each line beginning on a sector boundary. Generation of an LAS format pseudoimage file is not currently supported.

2.4 STATISTICS FILES

The cluster statistics are the principal output from CLASSY. They are recorded in a format designed to facilitate their use as input to the maximum likelihood classification program, MAXLIK.

The cluster statistics will be recorded in an ASCII file. The file will be divided into two parts, the first giving the cluster means, probabilities, and cluster-tree relationships, and the second, the covariance data.

2.4.1 CLUSTER MEANS AND PROBABILITIES

The first part of the file will contain the following records:

- A header containing the text "CLASSY statistics for nn clusters"
- Properly aligned column headers for the data records
- One data record for each cluster

Each data record will contain the following entries:

- Cluster serial number
- Serial number of parent cluster, or zero if not a subcluster
- A priori probability (weight) of cluster, to three decimal places
- Fraction of pixels for which this was the most probable cluster, to three decimal places
- Mean value for each image channel, to two decimal places

When a large number of channels are used (more than about 12), each data record will be divided into two, to fit the length of the printer line.

2.4.2 COVARIANCE DATA

The second half of the statistics file will be introduced by a record containing the title "Covariance Data." A covariance table for each cluster will then be presented. The first record of each table will give the cluster serial number and an equivalent radius value that represents the standard deviation of a spherically symmetric distribution having the same volume as the cluster. The covariance values will be given to two decimal places; each row of the covariance table will occupy one record. When a very large number of channels are used (more than about 16), each row will be split into two records to avoid overflowing the printer line.

2.5 DECISION LOG

The decision log provides a record of all decisions to split, merge, or eliminate tentative clusters and may also be used to report the tentative cluster statistics during processing. Normally, two versions of this log are

generated: one version is written to a disk file, and the other is displayed on the user's terminal. The amount of detail to be included in these two copies is controlled by the DISKLOG and CRTLOG parameters, respectively.

The decision log is formatted for a standard 80-column ASCII display. Except for the header, which is included only in the disk-file version, the two versions use identical formats. Specific contents are discussed in the following subsections.

2.5.1 DECISION LOG HEADER

When CLASSY begins processing, it opens the decision-log file, unless the user specified DISKLOG = NONE, and writes a header block. This block records the value of all the processing parameters, including those for which the user accepted the default value. Each parameter occupies a separate line.

2.5.2 CLUSTER STATUS REPORT

Every time the decision phase is entered, CLASSY will generate a message giving the iteration number of the main processing loop. If the DISKLOG or CRTLOG parameter is MEANS, FULL, or COVAR, statistics for each cluster in the current set of tentative clusters will also be written out, as follows:

- Cluster serial number
- Serial number of parent cluster
- Current a priori weight for cluster
- Fraction of samples for which this was the most likely cluster; i.e., the number of samples assigned to this cluster in the class-map pseudoimage
- Cluster mean values for each image channel

If the DISKLOG or CRTLOG parameter is FULL or COVAR, CLASSY also records the convergence behavior for the preceding MPP statistics-refinement phase. Convergence behavior for each iteration of the main MPP processing loop, which yields an updated set of trial means and covariances for each cluster, is reported on a separate line. Each line contains the following entries:

- Loop iteration number
- Maximum change in any component of the means for any cluster
- Number of iterations of inner loop, which adjusts a priori weights
- Maximum change in any weight value on the first, second, next to last, and last iteration of the inner loop

If the DISKLOG or CRTLOG parameter is COVAR, the covariance matrix for each current cluster will also be written. The format for each cluster is

- Header line giving cluster serial number and a measure of the spread of the cluster in terms of the radius of a hyperspherical distribution having the same volume as the actual, normally ellipsoidal, distribution
- Line of column headers
- Body of matrix, with each row of values preceded by the row number

If the number of channels is large enough that all matrix columns do not fit on a single line, the matrix is divided into square submatrices. Because the covariance is symmetric, submatrices lying entirely below the principal diagonal are not written out.

2.5.3 CLUSTER CREATION AND DELETION REPORTS

A one- or two-line decision log entry will be written every time one of the following occurs:

- A cluster is tentatively split into two subclusters
- A tentative separation into subclusters is confirmed
- A tentative separation into subclusters is rejected
- Two similar clusters are tentatively merged
- A tentative merge is confirmed
- A tentative merge is rejected
- A cluster is eliminated because it contains too few pixels

Each record will include the serial number(s) of the cluster(s) affected. If the DISKLOG or CRTLOG parameter specifies FULL or COVAR, the test values on which the decision was based will also be reported.

2.6 ERROR CONDITIONS

Most error conditions occurring during CLASSY execution will be trapped and reported by executive-level software. Invalid user parameters will be identified by the TAE command line or tutor mode modules, and MPP hardware problems will normally be flagged by the MPP Control and Debug Module (CAD). Certain image file input/output (I/O) errors, however, will be reported by the CLASSY software modules in which they are encountered. These various classes of errors, and appropriate corrective action, are discussed in the following subsections.

2.6.1 PARAMETER ERRORS

All fatal parameter errors are trapped by the TAE parameter passing routines. They are of three types:

- Missing parameter--The specified nondefaultable parameters have been omitted. The user must enter values for them before TAE will execute CLASSY.
- Parameter out of range--A numeric parameter is outside the permitted limits. These limits are specified in the "Help" file for CLASSY. A user who wishes to experiment with extreme values of the parameters may edit the file CLASSY.PDF to change these limits.
- Wrong number of parameters--The number of names of input image channels entered for parameter INIMAGES does not match the number specified by NCHAN. Either NCHAN or INIMAGES must be corrected.

2.6.2 MPP EXCEPTIONS AND ERRORS

The only MPP-related messages that do not need to be reported to the system manager are the following:

- Waiting for MPP--MPP currently in use by another task. The request for MPP access will be repeated automatically, with periodic repetitions of the "Waiting" message. The user may ask TAE to abort execution if he or she does not wish to wait.
- Got the MPP--MPP is now free; CLASSY processing is proceeding.

It should be noted that the MPP does not trap errors such as arithmetic overflow or divide-by-zero.

2.6.3 IMAGE FILE I/O ERRORS

In addition to errors reported by the VAX/VMS I/O services, CLASSY writes messages for four types of image I/O error:

- Input open failure--The channel and image file specified in the error message could not be opened. This may result from a typographical error when entering the file name or because the file has been deleted from the directory or catalog, is not in the default directory, or is on a disk pack not currently mounted.
- Input read error--Possible causes include an image label that overstates the number of lines in the image; this can be checked for noncataloged files using the DCL DIR/SIZE utility. Otherwise, the user should consult the system manager.
- Output open failure--The file for the class-map pseudoimage could not be opened. The user should check for an invalid file name and then consult the system manager.
- Image write error--An error occurred in writing the class-map pseudoimage. The user should consult the system manager.

CLASSY also generates self-explanatory warning messages if the input file contains fewer than 16,384 pixels, so that fill pixels must be generated or, if there are fewer than 128 pixels per line, so that sampling is not feasible and the Interactive Imaging System (IIS) display, if requested, will be rendered meaningless by line wraparound.

2.7 TIMING ESTIMATES

The execution time for CLASSY depends very much on how many iterations are required to achieve convergence to a stable result as well as on the number of clusters and number of

image channels. Normally, most of the time is spent by the MPP in refining statistics for a given set of trial clusters. When the host VAX computer is heavily loaded, however, the VAX time needed for split/merge decisions may be significant.

Very roughly, the time needed by the MPP for each iteration of the main processing loop is proportional to the current number of clusters and to the square of the number of image channels. If the image contains only well-separated clusters, the required number of iterations of the main loop will tend to be lower and convergence within each main iteration will be much faster than if a number of clusters are heavily overlapping. The user's choice of certain parameters specifying test thresholds, etc., will also affect processing time.

