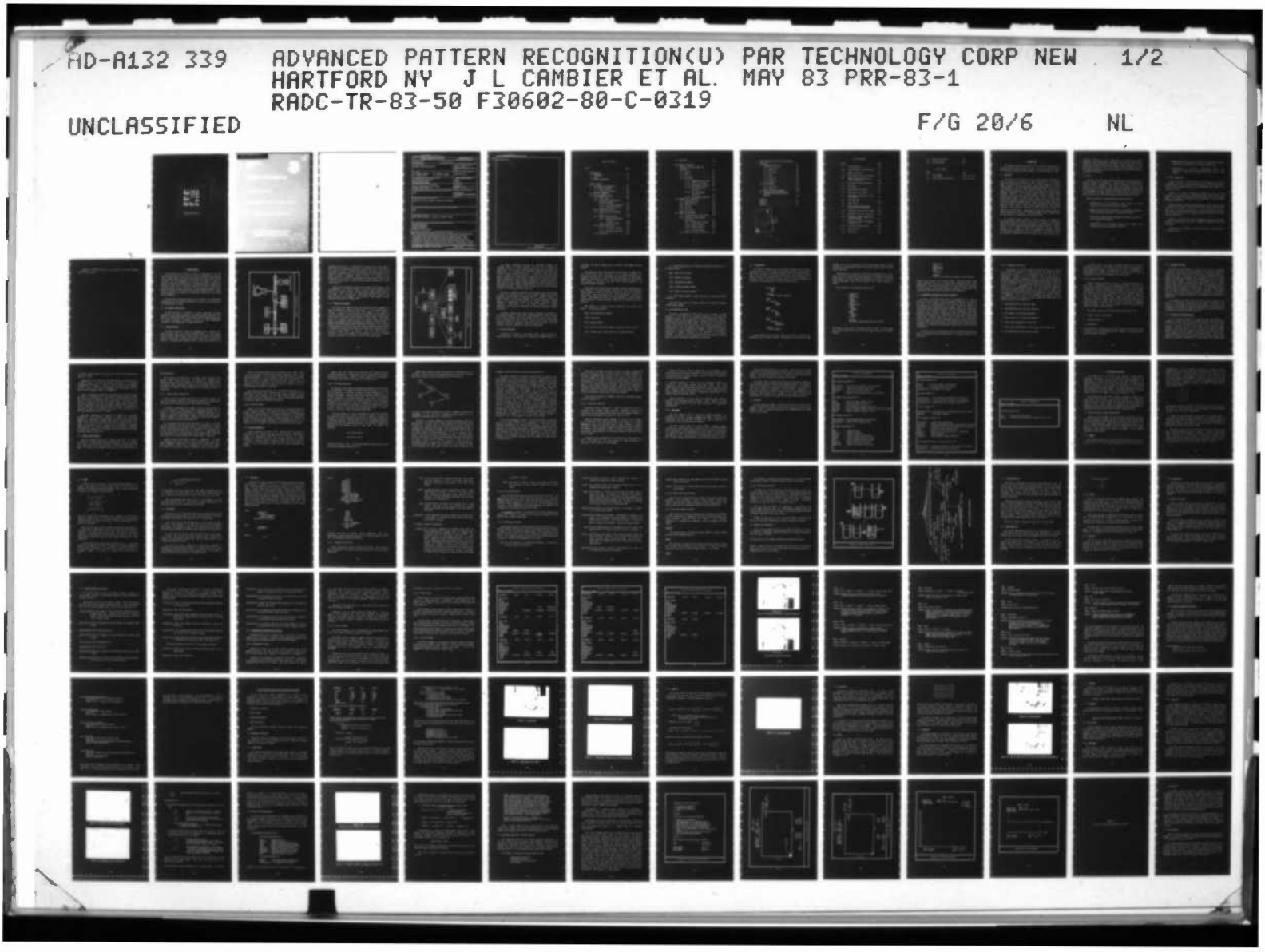
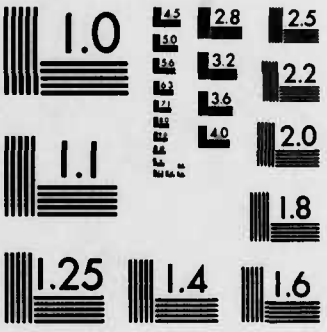
AD-A132 339 ADVANCED PATTERN RECOGNITION(U) PAR TECHNOLOGY CORP NEW 1/2
HARTFORD NY J L CAMBIER ET AL. MAY 83 PRR-83-1
RADC-TR-83-50 F30602-80-C-0319

UNCLASSIFIED F/G 20/6 NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

(12)

[redacted]

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER RADC-TR-83-50	2. GOVT ACCESSION NO. AD-A132 339	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) ADVANCED PATTERN RECOGNITION		5. TYPE OF REPORT & PERIOD COVERED Final Technical Report Sep 80 - Nov 82
		6. PERFORMING ORG. REPORT NUMBER 83-1
7. AUTHOR(s) Dr. James L. Cambier Dr. Stephen Barth Mr. William J. Reid Mr. Scott A. Barrett		8. CONTRACT OR GRANT NUMBER(s) F30602-80-C-0319
9. PERFORMING ORGANIZATION NAME AND ADDRESS PAR Technology Corporation Route 5 Seneca Plaza New Hartford NY 13413		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 62702F 45941822
11. CONTROLLING OFFICE NAME AND ADDRESS Rome Air Development Center (IRRE) Griffiss AFB NY 13441		12. REPORT DATE May 1983
		13. NUMBER OF PAGES 138
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) Same		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

Same

18. SUPPLEMENTARY NOTES

RADC Project Engineer: Frederick W. Rahrig (IRRE)

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Feature Extraction
Pattern Recognition
Image Processing
Artificial Intelligence

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The Advanced Pattern Recognition effort has provided additional enhancements to the RADC Image Processing System (IPS) for developing a variety of techniques in performing target detection and identifications. These extensions include the extraction of features based on edge and region information in addition to the normal pixel classification methods. This capability now permits both statistical and symbolic classification of pixels and objects. Further improvements provide the ability to develop and apply artificial intelligence in the form of rules for extracting features. (Cont'd)

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

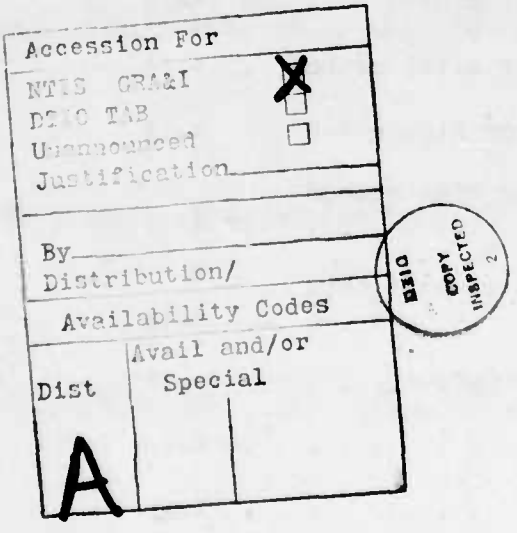
UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

cont.

automatically.

Table Of Contents



LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

This document describes the results of RADC contract F30602-80-C-0319, entitled Advanced Pattern Recognition (APR). The contract was performed by PAR Technology Corporation between September 30, 1980 and November 30, 1982.

1.1 BACKGROUND

The objective of the APR effort was to apply techniques developed for image processing and cartographic feature extraction to the automatic detection and identification of tactical and cultural targets from various types of imagery. This was accomplished through implementation of a variety of enhancements to the RADC Image Processing System (IPS). The IPS is an image processing testbed which is, except for the absence of a scanner/plotter subsystem, identical to the Automatic Feature Extraction System (AFES) developed by PAR under RADC contract F30602-78- C-0080 and delivered to the Defense Mapping Agency. The AFES design is based in part on earlier image processing and pattern recognition systems developed by PAR for RADC; these include the OLPARS and DICIFER systems. Under the Image Feature Manuscript Generation (IFMG) contract (RADC Contract F30602-78-C-0017) PAR developed DLMS-type feature manuscript generation software which was extended and enhanced in the APR effort. Target-detection algorithms developed under the Pattern Recognition Applications to Imagery Assessment contract (RADC F30602-78-C-0358) were also incorporated into the APR software.

The AFES/IPS software includes a large collection of system and applications modules which support a wide variety of functions. The operating system (UNIX) supports a multi-user environment, a tree-structured file system eminently suited to image processing, modular software structure, a complete software control system for system and applications programs, program development aids, documentation aids, and interfaces to all peripheral devices and subsystems. Applications software provided with the AFES supports pixel

measurement extraction; pixel classification via statistical pattern recognition; image preprocessing, enhancement, and filtering; image warping, resampling, and point positioning; and symbolic image processing via a rule-based inference system. All of these capabilities are available within the APR software configuration and have been augmented by a considerable number of enhancements and additions.

1.2 SCOPE

The scope of the APR effort includes extensions to the AFES software to support a variety of approaches to target detection and identification. Additional pattern recognition tools, previously developed as part of the OLPARS, were also implemented. These consist of measurement evaluation and structure analysis for use in target classification experiments. Software was also implemented to permit generation of DLMS-type feature manuscripts and feature analysis data tables including both cultural and tactical targets.

Specific extensions to the AFES developed under APR include:

- modification of the classification logic structure to permit multilevel classification of pixels and objects,
- extension of measurement extraction and classification software to permit statistical and symbolic classification of objects,
- extraction of objects based on edge-detection methods in addition to pixel classification methods,
- implementation of a new symbolic image processor (NEWSIP) which supports probabilistic inferential reasoning,

- implementation of the structure analysis and measurement evaluation facilities of the OLPARS within the IPS environment, and

- implementation of interactive, semiautomatic functions for generation of DLMS feature manuscripts and feature analysis data tables.

1.3 REPORT ORGANIZATION

Section 2 provides a brief overview of the IPS hardware and software configuration and, also, a top-level description of the APR enhancements to the IPS. More detailed descriptions of the enhancements may be found in the remaining sections.

Section 3 is devoted to symbolic processing and covers the data structures, data access routines, and data processing routines which support NEWSIP. This section also discusses and gives examples of the DLMS output generation capabilities.

Section 4 contains descriptions and examples of various pixel and region processing routines added to the IPS; these routines include edge detectors, edge filters, classifier output filters, and region measurement extractors. This section also describes the OLPARS measurement evaluation and structure analysis capabilities.

Appendix A, entitled "User Guide to Knowledge Engineering With NEWSIP," contains a tutorial-style description of NEWSIP and examples of rule base development.

Appendix B, entitled "NEWSIP Rules Syntax," provides a formal description of the rule syntax.

Appendix C, "APR Rule Bases," is a collection of rule bases developed under the APR contract.

2. SYSTEM OVERVIEW

The APR software was developed as a series of enhancements to the RADC Image Processing System (IPS), which, in turn, was developed by PAR Technology as the Automatic Feature Extraction System (AFES) under RADC Contract F30602-78-C-0080. The APR enhancements to the AFES are designed to support 1) automatic detection and identification of representative sets of features from multisource imagery and 2) production of graphic and tabular representations of those features. Earlier PAR work in this area includes the Image Feature Manuscript Generation (IFMG) effort (RADC Contract F30602-78-C-0017), in which various techniques were developed for generating, categorizing, and editing feature boundaries.

The subsections which follow provide a brief overview of the IPS hardware configuration and its processing capabilities, followed by a description of the APR enhancements to AFES.

2.1 IPS HARDWARE CONFIGURATION

The RADC IPS hardware is composed of two major subsystems, the master processor and the display workstation. The master processor functions as the vehicle for program development, data storage, and many processing operations. The display workstation configuration provides the main human-machine interface for the accomplishment of image exploitation.

2.1.1 Master Processor

The master processor is a PDP-11/70 minicomputer with a variety of I/O devices, storage units, and processing resources (Figure 2-1). Input imagery may be provided on magnetic tape, and tape drives are provided for access and copying of image data. Although not included in the IPS, the AFES design includes 1) a scanner/plotter subsystem which is linked to the processor via a

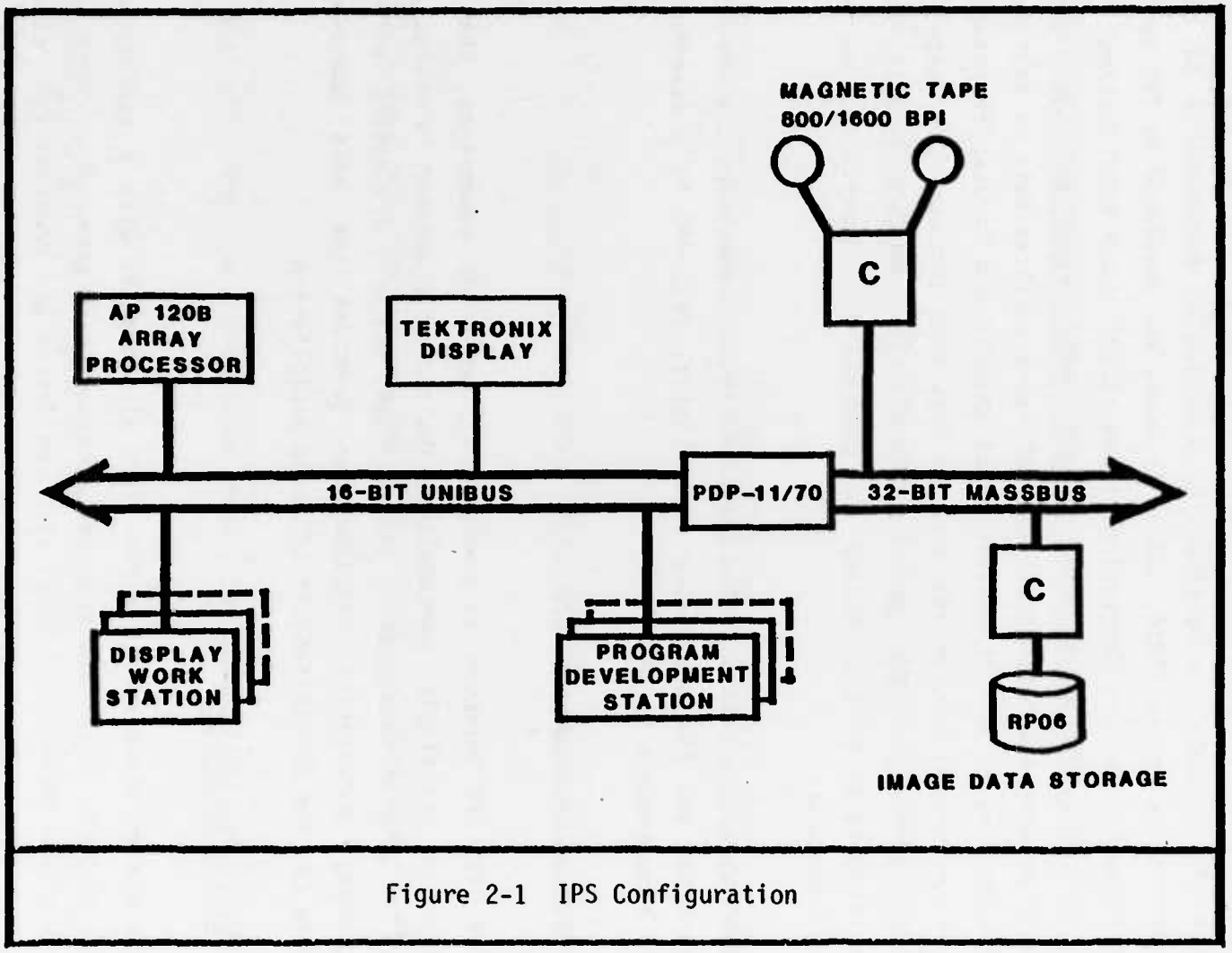


Figure 2-1 IPS Configuration

communication link and 2) a dual-ported disk system so that film, map, or chart data may be digitized and stored on the disk and, then, be utilized by the system as needed. A second large-capacity disk system stores source images and intermediate results of image processing functions executed on the master processor. Processing resources include, in addition to the capabilities of the PDP-11/70, a floating point array processor, which is used to perform certain types of tasks involving numerical computation on large blocks of data.

Associated with the master processor are a number of CRT terminals. They are designed for the user who wishes to edit and compile programs and to execute on the master processor, programs for which image display output is not needed. UNIX, the multi-user, time-sharing operating system for the master processor, can accommodate a large number of these terminals without noticeable degradation in response time.

2.1.2 Display Work Station

The Display Work Station (Figure 2-2) provides the full complement of image processing and interaction capabilities. A color display system is included, on which the user may view source imagery or the results of processing operations. Two high resolution monochrome display systems and a stereo viewer are provided to allow display of stereo imagery. Each display system has a trackball, hardware cursors with function buttons, and overlay memory to accommodate operator interaction and display of auxiliary data. A Hewlett Packard Random Scan Display accompanies the displays. This is used as a status display to provide the user with relevant information, such as status of background processes and the name of the image that is associated with a particular display channel. In addition, a Dunn color camera system is interfaced to the color display so that hardcopy of source or processed imagery is producible in an efficient, convenient, and timely manner.

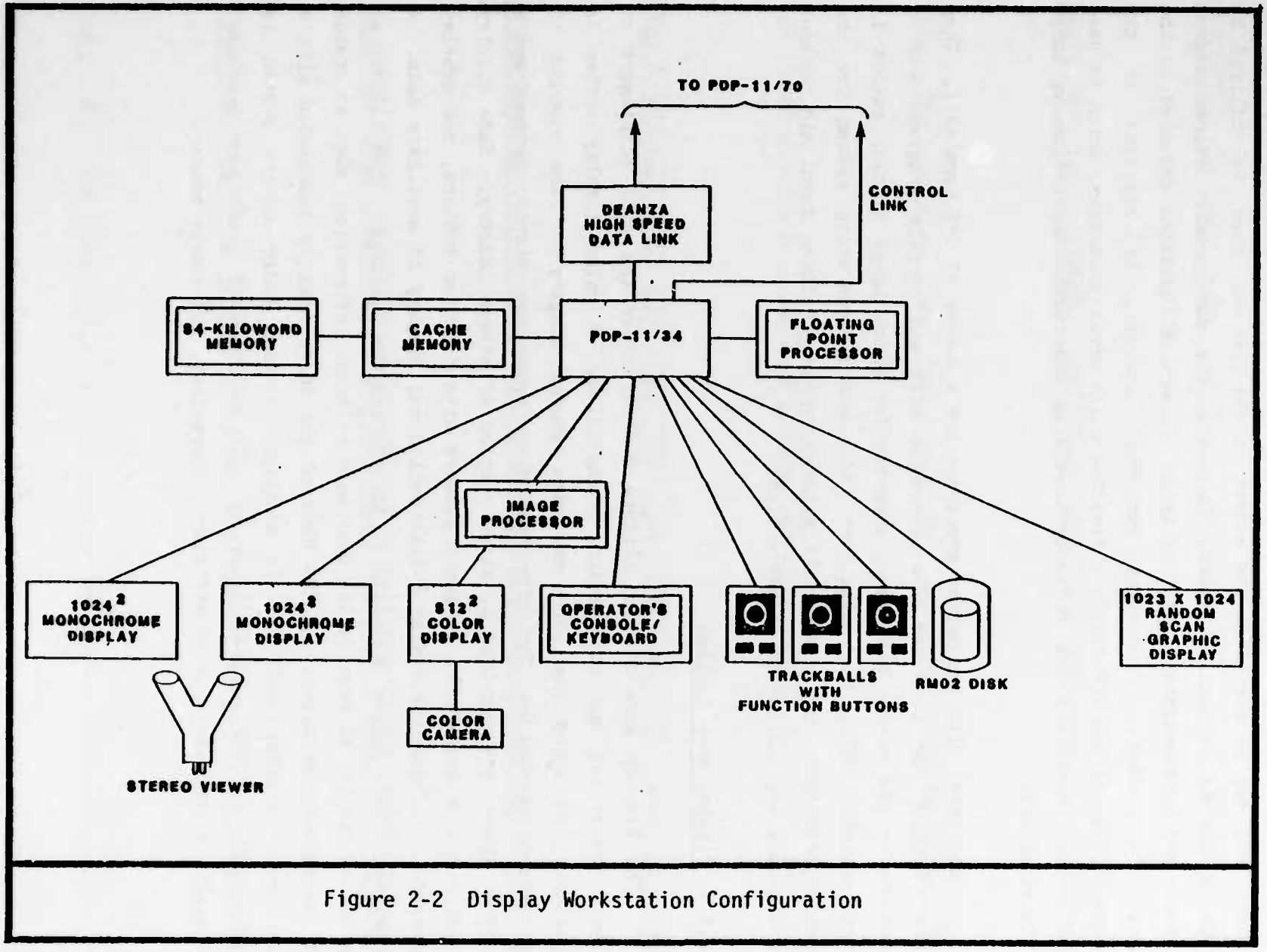


Figure 2-2 Display Workstation Configuration

The display configuration provides the environment necessary for integrated testing of image processing functions and for design and implementation of the types of software systems envisioned for production of digital maps. It is controlled by a PDP-11/34 display processor which also provides a minimal processing capability. In particular, operations which require frequent and/or random access to image data, but which do not perform complex computations, are well suited to execution on the display processor. These operations may include histogram computation, contrast modification, edge detection, simple geometric transformations, and other preprocessing or enhancement operations. Image data may be transferred to and from the master processor via a high-speed parallel data link.

Facilities for operator interaction for the display station are designed to minimize the knowledge required to use the system. Commands issued by the workstation user may refer to processes which are executed on either the master processor or the display processor. To simplify operations, incoming commands are automatically sorted by the display processor's command interpreter. Those which run on the display processor are executed immediately, while others are transferred to the master processor's command interpreter.

In general, programs which require operator interaction are executed on the display processor. In some cases a single command may 1) start a process on the display processor which will interact with the user to obtain input data or parameters and, then, 2) start a "batch" type process on the master processor to perform a computation using the user's input data. Most user interaction occurs via trackballs, cursors, and pushbuttons.

2.2 AFES/IPS PROCESSING

Processing in the AFES is of two general types: image processing and program development. These objectives are supported by commands available in

the system. Two types of commands exist in the AFES: UNIX commands and AFES commands.

UNIX commands provide a wide and powerful selection of user capabilities. They provide system status information, aid in program development, and manipulate files and data. UNIX commands are supported by the UNIX command language, also known as the shell. The reader can familiarize himself with UNIX commands by referring to the "IS/1 User's Guide" provided by the Interactive Systems Corporation.

AFES commands more closely support the goals of image processing than do UNIX commands. In image processing, AFES provides tools for Statistical Pattern Recognition, as well as tools for research in the field of Artificial Intelligence. Some AFES commands allow for photogrammetric processing of imagery. AFES commands also exist for input processing of imagery, general image manipulation, program development, administrative purposes, and utility functions.

AFES commands are arranged in menus according to the function they perform. The available menus are:

admin - AFES administrator commands

class - Classifiers

disp - Display commands

init - Display initialization commands (accessible on the 11/34 only)

input - Commands used to enter images into the AFES environment

itt - Command which make use of Intensity Transformation Tables (ITTs) on the DeAnza displays.

meas - Measurement extractors

mens - Mensuration commands

misc - Miscellaneous commands

prog - Program development commands

symb - Symbolic processing commands

tst - AFES testbed commands, including many used in Statistical Pattern Recognition.

The reader should refer to the "User's Manual for the AFES" for details concerning AFES commands.

2.3 APR ENHANCEMENTS TO AFES

The initial implementation of AFES allows a user to classify an image according to a pixel method and to perform segmentation of the image into homogeneous regions. This is basically the type of classification needed to determine Surface Material Category (SMC), a necessary entry for all features in the Feature Analysis Data Table (FADT). But, extraction of the types of features required for APR requires much more detailed segmentation and classification, as well as contextual information. Therefore, APR supports three kinds of statistical methods: pixel, region, and edge. All of these are independent of symbolic processing, but are available to the symbolic processor. In addition, pixel and edge methods have the capability to either create or modify the symbolic representation of the image.

2.3.1 Method Files

The concept of "method files" was developed on AFES to allow flexible experimentation within the paradigm of decision theoretic pattern recognition. Under AFES, the method file approach supported only "pixel" methods. "Pixel" methods describe the techniques used to classify pixels; measurements are extracted for each pixel using the window code, and a class is computed for each pixel. A typical method file for pixel classification is:

```
measurements:
      avg 3
      lapl

classifier:
      mahal  [optional arguments]

class:
trees
      regions:
            trees1
            trees2

class:
water
      regions:
            lake1
            river1
            river2

class:
urban
      regions:
            industry1
            residential1

comments:
    <user comments>
```

"Region" methods have been added under the APR effort to describe the techniques used to classify regions. Measurements are extracted for some

regions, and a class is computed for those regions. Region methods are thus analogous to pixel methods, except that regions are measured and classified rather than pixels.

"Edge" methods have been added to describe the techniques used to detect edges. In order to expand the method capability to incorporate "region" and "edge" methods, the current method file was modified to contain a new section called "method_type," where the user indicates either "pixel," "region," or "edge".

The following is a representative region method.

```
method_type:
 region
measurements:
 area
 length
classifier:
 mahal
(training set)
class:
 target
   regions:
 planes1
class:
 clutter
   regions:
clutter1

comments:
This method recognizes airplanes from clutter.
```

Edge methods do not include a "classifier" section; rather, the region edges are extracted directly and used to form the symbolic data base. An example follows:

```
method_type:
 edge
measurements:
wiener_edge f
edit:
 edge_thin3

comments:
This is an edge method using optimal Wiener filtering.
```

Another section called "edit" has been added to indicate automatic image editing techniques which may be applied after classification and before segmentation. These include "mode_filter" for pixel methods and "edge_thin" for edge methods. Since region methods are applied to images which have already been segmented, editing techniques are not appropriate.

2.3.2 Measurement Evaluation and Structure Analysis

Currently in the AFES system there is no capability for a user to evaluate the effectiveness of various measurement extractors. He does have the output of the confusion matrix, but this only lets him evaluate the effectiveness of the measurement set as a whole. Under APR a subset of On Line Pattern Analysis Recognition System (OLPARS) referred to as "measurement evaluation" and "structure analysis" has been implemented. Measurement evaluation allows the user to assess the accuracy of classification logic which has been developed for a particular set of pixel-based or region-based training data. Structure analysis can either be used to view classification results of region or pixel data in measurement space or be used to restructure training data for region methods only. The available output displays, such as histograms, are supported on the Tektronix display which is part of the IPS configuration.

It should be noted that measurement evaluation and structure analysis are based on trainsets which are representative of classes as defined in the method file.

2.3.2.1 Measurement Evaluation

At some stage in the pattern recognition process there exists a training set of vectors of dimension L for each class type. The eventual success of classification depends on the discriminatory power of this set of L measurements. In general, it is also desirable to use the minimum number of measurements to achieve a satisfactory solution. The two means of evaluating methods implemented in APR are the "discriminant measure" and the "probability of confusion measure". The discriminant measure is particularly useful for ranking a set of L measurements when the class conditional probability distributions are approximately unimodal. It is the fastest to compute, but can produce misleading results when the data classes are not unimodal. When non-unimodel data classes are involved, the probability of confusion measure is recommended. It is valid for any probability distribution, since it essentially measures the overlap of class conditional probabilities. The output options for either measure are:

1. Rank measurements for a specified class,

2. Rank measurements for a specified class pair,

3. Rank classes for a specified measurement,

4. Rank class pairs for a specified measurement,

5. Union of best measurements for each class,

6. Union of best measurements for every pairwise set of classes, and

7. Rank measurements distinguishing all classes.

In addition there is an option to display histograms which show the frequency of occurrence of the classes across the range of each measurement. These histograms are useful in determining the value of a measurement in discriminating the classes in the data set. The user has the option of displaying one or more of the histograms on the same display.

2.3.2.2 Structure Analysis

Structure analysis assists the analyst in determining the modality of each class as represented by the corresponding trainset. There are two primary uses for structure analysis in the APR system. One is to allow a user to sub-divide a trainset, as defined in his method, into two or more classes. Once the user views the plot of trainsets, he/she can restructure the plot into two or more sets representative of two or more classes (or throw one away) if it does not cluster nicely. These sub-classes could then be recombined by NEWSIP as deemed appropriate by the rules. The second use for structure analysis is to allow a user to cluster the regions of a segmented image into classes. This is one of the ways by which one may define test objects to be used in "region methods".

APR provides one-space and two-space plots of projections on the:

1. Original feature coordinate axis,

2. Arbitrary vectors, or

3. Eigenvectors.

The user may draw a piecewise-linear convex boundary for the two-space plots in support of the two goals mentioned above. Examples of structure analysis plots are provided in Section 4.

2.3.3 Classifier Training

The AFES has been designed to allow a user to train a classifier according to a method which is easily modifiable. This has been accomplished using method files, which specify the exact type of processing required for pixel classification. This capability was confined in the AFES to pixel-based methods (or simply pixel methods), where "get-region" was used to outline sample training regions. This concept has been extended with APR to region classification, whereby statistical pattern recognition is applied to entire regions, rather than to just individual pixels. The processing required to classify regions is specified in region methods. To obtain sample training sets for region classification, one executes "get-object". This program allows the user to specify entire regions or objects to be included in training sets for region classification. This program must be executed after segmentation.

A new command on the 11/70 allows a user to run structure analysis on the regions in the symbolic representation of the image. One could use this technique to assign regions to the proper region directories. The user has the capabilities of reviewing the regions thus defined on the DeAnza display and of removing any unwanted ones from the test set.

2.3.4 Classification and Segmentation

Due to the modular approach to classifiers which AFES utilized, no changes were required under APR to the actual classifier code itself, either in C to run on the 11/70 processor or in AP code to run on the array processor. There have, however, been changes to the "classify" command required at the shell level in order to support the "region" and "edge" methods. There is an "edit" command, which may be listed in the edit section of the method file which is applied to the classified output. There is also a "segment" command to convert the classified output of either a pixel or edge method into a symbolic representation. A major change in the system allows

the Symbolic Image Processor to execute any, or all, of the three types of methods on a region.

Segmentation, like classification and edge detection, may be performed on individual regions. The user may specify the region to be segmented (resegmented) as a parameter. Initially, before an image has been segmented even once, region 1 is defined as consisting of the entire image. The first time an image is segmented, region 1 is segmented.

Input for segmentation consists of the symbolic data files plus either the results of classification or the results of edge detection, possibly modified by editing. Segmentation occurs on the PDP-11/70 rather than on the PDP-11/34 (as was the case with the original AFES). This change permits invoking the segmenter from within the symbolic image processor without the need to transfer data from machine to machine. The output of segmentation is reflected in updates to the symbolic data files. This is the only way in which the segmenter makes public its results.

Edge-based segmentation is supported in addition to pixel-based segmentation. This edge-based technique is helpful when pixel-based segmentation is inadequate or inappropriate, as may be the case when source imagery comes from infrared sensors, when there is little texture in the image, or when there is only a single band of information. Edge-based segmentation may also be used for segmenting urban areas when it is impossible to recognize all possible land use classes, yet it will suffice to outline buildings, shadows, vehicles, etc.

2.3.5 Symbolic Data Handling

Images are described symbolically in symbolic data files. The symbolic data files for an image contain information sufficient to define the topology of the image and to define the attributes of regions and edges within the image. All access to the symbolic data files is coordinated through the

attribute manager.

One of the keys to effective use of symbolic image processing is an adequate symbolic image description. The adequacy of a symbolic image description scheme is, in turn, dependent on the file structure supporting it. The symbolic image representation scheme and file structure impose as few restrictions as possible on image processing while providing a maximum of versatility.

2.3.5.1 Symbolic Image Representation

Images in APR are represented symbolically as a semantic network. The nodes in the network correspond to the edges and regions in an image and are associated with the attribute values for those edges and regions.

The symbolic representation of an image is used for three purposes: (1) to aid in creation of training data sets of homogeneous regions, such as tanks; (2) to allow region classification of a segmented image; and (3) to respond to "artificial intelligence" requests from the New Symbolic Image Processor (NEWSIP) to extract region attributes, classify and resegment a region, or modify the symbolic image description.

The relationships among the regions and edges in an image are the most important components of the semantic network. The description of a region always includes those edges that make up its boundary, and the description of an edge always includes those regions on either side. If applicable, each region has pointers to its offspring regions and its parent region. Each edge has a pointer to its offspring edges and its parent edge, if applicable.

Offspring regions arise whenever regions are resegmented. The region being resegmented is divided into a set of child regions, each of which may be further resegmented. Edges cannot be resegmented in the same way as regions can, however, it is possible to define an edge that is part of another edge.

There are two special regions that are predefined in each image. Region 1 is the entire image. The first time the image is segmented, region 1 is the region being segmented. It is an ancestor of every other region in the image, except for region 0, the other predefined region, which corresponds to the rest of the world not present in the current image. Region 0 cannot be displayed like other regions, of course, as there is no data for it. It is included merely as a matter of convenience.

Each region and each edge has a set of attributes associated with it, expressed as attribute codes - attribute value pairs. Attributes (such as area, length, width, perimeter, etc.) are used to describe regions and edges in a manner that is well suited for rules. Most of the interesting properties of regions and edges are represented as attributes.

2.3.5.2 Attribute Manager

Central to the issue of symbolic data handling is the process referred to as the "attribute manager". All accesses to this data representation pass through the "attribute manager". This process performs many of the same type of functions which the "shell," the UNIX file system, and AFES modules currently handle in the pixel domain of image processing in the AFES. For example, the attribute manager determines if a measurement has been run for a region and, if not, extracts it and enters it as an attribute of the region.

2.3.6 Feature Recognition

Feature recognition is the crux of APR. Everything else in the system is subservient to it. Methods, training, structure analysis, classification, editing, and segmentation merely make up the front-end for feature recognition. Of course, they are all very important. Without such a battery of powerful techniques, feature recognition would have little chance of success. It is the presence of these tools that makes everything else possible.

Symbolic data files, together with image data files, comprise the input for feature recognition. Two approaches to feature recognition are supported: rule-based and statistical. Outputs of feature recognition are modifications to the symbolic data files and the creation of a DLMS data base.

2.3.6.1 Rule-Based Approaches

The symbolic image processing enhancements implemented within APR have been called NEWSIP. The NEWSIP rule-based system was designed to allow the use of probabilistic, inferential reasoning for image understanding in the AFES/RWPF environment. Like most expert system frameworks, NEWSIP consists of two main components, an inference engine and a rule base. The inference engine consists of the software for a general mechanism to draw and explain inferences. The rule base is a text file consisting of a set of conditions and actions which provide the knowledge for particular inferences to be made to solve a problem. In the field of Artificial Intelligence, the task of building a rule base is known as knowledge engineering.

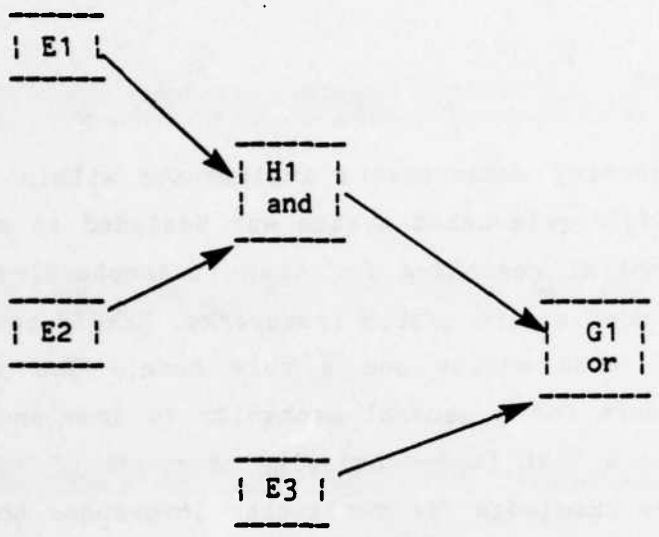
A rule base consists of a set of conditions and actions that describes the inferences which can be made for solving a problem. A single rule consists of an "If-Then" statement, relating a set of antecedent conditions to a consequent hypothesis. The certainty or truth of the consequent hypothesis depends upon that of the antecedent conditions. The antecedent conditions may themselves be the consequent parts of other rules. For example, the statements,

If E1 and E2, then H1,

If H1 or E3, then G1

represent two abstract rules. E1 and E2 are antecedent conditions for H1, and H1 and E3 are antecedent conditions for G1.

NEWSIP uses an inference network (inference net) representation for its rule base. If the two rules of the example above made up an entire rule base, they could be thought of as an inference net that can be drawn as:



The nodes in the network represent the individual antecedent conditions and consequents. E1, E2, and E3 are considered evidence nodes; H1 is an intermediate hypothesis node; and G1 is a goal node.

The labels of evidence, hypothesis, and goal are assigned by the user when building a rule base, but are usually reflected in the structure of the inference net. Evidence nodes are the leaves of the network, having no incoming arrows representing inferences. They are associated with some test or measurement to be performed on an image. Goal nodes are the roots or sinks of the network having no outgoing inferences. (As will be described later, however, a node like H1 may also be considered as a goal). The arrows between nodes of the inference net represent the antecedent consequent relationships. The labels of "and" and "or" applied to H1 and G1 show the kind of relationship between a node and its antecedents. In the probabilistic inference mechanism that NEWSIP uses, the arrows, or links between the nodes of the inference net, also represent paths along which probabilities are

propagated, and each node has a probability associated with it.

The control mechanism of the NEWSIP inference engine (as applied to the above example) can briefly be described as follows. Assuming that the rule base has been coded in the proper format, invoking the NEWSIP inference engine will cause 1) the rule base to be read in and parsed for syntactic correctness and (2) an inference net representation to be formed internally. Next, a goal node will be selected for consideration. (For this reason, NEWSIP is considered goal-directed.) In the example, only one goal node, G1, is available. Having selected G1, the control mechanism now tries to determine its certainty. To do this it looks for evidence nodes that affect G1. This process is recursive and depth-first. The immediate antecedents of G1 will be considered one at a time. If the first one considered is not an evidence node, its antecedent nodes will be examined. So in the example, H1 might be selected as an antecedent for G1, and then E1 could be selected as evidence for H1. At this point, whatever test or measurement associated with E1 is performed and the certainty of E1 is determined: E1 turns out to be either true or false. This certainty is now propagated through the inference net. Probabilities are updated for all of the nodes affected, directly or indirectly, by E1. In this example, H1 and G1 will be affected.

Now the control mechanism iterates. G1 still needs further consideration since other evidence remains to be investigated. H1 also has not been completely investigated, so it would be considered again. E1 has been tested, so the test for E2 would now be invoked. The resulting probability would be propagated, and the process continued with another iteration (which would this time investigate E3). After E3 has been considered, no further evidence remains to be investigated for G1, and NEWSIP would halt after having performed (or not performed, depending upon G1's probability) whatever action is associated with G1.

Three kinds of actions are currently possible: to apply a segmentation method to the current region, to delete a region from further processing, or to assign an AFES/RWPF region attribute value to the current region. The rules of a NEWSIP rule base are applied to one region at a time, and the one being considered at the moment is called the current region. Evidence tests are limited to the evaluation of conditions on sets of regions, or sets of edges, or some special predicate functions. Conditions are expressed as arithmetic relationships on region attribute values, and sets of regions are specified in terms of a quantity of neighboring, offspring, parent, edges, or included relationships to the current region being examined.

A more detailed description of NEWSIP is provided in the NEWSIP document attached as Appendix A.

2.3.6.2 Statistical Approaches

Statistical feature recognition is based on regions and, thus, is intimately tied to symbolic image processing. Statistical recognition of features may be either invoked from the shell, as an APR top-level command, or invoked from within NEWSIP, as an adjunct to or part of rules.

Input to a statistical feature recognizer is a region or set of regions to be classified. The input regions must be the result of a previous segmentation, which occurs naturally when the region classifier is invoked from NEWSIP. Rules are capable of invoking a statistical region classifier, for instance, to find airplanes or tanks. Of course, if desired, a user can also invoke a region classifier if he knows about specific target regions that are to be input.

A method of region classification must be specified for classification to occur. This method will have been defined and trained previously, as described in earlier sections of this document.

Output of statistical region classification is the assignment of new class attributes to the regions classified. The new class types are assigned by interacting with the attribute manager. These new classes are examined by querying the attribute manager.

Features recognized in this way are not immediately suitable for inclusion in the Feature Analysis Data Table (FADT). Mere region classification does not provide enough information to fill out an entry in the FADT. Therefore, region classification will not by itself effect the FADT. However, NEWSIP may use the results of region-based classification in its decision logic when producing the FADT.