Worst-case upper limits for MPP time for one iteration of the main loop are estimated at 0.1 second per cluster for 4-channel imagery and 1 second per cluster for 16-channel imagery. For Landsat-type imagery containing between 10 and 30 meaningful clusters, it is expected that the first 5 or 6 iterations of the main loop will divide the image into as many as 30 trial clusters. In addition, at most 10 to 15 further iterations will be needed to reach a reasonably stable set of distributions. Maximum total MPP times are thus unlikely to exceed 1 minute for 4-channel imagery and 10 minutes for 16-channel imagery.

SECTION 3 - PROGRAM STRUCTURE

3.1 CLASSY MODULES

Source code for CLASSY is divided into 10 modules, of which 7 execute in the host VAX computer and 3 in the MPP. The five modules that are called directly by the main program correspond to the five main functions depicted in Figure 1-1. The 10 modules are listed below.

VAX Modules

CLASSY	Main program
CLINIT	Called by CLASSY to get parameters and initialize tables
CLLDPX	Called by CLASSY to load image pixels into ARU memory planes
CLSTAT	Called by CLASSY to reformat data for the MPP and invoke MPP-resident code
CLRPRT	Called by CLASSY to report final cluster statistics and, optionally, convergence behavior and intermediate cluster statistics
CLAJST	Called by CLASSY to adjust tentative set of distributions by examining refined statistics for current clusters to identify clusters that should be split, merged, or eliminated
SPLIT1	Called by CLAJST to compute trial means and covariances for the subclusters of one cluster that is being tentatively split

MPP Modules

MPPSTT	Called by CLSTAT to refine the weights, means, and covariances of a trial set of clusters and compute statistics needed for split/merge decisions
GETPROBS	Called by MPPSTT to compute relative probabilities of cluster membership for each sample (pixel) that maximize likelihood function for a given set of cluster means and covariances

DECSTAT	Called by MPPSTT to compute skew and kurtosis values used by CLAJST to identify clusters that are candidates for splitting and similarity measures used to accept or reject tentative split/join decisions

The following subsections provide a pseudocode overview of the function of each of the 10 modules.

3.1.1 CLASSY--MAIN PROGRAM

```
BEGIN
Initialize DR780 for data transfers between VAX and ARU
Get processing parameters and initialize tables
Extract pixels from input image and load into ARU memory
DO UNTIL stable set of clusters is found DO
   Refine statistics for tentative set of clusters
   Report statistics for current clusters
   Modify cluster set by splitting and merging
END DO
Save final cluster statistics in disk file
If requested, save class-map pseudoimage on disk
END
```

3.1.2 CLINIT--READ PARAMETERS AND INITIALIZE TABLES

```
BEGIN
Get all parameters or their defaults from TAE parameter-
  passing interface
Compute test thresholds for skew, kurtosis, and likeli-
  hood ratios
Initialize cluster tree tables
IF decision log file specified
THEN
   Copy parameters to log file
END IF
END
```

3.1.3 CLLDPX--LOAD IMAGE PIXELS INTO ARU MEMORY

```
BEGIN
Open image files and determine image size
Create sampling table specifying pixels to be taken from
  each line of input image
Extract image pixels and load into ARU memory
END
```

3.1.4 CLSTAT--DETERMINE STATISTICS FOR TRIAL SET OF CLUSTERS

```
BEGIN
Convert estimated mean values for each cluster into scaled
  format for MPP
Load estimated covariances into MPP stager memory
IF class-map pseudoimages are to be displayed
THEN
  Load color triplets into display lookup tables
END IF
Refine statistics using MPP
Extract updated cluster statistics, additional statistics
  needed for split/merge decisions, and class-map pseudo-
  image from MPP stager memory
END
```

3.1.5 MPPSTT--USE MPP TO REFINE CLUSTER STATISTICS

```
BEGIN
IF only one cluster
THEN
   Set cluster weight and all relative probabilities to 1
   Compute mean value vector
   Compute covariance matrix
ELSE
   DO UNTIL cluster means become stable
      Invert covariance matrices
      Compute relative probabilities of each sample relative
        to each cluster, using current estimates of means and
        covariances
      IF cluster maps are to be displayed
      THEN
         Assign each pixel to its most likely cluster
         Send cluster map to IIS display
      END IF
      Update mean value vectors using new weights
      Update covariance matrices using new means
      Record change in means in convergence-report table
   END DO
END IF
Load covariance matrices into stager memory for subsequent
  retrieval by VAX
Invert covariance matrices
Load inverse covariance matrices into stager memory
Compute statistics needed for split/merge decisions
Load decision statistics into stager memory
Assign each pixel to its most likely cluster
Load cluster map into stager memory
END
```

3.1.6 GETPROBS--GET RELATIVE PROBABILITIES FOR EACH SAMPLE

```
BEGIN
Reformat inverse covariances using hexadecimation
DO FOR each cluster
   Compute probability density functions for each pixel
     relative to each cluster using current cluster means
     inverse covariances, and a priori cluster weights
   Compute relative probability for each PE, using partial
     sums of densities over cluster subsets
END DO
DO UNTIL stable a priori probabilities are obtained
   DO FOR each cluster
      Sum relative probabilities over all samples
      Sum differences between relative probabilities and
        a priori probabilities over those samples for which
        difference is positive
      Update estimates of relative probabilities
   END DO
   Compute new a priori probabilities
   Enter maximum improvement into convergence report
END DO
Compute normalization factors for means, etc.
END
```

3.1.7 DECSTAT--COMPILE SPLIT/MERGE DECISION STATISTICS

```
BEGIN
Hexadecimate inverse covariance matrices to facilitate rapid
  extraction via corner point module
DO FOR eacn cluster that does not have subclusters
   Compute contraction of two components of skew/kurtosis
     tensors using inverse covariances as metric
   DO FOR each channel
      Compute corresponding component of trace of skew tensor
      Compute corresponding row of trace of kurtosis tensor
   END DO
END DO
DO FOR each cluster that does have subclusters
   Compute subcluster likelihood ratio
   Compute sum of squares of subcluster probability differ-
     ence ratios
END DO
Normalize tensor components for each cluster by dividing by
  sum of relative probabilities
END
```

3.1.8 CLRPRT--REPORT PROGRESS AND FINAL CLUSTERING RESULTS

```
BEGIN
IF final clusters not yet attained
THEN
   Report results of last statistics-refinement phase
ELSE
   Write final cluster statistics to statistics file
   Write class-map pseudoimage to disk, if wanted
   IF decision log being recorded and final decision
     iteration altered clusters
   THEN
     Record final statistics in decision log file
   END IF
END IF
END
```

3.1.9 CLAJST--ADJUST CLUSTERS BY SPLITTING AND MERGING

```
BEGIN
Scan cluster tree to identify clusters that should be
  tentatively split or merged and confirm or reject previous
  tentative split/merge decisions
Squeeze out unused cluster tree nodes
Compute trial parent means and covariances for pairs of
  clusters being tentatively merged
Compute trial subcluster means and covariances for clusters
  being tentatively split
END
```

3.1.10 SPLIT1--SPLIT ONE CLUSTER INTO TWO SUBCLUSTERS

```
BEGIN
Rotate skew and kurtosis to frame with unit covariance matrix
  and diagonal kurtosis
Make first guess at subcluster statistics
IF steepest descent refinement requested
THEN
   DO FOR specified number of iterations
      Compute derivatives of objective function
      Adjust relative means, covariances, and weights
      Adjust step size
   END DO
END IF
Rotate trial statistics back to image frame
END
```

3.2 PROCEDURE HIERARCHY

The overall hierarchy of procedures within these modules is shown in Table 3-1, which also briefly describes the function of each.