Region classification is most useful for recognizing small features, especially when complicated shapes are present. Tanks, airplanes, and other military vehicles seem to be the best candidates for region-based classification.

2.3.7 DLMS Output

The final output of APR is a FADT and a feature manuscript, two components of the DLMS. The FADT is generated by NEWSIP. This can be done interactively with the operator performing the feature recognition or automatically with rule-based feature recognition.

The FADT contains fields for surface material category, height, structures per square nautical mile, feature code, etc. Most of these fields can be determined by rules, especially since some values are standardized. For example the height of Surface Material Category (SMC) 6 features (water) is always set to zero. The orientation of non directional features is set to 360. In fact, the DLMS specifications are well suited for encoding in rules.

Feature manuscript generation is very simple. There is no need to create a new file for the feature manuscript since all needed information is already present. The manuscript itself is a set of graphics overlays delimiting the features, which are displayed in the DeAnza overlay memory.

FAC numbers, used to associate FADT entries with the feature manuscript, can be displayed using the DeAnza display annotation memory. The annotation memory can be blanked out, if desired, allowing the operator to see features that might, otherwise, be obscured by the FAC numbers. Additionally, if desired, the operator can display the image together with the manuscript. The manuscript then appears to be overlaid on the image, allowing quick verification of the accuracy of the FADT and the image feature manuscript.

2.4 APR MENUS

The new capabilities added to the AFES in the course of the APR effort consist of modifications to some existing commands and, also, addition of new commands. Table 2-1 contains descriptions of all commands added to the AFES during the APR contract.

Table 2-1 APR Commands

Page 1 of 3 Pages

*** Program Development ***

mod_afes(mda)	- start parallel development of an afes routine
apr_sys(aps)	- modify the apr makefile, menu, help...
aprupdate	- update the apr "mods" directory
acc	- compile file with apr/afes includes and libraries

*** Symbolic Data Manipulation ***

newsip	- execute new Symbolic Image Processor
init_symb	- initialize symbolic data files
prt_symb	- print information on symbolic data files
edit_symb	- edit and manipulate symbolic data files via attribute manager
segment	- segment an image or resegment a region

*** Symbolic Methods ***

enter_symb(ens)	- create symbolic method and enter rules
mod_symb(mds)	- modify rules for symbolic method

*** Region Measurements ***

area	- compute the area
perimeter	- compute the perimeter
p2overa	- perimeter squared divided by area
length	- length of minimum enclosing rectangle
width	- width of minimum enclosing rectangle
ave_inten	- average intensity of the region
moment_bnd	- moment of the boundary of the region

Table 2-1 APR Commands

Page 2 of 3 Pages

```
*** Edge Measurements ***

length          -   the actual length of the chain code
distance        -   the distance between the endpoints
edge_complexity - the complexity of an edge

*** Region Methods ***

get_object(gto)    - select object training samples (11/34 program)
load_reg_trn(lrt)  - load object region training files from mask files
obj_reg_edit(ore)  - examine and/or edit obj_region training files

*** Edit Programs ***

mode_filter (mdf) - mode filtering of a classified image (noise cleaning)
edge_thin         - thin edges iteratively

*** Measurement Extractors ***

wiener_edge     - optimal wiener edge detector
sobel_dir       - sobel gradient operator with direction added for filtering
dir_filter      - directional filter, must follow sobel_dir or some filter
mdmf            - maximal directional matched filter
edge_fill       - edge filler, must follow sobel_dir or some filter
squeeze_i       - image compression
expand_ipr      - image expansion by pixel replication


*** Measurement Evaluation / Structure Analysis ***

meas_eval(mev)  - measurement evaluation via discriminant and probability
structure(str)  - one-and two-space structure analysis
```

Table 2-1 APR Commands

Page 3 of 3 Pages

*** DRLMS Generation ***

fadt	- feature analysis data table
bld_man	- interactive feature manuscript generation (PDP-11/34)

3. APR SYMBOLIC PROCESSING

The primary goal of the APR effort was expansion of AFES symbolic processing capabilities; this was accomplished through expansion and modification of software at many levels of the AFES architecture. The implementation of APR is most easily described from the perspective of symbolic processing, as this provides an appreciation of the factors motivating the various software changes. The following paragraphs discuss the major components of the APR software. A detailed description of the new symbolic image processor, NEWSIP, is provided in Appendices A and B.

The principal data items used in symbolic processing - regions, grid cells, edges, and attributes - are described in Section 3.1. Section 3.2 describes the layout of the symbolic data files and how the principal data items are stored. Finally, Section 3.3 describes the modules that were developed to access the data stored in the symbolic files and to generate the DLMS-type feature manuscript and feature analysis data table (FADT).

3.1 REGIONS, GRID CELLS, EDGES, AND ATTRIBUTES

Regions, grid cells, edges, and attributes are the principal data items that are represented in Symbolic Data Files. Regions, edges, and grid cells are numbered starting at zero and are often referred to as r0, r1, e0, e1, g0, g1, etc. Regions and edges together suffice to describe the topology of an image, while grid cells provide a direct representation of an image. Attributes further describe the regions and edges. A more detailed explanation of each follows.

3.1.1 Regions

The following definition of a region is overly-simplified and, thus, not totally correct, but it will serve our purposes until a better definition is

presented later. A region is simply a set of pixels (in an image) which are CONNECTED and belong to the same CLASS. Specifically, the pixels in a region must be 4-connected. That means they must be next to one another to the right, left, top, or bottom; diagonals do not count. The class of the pixels is usually determined by CLASSIFYING the image; this assigns a class to each pixel. The class of a region is the class of its component pixels. Consider the following image composed of 16 pixels. Each pixel is labeled with its class, as shown below:

```
A  A  B  B
A  A  B  B
C  C  A  A
C  C  A  A
```

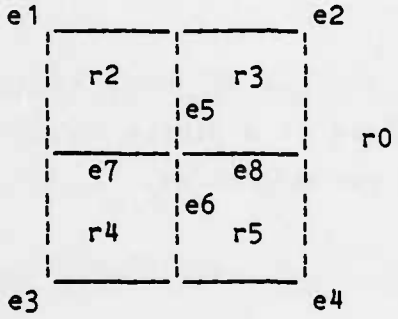
In this example there are 4 regions. One is class B, one is class C, and two are class A. Notice that the two class A regions are not 4-connected; therefore, they do not form a single region.

A region can have any size from a single pixel up to an entire image. A single pixel is usually too small to correspond to a meaningful feature, so such small regions are often ignored by rules. Also, very large regions may contain many features, and therefore it is often desirable to reclassify a large region to break it into smaller regions.

There are two special regions whose definitions differ from that given above. These are region 0 and region 1. Region 1 is defined to be the entire image. Of course, it is 4-connected, but its pixels do not all have the same class. Therefore, the class of region 1 is not defined. Region 1 can be considered the ancestor from which all other regions are derived. Region 0, on the other hand, is defined to be the entire world not in Region 1.

3.1.2 Edges

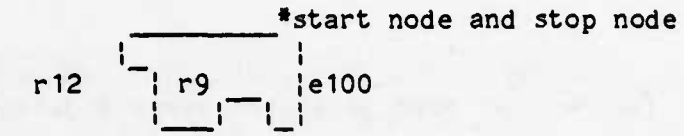
An edge is defined as follows: An edge separates every adjacent pair of regions. Each edge has two distinct regions next to it: one on each side. Edges may have arbitrary length and complexity. Below is shown the same image as before, with the edges and regions labeled:



There are 5 regions here, and 8 edges. Edge 1 separates r2 and r0, edge 5 separates r2 and r3, etc. It is usual for there to be more edges than regions, although this need not be the case. As can be seen, edges 1, 2, 3, and 4 are not straight.

Each edge has a start node and a stop node, with each node defined by a row number and a column number. They are not shown above for the sake of readability. The start and stop nodes give each edge an orientation; therefore, one can speak of the "left region" of an edge and, also, of the "right region". Again, each edge will have a single leftregion and a single rightregion.

A single region may have many edges associated with it. In the above example, r5 has edges e4, e6, and e8. The boundary of a region is composed of all the edges of that region. In this definition, boundary and edge are not synonymous. However, it is possible for a region to have a single edge, as in:



In this example r9 has but a single edge, e100. Edge 100 separates r12 and r9. Since e100 is the only edge for r9, it must form a closed boundary or loop. Therefore, the start node of e100 must be the same as its stop node.

There is one special edge, edge 0, that is in every image. It is the edge that separates r0 and r1. Since there is a single region inside e0, e0 must form a loop and have the same start and stop nodes.

3.1.3 Grid Cells

Grid Cells are 16 x 16 pixel bins which cover the entire image and have as attributes the objects which cover their constituent pixels. Grid cells are numbered consecutively, starting with cell 0 in row major order.

A grid cell has only three attribute types: point, lineal, and areal. The value of the attribute is the number of the object of the indicated feature type. As an example, if region 100 has pixels in grid cells 225 and 226, then G225 and G226 would each have an areal attribute with the value 100.

Grid cells allow more than one object to occupy a pixel or set of pixels in an image. This provides an overlay capability and a mechanism for treating, as half-siblings, objects which cover the same pixels but were created using different methods.

Grid cells also provide a mechanism for object-driven targeting. A rectangle (the minimum bounding rectangle for a region, perhaps) in one image may be mapped onto another image to define a search window. This mechanism is useful in symbolic stereo correlation and change detection.

3.1.4 Attributes

Attributes are lists of properties associated with every region and edge. Each attribute is composed of two parts, an attribute name and attribute value. Attribute names are simply character strings up to 19 characters in length. Attribute values may be long integers, floating point numbers, or character strings up to 19 characters in length. A region may use the same attribute name several times, so that the attribute is multi-valued. In this case the attribute values must be compatible: that is, the same attribute name cannot have both string and numeric values. By using attributes, almost any kind of data about regions and edges may be represented and stored. The following list illustrates some of the attributes that were found in an actual Symbolic Data Base.

```
edge 0
                rightregion 0
                 leftregion 1
                     length 2048.000000
                   distance 0.000000
            edge_complexity 270.000000



edge 1
                rightregion 2
                 leftregion 3



region 0
                        edge 0
```

```
region 1
                edge 0
            offspring 2
            offspring 3
            offspring 4
            offspring 5
            offspring 6
            offspring 7
            offspring 8
            offspring 9
              length 512.000000
                area 262144.000000
           perimeter 2048.000000
           ave_inten 130.212173
          moment_bnd 313175040.000000
               width 512.000000
               holes 0


region 2
                edge 1
                edge 3
                edge 5
                edge 7
              parent 1
               class null
              method pixel

                area 11143.000000
           ave_inten 17.022974
               label house
               holes 1
```

Attributes come from six different sources: segmentation, grid cells, measurements, feature manuscript generation, the attribute manager, and the user. An explanation of each source follows.

3.1.4.1 Segmentation Attributes

During segmentation, several attributes are defined. These attributes may be considered to be the default attributes because they will be defined in every image.

class The class attribute is a string-valued attribute. Every region (with the exception of r0 and r1) will have a class value. Naturally, the value of this attribute is the class of the region.

method The method attribute is also a string-valued attribute. Every region (with the exception of r0 and r1) will have a method value signifying the segmentation method that was used to define it. The class and method are related, since the class name must be in the method file.

edge The edge attribute is present for all regions and is a long integer. There may be many edge attributes for a single region; each indicates the edge number of one of the edges of the region.

rightregion A single rightregion attribute is present for all edges and is a long integer. It indicates the region number of the region to the right of the edge.

leftregion Similar to rightregion.

offspring When a region is segmented, the subregions that result are called "offspring" of the original region. The original region is assigned offspring attributes for each subregion. The attribute itself is a long integer whose value is a region number. Edges may also have offspring attributes. This occurs when, by segmenting a region, an edge gives rise to one or more subedges. Each subedge is considered to be an offspring of the original edge. Inherent in the concept of "offspring" is the notion that the offspring are derived from the original and are entirely contained within the original. However, it should NOT be assumed that the offspring of a region or edge can be summed

to produce the original.

parent The parent of a region or edge is the inverse of offspring. Each offspring will have a parent attribute that is a long integer.

3.1.4.2 Grid Cell Attributes

There are two attributes which arise from definition of grid cells. The areal attribute applies only to grid cells and is the number of a region which contains pixels in the grid cell. A grid cell must have at least one areal attribute. Before segmenting, all grid cells have one areal attribute with the value 1. After segmentation, the parent region will be replaced by its offspring in all of the grid cells which it occupies.

The resides-in attribute is the inverse of the areal attribute and applies only to regions. The value of the attribute is the number of one of the grid cells in which it has pixels.

3.1.4.3 Measurement Attributes

Measurement attributes are attributes that are not necessarily used in a particular Symbolic Data Base, but which can be computed by specific programs or routines. These attributes are computed only if they are needed, and then they are stored so that it will not be necessary to recompute them in the future. It is illegal to attempt to extract any of the following measurements for region 0, since they are meaningless for this region.

area The area attribute is a floating point attribute. It contains the area of a region in square meters.

perimeter The perimeter attribute is also a floating point attribute. It contains the perimeter of a region in meters.

p2overa This attribute is the ratio of perimeter squared to area. It is a dimensionless floating point quantity.

length This attribute is a floating point attribute that applies both to regions and edges. For regions it is defined as the length in meters of the longer side of the minimum enclosing rectangle of the region (minimum enclosing rectangle is the rectangle that completely encloses the region and has minimum area). For edges it is defined as the length in meters of the chain-encoded edge.

ave_inten This attribute is the average intensity of the pixels in a region. It is a floating point number.

moment_bnd This floating point attribute is the moment of the boundary of a region. It is computed as the second moment of rotation of the boundary about the center of mass of the boundary. It equals the moment of inertia of the boundary. The units are meters squared.

holes This long integer attribute is the number of holes in a region. The number of holes is in no way related to the number of offspring.

distance This floating point number is the distance in meters between the nodes of an edge. In general it will be less than the length (see above), unless the edge is perfectly straight. Note that for a closed edge, the distance will be zero, since the start and stop nodes coincide.

edge_complexity This attribute is a measure of the complexity of an edge. It is a dimensionless floating point number.

centroid This attribute is a long integer that gives the weighted average column of the region.

centroid This attribute is a long integer that gives the weighted average row of the region.

3.1.4.4 Feature Manuscript Attributes

There are six attributes that are created during feature manuscript generation. They are "labelr," "labelc," "key1r," "key1c," "key2r," and "key2c". These attributes define locations of feature labels and key lines and are used solely for creating and displaying the feature manuscript.

3.1.4.5 Attribute Manager Attributes

Some routines in the attribute manager create attributes as side-effects of object measurements or computations which may be requested through rule-based processing. These attributes are created to avoid redundant computations of commonly-used information. They are computed only when needed.

Included

This attribute has as its values the region numbers of sibling regions which are wholly contained within it.

Oldsdf

This attribute is created by the transfer routine which copies a region and its edges from one symbolic image to another. Its value is the string representing the name of the symbolic image from which it was copied.

oldnum

This attribute is associated with oldsdf and has as its value the number of the region or edge in the symbolic image from which it was copied.

3.1.4.6 User-Defined Attributes

In addition to the above attributes, the user may define his own. The attribute names that a user chooses should be distinct from the ones given above. The type of values an attribute may hold is determined by usage. For example, if one created an attribute with a string value, all subsequent references to that attribute will be treated as strings. User-defined attributes may arise from rules or from symbolic editing.

Rules are used to determine the operations to be performed during Symbolic Image Processing. Some rules, called Feature Assignment Rules, are able to assign attribute values to regions. The attribute values are added to the Symbolic Data Base if the rule is invoked.

Symbolic editing allows a user to directly modify the Symbolic Data Files. The "edit_symb" program may be executed to perform such editing.

3.2 SYMBOLIC DATA STRUCTURES

There are seven Symbolic Data Files in a Symbolic Data Base. The files are region_hdr, region_ident, edge_hdr edge_ident, atr, atr_key, and grid_hdr, and are located in directory

/w/<user>/<photo>/<view>/<frame>/.symb_methods/<symbolic method name>

Figure 3-1 shows the layout of the Symbolic Data Files and the relationships among them. Figure 3-2 illustrates the entire APR Directory Structure. A detailed description of each follows.