3.3 SUPPORT ROUTINES UNIQUE TO CLASSY

In addition to the routines shown in Table 3-1, CLASSY uses a number of routines whose functions are primarily to support tne main algorithmic procedures. Many of these routines are general-purpose VAX, TAE, or MPP procedures. Some additional support routines have, however, been written specifically to support CLASSY. These routines can be divided into three groups: VAX cluster tree manipulations, VAX arithmetic routines, and MPP array arithmetic procedures.

3.3.1 CLUSTER TREE MANIPULATION ROUTINES

Whenever the tentative set of clusters is to be altered by splitting one cluster into subclusters, merging two or more clusters, or eliminating trial clusters, the cluster tree tables must be updated and relinked to reflect the change. These activities are facilitated by the following procedures and functions, all of which are called only by module CLAJST.

Name	Function
freenode	Procedure to free a cluster tree node and make it available for reuse
getnode	Function to find first free node
getcolor	Procedure to assign color codes to newly created node(s)
subweights	Function to sum the weights of all subclusters of a particular cluster

3.3.2 VAX ARITHMETIC PROCEDURES

Parallel Pascal permits performing arithmetic operations between arrays without explicitly specifying loops over each element. However, these automatic array operations assume

Table 3-1. CLASSY Procedure Hierarchy (1 of 2)

Procedure	Function
classy	Main program, in module CLASSY
clinitmain	Main procedure for module CLINT
getparm	Get parameters via TAE interface
setthr	Compute thresholds for split/merge tests
inittree	Initialize cluster tree tables
initlog	Initialize processing log file
clldpxmain	Main procedure for module CLLDPX
imgopen	Open input image files and get sizes
offsets	Compute image-sampling offsets
getsamp	Load image samples into ARU memory
clstatmain	Main procedure for module CLSTAT
setsubdim	Set dimension of ARU subarrays for covariance matrices, etc.
scalemeans	Convert mean values to scaled integers
ldcovar	Load covariance matrices into ARU
setlut	Set color lookup tables for display
mppsttmain	Main procedure for module MPPSTT
getmeans	Compute means for all clusters
getcovar	Compute covariances for all clusters
invcovar	Invert covariance matrices
gpmain	Main procedure for module GETPROBS
rpestim	Estimate sample relative probabilities from means, covariances, and weight
rpadjust	Adjust estimated probabilities to maximize likelihood function
makemap	Generate class-map image
dcmain	Main procedure for module DECSTAT
skewkurt	Compute skew and kurtosis arrays
simmeas	Compute cluster similarity measures
likerat	Compute subcluster likelihood ratios
probdiff	Compute probability difference function
floatmeans	Convert scaled means back to real
getarustats	Get arrays returned from ARU
clrprtmain	Main procedure for module CLRPRT
wrtheader	Write header for processing log report
wrtreport	Report one processing iteration
wrtstats	Write final cluster statistics to file
wrtimage	Write class-map image to file

Table 3-1. CLASSY Procedure Hierarchy (2 of 2)

Procedure	Function
clajstmain	Main procedure for module CLAJST
scantree	Scan cluster tree for clusters to be split, merged, or eliminated
elim	Eliminate very small cluster
trysplit	Test cluster for tentative splitting
separate	Eliminate parent and retain subclusters
sublim	Eliminate subclusters and retain parent
trymerge	Test clusters for tentative merger
packtree	Squeeze out unused cluster tree nodes
domerge	Create parent for trial merger
dosplit	Create subclusters for trial splitting
splitlmain	Main procedure for module SPLIT1
firstguess	Estimate statistics of new subclusters
descend	Refine estimates using steepest descent
derivs	Compute derivatives needed for descent
bldpair	Finish setting up subcluster statistics

that each array contains meaningful data in all its elements. Because CLASSY will frequently be executed using many fewer than the maximum number of image channels, and because a number of matrix operations take time proportional to the square or cube of the number of channels, special routines have been written that use explicit loops over the actual number of channels.

The initial implementation of Parallel Pascal does not provide functions for finding the maximum and minimum of two scalars. Because these functions are needed for a number of computations, temporary versions have been written for CLASSY. The resulting VAX arithmetic functions and procedures are listed below.

Name	Called By	Function
matprod	CLAJST, SPLIT1	Procedure to form product of two matrices
dotprod	CLAJST	Function to compute scalar (dot) product of two vectors, using a matrix as metric tensor
mvprod	SPLIT1	Procedure to left multiply a vector by a matrix
anticom	SPLIT1	Procedure to compute anticommutator of two matrices
realmax	SPLIT1	Function returning maximum of real scalars
realmin	SPLIT1	Function returning minimum of real scalars

3.3.3 MPP ARRAY ARITHMETIC PROCEDURES

To maximize the number of clusters and image channels that can be supported, CLASSY must allocate as few ARU memory planes as possible for temporary storage. The Parallel Pascal code generator, however, allocates planes for all parallel arrays declared within a procedure at the time that the procedure is entered, and it does not deallocate them until the procedure is exited. It is therefore awkward to

reuse the same set of planes for different sets of intermediate results if the results are of different data types. For this reason, a number of very simple computations have been broken out as separate procedures or functions, so that planes needed for intermediate results can be immediately deallocated upon completion of the procedure. These procedures are listed below.

Name	Called By	Function
blknorm	MPPSTT, DECSTAT	Normalize a block of real values in ARU
zp	MPPSTT, DECSTAT	Multiply one component of each image sample by relative probability values for one distribution
zpz	MPPSTT	Multiply results of zp by another image component
pr	MPPSTT, GETPROBS	Multiply a real array by relative probability values for one distribution
zz	GETPROBS, DECSTAT	Multiply one component of each sample by another component
zmb	DECSTAT	Multiply real array by one image component and initiate block summation over all processing elements (PE)s

3.4 GENERAL-PURPOSE VAX SUPPORT ROUTINES USED BY CLASSY

In addition to the procedures and functions written specifically for CLASSY, a large number of standard support routines are used by both VAX and MPP modules. The general-purpose VAX procedures used by CLASSY can be divided into the following groups:

- Parallel Pascal host call interface
- ARU-VAX data transfer procedures
- TAE parameter-passing services
- TAE image I/O routines
- Miscellaneous TAE services
- VAX/VMS system services

- IIS image display procedures
- IMSL mathematical function library

The routines in each group that are used by CLASSY are described briefly in the following subsections. They are given the names by which they are known to VAX/VMS. The actual calls from CLASSY append "$h" to these names to use the HOSTCALL facilities.

3.4.1 PARALLEL PASCAL HOST CALL INTERFACE

The Parallel Pascal code generator does not follow standard VAX conventions for calling subroutines. As a result, all calls to standard VAX support software must be preprocessed to convert arguments to the format expected by the called module.

This conversion is done by a standard module. The Macro source of this module contains a specification line for each called subroutine, giving its standard VAX library name, the name by which it is called from the Parallel Pascal module, and the type of each argument. The VAX Macro Assembler, using a macro definition contained in the source module, builds a compact argument-descriptor table for each subroutine. At run time, these argument descriptors are used to carry out the argument reformatting. This module is summarized below.

Name	Function
HOSTCALL	Invoke host computer (VAX) functions and procedures, written in standard VAX/VMS languages, after first reformatting arguments to conform to the VAX procedure-calling standard

3.4.2 ARU-VAX DATA TRANSFER PROCEDURES

In general, CLASSY transfers data between VAX memory and ARU array memory by using the MPP stager memory as an intermediate buffer that can be loaded by commands from either the

VAX-resident code or the main control unit (MCU)-resident code. The transfer is achieved using the VAX modules listed on the following page.