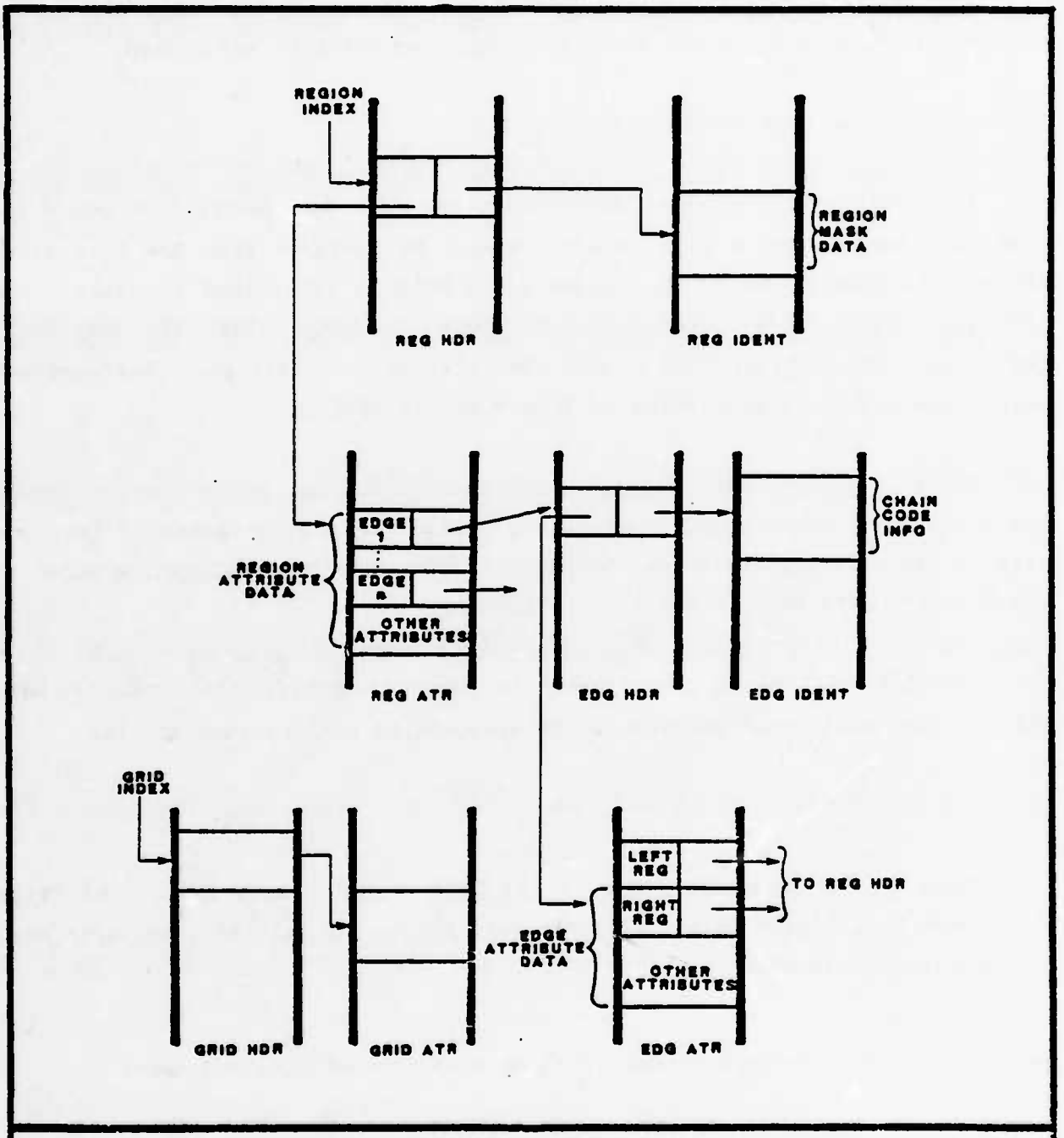


Figure 3-1 Symbolic Data Files

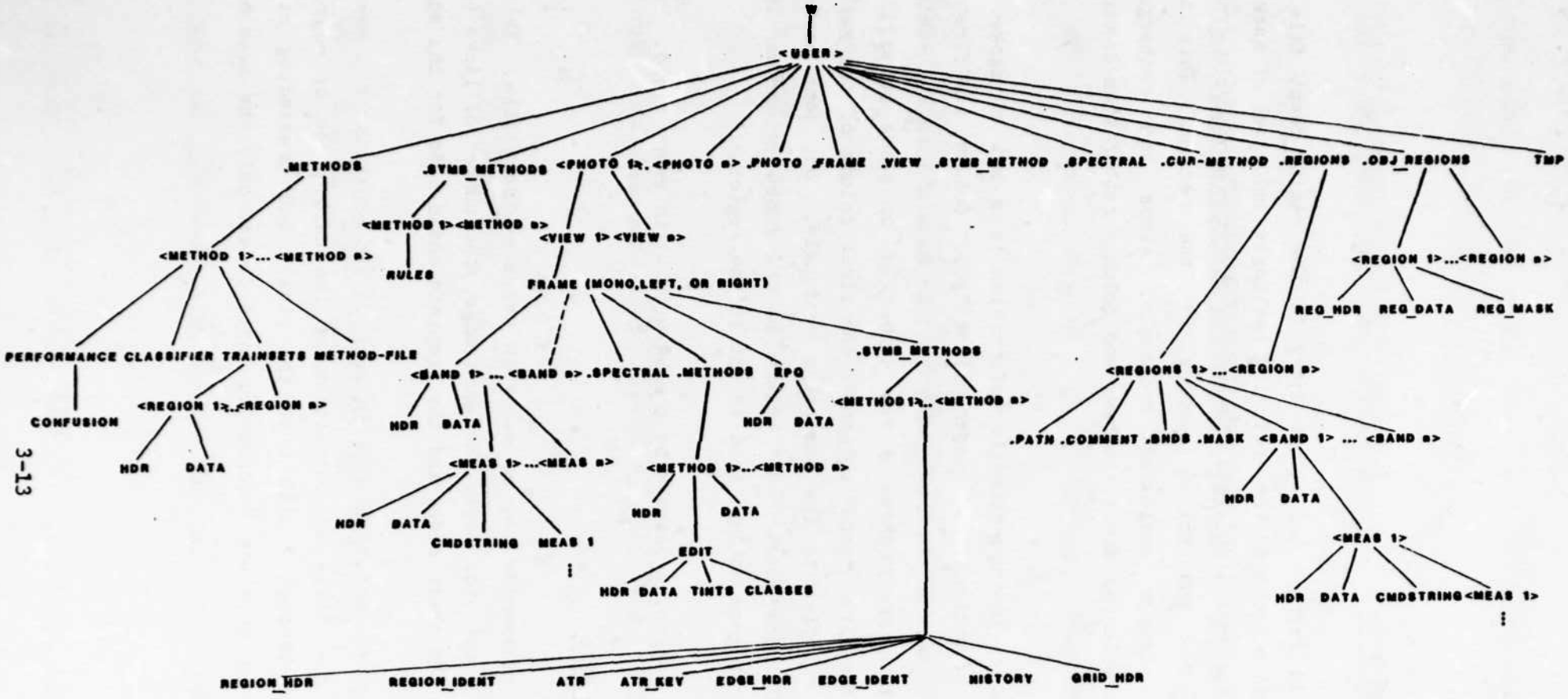


FIGURE 3-2 APR DIRECTORY STRUCTURE

3.2.1 Region Ident File

Each region is represented by an entry in the region_ident file. The entry for each region contains the enclosing rectangle and a set of mask codes for the region. The enclosing rectangle has the lowest row and column numbers and the highest row and column numbers for the region. This is not necessarily the minimum enclosing rectangle, since the rectangle is constrained to be oriented along the rows and columns rather than assuming an arbitrary orientation.

Following the enclosing rectangle information is a set of masks. Each mask is a triplet of numbers: row number, first "good" column, and first "bad" column. The row number is the row number of this mask within the rectangle. The first mask will always have a row number of 0, since it will be at relative row 0. The first "good" column is the first column of the region on this row, again relative to the enclosing rectangle. So, some masks will start at column 0. The first "bad" column is the first column on the row (following the first good column) that is not in the region.

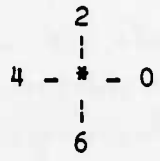
The end of a region is denoted by a mask with -1 in every field.

3.2.2 Edge Ident File

Each edge is represented by an entry in the edge_ident file. This file contains the start and stop nodes for each edge, the number of links (pieces of chain code) in the chain code, and the complete chain code for the edge.

The chain code describes an edge in detail. It consists of a series of numbers; each number tells the direction of the next piece of edge. The numbers go from 0 through 7 with 0 to the right, and proceeding counter-clockwise. Since we only use 4-connected chain codes, only the even numbers are used.

Chain Code Directions



3.2.3 Atr File

The atr file contains attributes for all regions, edges, and grids. The attributes for each region, grid, and edge are kept separate because each region grid and edge has a pointer into the atr file. This pointer points to a chunk of seven attributes. This chunk, in turn, points to another chunk of seven attributes, etc. Thus the atr file consists a series of linked lists, one linked list for each region, grid, and edge.

There are special codes for unused attributes and for the end of a linked list. An unused attribute is denoted by a -2 in the attribute code field. This may be the result of an earlier attribute deletion. The end of an attribute list is denoted by a -1 in the attribute code field.

This scheme allows for the deletion of attributes and the reuse of empty space. It also permits the continual addition of attributes to the Symbolic Data Base. It is common to find over 30,000 attributes in an atr file.

3.2.4 Hdr Files

There are three hdr files present, region_hdr, edge_hdr, and grid_hdr. Region_hdr and edge_hdr contain two pointers, one for the appropriate ident file and one for the atr file. The pointers can be used directly by disk routines to "lseek" to the correct places in the ident and atr files. The grid_hdr file contains a pointer into the atr file.

3.2.5 Atr Key File

The atr_key file contains the attribute names and formats. Each name is allocated 19 characters, plus one character for the format. String attribute values are also stored in this file. Attribute codes are merely pointers into the file. The contents are ordinarily read into memory at the start of processing to improve efficiency and minimize the number of file operations required.

3.3 SYMBOLIC DATA ACCESS

Access to symbolic data can be achieved at several levels. The lowest level of access uses the Symbolic Data Files directly. This access is provided by a set of subroutines in the file "symb.c". Some fairly complex processing and pointer manipulation occurs inside these routines, so it is best to leave the complexity inside them and treat these routines as the primitive procedures.

At the next level are programs that use the routines in "symb.c". This includes the segmentation programs "seg_pass0.c" through "seg_pass4.c," "init_symb," "prt_symb," "bld_man," and "fadt" and the Attribute Manager "am". These programs tend to function independently of the actual layout of the Symbolic Data Files.

The highest level programs are those that the user actually uses to access symbolic data. They are "edit_symb" and "newsip," both of which use the Attribute Manager. These are the programs the user actually works with and which require no knowledge of the details of symbolic data.

3.3.1 Low-Level Symbolic Data Access

All low-level symbolic access is provided by routines in "symb.c". A list of symb.c routines is provided below along with a brief description of the symb.c functions.

Three kinds of routines are provided in symb.c. Some routines merely open files for writing, possibly checking their size. Others read symbolic data, and the rest write symbolic data. The routines that open files are:

init_sdf(symb_dir) Allocate space for storing necessary information about the symbolic data files. Initialize the data in the structure. Returns a pointer to the newly allocated storage for use by all other routines in symb.c.

open_rh(sdf_ptr) Open the region_hdr file and determine the highest region present.

open_ri(sdf_ptr) Open the region_ident file.

open_eh(sdf_ptr) Open the edge_hdr file and determine the highest edge present.

open_ei(sdf_ptr) Open the edge_ident file.

open_at(sdf_ptr) Open the atr file.

open_ky(sdf_ptr) Open the atr_key file and determine the highest key number present.

openall(symb_dir) Do all of the above at once for symbolic directory symb_dir. Returns sdf_ptr for use by subsequent accesses.

The "init_sdf" routine returns a pointer to a structure. This pointer must be used for all the other routines. It is called "sdf_ptr" above. "symb_dir" is a pointer to a character array containing the name of the symbolic directory. The following routines read symbolic data files once they have been opened:

rrect(sdf_ptr,~region) rrect reads the minimum enclosing rectangle information for the given region.

rmask(sdf_ptr) rmask reads mask entries.

rcinfo(sdf_ptr,~edge) rcinfo reads information about the chain code for an edge from the edge_ident file, but does not read in the chain code itself.

rccode(sdf_ptr,~edge) rccode executes rcinfo and, additionally, reads in the chain code.

ratrcod(sdf_ptr,~r_e,~reg_edg,~a_code) ratrcod reads in an attribute from the atr file for either a region or an edge.

lookup(sdf_ptr,~a_name) lookup tries to match the attribute name pointed to by its argument with the attribute names in the atr_key file.

The following routines write into one or more symbolic data files:

wrect(sdf_ptr,~region) wrect writes the minimum enclosing rectangle for the named region.

wmask(sdf_ptr) wmask writes a mask entry.

wccode(sdf_ptr,~edge) wccode writes information about the chain code for an edge into the edge_ident and writes the chain code itself.

watrcod(sdf_ptr,~r_e,~reg_edg) watrcod writes an attribute into the atr file for either a region or an edge.

addkey(sdf_ptr,~keyname,~format) addkey adds the named key to the atr_key file with the named format.

xatrcod(sdf_ptr,~r_e,~reg_edg~,a_code) xatrcod deletes an attribute from the atr file for either a region or an edge.

xatrval(sdf_ptr,~r_e,~reg_edg~,a_code~,atr_val) xatrval deletes an instance of an attribute with the value specified by atr_val.

catrcod(sdf_ptr,~r_e,~reg_edg~,a_code~,atr_val) catrcod changes the value of an attribute from the atr file for either a region or an edge.

3.3.2 Intermediate-Level Symbolic Data Access

Intermediate-level access to symbolic data is provided by a variety of programs that use "symb.c". Included are some segmentation programs, some utility programs, some DRLMS programs, and the Attribute manager.

3.3.2.1 Segmentation Programs

Segmentation programs are the most important programs that use the Symbolic Data Files, since they do most of the creation of symbolic data. This includes "seg_pass0.c," "seg_pass1.c," "seg_pass3.c," and "seg_pass4.c".

"seg_pass0.c" and "seg_pass1.c" perform the first pass of segmentation for edge-based methods and pixel-based methods, respectively. These programs assign each pixel to one of many regions. Since they know about all regions,

it is only natural that they should add as much as possible to the Symbolic Data Files. This is limited to "parent," "offspring," "method," "areal," "resides_in," and "class" attributes for regions. "seg_pass1.c" adds the names of all classes to the atr_key file; "seg_pass0.c" adds the class "object" to it because, when performing edge-based segmentation, that is the only class.

"seg_pass3.c" adds mask codes to the region_ident file and assigns the areal and resides_in attributes.

"seg_pass4.c" does all edge processing for segmentation. It adds the edges to the edge_ident file, gives them leftregion and rightregion attributes, gives them a parent if necessary, attaches edge attributes to regions, and attaches offspring attributes to any edges that have yielded offspring.

3.3.2.2 Utility Programs

Some utility programs exist for initializing and viewing Symbolic Data Files. This category includes "init_symb" and "prt_symb".

"init_symb" is used to remove all symbolic data and to create new files containing only the most elementary data. It calls on "init_r_e.c" to actually create the data. This program creates a mask for r1 and a chain code for e0. It gives both r0 and r1 an attribute of "edge 0". It makes r0 the rightregion of e0, and r1 the leftregion. This is an initialized Symbolic Data Base.

"prt_symb" is used to print the symbolic data for regions and edges. It calls on "prt_symb.c" to perform the actual symbolic data access. Valid input is of the form "r 0," "e 1," "g 1," etc. For regions, it prints the enclosing rectangle, masks, and all attributes. For edges, it prints the start and stop nodes, chain length, and all attributes. For grid cells, the occupant types

and numbers are printed. The actual chain code is not printed.

3.3.2.3 DRLMS Programs

DRLMS programs such as fadt and bld_man produce a Feature Analysis Data Table and a Feature Manuscript, respectively. These programs must use symbolic data to display it and, also, to store facts about the regions and how to label them.

"fadt" produces a tabular listing of certain attributes for inclusion in an FADT. This program is limited to reading attribute values. It does not modify the Symbolic Data Files in any way. A typical FADT is shown in Table 3-1.

"bld_man" produces a feature manuscript for displaying. It uses symbolic data to display region outlines, and, also, to record the locations of labels that the user might place on the display. Figure 3-3 is a photograph of the bld-man manuscript editing function, in which the user places labels and key lines on the manuscript and modifies the feature type if desired. The feature being edited is outlined in red. Figure 3-4 is the finished manuscript.

3.3.2.4 Attribute Manager

The Attribute Manager is an intelligent interface between the Symbolic Data Files and the Symbolic Image Processor. It acts as a data base manager, performing information retrieval, deletion, and updates. A list of the commands (and a short description of each) follows:

Table 3-1 Feature Analysis Data Table

Page 1 of 3 Pages

ATTRIBUTES	FAC 0	FAC 1	FAC 2	FAC 3
featuretype				
smc				
height				
nostructures				
percenttrees				
percentroof				
featureid			null	background
orientation				
length		512.000012	512.000012	507.000006
width				
level				

ATTRIBUTES	FAC 4	FAC 5	FAC 6	FAC 7
featuretype				
smc				
height				
nostructures				
percenttrees				
percentroof				
featureid	plane	plane		
orientation				
length	46.837170	36.000001	4.000000	15.000000
width			4.000000	
level				

ATTRIBUTES	FAC 8	FAC 9	FAC 10	FAC 11
featuretype				
smc				
height				
nostructures				
percenttrees				
percentroof				
featureid		plane	plane	plane
orientation				
length	19.000000	50.864949	51.000001	60.044686
width				
level				

Table 3-1 Feature Analysis Data Table

Page 2 of 3 Pages

ATTRIBUTES	FAC 12	FAC 13	FAC 14	FAC 15
featuretype				
smc				
height				
nostructures				
percenttrees				
percentroof				
featureid	plane	building		
orientation				
length	41.000001	179.000003	11.000000	7.000000
width				
level				

ATTRIBUTES	FAC 16	FAC 17	FAC 18	FAC 19
featuretype				
smc				
height				
nostructures				
percenttrees				
percentroof				
featureid				plane
orientation				
length	14.532860	15.000000	15.000000	96.0000011
width				
level				

ATTRIBUTES	FAC 20	FAC 21	FAC 22	FAC 23
featuretype				
smc				
height				
nostructures				
percenttrees				
percentroof				
featureid				plane
orientation				
length	12.000000	12.000000	10.000000	96.000001
width				
level				

Table 3-1 Feature Analysis Data Table

Page 3 of 3 Pages

ATTRIBUTES	FAC 24	FAC 25	FAC 26	FAC 27
featuretype				
smc				
height				
nostructures				
percenttrees				
percentroof				
featureid				
orientation				
length	5.000000	11.000000	15.000000	3.000000
width				
level				

ATTRIBUTES	FAC 28	FAC 29
featuretype		
smc		
height		
nostructures		
percenttrees		
percentroof		
featureid		
orientation		
length	3.000000	2.000000
width		
level		

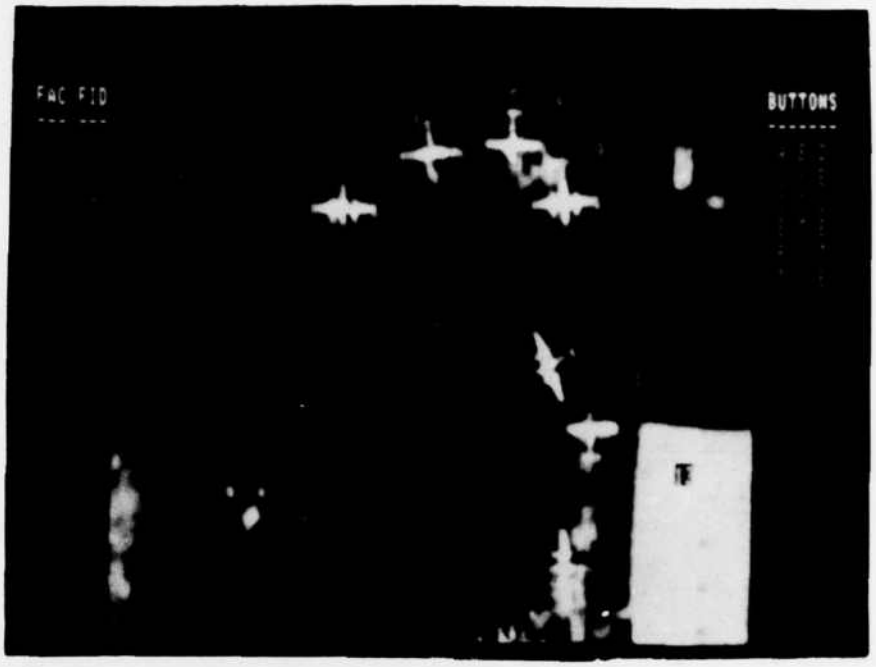


Figure 3-3
Feature editing being applied to feature manuscript

Figure 3-4

Final edited feature manuscript

NAME: put

SYNTAX: (put <-r region> || <-e edge> || < - g grid > attribute_name value)

FUNCTION: Assign a value to an attribute of a region or edge.
If attribute_name is not in the atr_key file, it is added.

NAME: get

SYNTAX: (get <-r region> || <-e edge> || < -g grid > attribute_name)

FUNCTION: Return the values assigned to an attribute of a region or edge.
If attribute_name is not in the atr_key file, it is added.
If there is no value assigned, but there is a region or edge measurement extractor for the attribute, it will be run, otherwise "()" is returned.

NAME: change

SYNTAX: (change <-r region> || <-e edge> || < -g grid > attribute_name value)

FUNCTION: Change the value of an attribute of a region or edge.
If there are several values for the region or edge, only the first is changed. If there is no value, change is identical to put.

NAME: deleteall

SYNTAX: (deleteall <-r region> || <-e edge> || < -g grid > attribute_name)

FUNCTION: Delete all the values of an attribute of a region or edge.

NAME: deletespec

SYNTAX: (deletespec <-r region> || <-e edge> || < -g grid >
attribute_name value)

FUNCTION: Delete the values of an attribute of a region or edge which have a specific value.

NAME: xmt

SYNTAX: (xmt method [regions])

FUNCTION: Execute the method for the regions given. This method may be either a region method in which the class type of the region would be returned, or a pixel or edge method followed by segmentation. If no regions are given, region 1 is used for pixel methods and all regions in the image are used for region methods.

NAME: merge

SYNTAX: (merge <-r regions> || <-e edges>)

FUNCTION: Merge the list of regions [edges] given to form a new region [edge]. When edges are merged, the list of edges must form a single, non-self intersecting chain or loop.
The old regions or edges remain intact.

NAME: newedge

SYNTAX: (newedge row1 col1 row2 col2)

FUNCTION: Create a new edge in the symbolic data file going from (row1,col1) to (row2,col2).

NAME: newregion

SYNTAX: (newregion edges)

FUNCTION: Create a new region in the symbolic data file that contains the area bounded by the edges listed.

NAME: nodes

SYNTAX: (nodes edge)

FUNCTION: Return the nodes of the edge specified.

NAME: deleteregion

SYNTAX: (deleteregion <region #>)

FUNCTION: Delete all references to the named region and its edges. The mask codes for the deleted region will be merged with those of the neighboring region. Deletion will only be successful if the updated region mask is shorter than or equal to the original mask representation.

NAME: cells

SYNTAX: (cells row1 col1 row2 col2)

FUNCTION: Get the list of grid cells which cover all the pixels within the rectangle described by the corner points (row 1, col1) and (row 2, col2). The rectangle described is clipped if part of it falls outside of the image.

NAME: included

SYNTAX: (included region)

FUNCTION: Get the list of regions which are siblings of the region and are wholly contained within it.

NAME: transfer

SYNTAX: (Transfer region target symbolic directory)

FUNCTION: To copy a region and its bounding edges to a second symbolic image.

NAME: XYZ

SYNTAX: (XYZ Lrow Lcol Rrow Rcol)

FUNCTION: To return latitude, longitude, elevation and residual parallax for a pair of matching points between two images in a stereo pair.

NAME: Background

SYNTAX: (Background symbolic image directory)

FUNCTION: Builds the background region (region 1) in a geographic symbolic image from the exterior edges of the matched regions which have been transferred into the image.

Two of the AM commands allow the creation of new regions and edges. The "newregion" command allows a new region to be defined based upon a set of edges, and the "newedge" command allows a new edge to be defined from its endpoints. The question arises: how well does this agree with the definition of region and edge? It does not, which leads to a redefinition of regions and edges.

The original definitions of regions and edges were given in section 3.1. The original definitions are correct only for regions and edges that are derived as a result of segmentation. Normally, that will be all or nearly all regions and edges. However, it is possible to define new regions and edges using the Attribute Manager.

The "newedge" command creates a new edge from the endpoints. Such an edge has a start node, a stop node, and a chain code representation. It does not have a rightregion or a leftregion. The original definition of edge

required that every edge separate two regions. However, this constraint cannot possibly be met when new edges are defined by the AM. Therefore, we will simply drop the region separation requirement for new edges.

The "newregion" command creates a new region from its edges. Such a region has a mask representation and a set of edges, but it will not necessarily consist of pixels of the same class. Therefore, we will drop the class requirement for new regions.

3.3.3 High-Level Symbolic Data Access

High-level access to symbolic data can be had by using "edit_symb" and "newsip". "edit_symb" is an interface from the user to the Attribute Manager. The syntax is identical to that for the AM, so we will not discuss "edit_symb" further here.

"newsip" provides access to symbolic data through the use of rules. Rules encode knowledge about image interpretation. There are several types of rule; one type tests the Symbolic Data Base to see if a particular value or range of values is present for some attribute. The rules that perform testing are called Evidence Rules. Another type adds assertions to the Symbolic Data Base, rules of this type are called Feature Assignment Rules. By using Evidence and Feature Assignment Rules, testing and updating symbolic data is made easy, and the implementation details are totally hidden from the rule-writer. The following are examples of Evidence and Feature Assignment Rules:

```
(evidence Urban
        text "the current region has class urban"
        prior 0.1
        action ( (test (current) with (class = urban)) )
)
```

```
(evidence Offspring_of_Firstregion
        text "current region is an offspring of region 1"
        prior 0.01
        action ( (test (= 1 parent) with (firstregion) ) )
)


(evidence Urban_Neighbor
        text "region has 1 urban neighbor"
        prior 0.2
        action ( (test (= 1 neighbor) with (class = urban) ) )
)


(evidence No_Nonurban_Neighbor
        text "region has 0 nonurban neighbors"
        prior 0.2
        action ( (test (= 0 neighbor) with (class <> urban) ) )
)


(goal Label_House
        text "label current region with house tag"
        prior 0.01
        antecedents (and House_Shape House_Size Residential_Region)
        action ( (assign label house) )
)


(goal Assign_SMC6
        text "water features must be SMC (Surface Material Category) 6"
        prior 0.5
        antecedents (and Water_Class)
        action ( (assign smc 6) )
)
```

Each Evidence Rule is composed of the word "evidence," the rule name, a short description, the prior probability and the action to be performed, in these cases a test. Each Feature Assignment Rule is composed of the word "goal",

the rule name, a short description, the prior probability, a set of antecedents, and the feature assignment action to take. As can be seen, no knowledge of what is actually in the Symbolic Data Base is actually required, thus making "newsip" the highest level of symbolic data access.

4. PIXEL PROCESSING OPERATIONS AND MEASUREMENT EVALUATION

The APR effort has included implementation of a number of pixel processing and region and pixel measurement extraction commands. In the subsections which follow we provide descriptions of these commands and photographs which illustrate typical output from some of the operations. The APR menu sections containing these commands are:

- Measurement Extractors
- Edit Programs
- Region Measurements
- Edge Measurements

The last subsection discusses structure analysis and measurement commands.

4.1 MEASUREMENT EXTRACTORS

Measurement extractors implemented within the APR effort include five edge extraction and filtering operators, plus two commands for spatial compression and expansion of images.

4.1.1 wiener_edge

This command extracts edges using optimal wiener filtering. The program "wiener_edge" is an interface to the program "efilt," which does the actual filtering. "wiener_edge" is called with a parameter f, m, or c designating fine, coarse, or smooth filtering. For each case efilt is executed with suitable parameters:

parameter	coarse	fine	medium
overlap	no	no	no
a01	-.4815	-.1944	-.3075
a11	-.0102	-.1929	-.2046
b01	-.1222	-.0381	-.0676
b11	-.7556	-.9238	-.8648
A	.88	.084	.034
radius	2.0	2.0	2.0
offset	0	0	0
output format	s	s	s

Next edge is called with appropriate parameters.

parameter	coarse	fine	medium
factor	.2	.2	.2
thresh	8000	20	5

"Efilt" applies a two dimensional recursive filter consisting of the sum of four sections each coming from one quadrant of the image.
Each section consists of:

```
y(i,j)=A*{
       -a10*(y(i-1,j)+y(i,j-1))-a11*y(i-1,j-1)
       +x(i,j)
       +b10*(x(i-1,j)+x(i,j-1))+b11*x(i-1,j-1)
          }
```

which has Z transform

$$H(Z1,Z2)=A\,\frac{1+b10*(Z1^{-1}+Z2^{-1})+b11*Z1^{-1}Z2^{-1}}{1+a10*(Z1^{-1}+Z2^{-1})+a11*Z1^{-1}Z2^{-1}}$$

efilt first opens the input and output directories and makes sure the input directory header is in good order. Next it reads in the filter parameters. It then carries out the filtering operation moving down through the image one line at a time.

i) initialize the first previous output line for
both the left to right and right to left sections
to zero.
ii) read in first line of image
iii) for each line of the image
a) read another input line
b) process left to right section
c) process right to left section
d) sum left to right and right to left and output it

Then it carries out the filtering operation moving up through the image one line at a time adding result of up pass to down pass.

i) initialize as in down pass
ii) read in bottom line
a) read in input for next line up
b) read in output from down pass for this line
c) process left to right
d) process right to left
e) sum left to right and right to left and add
to sum from down pass.

"Edge" carries out the thresholding portion of the wiener edge operator. For each 3 by 3 window, this program takes in the filter output and calculates the following:

```
dif1=abs(x(0,0)-x(2,2))
dif2=abs(x(1,0)-x(1,2))*1.4
dif3=abs(x(2,0)-x(0,2))
dif4=abs(x(2,1)-x(0,1))*1.4
dif=max(dif1,dif2,dif3,dif4)
sum=sum of all elements in 3 by 3 window
```

If the quantity (dif*dif-factor*sum*sum) is greater than thresh the point is declared as an edge point.

Operation of the wiener edge operator is shown in Figures 4-2 and 4-3 for the input image of Figure 4-1. Figure 4-2 has wiener edge pixels for the "f" filter option shown in green, while Figure 4-3 has edge pixels for the "c" option in red. Figure 4-4 shows both the "c" and "f" outputs superimposed and magnified so that individual pixels may be visualized. Those pixels depicted in yellow are edge pixels detected at both c and f resolutions. The differences between the c and f options are readily apparent.

Figure 4-1 Test image

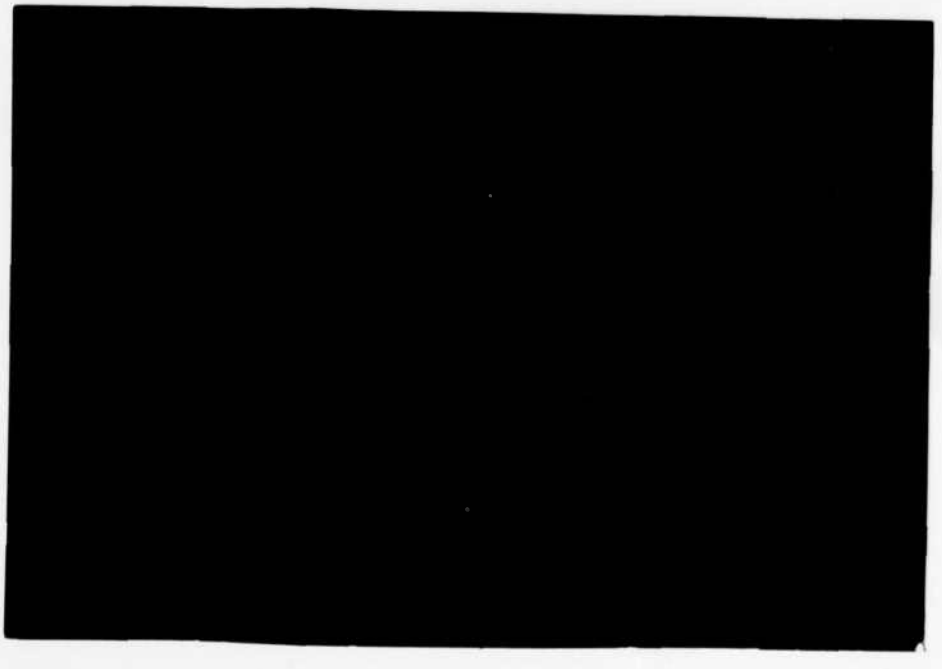


Figure 4-2 wiener_edge with f option

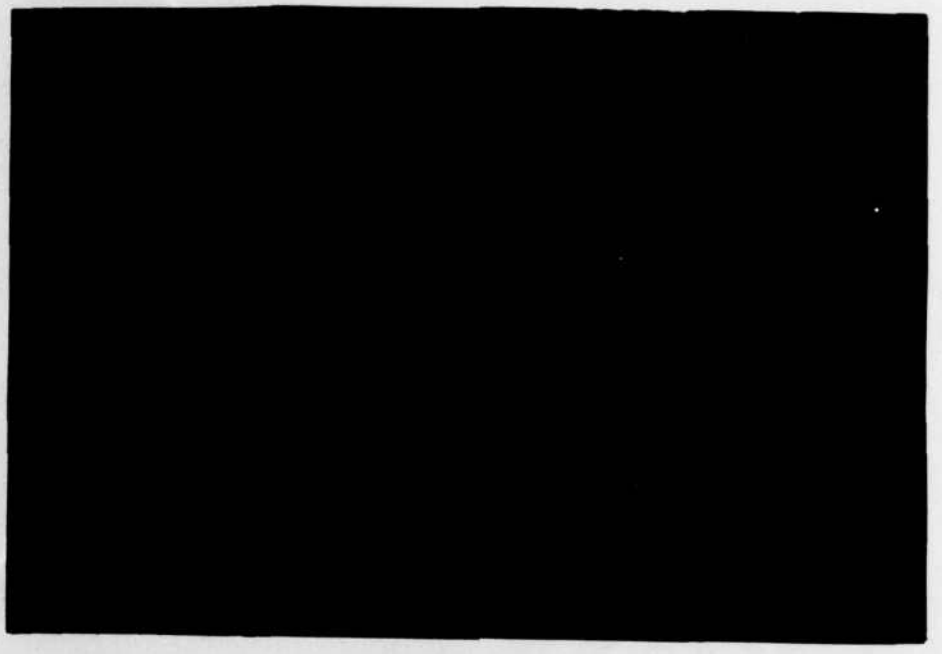


Figure 4-3 wiener_edge with c option

Figure 4-4 wiener_edge f (green) and c (red) superimposed

4.1.2 sobel_dir

This routine outputs the sobel edge detection magnitude and direction as a two element vector for each pixel, with both the magnitude and the direction represented as characters. The sobel magnitude is defined as:

$$Fs(x,y) = .125 \left[\left| f(x,y) \langle * \rangle \begin{vmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{vmatrix} \right| + \left| f(x,y) \langle * \rangle \begin{vmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{vmatrix} \right| \right]$$

where: Fs(x,y) is the Sobel value at (x,y).
f(x,y) is the 3 x 3 input window centered at (x,y).
<*> denotes "convolution".

Therefore in the 3 x 3 window : a b c
d e f
g h i

this equation in effect yields:

$$Fs(x,y) = .125 \left[|a + 2b + c - g - 2h - i| + |a + 2d + g - c - 2f - i| \right]$$

Output also includes the Sobel direction which is defined as:

$$Fs(x,y) = \arctan2 \left[f(x,y) \langle * \rangle \begin{vmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{vmatrix} , f(x,y) \langle * \rangle \begin{vmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{vmatrix} \right]$$

Since sobel_dir generates a vector for each point, with each point giving rise to two characters, it cannot be displayed in a meaningful fashion. Figure 4-5 shows the magnitude portion of the sobel operator as white pixels. It should be compared with Figures 4-2 and 4-3 to see the differences between the wiener and sobel operators.

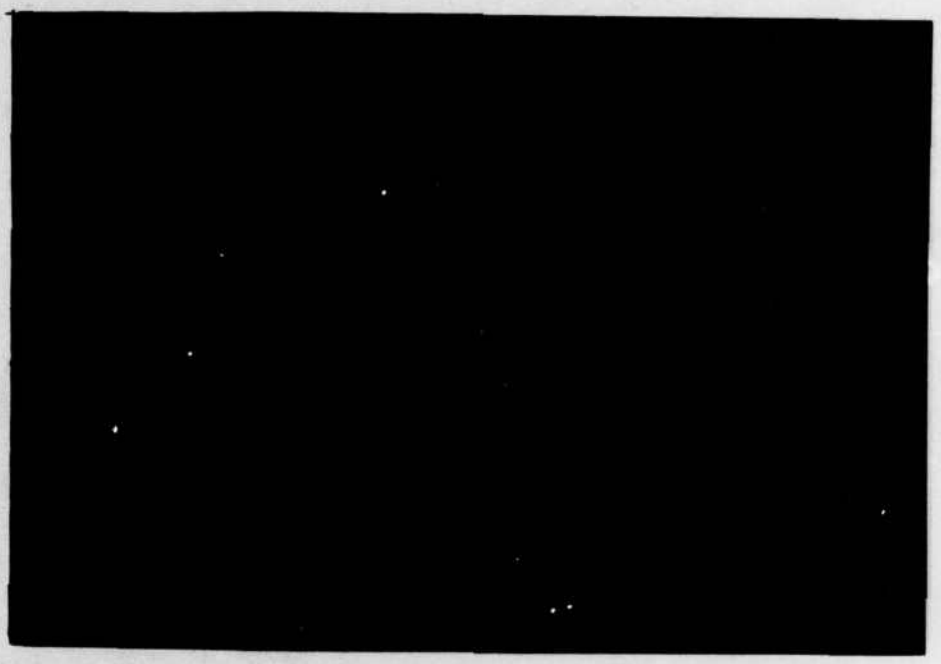


Figure 4-5 sobel magnitude

4.1.3 dir_filter

"dir_filter" implements a directional filter. Its input is edge magnitudes and directions at each pixel. The filter will strengthen or weaken the edge values depending on the degree of corroboration provided by neighboring edge strengths and directions. The single parameter input by the user specifies the size of the neighborhood to be used.

Each pixel's edge magnitude is examined to see if it has been identified as an edge pixel. If so, the surrounding pixels are examined in order to identify other edge pixels having the same edge orientation (direction). The magnitude of edge pixels whose direction is consistent with neighboring pixels is strengthened, and the pixel is assigned an average direction for the local neighborhood.

"dir_filter" is run on the output data file produced by "sobel_dir". Its single parameter is the dimension of the square window over which the local edge directions and magnitudes are to be examined. The window dimension must be an odd number. The output of "dir_filter" is another magnitude-direction file with format identical to that of the input file.

4.1.4 mdmf

"mdmf" is a maximal directional edge filter. Its input file is a thresholded edge point image. It matches the edge points within a local window to an edge mask which is rotated through 360 degrees, and which includes orthogonal suppression. An example of the filter is given below. The input parameters for "mdmf" include the lengths of the edge arm and suppression arm of the filter and, also, the value of the edge significance threshold. For example, the mask for edge arm length = 7 and suppression arm = 7 looks like this:

```
 0  0  0  0  0  0  0  0  0
 0  0  0  0 -1  0  0  0  0
 0  0  0  0 -1  0  0  0  0
 0  0  0  0  0  0  0  0  0
 0  1  1  1  1  1  1  1  0
 0  0  0  0  0  0  0  0  0
 0  0  0  0 -1  0  0  0  0
 0  0  0  0 -1  0  0  0  0
 0  0  0  0  0  0  0  0  0
```

This filter is rotated through 16 directions. If the value of the filter in its maximal direction is greater than the significance threshold, the direction number is written as the output value; if not, a 0 is output. The result is that isolated edge pixels are removed and edge points which are collinear with other edge points are retained.

The operation of "mdmf" can be illustrated by running the AFES program "athres" on the output of the sobel operator of Figure 4-5. The athres output is in Figure 4-6. The output of mdmf is shown in Figure 4-7, with output pixels overlaid in green on the red athres output pixels.

4.1.5 edge fill

This program implements an edge filler. It only affects pixels that are not yet classified as edge, but for which the neighboring edge pixels indicate the possibility of an edge. The edge strength and direction are computed based upon weighted sums of neighbor edge strengths and directions.

The window dimension for "edge_fill" must be odd, and its input must consist of vectors of dimension 2 consisting of edge magnitude and direction, such as that produced by "sobel_dir". In addition to input and output file names, "edgefill" accepts a window dimension parameter and a threshold for testing the strength of the edge likelihood.

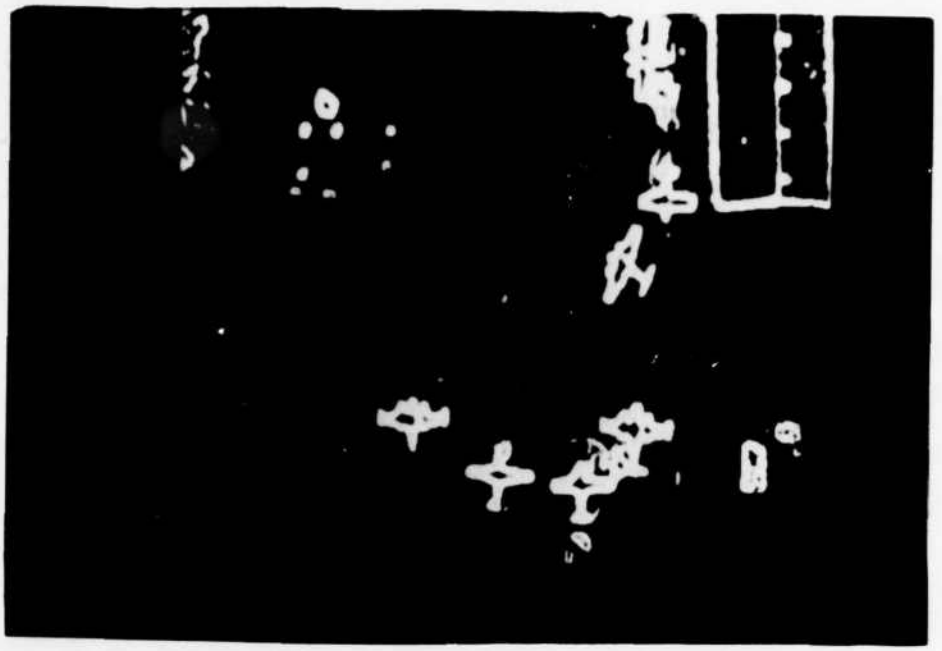


Figure 4-6 athres output

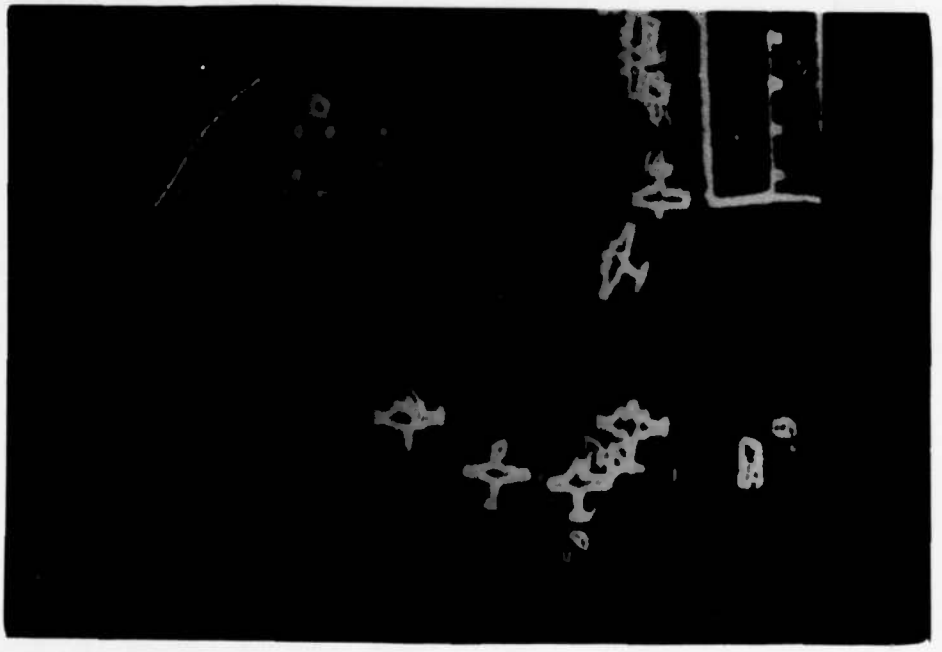


Figure 4-7 athres output (red) with mdmf output (green) superimposed

4.1.6 squeeze_i

"squeeze_i" reduces the resolution of an image by averaging pixels together. Its use in APR is to generate reduced resolution images when it is desired to do a "gross" classification of an image into classes, such as urban and rural. Its syntax is

squeeze_i <input image> <output image> <row factor> <col factor>

4.1.7 expand_ipr

This program does the opposite of "squeeze_i" in that it accepts row and column expansion factors and expands an image by an integral amount. Syntax is

expand_ipr <input image> <output image> <row factor> <col factor>

4.2 EDIT PROGRAMS

The Edit Programs are used to process classification output data. They are designed to operate on the image which results from a classification procedure. Classifier output exists for the current working image. The two commands which are illustrated are "mode_filter," which removes extraneous small regions from a classified image, and "edge_thin," which thins edges and removes isolated edge pixels.