Name	Called By	Function
IDENTBUFF	CLASSY	Initialize DR780 channel driver and define VAX buffer area for data transfers
OPENVCU	CLSTAT	Open virtual channel through MPP stager
LDARU	CLLDPX, CLSTAT	Load ARU memory from VAX array
RDSTAGER	CLSTAT	Read data, placed in stager by MPP, into VAX memory
CLOSVCU	CLSTAT	Close virtual channel through MPP stager

3.4.3 TAE PARAMETER PASSING SERVICES

CLASSY module CLINIT uses standard TAE facilities to obtain user parameters. The routines called are listed below.

Name	Function
XRINIM	Receive parameter block from TAE
XRFILE	Get file name parameter
XRINTG	Get integer parameter
XRREAL	Get real parameter
XRSTR	Get character-string parameter
XRATTR	Get parameter attributes

3.4.4 TAE IMAGE I/O ROUTINES

CLASSY reads and writes images using standard TAE image I/O routines. To permit reading images in standard LAS image format also, in which the header record containing image size data is located in a separate file, the standard TAE routine for opening an input image (XIOPIN) has been replaced by a modified version (LIOPIN). LIOPIN automatically recognizes images in LAS format and extracts the required

size values from the header file. The routines used to read and write images are listed below.

Name	Called By	Function
LIOPIN	CLLDPX	Open TAE or LAS format image file for input
XIOPOU	CLRPRT	Open image file for output in standard TAE format
XIREAD	CLLDPX	Initiate reading of one block of imagery from disk file into VAX memory
XIWRIT	CLRPRT	Initiate writing of one block of imagery from VAX memory to disk file
XIWAIT	CLLDPX, CLRPRT	Wait for completion of previous image-file read or write
XICLSE	CLLDPX, CLRPRT	Close image file on completion of data transfer

3.4.5 MISCELLANEOUS TAE SERVICES

CLASSY uses one additional TAE service to report unrecoverable I/O errors and terminate processing. This service is summarized below.

Name	Called By	Function
XZEXIT	CLLDPX, CLRPRT	Terminate processing with status report

3.4.6 VAX/VMS SYSTEM SERVICES

CLASSY uses one standard VMS service to associate a Parallel Pascal text file with a standard VAX file descriptor. This service is summarized below.

Name	Called By	Function
$CRELOG	CLINIT, CLRPRT	Assign file name to logical device

3.4.7 IIS IMAGE DISPLAY SERVICES

If the user chooses to monitor the progress of the clustering algorithm by displaying class-map pseudoimages on the IIS image display, the lookup tables used to assign a color

triplet to each cluster must be updated each time clusters are added or deleted. The actual transfer of tables to the IIS is done using the routine summarized below.

Name	Called By	Function
LOADLUT	CLSTAT	Load IIS image display lookup tables

3.4.8 IMSL MATHEMATICAL FUNCTION LIBRARY

CLASSY uses one function from the IMSL library of mathematical and scientific subroutines, as summarized below.

Name	Called By	Function
EIGRS	SPLIT1	Determine eigenvalues and eigenvectors of a real, symmetric matrix

3.5 GENERAL-PURPOSE MPP SUPPORT ROUTINES USED BY CLASSY

The general-purpose MPP procedures used by CLASSY can be divided into the following groups:

- MPP mathematical functions
- Data routing within ARU subarrays
- Data transfers between ARU and MCU
- ARU external data transfers

3.5.1 MPP MATHEMATICAL FUNCTIONS

In addition to the intrinsic Parallel Pascal mathematical functions, CLASSY uses several additional procedures to carry out standard mathematical functions, as listed below.

Name	Called By	Function
blksum/ bsmopn/ bsmcls	MPPSTT, GETPROBS, DECSTAT	Sum values over all PEs, using partial summation by blocks
bsmnum	MPPSTT, DECSTAT	Generate, for each PE, the serial number of the blocked sum stored at that PE
insert	MPPSTT	Insert a set of contiguous ARU planes from one array into another array

Name	Called By	Function
minvpd	MPPSTT	Invert positive definite matrices
scale	MPPSTT, GETPROBS, DECSTAT	Scale integer or cardinal arrays
scalrr	MPPSTT, GETPROBS, DECSTAT	Scale real arrays by incrementing exponent

3.5.2 DATA ROUTING WITHIN ARU SUBARRAYS

CLASSY packs covariance matrices for all clusters into a single set of ARU planes by using a different subarray of PEs to contain each matrix. Manipulation of rows and columns within each subarray is facilitated by the standard procedures listed below.

Name	Called By	Function
rchtrn	MPPSTT	Half transpose PE subarrays
rcnum	MPPSTT	Generate subarray row and column numbers at each PE

3.5.3 DATA TRANSFERS BETWEEN ARU AND MCU

CLASSY calls a number of standard MPP procedures to transfer data between the ARU memory planes and arrays in MCU memory, as listed below.

Name	Called By	Function
extblk	MPPSTT, GETPROBS	Extract sums over all PEs that were computed by partial summation over blocks
exthdm	GETPROBS, DECSTAT	Extract values from arrays that have been reformatted using hexadecimation
getlpe	MPPSTT, GETPROBS, DECSTAT	Extract a data value from a single PE, specified by its row and column number
hxdcmt	MPPSTT, GETPROBS	Reformat ARU planes to permit rapid data extraction via the corner point module

Name	Called By	Function
ldsuba	MPPSTT	Load a subarray of PEs from an MCU array

3.5.4 ARU EXTERNAL DATA TRANSFERS

Although the host VAX is given most of the responsibility for setting up and carrying out data transfers between the MPP and other devices, only the MPP-resident modules of CLASSY know when a given ARU array is ready for transmission. Accordingly, standard MPP procedures are needed to initiate external data transfers from the ARU. The procedures used by CLASSY are listed below.

Name	Called By	Function
ldstager	MPPSTT	Load a set of ARU planes into the stager buffer
TBD	MPPSTT	Transfer a set of planes from the ARU to the image memory of the IIS image display

SECTION 4 - INTERNAL DATA STRUCTURES

All arrays and variables used by CLASSY modules are defined briefly in the source code. In most cases, their usage is obvious and will not be described here. The arrays used to record the hierarchy of clusters and subclusters may, however, be somewhat confusing and are therefore discussed.

In addition, because the processing algorithms and data storage philosophy reflect ARU and MCU memory limitations, some notes on MPP memory usage and allocation are included. No attempt has been made to minimize use of VAX memory because Parallel Pascal takes full advantage of the virtual memory facilities of the VAX/VMS operating system.

4.1 CLUSTER TREE TABLES

The repeated splitting and merging of tentative clusters leads to a hierarchical structure of clusters, which can conveniently be represented as a tree. A fictitious root node, node 0, is the parent of one or more actual clusters, which in turn may serve as the parent node(s) for additional clusters.

Whenever a cluster is tentatively split, two new nodes are created for the subclusters. Both new nodes point back to the parent cluster. The parent contains a pointer to one of the new nodes, which in turn contains a link pointer to the second new node.

When two clusters are found to be so similar that they are to be tentatively merged, a new node is created for the merged cluster. This is inserted in the cluster tree hierarchy at the position occupied by one of the two clusters and contains a pointer to it as a subcluster. The node containing the other cluster, together with any subclusters, is relinked as a second subcluster of the new cluster.

After the MPP processing has refined the estimated statistics and computed the likelihood ratios and average probability differences between each parent cluster and its set of subclusters, CLASSY may decide to delete either the parent cluster or its subclusters. The corresponding cluster nodes are then unlinked from the cluster tree and made available for reuse.

To simplify scanning of the tree tables by the MPP modules, the cluster tree is packed after each iteration of the decision phase, so that all free nodes occur after the last node currently in use.

The tree structure is recorded in a set of arrays, each of which has one entry per node, as listed below.