4.2.1 mode_filter

Mode filtering involves replacing a pixel's class with the most frequently occurring class in a window. The arguments depth and width specify the window size. If the -old option is used, the original classification file is used. Otherwise, the default is the edited version, if it exists. In either case, the result is output to the edit directory.

Figure 4-8 is the output from a pixel method which will serve as our input to mode_filter. The background region is blue; potential objects of interest are shown in green. The results from mode_filter are in Figure 4-9; note the small "holes" in the building in the upper right have been removed. There are a number of other very small regions in the classifier output which do not appear in the edited version.

4.2.2 edge_thin

The "edge_thin" program is similar to "mode_filter," except that it operates on images classified via edge methods. It reduces the width of edges to a single pixel by changing the class of certain pixels from "edge" to "non-edge". edge_thin is called repeatedly until no more pixels can be changed or until the maximum number of iterations is reached. If no maximum number of iterations is given, the process will repeat until no more pixels can be changed. If the -old option is given, the classified output is used; otherwise, the edit directory file is used. In either case, results are stored in the edit directory. The maximum number of edges must follow the -old flag if both are present.

If called without a limiting number of iterations, "edge_thin" is guaranteed to remove all edges that are not part of a closed region. However, this is more processing than is usually necessary. Segmentation may benefit just as well from a partial edge thinning, say 1 to 3 iterations. Limiting the iterations will not necessarily remove all unnecessary edges, but it will reduce the thickness of those edges that are left and be more time-efficient. The great savings in time may make it worthwhile to limit the iterations.

The program examines 3x3 windows and decides if the center pixel should be changed from edge to non-edge (since this is edge thinning, we are only interested in cases where the center pixel is an edge). There are two conditions that must be satisfied in order to delete the center pixel. Let each pixel in the window have a number as follows:

Figure 4-8 Classifier output for pixel method

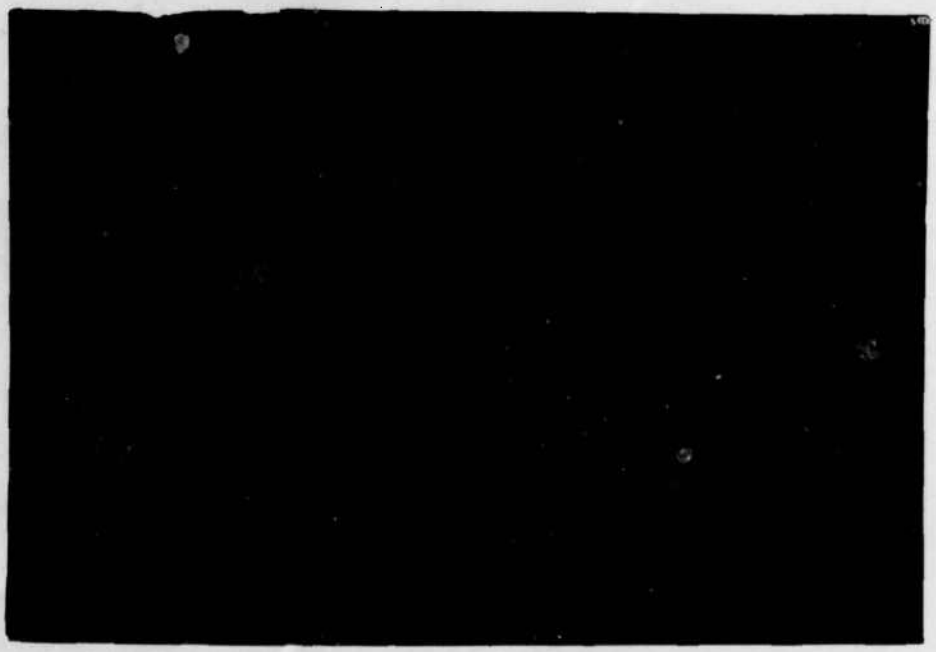


Figure 4-9 mode filter output for Figure 4-8

```
8 1 2
7 0 3
6 5 4
```

(We are assuming throughout that pixel 0 is an edge.)

The conditions are:

1. By deleting the center pixel we do not create any new regions. So if we have:

```
. * .
* * *
. * .
```

Where the * indicates an edge and . indicates non-edge. Now if we delete pixel 0 we will get:

```
. * .
* . *
. * .
```

In this case a new region has been created consisting solely of pixel 0. The center pixel should not have been deleted.

The condition can be stated as:

At least one of pixels 1, 3, 5, 7 must not be an edge in order to delete pixel 0.

2. By deleting the center pixel we do not merge any old regions. This can be restated by saying that the connectivity of the edge pixels does not change by deleting pixel 0. For example:

```
. * *
. * .
. . .
```

Here the connectivity is 1. All edge pixels are connected. If pixel 0 is deleted, the connectivity is still 1. Therefore it can be deleted. However, if we have:

```
. * *
. * .
* . .
```

The connectivity here is also 1. But if pixel 0 is deleted, the connectivity will be 2 (pixels 1 and 2 will be a connected component and pixel 6 will be a connected component). Therefore the center pixel cannot be deleted.

There is a special case where the center pixel can be deleted even though the connectivity of the edges changes. That is when the only edge pixel in the window is the center one.

The only parameter used by "edge_thin" is the maximum number of iterations

which will be applied. It is usually instructive to just run one iteration and then compare the input and output images. Figure 4-10 is the classifier output from an edge method using the wiener edge operator. The image is shown in red with the edge pixels overlaid in blue. After one iteration of edge_thin we get the results shown in Figure 4-11 with, again, the remaining edge pixels shown in blue on a red background image. The effects of edge thinning are obvious.

4.3 REGION AND EDGE MEASUREMENTS

The region and edge measurements for a segmented image are computed by the attribute manager. The easiest way to observe the action of these measurements is with "edit_symb," which is an interface to the attribute manager. There are two commands in "edit_symb" which can be used to determine where an edge and region are and what the computed attributes of that region and edge are.

The measurements available include

```
*** Region Measurements ***

area         - compute the area
perimeter    - compute the perimeter
p2overa      - perimeter squared divided by area
length       - length of minimum enclosing rectangle
width        - width of minimum enclosing rectangle
ave_inten    - average intensity of the region
moment_bnd   - moment of the boundary of the region
x_centroid   - x centroid position of the region
y_centroid   - y centroid position of the region
holes        - number of holes in a region

*** Edge Measurements ***

length       -   the actual length of the chain code
distance     -   the distance between the endpoints
edge_complexity - the complexity of an edge
```

These are all self-explanatory except for "moment_bnd" and "edge_complexity".

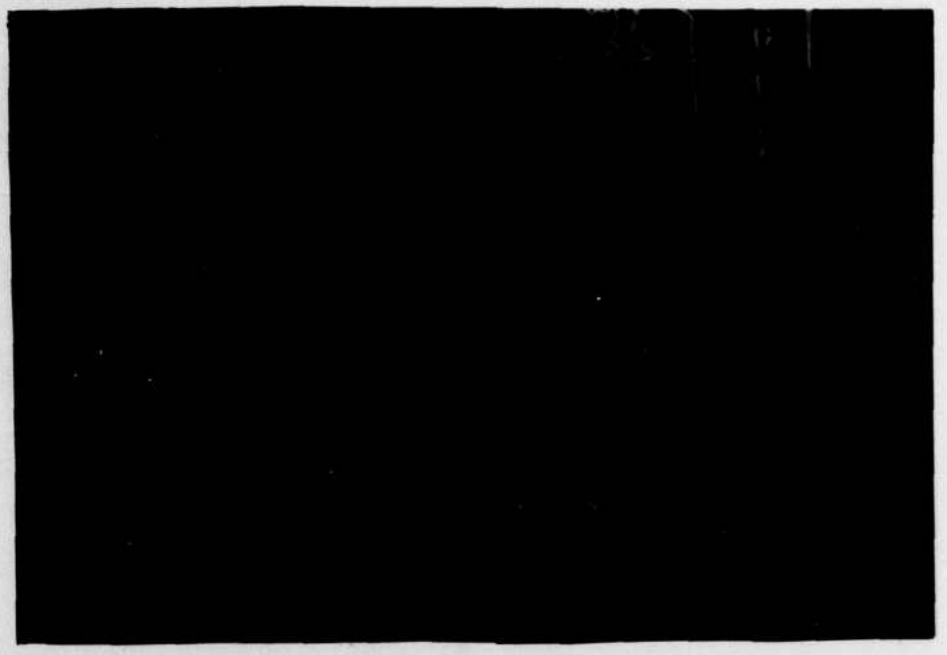


Figure 4-10
Classifier output for edge method using wiener_edge

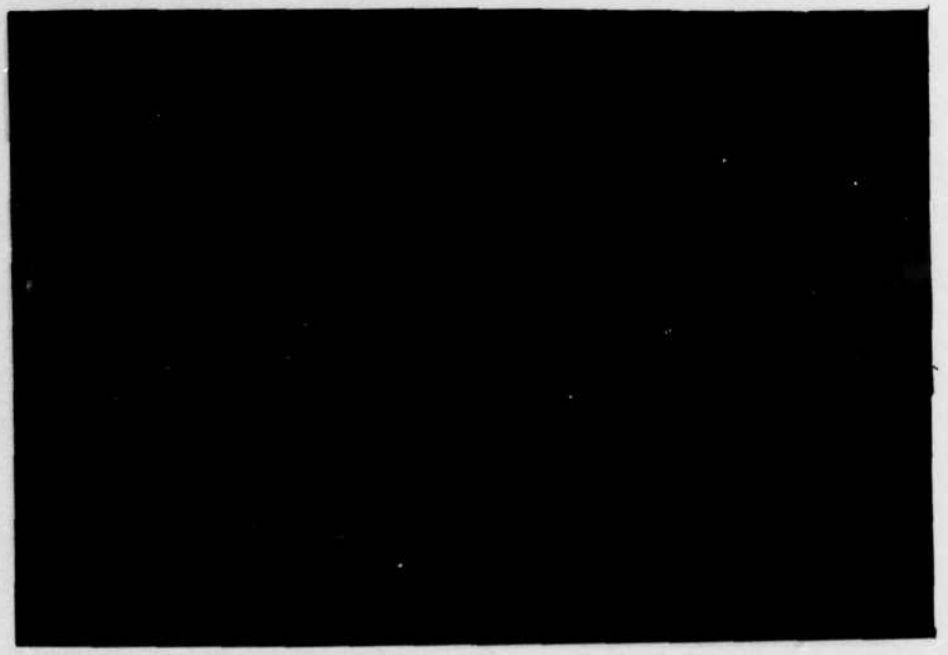


Figure 4-11 Results of iteration of edge_thin on Figure 4-10

"moment_bnd" computes the rotational moment of an object boundary about the "center of mass" (COM) of that boundary. The simplest way to do this would be to compute the COM, then go back and compute the moment. However, due to the following, we can do it with only one pass of the chain data.

I = sum of D*D, where D = distance from boundary point to COM
I = rotational moment.

= sum((x - xc)^2 + (y - yc)^2) '^' denotes exponentiation.
sum denotes summation.
(x,y) = coords of a boundary pnt.
(xc,yc) = coords of COM

= sum(x^2 - 2x * sum(x)/N + (sum(x))^2/N + ... N = no. of bound points.

= sum(x^2) - 2 * (sum(x))^2/N + (sum(x))^2/N + ...

= sum(x^2) - (sum(x))^2 / N + sum(y^2) - (sum(y))^2 / N

"edge_complexity" is based on an AFES program called "vector," which converts the chain-coded boundaries generated by region_bnd to vector boundaries. Edg_cmplx converts the chain-code to vectors to smooth it. It then computes the sum, for all the angles, of:

angle * small / large

where small is the length of the smaller of the two vectors adjacent to the angle and large is the length of the larger.

Chain code is converted to vector code in one of the two following fashions:

Method 1 (last good point): The vector begins at the first coordinate pair (either the first point of the boundary or the last vector's endpoint). Each succeeding chain code increment defines a new endpoint for the vector. If, when drawing a straight line from the start point to the end point, any intervening coordinate pair has a distance from that line greater than a user definable delta, extension of that vector terminates, and the endpoint is moved back to the previous point. The default delta is 1.0. Any delta less than half the square root of 2 (0.707) will result in vector code that is identical to chain code. As deltas get greater than 1.0, the vectors form even rougher approximations to the original boundary. The attribute manager for APR uses a delta of 2.0 when it calls "edg_complex".

Method 2 (worst midpoint method): Proceeds as in method 1, except that the endpoint is deemed to be the point that was the farthest from the line.

Method 1 generally smooths better, whereas method 2 will follow the contours more closely. The current implementation of the APR attribute manager calls "edg_cmplx" with the first option (last good point).

4.4 MEASUREMENT EVALUATION / STRUCTURE ANALYSIS

The measurement evaluation and structure analysis capabilities are available through top-level programs called "meas_eval" and "structure," respectively. Measurement evaluation is used to analyze the quality of the training data which has been collected prior to region classification. Structure analysis is used to examine and edit classification results.

The top level of the meas_eval program presents a menu:

0 Explanation of Selections
1 Discriminant Measures
2 Probability of Confusion Measure
3 Exit Program

The discriminant measure refers to the ability to distinguish objects of two different classes. The higher the value, the better the distinction between those objects. It is calculated as the ratio of the distance between the centers of the two distributions (in feature space) squared to the sum of each object's spread ("variance" from average).

The probability of confusion measure represents the overlapping of distribution in feature space. The value of confusion is smaller for less overlap. Thus, a value of zero confusion between two classes indicates that there exists very little overlap between their distributions.

Each measure has a menu that describes its output options. Figure 4-12 shows an example of terminal output for the discriminant measure in which three measurements are ranked for a simple target vs. background discrimination.

The structure command allows the user to examine and edit training region data for the current object method for the purposes of improving classification results. Output plots for structure are sent to the Tektronix display. Note that the "restructure" option cannot be used for pixel methods, only for region methods. Figures 4-13 through 4-16 are typical structure output. Figure 4-13 is a scatter plot containing samples of class "light" (labeled "A") in the upper right corner and class "background" (labeled "B," but heavily overlapping) very near the origin. Figure 4-14 is a cluster plot in which local clusters of samples are identified by a single letter. A group of "A" clusters occurs in the upper right corner and the "B" cluster appears near the origin. Figure 4-15 is a combined histogram of A and B samples for one measurement, avg 3. Note the single "B" peak near zero on the horizontal intensity bin scale and the "A" peak near 255 on the vertical count-per-bin scale. Finally, Figure 4-16 presents separate histograms of avg 3 for the two classes, A and B. B (background) has a peak near zero on the horizontal intensity bin scale and A (light) has a peak near 255 on a horizontal count-per-bin scale. The separation is readily apparent.

```
Measurement Evaluation Categories

0 - Explanation of Categories
1 - Discriminant Measures
2 - Probability of Confusion
3 - Done

Category = 1

Output Menu For "Discriminant Measures"

0 - Explanation of Categories
1 - Rank Measurements for a specified Class
2 - Rank Measurements for a specified Class Pair
3 - Rank Classes for a specified Measurement
4 - Rank Class Pairs for a specified Measurement
5 - Union of Best Measurements for each Class
6 - Union of Best Measurements for every pairwise set of
    Classes
7 - Rank Measurements Distinguishing all Classes
8 - Select New Measurement Category

Category = 7

Rank measurements distinguishing all classes

M(Xp)                                   Measurement
-----                                   -----------

94827,375000                            maxint 3 3
10707,809570                            avg 3
7350,170898                             minint 3 3
```

Figure 4-12 Discriminant Measure Output

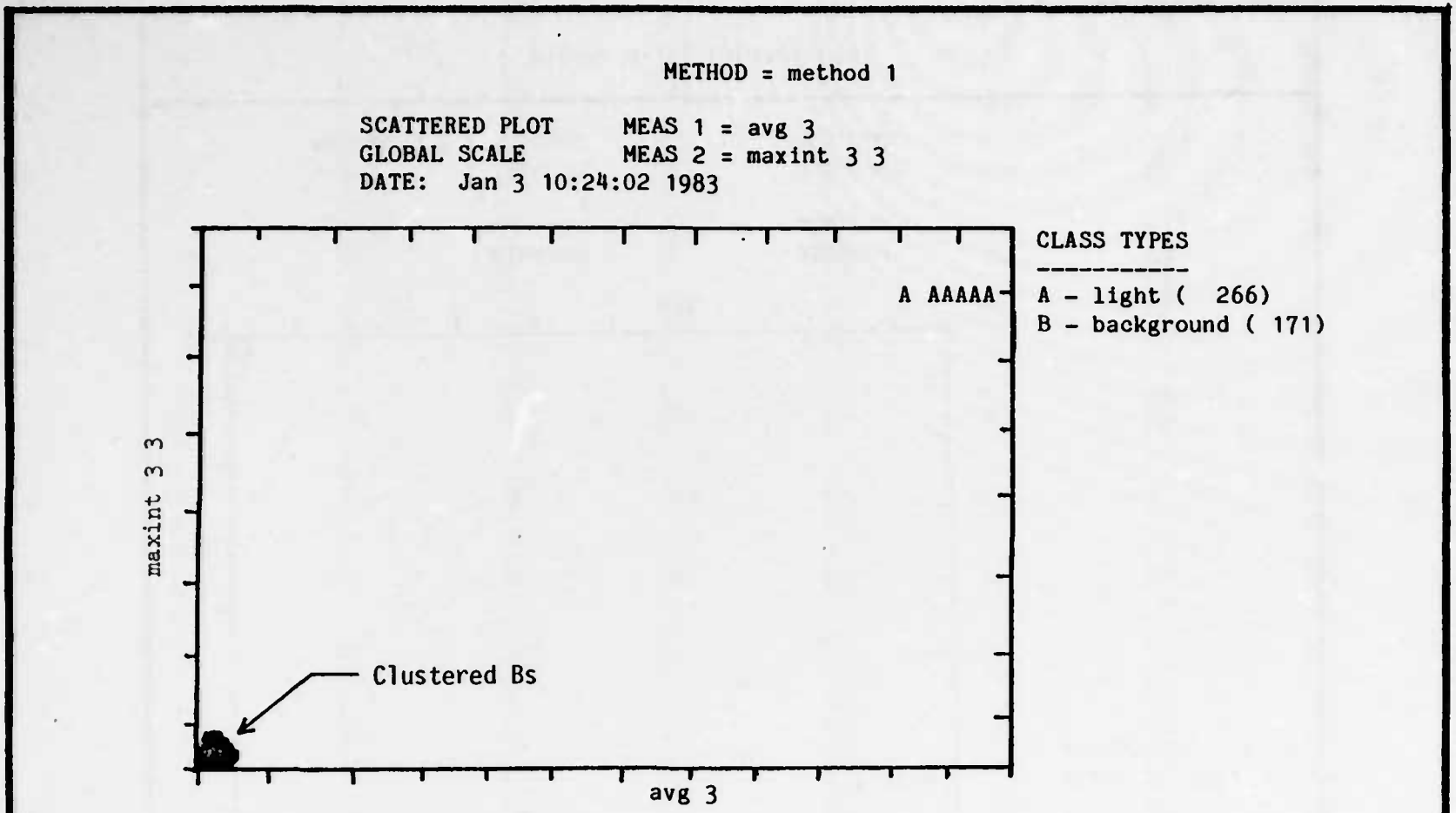
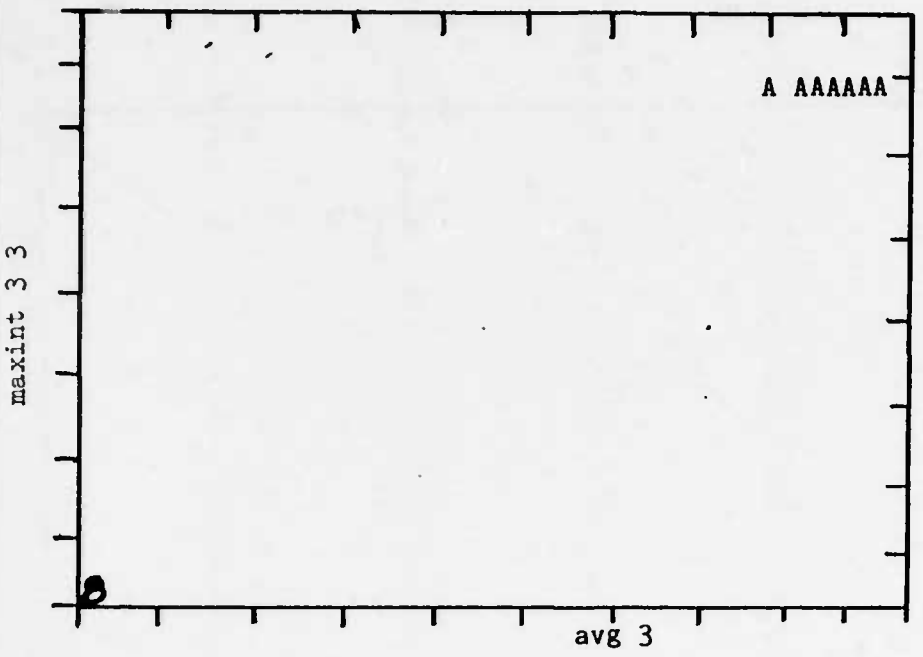