Name	Contents
Parent	Number of the node containing parent for this node; 0 denotes the root node, -1 indicates node is not in use.
Sub	Number of node containing leftmost subcluster.
Link	Number of node containing closest sibling on right; 0 indicates this is rightmost sibling. A negative value preserves the previous link of a node that has been deleted but whose siblings may not yet have been fully processed by procedure scantree in module CLAJST.
Oldnode	Number of node to which cluster was assigned before last packing of node list. This is needed because means, covariance, skew, and kurtosis data are not moved to new locations when the nodes are packed.
Remap	New node number corresponding to old node in class-map pseudoimage array.
Serial	Serial number assigned when cluster was first tentatively created. Not altered by node packing; used as reference number in decision log reports of split/merge decisions and current cluster statistics.

Name	Contents
Creation	Iteration of main processing loop during which cluster was first tentatively created. This array is not referenced in the current version of CLASSY but may prove useful if code is added to reduce repeated creation and deletion of subclusters of a given cluster.
Color	Image display color triplet assigned to cluster.
Palette	Color usage record; set to 1 if color is assigned to an active node; otherwise, set to 0.

4.2 MPP MAIN CONTROL MEMORY USAGE

CLASSY uses approximately 45,000 of the available 65,000 bytes of memory in the MPP Main Control Unit. Most of this memory is occupied by program code.

The inverse covariance matrices, which would occupy nearly 30,000 bytes for the maximum allowed number of channels and classes, are not stored in MCU memory. Instead, they are kept in a set of ARU memory planes. When needed as scalar values in computations, values for one cluster at a time are transferred into a local array in MCU memory.

to reduce ARU computation time, two types of data are stored in MCU memory as scaled integers rather than floating point values. These arrays are listed below.

Name	Contents
Means, Oldmeans	Mean values for each cluster; 16-bit integer, consisting of the sign bit (always 0), an 8-bit integer part, and a 7-bit fraction
Oldwghts, Newwghts	A priori cluster weights; 12-bit cardinal, normalized so that the maximum value, 4095, represents 1.0

4.3 ARRAY UNIT DATA STRUCTURES

Because CLASSY needs to reference image values for a large number of channels and relative probabilities for a large number of clusters repeatedly, data are packed very tightly

into the ARU memory planes and the planes must be reused as much as possible. Array space is permanently allocated for three sets of data; namely, raw image values, cluster relative probabilities, and inverse covariance matrices. The remaining planes are reused repeatedly for different data in different modules and procedures. In some cases, it has been necessary to restructure a computation slightly or to break out two or three lines of code as a separate procedure or function, to avoid having too many temporary arrays allocated at one time. ARU memory is allocated as summarized below.

Planes	Array Name	Contents
0	-	Reserved for system use
1-168	Image	Image data, 21 channels of 8 bits each
169-556	Relprob	Relative cluster probabilities, scaled 12-bit cardinals (representing values between 0.0 and 1.0) for 32 clusters
557-588	Matrix	Inverse covariance matrices; each matrix occupies a different subarray of PEs
589-876	(Various)	Temporary storage explicitly allocated by CLASSY modules
877-974	-	Temporary storage automatically allocated by Parallel Pascal code generator and computational primitives
975-1023	-	Reserved for system use

SECTION 5 - NOTES ON MATHEMATICAL ALGORITHMS

The MPP implementation of CLASSY uses basically the same mathematical procedures as the original Lennington and Rassbach version (References 1 and 2). To take advantage of the parallel structure of the MPP and work within the available number of ARU memory planes, however, some of the mathematical formulas have been restructured. The versions of the algorithms used for the MPP implementation are outlined in the following subsections.

5.1 MULTIVARIATE NORMAL CLUSTER STATISTICS

The CLASSY algorithm assumes that a set of multichannel image measurements can be meaningfully grouped into an initially unknown number of clusters. For each cluster, the probability density function is a multivariate normal distribution, defined by a mean value vector, a covariance matrix, and an a priori cluster probability. The distributions for different clusters may overlap, in which case, measured image samples that lie in the overlap region are assigned partly to each of the overlapping clusters in proportion to the values of the probability density function and the a priori probability for each cluster.

The statistics refinement phase of CLASSY assumes a particular set of clusters and initial estimates of their means, covariances, and a priori probabilities. The initial estimates are updated by a series of iterative corrections until a reasonable approximation to a (local) maximum of the likelihood function is achieved.

The statistics that define each multivariate-normal distribution are defined in Figures 5-1 and 5-2. The set of measurements for d spectral channels at each of N image samples are represented by a set of N sample vectors, as defined by Equation (1-1) in Figure 5-1. The probability density for

A set of m multivariate normal distributions are to be fit to N sets of measurements, each of which consists of d channels. Let the measured values for sample s be represented by the vector

$$\mathbf{X}_s = (x_{s1}, x_{s2}, \ldots, x_{sd})^T \tag{1-1}$$

For a given set of distributions described by the mean vectors $\boldsymbol{\mu}_1$ through $\boldsymbol{\mu}_m$ and the covariance matrixes $\boldsymbol{\Sigma}_1$ through $\boldsymbol{\Sigma}_m$, the probability density for $\mathbf{X}_s$ relative to distribution i is

$$P_{is} = (2\pi)^{-d/2} (\det \boldsymbol{\Sigma}_i)^{-1/2} e^{-(\mathbf{Z}_{is}{}^T \boldsymbol{\Sigma}_i{}^{-1} \mathbf{Z}_{is})/2} \tag{1-2}$$

where the displacement of sample s from the distribution is given by the vector

$$\mathbf{Z}_{is} = \mathbf{X}_s - \boldsymbol{\mu}_i \tag{1-3}$$

The relative probability that sample s belongs to the distribution described by ($\boldsymbol{\mu}_i$, $\boldsymbol{\Sigma}_i$) rather than to one of the other distributions is given by

$$P_{is} = a_{is}\, \rho_{is}/\rho_s \tag{1-4}$$

where values a_1 through a_m represent the *a priori* probabilities for distribution 1 through m, respectively, and the average probability density for sample s is given by

$$\rho_s = \sum_{i=1}^{m} a_i\, \rho_{is} \tag{1-5}$$

5909157-1/84

Figure 5-1. Statistics for a Single Multichannel Measurement

The set of distributions that maximize the likelihood function over the N sets of measurements obey the relationships

$$a_i = (1/N) \sum_{s=1}^{N} P_{is}, \quad i = 1, 2, \ldots, m \tag{2-1}$$

$$\boldsymbol{\mu}_i = (1/Na_i) \sum_{s=1}^{N} P_{is}\mathbf{X}_s, \quad i = 1, 2, \ldots, m \tag{2-2}$$

$$\boldsymbol{\Sigma}_i = (1/Na_i) \sum_{s=1}^{N} P_{is}\mathbf{Z}_{is}\mathbf{Z}_{is}^T, \quad i = 1, 2, \ldots, m \tag{2-3}$$

If the set of distributions is only an approximation to the maximum likelihood case, the approximations can be iteratively refined by repeated evaluation of Equations (1-2), (1-4), and (2-1) through (2-3).

For nonoverlapping distributions, convergence is fairly rapid. In particular, Equation (2-1) immediately gives the new value of a_i for a new set of $\boldsymbol{\mu}_i$ and $\boldsymbol{\Sigma}_i$. For overlapping distributions, on the other hand, Equation (2-1) gives very slow convergence. Lennington and Rassbach (Reference 1, equation 28) introduced a modified form of Equation (2-1) that accelerates convergence when overlap is present. Their formula for the updated estimate a_i' in terms of the old value a_i can be written in the form

$$a_i' = a_i \frac{\sum_{P_{is} > a_i} (P_{is} - a_i)}{(1 - a_i)[a_i N - \sum_{s=1}^{N} P_{is}] + \sum_{P_{is} > a_i} (P_{is} - a_i)} \tag{2-4}$$

5090(57*) 84

Figure 5-2. Clusters Statistics

cluster i at sample s is calculated using Equations (1-2) and (1-3). The relative cluster probability, which represents the relative probability that sample s belongs to cluster i rather than one of the other clusters, is then calculated using Equations (1-4) and (1-5).