Figure 4-13 Scatter Plotter

METHOD = method 1

CLUSTER PLOT MEAS = avg 3
GLOBAL SCALE MEAS = maxint 3 3
DATE: Jan 3 10:22:14 1983



CLASS TYPES

A = light (266)
B = background (171)

	MINIMUM	MAXIMUM
avg 3	: 16.0000	255.0000
maxint 3 3	: 20.0000	255.0000

Figure 4-14 Cluster Plot

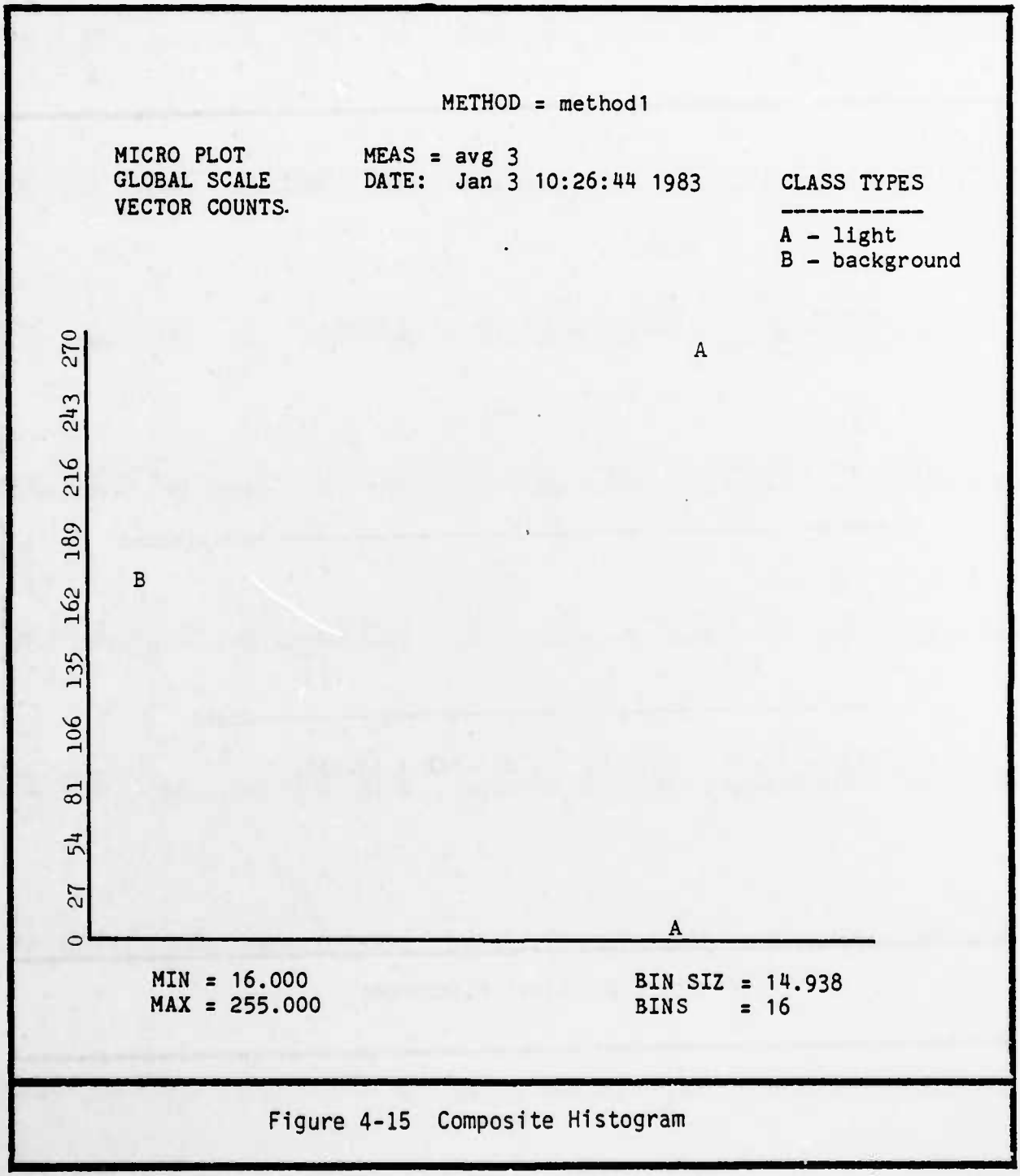


Figure 4-15 Composite Histogram

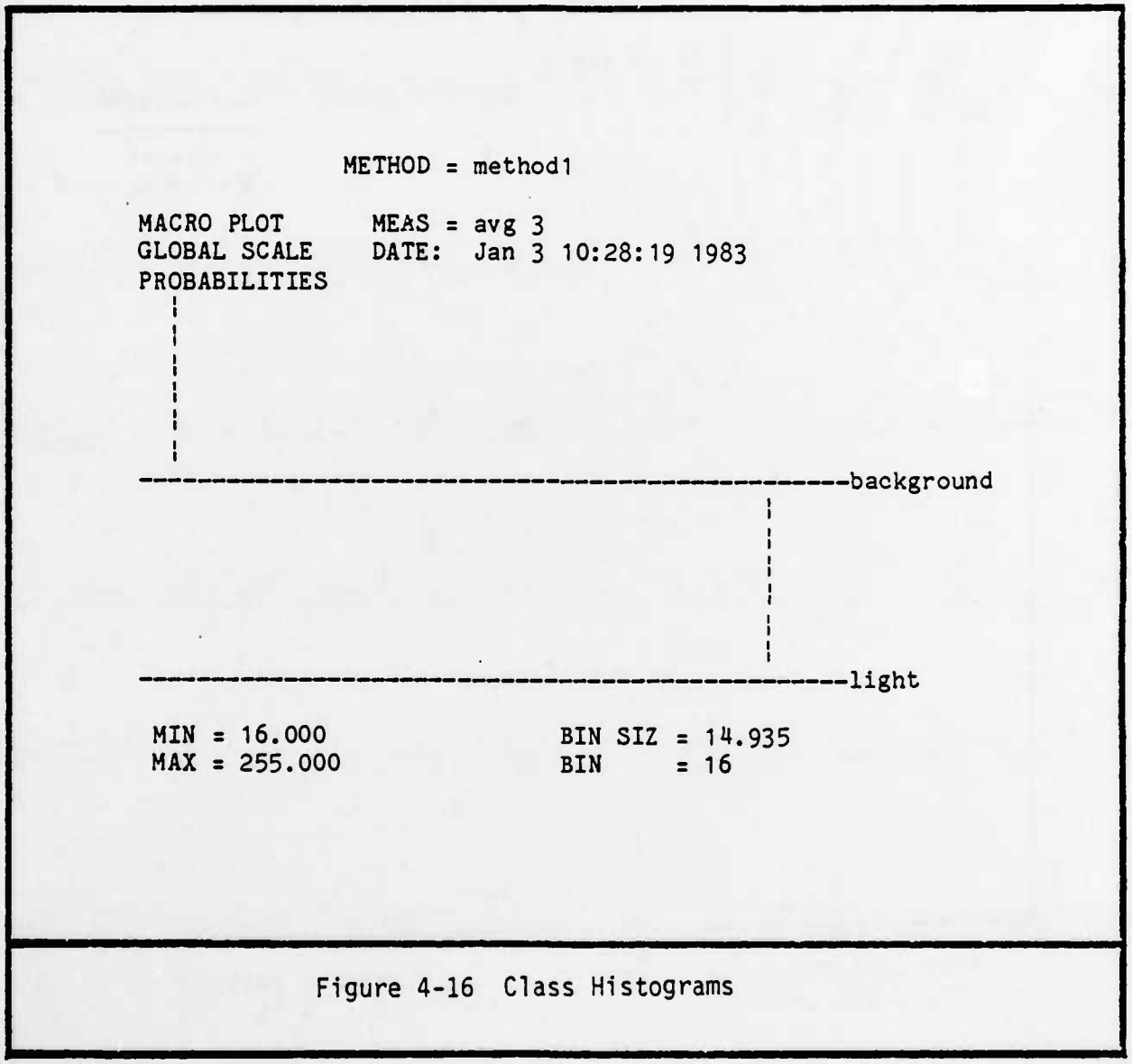


Figure 4-16 Class Histograms

APPENDIX A

User Guide to Knowledge Engineering With NEWSIP

A.1 INTRODUCTION

The NEWSIP rule-based system was designed to allow the use of probabilistic inferential reasoning for image understanding in the AFES/RWPF environment. Like most expert system frameworks, NEWSIP consists of two main components, an inference engine and a rule base. The inference engine consists of the software for a general mechanism to draw and explain inferences. The rule base is a text file consisting of a set of conditions and actions which provide the knowledge for particular inferences to be made to solve a problem. In the field of Artificial Intelligence, the task of building a rule base is known as knowledge engineering.

This document provides an introduction to knowledge engineering with NEWSIP. Information is provided for an understanding of how to build a rule base that provides specific problem-solving knowledge for NEWSIP. In particular, the general methodology of rule base construction is described (with examples given) and a detailed definition of rule base syntax is provided.

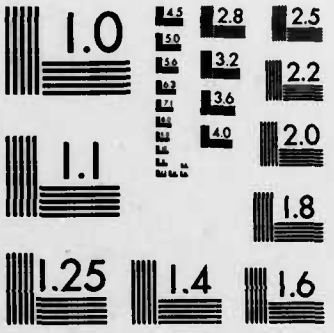
A.2 BASIC CONCEPTS

Before providing an example of how specific rules can be developed for NEWSIP, it is necessary to describe the basic concepts that the system employs for representing and using a rule base.

A rule base consists of a set of conditions and actions which tell what inferences can be made for solving a problem. A single rule consists of an If-Then statement relating a set of antecedent conditions to a consequent hypothesis. The certainty or truth of the consequent depends upon that of the antecedent conditions. The antecedent conditions may themselves be the consequent parts of other rules. For example, the statements,

AD-A132 339 ADVANCED PATTERN RECOGNITION(U) PAR TECHNOLOGY CORP NEW HARTFORD NY J L CAMBIER ET AL. MAY 83 PRR-83-1 RADC-TR-83-50 F30602-80-C-0319 2/2

UNCLASSIFIED F/G 20/6 NL

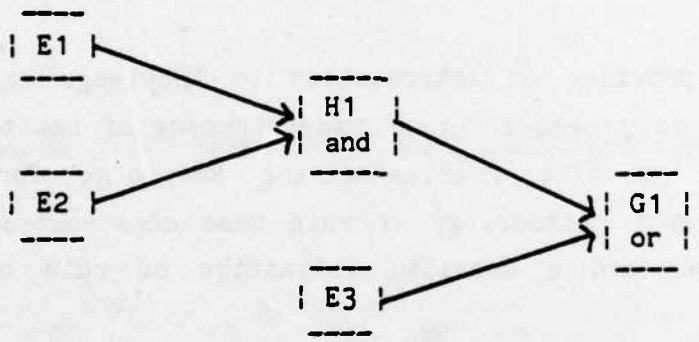


If E1 and E2, then H1, and

If H1 or E3, then G1

represent two abstract rules. E1 and E2 are antecedent conditions for H1, and H1 and E3 are antecedent conditions for G1.

NEWSIP uses an inference network (inference net) representation for its rule base. If the two rules of the example above made up an entire rule base, they could be thought of as an inference net that can be drawn as:



The nodes in the network represent the individual antecedent conditions and consequents. E1, E2, and E3 are considered evidence nodes; H1 is an intermediate hypothesis node; and G1 is a goal node.

The labels of evidence, hypothesis, and goal are assigned by the user when building a rule base, but are usually reflected in the structure of the inference net. Evidence nodes are the leaves of the network, having no incoming arrows representing inferences, and are associated with some test or measurement to be performed on an image. Goal nodes are the roots or sinks of the network, having no outgoing inferences, and are associated with some action to be taken such as identifying a feature or segmenting a region. Hypothesis nodes have both incoming and outgoing inferences. (As will be

described later, however, a node like H1 may also be considered as a goal). The arrows between nodes of the inference net represent the antecedent consequent relationships. The labels of "and" and "or" applied to H1 and G1 show the kind of relationship between a node and its antecedents. In the probabilistic inference mechanism that NEWSIP uses, the arrows, or links between the nodes of the inference net also represent paths along which probabilities are propagated, and each node has a probability associated with it.

The control mechanism of the NEWSIP inference engine as applied to the above example can briefly be described as follows. Assuming that the rule base has been coded in the proper format, which will be described later, invoking the NEWSIP inference engine will cause the rule base to be read in and parsed for syntactic correctness and will also cause an inference net representation to be formed internally. Next, a goal node will be selected for consideration. (For this reason NEWSIP is considered goal-directed.) In the example, only one goal node, G1, is available. Having selected G1, the control mechanism now trys to determine its certainty. To do this it looks for evidence nodes that affect G1. This process is recursive and depth-first. The immediate antecedents of G1 will be considered one at a time. If the first one considered is not an evidence node, then its antecedent nodes will be examined. So, in the example, H1 might be selected as an antecedent for G1 and, then, E1 could be selected as evidence for H1. At this point, whatever test or measurement is associated with E1 is performed and the certainty of E1 is determined, turning out to be true or false: that is, probability 1 or 0. This certainty is now propagated through the inference net. Probabilities are updated for all of the nodes affected directly or indirectly by E1. In this example H1 and G1 will be affected.

Now the control mechanism iterates. G1 still needs further consideration since other evidence remains to be investigated for it. H1 has also not been completely investigated, so it would be considered again. E1 has been tested, so the test for E2 would now be invoked. The resulting probability would be

propagated and the process continued with another iteration (which would this time investigate E3). After E3 has been considered, no further evidence remains to be investigated for G1, and NEWSIP would halt, after having performed whatever action is associated with G1 (or not performed it), depending upon G1's probability.

The specific actions and tests that can be specified in an NEWSIP rule base are described later in this document. Only two kinds of actions are currently possible: to apply a segmentation method to the current region or to assign an AFES/RWPF region attribute value to the current region. The rules of an NEWSIP rule base are applied to one region at a time; the one being considered at the moment is the current region. Evidence tests are limited to the evaluation of conditions on sets of regions or some special predicate functions. Conditions are expressed as arithmetic relationships on region attribute values, and sets of regions are specified in terms of a quantity of of neighboring, offspring, parent, or included relationships to the current region being examined.

Other details of the NEWSIP control mechanism such as the user-command interface will be described in other documentation. Subsequent sections of this tutorial will describe how goal actions and evidence tests may be specified, how probabilities are assigned, and how goals and antecedents for investigation are selected. The next section presents an example of how rules are developed.

A.3 Example of Rule Base Development

The example of rule base development presented in this section addresses the problem of finding in an image large and small regions which have a class of water, urban, or vegetation as determined by a segmentation method called wuv_finder. This example is fairly artificial but serves to illustrate an overview of the process of knowledge engineering with NEWSIP.

The first step is to define the problem or what it is we want the rules to determine. In this example we want to apply the wuv_finder segmentation method to the initial region if it is very large and, then, decide if it or any of the resulting offspring regions are large or small.

Having defined the problem, we can begin building a rule base to solve it. The general approach is to 1) determine what actions are to be executed by the rule base (actions which will correspond to the goal nodes in the inference net) and then 2) define the conditions that determine when to perform these actions. There are three actions that our rules will perform:

segment a region using wuv_finder,

assign the label of 'large' to a region, and

assign the label of 'small' to a region.

The considerations for deciding among these actions may be represented in the following conditional statements:

If the region being examined is the first region,
and has a large area,
and has a large perimeter,
then use wuv_finder to segment it.

If the region being examined has either a large area
or a large perimeter,
then label it as large.

If the region being examined has a small size

and is not considered large,
then label it as small.

Of course, there are many other conditions that could have been used and, also, many ways they could be combined in such statements. Building some small alternative rule bases for this problem is recommended as a useful first exercise at knowledge engineering with NEWSIP.

A useful next step is to specify the above statements in a slightly more formal fashion, naming the antecedent conditions and consequents, indicating what tests will be required, the names of the labels to be used, and the logic of the antecedent-consequent relationships. The more formal specification is:

```
If      first_region
        the region being examined is the first region
        test for R1
        And
        large_area
        the region being examined has a large area
        test for area >= 100,000
        And
        large_perim
        the region being examined has a large perimeter
        test for perimeter >= 400
Then    wuv_finder
        uses wuv_finder to segment the region
        segment using wuv_finder

If      large_area
        Or
        large_perim
Then    large_region
        the region is fairly large
```

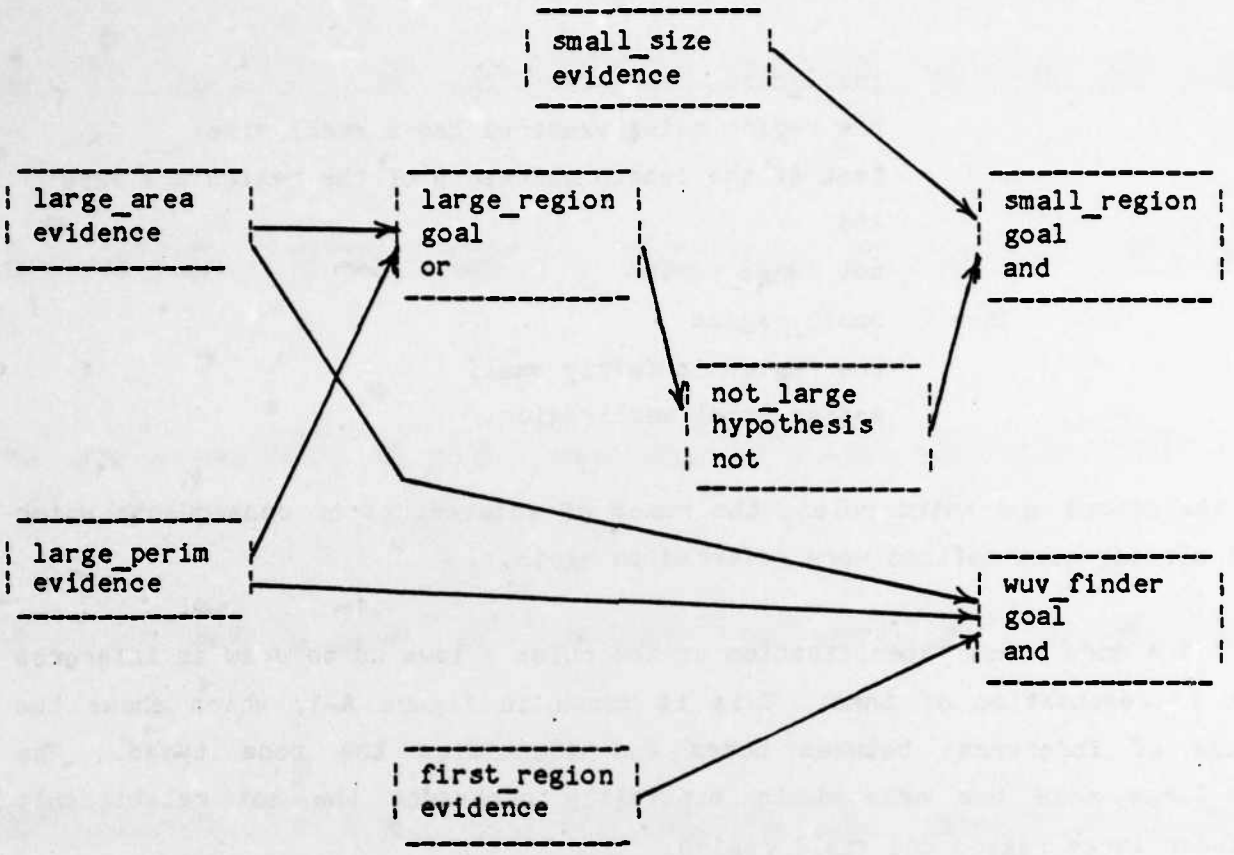


Figure A-1. Example of NEWSIP Inference Net Representation

We can also take (before using and debugging the rules) the final representation step, which is to code the inference net in the NEWSIP representation language. The syntax for this language is described in Section A.6, and the result for our sample rules is shown in Appendix B. Once coded, the rules can be used with the NEWSIP software for debugging the rule base. The above example gives an overview of the process of knowledge engineering with NEWSIP. The next section provides a general methodology for this process.