The relative cluster probabilities computed for all samples are then used to compute new estimates of the mean value vector and covariance matrix for each of the m clusters using Equations (2-2) and (2-3) in Figure 5-2.

Although improved estimates for the a priori cluster probabilities could be obtained using Equation (2-1), each iteration of the calculation gives only a small correction to the estimate when there is substantial overlap between clusters. Lennington and Rassbach (Reference 1) showed that convergence can be significantly speeded using a modified expression for the a priori cluster probability. This expression, rewritten to facilitate evaluation using the MPP, is given by Equation (2-4).

The true, continuously varying image is approximated by a set of integer-valued samples, each of whose components can take on at most 256 different values. There is thus a danger that a very compact cluster might collapse into a single point in feature space; i.e., a cluster all of whose members have identical values. To avoid this, a small positive constant, defined by the user parameter SPREAD, is added to each diagonal element of each covariance matrix before the inverse covariance matrices are computed.

5.2 MPP STATISTICS FOR SPLIT/MERGE DECISIONS

After the means, covariances, and a priori probabilities of a trial set of clusters have converged to a stable set of values, the MPP is used to compile additional statistics. These statistics help identify which clusters should be

tentatively split and confirm or reject previous split and merge decisions; they are defined in Figure 5-3.

If a given cluster has no subclusters, it is a potential candidate for splitting. The decision whether to split the cluster, and if so, how, uses the traces of the skew and kurtosis tensors, given by Equations (3-1) and (3-2) in Figure 5-3. If a cluster has already been tentatively split into subclusters, the likelihood ratio between the parent and the combination of its subclusters (given by Equation (3-3)) and a measure of similarity between the parent and its subclusters (given by Equation (3-4)) must be compiled. Both of these equations have been rewritten from the form used by Lennington and Rassback to (1) make them depend on the relative cluster probabilities of eacn sample and the a priori probabilities of the clusters, rather than on the probability density values, and (2) use subcluster probabilities relative to all clusters, rather than just relative to the parent. The bias term in Equation (2-3) may be specified by user parameter LBIAS.

5.3 TEST FOR CLUSTER SIMILARITY

During a typical processing run, the first few iterations of CLASSY's main loop serve to split the data into a large number of trial clusters. As the result of successive refinements of cluster parameters, however, it is quite likely that some clusters will become very similar to other clusters, even though they were originally split off from different parent clusters. When two such clusters have become sufficiently similar, they should be merged into a single cluster.

Rather than using the MPP to compute sample-by-sample similarity measures between all possible pairs of clusters, CLASSY uses a crude similarity test, based solely on overall cluster statistics, to identify pairs of clusters that are

Measures of the deviation of the samples in a cluster from the multivariate normal distribution are derived from two statistics, the trace of the skew tensor,

$$\mathbf{S}_i = (1/Na_i) \sum_{s=1}^{N} \mathbf{Z}_{is} (\mathbf{Z}_{is}{}^T \boldsymbol{\Sigma}_i{}^{-1} \mathbf{Z}_{is}) P_{is} \tag{3-1}$$

and the trace of the kurtosis tensor,

$$\mathbf{K}_i = (1/Na_i) \sum_{s=1}^{N} \mathbf{Z}_{is}\mathbf{Z}_{is}{}^T (\mathbf{Z}_{is}{}^T \boldsymbol{\Sigma}_i{}^{-1} \mathbf{Z}_{is}) P_{is} \tag{3-2}$$

The logarithm of the likelihood ratio between a parent cluster i and the set of its m_i subclusters is given by

$$\ln \Lambda_i = -(m_i - 1)(2d + b) + \sum_{s=1}^{N} \ln\{(\sum_{j=1}^{m_i} P_{i_j s})/P_{is}\} \tag{3-3}$$

where b is a bias term, on the order of 1.

A sample-by-sample measure of similarity between a parent cluster and its subclusters is

$$E_i = (1/N) \sum_{s=1}^{N} \{(P_{is}{}' - P_{is})/(P_{is}{}' + P_{is})\}^2 \tag{3-4}$$

where the combined probability of the subclusters, normalized to the same *a priori* probability as the parent, is

$$P_{is}{}' = \{a_i/ \sum_{j=1}^{m_i} a_{i_j}\} \sum_{j=1}^{m_i} P_{i_j s} \tag{3-5}$$

5090157*1/84

Figure 5-3. Statistics To Support Split and Merge Decisions

candidates for merger. The similarity measure used is given in Equation (4-1) in Figure 5-4.

5.4 ESTIMATION OF STATISTICS FOR NEWLY CREATED CLUSTERS

When two clusters have been adjudged sufficiently similar to warrant a trial merger, the initial statistics for their new parent cluster can be readily computed from their means and covariances using Equations (4-2) and (4-3). Finding trial statistics for the new subclusters of a cluster that is to be tentatively split, on the other hand, is quite difficult. As Lennington and Rassbach point out (References 1 and 2), the number of equations relating the subcluster means, covariances, and a priori weights to the parent means, covariances, and traces of skew and kurtosis tensors is one less than the number of statistics components to be determined, so an exact solution is not possible. Furthermore, the available equations are not linear, so even approximate methods of solution tend to become complex.

The MPP version of CLASSY uses the same procedure as the Lennington and Rassbach version to generate trial subcluster statistics. The skew and kurtosis tensors are first rotated to a coordinate frame in which the covariance matrix is identical to the unit matrix and the trace of the kurtosis tensor is diagonal. The difference vector between the two subcluster means is then estimated by assuming that it is along the (rotated) coordinate axis for which the kurtosis deviates most markedly from the expected value for a multivariate-normal distribution. The difference between the covariance matrices for the two subclusters is also estimated. These arrays are then rotated back to the coordinate frame of the image and used to compute the trial subcluster statistics.

As an option, the user may attempt to refine these initial statistics estimates using the method of steepest descent.

An alternative measure of similarity between clusters i and j, based solely on the overall cluster statistics rather than individual sample values, is given by

$$S_{ij} = \{ (\mu_i - \mu_j)^T \frac{a_i\Sigma_i^{-1} + a_j\Sigma_j^{-1}}{a_i + a_j} (\mu_i - \mu_j)$$

$$+ A \sum_{k=1}^{d} [\ln (\Sigma_i)_{kk} - \ln(\Sigma_j)_{kk}]^2 \} / \{ 1 + B [\frac{a_i}{a_j} - \frac{a_j}{a_i}]^2 \} \quad (4\text{-}1)$$

where d again denotes the number of channels.

The means and covariances for a cluster k formed by merging clusters i and j are

$$\mu_k = \frac{a_i}{a_i + a_j} \mu_i + \frac{a_j}{a_i + a_j} \mu_j \quad (4-2)$$

and

$$\Sigma_k = \frac{a_i}{a_i + a_j} \Sigma_i + \frac{a_j}{a_i + a_j} \Sigma_j + \frac{a_i a_j}{(a_i + a_j)^2} (\mu_i - \mu_j)(\mu_i - \mu_j)^T \quad (4-3)$$

5090157-1/84

Figure 5-4. Formulas for Trial Merger of Clusters

This procedure, which is performed on the VAX, creates an objective function by taking a weighted sum of differences between the actual covariance, skew, and kurtosis components of the parent cluster and the components computed using the trial subcluster weights, means, and covariances. The derivatives of this objective function with respect to the subcluster-statistics components provide estimates of the corrections to be applied. The number of iterations of the procedure is specified by user parameter SPLITER.

Experiments with limited amounts of synthetic imagery suggest that 100 or more iterations of the steepest descent procedure are needed to improve significantly on the initial subcluster statistics, and that the ultimate improvement may be rather small. Because the MPP statistics refinement phase is quite fast, it is not clear under what circumstances, if any, it is worthwhile to use the VAX to refine the initial estimates.

APPENDIX - CLASSY PROGRAM MODIFICATIONS

This appendix covers the following topics:

- Status of CLASSY testing and debugging
- Actions to be taken to correct known problems and deficiencies
- Suggestions for enhancements
- Code to be inserted into various CLASSY and support modules to correct problems and/or add enhancements

Questions may be addressed to Dick White, telephone (919) 781-7292 or (919) 781-4963.

A.1 STATUS OF CLASSY TESTING AND DEBUGGING

The entire CLASSY program has been exercised with two different test images, generated by MLVRGN, as follows:

- Single broad multivariate distribution, with three spectral channels
- Two broad, slightly overlapping distributions, with five spectral channels

The first test image yielded acceptable results, except that its kurtosis somewhat exceeded the threshold for trial splitting. It is not clear whether this is because the distribution generated by MLVRGN was not "perfectly random" (due to rounding of image components to integers and/or imperfections in the FORTRAN RAN function) or because the default value of the CONLEVEL parameter is unrealistic.

The second test image underwent several iterations without problems although one of the two distributions once again yielded kurtosis values above the splitting threshold. Eventually, the program aborted because of an error in the primitive XCHGRP, called by BLKSUM. The identical error

occurred on several consecutive tests but could not be made to occur for standalone tests of XCHGRP, neither on the array unit (ARU) nor on the simulator. This testing was done at the end of January, just before a major failure of the Massively Parallel Processor (MPP).

All parts of the CLASSY code have been exercised except the following:

- CLAJST/trymerge, CLAJST/domerge--Sufficiently similar clusters did not occur during testing. To force occurrence of merging, the default value of parameter MERGETHR should be increased; decreasing BCOEFF may also help.
- Calls to HXDCMT, EXTBLK, and EXTHDM in modules MPPSTT, GETPROBS, and DECSTAT--These calls have been replaced by temporary Parallel Pascal versions of EXTBLK and EXTHDM pending completion of testing of the corresponding primitives.
- Subroutines to load color lookup tables into the Interactive Imaging System (IIS) display in module CLSTAT and to copy class-map pseudoimages from ARU memory to the IIS in module MPPSTT--The required subroutines have yet to be written.

Unit testing was also done for all MPP primitives except HXDCMT, EXTBLK, and EXTHDM. Because of repeated hardware problems, systematic testing of all combinations of options and a full range of data types and lengths was not completed. Other programs using these primitives may encounter errors; as far as is currently known, the primitives work correctly for the options and data types required by CLASSY.

HXDCMT has been checked out for reformatting 16 planes. Because very little additional code is used to handle more

than 16 planes, further testing of HXDCMT should be simple and is unlikely to uncover major problems.

EXTBLK has encountered problems in retrieving valid data via the "Read Corner Points" instruction; these problems seem to be in module RCORN.PCU. Introducing a "no op" instruction solved part of the problem, suggesting that more than one ARU cycle is needed for the corner point values to be propagated from the boards containing the corner-point processing elements (PEs). The remaining problem is thought to occur because, unlike other instructions that use fields 3 and 4, the RCORNER instruction acts on the current contents of field 3 regardless of the value of the W bit in the common register. A solution is suggested in Section A.4.1.

Testing of EXTHDM was not begun. This is a rather messy routine, using a large number of registers and for temporary pointers, so problems are likely.

A.2 KNOWN PROBLEMS AND DEFICIENCIES

The following actions should be taken to correct known problems and deficiencies:

- XCHGRP--When exchanging adjacent groups of rows or columns for 32-bit real data arrays, the low-order 2 bits were partly incorrect, whereas all higher order bits were exchanged properly. This suggests that the main loop in XCHGRP.PCU is working correctly. However, even if register contents on entry to the main loop are assumed completely unknown, the contents of the second plane in the results can be partially predicted and do not agree with the observed results. CSC documented the problem and provided procedures to GSFC for breaking at that point in CLASSY where the problem occurs. If it is still present, a check should be made to determine whether the problem depends on the location of the XCHGRP code in PE control memory. Reinstating

a Control and Debug Module (CAD) capability to single step through PECU code would also be helpful.

- EXTBLK--As stated above, there seems to be a problem in RCORN.PCU. A code change is suggested in Section A.4.1.

- Convergence report--For parameter MAXMITER greater than 20, CLRPRT does not correctly print out the convergence behavior for the statistics-refinement phase of CLASSY. A code change is suggested in Section A.4.2.

- IIS display interface--The same MCU code used by program MAXLIK to transfer class-map pseudoimages from ARU memory to the IIS display can be used, with minor modifications to allow for the single 128-by-128 image generated by CLASSY. The module for loading color lookup tables, Loadlut, should be very similar to existing subroutines used by Land Analyses System (LAS) software; a Parallel Pascal callable front end will need to be added.

- Processing log disk format--Because the default field length in Parallel Pascal standard output is very long, the header record for the processing log file, which records the values of user parameters, is badly formatted. Explicit field lengths need to be specified in procedure initlog in module CLINIT.

- Transportable Applications Executive (TAE) Catalog Manager interface--Although the VAX/VMS operating system already provides many of the features that the Catalog Manager added to the PDP-11 RSX environment, all standard LAS programs are designed to work with the Catalog Manager. MPP image applications intended to interface with LAS routines must therefore work with the dual file name facility (TAE names plus non-mnemonic host names) required by Catalog Manager and LAS Label Services routines. Specific changes in CLASSY are the specification of file names as "filename"

rather than "string" type parameters in CLASSY.PDF and CLINIT; the use of XRFILE instead of XRSTR in CLINIT to get the parameters from TAE; passing host names and lengths from CLINIT to CLLDPX and CLRPRT; and reading LAS-format image files in CLLDPX. A code change is suggested in Section A.4.3.

- LAS-format images--CLASSY needs to be able to read images in the standard format produced by the LAS Image Input/Output (I/O) package. This format records header information in a Data Descriptor Record (DDR) located in a different file from the image. The standard TAE XI file I/O routines, on the other hand, look for header data in the first record of the image file. Routine LIOPIN.FOR has been written, but not tested, to determine automatically whether an input image is in TAE or LAS format; in the latter case, it receives required header data from the DDR file. This routine must be used with a corrected version of the TAE routine XIOPIN (as of February 1, 1985, XIOPIN did not properly assign event flags for use by XIREAD and XIWAIT). Writing output images (for CLASSY, only the class-map pseudoimage) in LAS format will be more difficult; the current version of CLASSY will not attempt to do so. Files LIOPIN.FOR and PPXTRN.MAR contain the required external modules.

- Final cluster statistics format--The current report written by procedure wrtstats in module CLRPRT is not compatible with that produced by existing LAS statistics-generating routines. To facilitate future use of these statistics by MAXLIK and similar programs, this format should be changed. Larry Novak or Hampapuram Ramapriyan of GSFC may be able to supply a description of the correct format.

- Image-generator routine--The current version of MLVRGN does not use buffer space efficiently. A better

version (which matches the pseudocode in the program documentation) is provided in file MLVRGN.UPD; this version has not been checked out.

A.3 SUGGESTIONS FOR ENHANCEMENTS

Based on the very limited experience with test data so far, the following enhancements appear desirable:

- Reducing effect of low-probability pixels--The present method of computing the likelihood ratio between a cluster and its subclusters assumes that a 0 value of the relative probability is very much less than any finite value. An actual ratio of either 0 or infinity is replaced by a value (1/1024 or 1024, respectively) whose logarithm has a relatively large absolute value. Because the probabilities are in fact represented as scaled integers between 0 and 4095, it is unlikely that a rounded probability value of 1 or 2 really represents a much stronger cluster membership than a rounded value of 0. It is therefore suggested that 0.5 be added to the rounded relative probability values before taking their ratio. This will avoid heavily biasing the final likelihood values by contributions from pixels that have low relative probabilities with respect to both the parent cluster and its subclusters. A suggested alternate version of procedure likerat in module DECSTAT is given in Section A.4.4.

- Ad hoc adjustment of splitting threshold--The multivariate normal arrays generated by MLVRGN tended to have kurtosis values somewhat above the threshold for trial splitting computed using a 99-percent confidence level for parameter CONLEVEL. This may be because MLVRGN is not producing a truly random distribution. It may, however, be desirable (also when using real imagery) to be able to adjust the splitting threshold arbitrarily either to encourage or discourage splitting without explicit reference to the

statistical models reference by CONLEVEL. A new parameter, SKKMULT, is suggested to follow CONLEVEL immediately in the list of parameters. Procedure setthr in module CLINIT would multiply the three skew/kurtosis thresholds (variables Skwthr, Trkthr, and Urkthr) by SKKMULT; its default value would be 1.0.

- Anti-thrashing logic--There seems to be a danger that a cluster having skew or kurtosis values above the threshold for trial splitting but yielding a subcluster-to-cluster likelihood ratio small enough that the split is rejected will again yield skew or kurtosis values above the threshold. To avoid an endless cycling between trial splitting and split rejection, a record needs to be kept of clusters for which splitting has already been rejected. Because adjustments of the statistics and membership of other clusters may cause some drift in the given cluster, allowance needs to made for retrying the split if major changes occur. The following logic is suggested:

In module CLASSY, define

```
var
   Parstats:  array[1..MAXCLUST,1..2*MAXCHAN] of real;
   Nosplit:  array[1..100,1..2*MAXCHAN] of real;
   Rejspltcnt:  integer; (* Initialized to 0 *)
```

and include these variables in the argument list for CLAJST.

CLAJST/dosplit:

```
IF cluster successfully split
THEN
   Save means and diagonal covariance values of parent
     cluster in Parstats[Node]
END IF
```

```
CLAJST/sublim:

IF Creation[Pnode] < Creation[Sub[Pnode]]
THEN
   Copy means and diagonal covariances from
     Parstats[Oldnode[Pnode]] into next available entry in
     Nosplit
   Increment Rejspltcnt by 1
END IF

CLAJST/trysplit:

IF cluster meets skew/kurtosis tests for splitting
THEN
   Set Divisor = 1.0
   DO FOR each record in Nosplit
      Compute sum of squares of differences between means in
        Nosplit record and means of cluster being examined
      IF difference less than Nchan*Mdiffmax (where Mdiffmax
        is on the order of .01)
      THEN compute sum of squares of logs of ratios of
        diagonal covariance elements
         IF sum less than Nchan*Cdiffmax (where Cdiffmax is
           on the order of .01)
         THEN multiply divisor by Divmult (where Divmult is
           on the order of 2.0)
         END IF
      END IF
   END DO
   Divide skew/kurtosis test ratios (Srat, Trat, and Urat)
     by Divisor
   IF one or more test ratios > 1.0
   THEN set flag for trial split
   END IF
END IF
```

- Increased number of clusters--The easiest way to increase the number of clusters to 64 would be by swapping the relative cluster probabilities for half the clusters out to the stager memory. To fit inverse covariance arrays for 64 clusters into a single set of 32 ARU memory planes, the maximum number of channels would be reduced to 16. This would free up 8*(21-16) = 40 planes currently used for raw image data and thereby provide the needed extra storage for the 32-fold partial sums in module GETPROBS. Except for procedure rpadjust in module GETPROBS, the swapping of data

to and from the stager could be fully overlapped with the computations that use the relative probabilities.

A.4 CODE MODIFICATIONS

A.4.1 RCORN.PCU

```
;  At label P$RCORNM insert after MODPE
+        LOAD,IFO P,FROMNORTH     ; Suppress possible corner-point
                                  ;  load
```

A.4.2 CLRPRT.PP

```
{procedure mppconvrg within procedure wrtreport -- replace by:  }
    (* Local procedure to report MPP statistics convergence
       behavior *)
    procedure mppconvrg(var Device : text);
begin
  writeln(Device);
  writeln(Device, 'Convergence behavior for MPP',
          ' statistics refinement:');
  writeln(Device);
  writeln(Device, 'Iteration   Maximum    Iterations   ',
          'Maximum Weight Change for Iteration');
  writeln(Device, 'of Means    Change     of Weights   ');
  writeln(Device, '  Loop      in Means    Loop (N)    ',
          '   1         2        N-1        N');
  writeln(Device);
  Jstart := 4;
  Kstop := Convrep[1];
  if (Kstop > REPLNG div 6) then
      Kstop := REPLNG div 6;
  for K := 1 to Kstop do
    begin
      Temp := Convrep[Jstart-2] / 128.0;
      if K < Kstop then
          write(Device, K : 5, Temp : 13:2,
                  Convrep[Jstart-1] : 11, '         ')
      else
          write(Device, Convrep[1] : 5, Temp : 13:2,
                  Convrep[Jstart-1] : 11, '         ');
      for J := Jstart to Jstart + 3 do
        begin
          Temp := Convrep[J] / 4096.0;
          write(Device, Temp : 9:4);
        end;
      writeln(Device);
      Jstart := Jstart + 6
    end;
end;  (* of local procedure mppconvrg *)
```

A.4.3 CLASSY.PDF

```
!  To support Catalog Manager, replace parameter definitions by
PARM  INIMAGES  TYPE=FILE        COUNT=2:21
PARM  STATFILE  TYPE=FILE
PARM  OUTIMAGE  TYPE=FILE        COUNT=0:1
PARM  LOGFILE   TYPE=FILE        DEFAULT=CLASSY.LOG
```

A.4.4 DECSTATS.PP

```
DECSTATS.PP
{Replace procedure likerat with following code:}

procedure likerat;        (* Compute contribution of each sample
                             to ratiobetween likelihood that
                             single best fit to data and
                             likelihood that subclusters are
                             instead best fit *)
(* Arguments:
*    Clust       Number (node) of parent cluster.
*    Psum        Array containing, for each sample, the sum of
*                relative probabilities over all subclusters.
*    Ratio       Array to receive likelihood ratio values.
*
*  Global references:
*   Read only - Relprob
*)
var
  Parprob : MPPINT;    (* Rescaled relative prob. for parent *)
  Subprob : MPPINT;    (* Rescaled sum of relative
                          probabilities for subs *)
begin
  scale1(Psum, Subprob, 0, 13, 18, 0, 0);  (* 30-bit fract. *)
  scale1(Relprob[Clust], Parprob, 0, 12, 1, 0, 0);(* 13-bit *)
  Subprob := Subprob + 131072;  (* Deemphasize samples with *)
  Parprob := Parprob + 1;      (*  low prob. for all clusts. *)
  Subprob := Subprob div Parprob;      (* 17-bit fraction *)
  Ratio := Subprob;
  scalrr(-17, Ratio);
  Ratio := ln(Ratio)

end;  (* of procedure likerat *)
```

REFERENCES

1. R. K. Lennington and M. E. Rassbach, Mathematical Description and Program Documentation for CLASSY, an Adaptive Maximum Likelihood Clustering Method, Lockheed Electronics Company, Inc., April 1979.

2. R. K. Lennington and M. E. Rassbach, CLASSY--An Adaptive Maximum Likelihood Clustering Algorithm, Proceedings of tne Ninth Annual Meeting of the Classification Society, May 1978.