A.4 KNOWLEDGE ENGINEERING METHODOLOGY

It is convenient to think of knowledge engineering with the NEWSIP system as the process of building an inference net. The goal of this process is to produce a text file containing a description of the rules for solving a problem written in the NEWSIP rule representation language. General steps in this process are:

1. Identify the problem, what needs to be done, what is the expected outcome in terms of existing AFES/RWPF data structures, etc.

2. Determine the problem structure, what goal actions are to be taken, what kinds of evidence tests will be required, and what some of the intermediate hypotheses are, if any.

3. Determine the kinds of relationships between antecedents and consequents of the rules, whether rules should involve "and," "or," or "not" logic, or whether they should use Bayesian inference and assign rough values for Bayesian weights and prior probabilities.

4. Code the rules in the NEWSIP inference net representation language.

5. Debug the rule base, modifying or extending the inference net, and adjusting Bayesian weights and prior probabilities.

Clearly, the process described in these steps is incremental and iterative, especially in steps 2 through 5. Chunks of knowledge, in the form of single or small sets of rules, can be added to an inference net and debugged, gradually building up the inference net to handle all aspects of the problem as defined in step 1.

One difficult part of the rule building process is to determine the kind of mechanism for representing inferences; i.e. whether particular rules should have their antecedents in an "and", "or", "not" or Bayesian relationship. In general, an "and" relationship is used when all of the antecedents are required to determine the truth of the consequent, an "or" relationship is used when only one antecedent is required, a "not" relationship when the truth of the consequent depends on the antecedent being false, and a Bayesian relationship when independent antecedents affect the consequent in different ways. The next section defines the mathematics behind these different types of inference mechanism.

A.5 PROBABILITY ASSIGNMENTS

NEWSIP uses a probabilistic inference mechanism based on the ideas used in PROSPECTOR [Duda, Hart, et al] and AL/X [Michie]. The nodes in the inference net may be thought of as representing events in some probability space.

Some events are associated with observable evidence that determines their probability; in the case of NEWSIP, these events are represented by evidence nodes, and the observable evidence is the test action associated with the node. Each node has a prior probability associated with it which represents the probability that the event associated with the node will occur given that no observable evidence has been collected. As evidence is collected, the probabilities of the nodes are updated according to several rules of inference. These rules tell how the probability of a node can be determined from that of its antecedents. Let A_i, $i = 1 \ldots n$, be antecedent nodes for node B. Then the rules of inference for "and," "or," or "not" nodes are:

If B is an "and" node:

$$Prob(B) = \min [Prob(A_i)] , i = 1 \ldots n$$

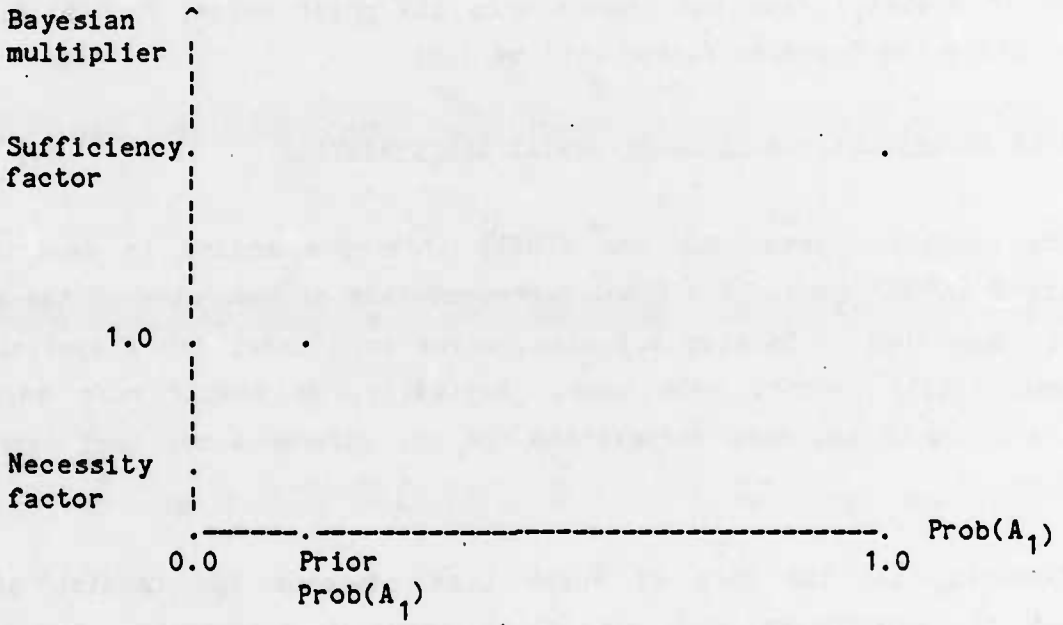
If B is an "or" node:

$$Prob(B) = \max [Prob(A_i)] , i = 1 . . n$$

If B is a "not" node (B can have only one antecedent):

$$Prob(B) = 1 - Prob(A_1)$$

For Bayesian nodes, the probability of B is updated independently as the probability of each antecedent node changes. The updated Prob(B) is determined by multiplying it by a Bayesian factor, which is calculated by a piece-wise linear interpolation between the Necessity and Sufficiency factors provided for an antecedent. The interpolation function looks like:



The interpolation formulas are

For $Prob(A_i) < Prior\ Prob(A_i)$:

$\{Prob(A_i) * [(1.0 - Necessity) / (Prior\ Prob(A_i))]\}$

+ Necessity

For $Prob(A_i) > Prior\ Prob(A_i)$:

$$\{[Prob(A_i) - Prior\ Prob(A_i)] * [(Sufficiency - 1.0)/(1.0 - Prior\ Prob(A_i))]\} + 1.0$$

If $Prob(A_i)$ is 0.0, Prob(B) will be multiplied by the Necessity factor for A_i. If $Prob(A_i)$ is 1.0, Prob(B) will be multiplied by the Sufficiency factor for A_i. A low Necessity factor means that, if A_i is false ($Prob(A_i)$ = 0), B is also false (Prob(B) is greatly reduced). A high Sufficiency factor means that if A_i is true ($Prob(A_i)$ = 1.0), B is also true (Prob(B) is greatly increased). (Hence, the names of the Necessity and Sufficiency factors suggest the corresponding concept of necessary and sufficient conditions in logic). If $Prob(A_i)$ does not change from its prior value, Prob(B) will not change (since the Bayesian factor will be 1.0).

A.6 RULE REPRESENTATION LANGUAGE SYNTAX AND SEMANTICS

The complete syntax for the NEWSIP inference engine is described in Appendix B in BNF form. The final representation of the rules of the example that was presented in Section A.3 also serves as a model for a syntactically and semantically correct rule base. Basically, an NEWSIP rule base file contains comments and node definitions for the inference net that represents the rules.

Comments take the form of Franz Lisp comments and consist of text enclosed in parentheses with the first word of "comment". For obscure reasons, comment text can contain any characters except commas (,) and single quotes (').

Node definitions are also contained in parentheses and consist of several parts. Required parts for all nodes are the node type, node name, text describing the node, and prior probability for the node. Node type must be goal, hypothesis, or evidence. A node name is any user-defined name for the node, but it must start with an alphabetic character. The text part of a node

definition consists of the word text followed by any user-defined text describing, enclosed in double quote characters ("), what the node represents. The prior probability is the prior probability that the event or statement represented by the node will occur or is the case. It should be consistent with the rules defined in the previous section, the priors of antecedent and consequent nodes that it is linked to. It is defined by the word "prior," followed by a number between 0 and 1.

Optional parts of a node definition are the specifications of antecedents, consequents, and actions. The antecedent specification for a node consists of the word "antecedents" followed by a list which has a format depending upon the logical relationship of the antecedents to the node being defined. For "and" or "or" nodes, the antecedents list consists of the word "and" or "or" followed by a list of names of the nodes that represent the antecedents. For "not" nodes, the antecedents list is just the word "not" followed by the name of a single node. For Bayesian nodes, the antecedent list contains one or more Bayesian link definitions which are triplets enclosed in parentheses consisting of the antecedent node name and the Necessity and Sufficiency factors for the Bayesian link. A node can have only one antecedents list associated with it and, hence, can only be of one logical type.

The consequents specification for a node consists of the word "consequents" followed by a list of the names of the nodes that represent consequents of the node being defined. The type of relationship that the node has with the consequent must be specified in the antecedents part of the consequent node definition; that is, the consequent part of a node definition exists merely to allow the knowledge engineer to introduce clarity and redundancy into the inference net definition.

The action part of a node definition is required for goal and evidence nodes, but not for hypothesis nodes. It consists of the word "action" followed by parentheses enclosing the action specification. For evidence

nodes, the action part should specify some tests to perform to determine the probability of the node, which, at the moment, can only turn out to be either 0 or 1. More than one test may be specified, and the probability of the evidence node will be set to 1 only if all tests are true. Each test specification consists of the word "test" followed by a single condition to be tested. Test conditions take the form of (<region spec>) with <region quals>. <Region Spec> may be either current, meaning the current region being examined, or some specification of a relationship to some count of neighbors, parent, offspring, or included regions for the current region. <Region quals> may be a single qualification or a list of single qualifications separated by the words "and" or "or". For an "and" list, the test returns true only if each qualification is true for an "or" list, the test returns true if one of the qualifications is true. A single qualification may be a SIP predicate or an expression of a SIP predicate or AFES/RWPF region attribute in a relationship to one of its values. A list of current SIP predicates is provided in Appendix B. Previously-defined, valid, AFES/RWPF attribute names are listed in the available on-line documentation.

Some examples of evidence action specification will clarify the above discussion. The syntactically correct test specifications below are preceded by text describing what is being tested.

```
test whether the current region has a perimeter greater than
1000 meters
( test (current) with (perimeter > 1000) )

test whether 2 or more neighbors of the current region
have perimeters greater than 1000 meters
( test ( > = 2 neighbor) with (perimeter > 1000) )

test whether any offspring of the current region have
areas less than 100 square meters
( test ( > 0 offspring) with (area < 100) )
```

```
test whether no included region has area < 100 and
perimeter < 40
( test ( < = 0 included) with (area < 100)
                          and  (perimeter < 40) )

same test as above expressed as two separate tests
( test ( < = 0 included) with (area < 100) )
( test ( < = 0 include) with (perimeter < 40 )
```

The action part for goal nodes may specify either a segmentation method to use on the current region or a set of attribute value assignments for the current region. Segmentation goals contain an action specification consisting of the phrase "segment using" followed by the name of a segmentation method. Segmentation methods are user-defined.

Attribute assignment goals have an action specification consisting of a series of one or more assign statements. Each statement has the format of the word "assign" followed by the name of an AFES/RWPF attribute followed, in turn, by the value to be given the attribute. AFES/RWPF attribute names may be user-defined, (for example, the "label" attribute defined in the example in Section A.3).

A.7 CONCLUSION

The above document provides some insight into how to do knowledge engineering with NEWSIP. The important aspects of this process are 1) that it is incremental, with small chunks of knowledge being added to the system over time, and 2) that it is iterative, with rules being gone over many times to achieve the exact desired results. Working with any formalism requires time for learning its details. Much trial and error will be required, along with an understanding of this document, in order to make satisfactory use of NEWSIP.

A.8 REFERENCES

The following are the main references utilized in developing this appendix:

Duda, R.O. Hart, P.E., and Gaschnig, J.,
"Model Design in the PROSPECTOR Consultant System for Mineral Exploration,"
Expert Systems in the Micro-Electronic Age
(ed. D. Michie) Edinburgh: Edinburgh University Press, 1977 pp. 153-167

Michie, D., and Paterson, A.,
AL/X User Manual,
Oxford: Intelligent Terminals Ltd., 1981

APPENDIX B

NEWSIP Rules Syntax, SIP Predicates, and Sample Rule Base

B.1 NEWSIP RULES SYNTAX

Below is a complete BNF description of the syntax for defining a rule base to be used by the NEWSIP inference engine. Instead of completely specifying some of the terminal symbols, generic descriptions have been used to define these symbols.

Any symbols, except |, ::=, <, and >, may represent something that must be included in the syntax, e.g. (or). Spaces count except those following the symbols (or " or those inserted before a) or ".

Rule Base Definition:

```
<node set>      ::= <node def> <node set> |
                    <comment> <node set> |
                    <node def>

<comment>       ::= ( comment <comment text> )

<comment text>  ::= any string of characters allowed
                    within Franz LISP comments
```

Inference Net Node Definition:

```
<node def>      ::= ( <node type> <node name> <node desc> )

<node type>     ::= goal | hypothesis | evidence

<node name>     ::= <name>
```

```
<node desc>     ::=  <text>                                    <prior>
                          <antecedents>
<consequents>                          <action>

<text>          ::= text " <char string> "

<char string>   ::=  any string of characters except the
                     " character

<prior>         ::= prior <probability>

<probability>   ::= a <number> between 0 and 1

<number>        ::= a real number
```

Antecedents Definition:

```
<antecedents>   ::= antecedents ( <antecs list> ) |
                    <>

<antecs list>   ::= and <node list> |
                    or <node list> |
                    not <node name> |
                    <Bayesian list>

<node list>     ::= <nodename> <node list> |
                    <nodename>

<Bayesian list> ::= <Bayesian link> <Bayesian list> |
                    <Bayesian link>

<Bayesian link> ::= ( <node name> <Sufficiency> <Necessity> )
```

```
<Sufficiency>   ::= <number>

<Necessity>     ::= <number>
```

Consequent Definition:

```
<consequents>   ::= consequents ( <node list> ) |
                    <>
```

Action Definition:

```
<action>        ::= action ( <action spec> ) |
                    <>

<action spec>   ::= segment using <seg method> |
                    <assign series> |
                    <test series>

<assign series> ::= ( assign <assign spec> ) <assign series> |
                    ( assign <assign spec> )

<assign spec>   ::= <attribute> <attr value>

<attribute>     ::= AFES/RWPF region or edge attribute name

<attr value>    ::= <number> |
                    <symb attr val>

<symb attr val> ::= AFES/RWPF region or edge attribute symbolic value

<test series>   ::= ( test <test spec> ) <test series> |
```

```
                    ( test <test spec> )

<test spec>     ::= ( <region spec> ) with <region quals>

<region spec>   ::= current |
                    <rel> <region quant> <region rel>

<region quant>  ::= an integer >= 0

<region rel>    ::= neighbor |
                    parent |
                    offspring |
                    included

<region quals>  ::= <and reg quals> |
                    <or reg quals> |
                    ( <region qual> )

<and reg quals> ::= ( <region qual> ) and <and reg quals> |
                    ( <region qual> )

<or reg quals>  ::= ( <region qual> ) or <or reg quals> |
                    ( <region qual> )

<region qual>   ::= <attribute> <rel> <attr value> |
                    <predicate> |
                    <predicate> <rel> <pred value>

<predicate>     ::= SIP predicates yet to be defined

<pred value>    ::= <number> |
                    <symb pred val>
```

```
<pred value>     ::= <number> |
                     <symb pred val>

<symb pred val> ::= SIP predicate symbolic value

<rel>            ::= < | > | = | <= | >= | <>
```

B.2 SIP PREDICATES

unsegmented: returns true if the region has not had a segmentation method used on it yet.

firstregion: returns true if the current region is R1

B.3 SAMPLE NEWSIP RULE BASE

```
(comment Rule Base to Determine Large and Small Regions
         Used to test interface between SMASC Inference Engine and
         SIP/AFES
        )

(comment Segmentation rules
        )

(comment Goal for wuvfinder
        )
(goal wuv_finder
        text "segment region into water, urban, and vegetation classes"
        prior 0.01
```

```
        antecedents (and large_area large_perim first_region)
        action (segment using wuvfinder)
)

(evidence first_region
        text "current region is the first one being analyzed"
        prior 0.1
        action (
                (test (current) with (firstregion) )
               )
)

(comment Feature assignment rules
        )

(comment Goal for large regions
        )
(goal large_region
        text "region is fairly large"
        prior 0.01
        antecedents (or large_area large_perim)
        action  ((assign label largeregion))
)

(evidence large_area
        text "region has a large area, measured in square meters"
        prior 0.01
        action ( (test (current) with (area > 16000) ) )
)

(evidence large_perim
        text "region has a large perimeter, measured in meters"
        prior 0.01
```

```
        action ( (test (current) with (perimeter > 1000) ) )
)

(comment Goal for small regions
        )
(goal small_region
        text "region is fairly small"
        prior 0.01
        antecedents (and not_large small_size)
        action ((assign label smallregion))
)

(hypothesis not_large
        text "region is not very large"
        prior 0.01
        antecedents (not large_region)
)

(evidence small_size
        text "length and width of region are fairly small"
        prior 0.01
        action ( (test (current) with (length < 250) )
                 (test (current) with (width < 250) )
               )
)
```

APPENDIX C

APR Rule Bases

C.1 INTRODUCTION

This appendix contains sets of rule bases developed under the APR effort for recognition of various types of features. Also included are method files specifically referenced in the rules. Specific procedures for creation of rules and application of rule bases are provided in the APR Test and Implementation Plan and in the APR Final Report, to which this appendix is attached.

The method files and rule base described here were developed by PAR Technology Corporation with the aid of Mr. Andrew Douglas, who provided support in DLMS feature description under a Rome Research Corporation subcontract, and Mr. Paul Hopkins, a consultant under the APR effort from the State University of New York College of Environmental Science and Forestry in Syracuse, New York.

C.2 AIRCRAFT RULE BASE

A rule base for recognizing aircraft in vertical imagery was developed. It uses a pixel method, called "method 1," for segmentation of the image prior to application of rules. This method is as follows:

```
method_type:
        pixel
measurements:
        avg 3 3
        maxint 4 4
        minint 4 4
classifier:
        mahal
(training set)
class:
```

```
        object
regions:
        light 1
        light 2
class:
background:
        regions:
dark 1
edit:
        mode_filter 3 3
comments:
        This is a simple method for easily detected objects.
```

The rule base is the following:

```
(goal Initial_Segmentation
        text "segment region 1 with method 1"
        prior 0.00001
        antecedents (and First_Region)
        actions (segment using method 1)
)

(evidence First_Region
        text "this is the first region"
        prior 0.001
        action ( (test (current) with (firstregion)) )
)

(goal Assign_plane_featuretype
        text "region is a plane point feature"
        prior 0.01
        antecedents (and Is_a_plane)
        action( (assign featuretype point) )
```

```
)

(goal Assign_plane_featureid
        text "region has feature type plane"
        prior 0.01
        antecedents (and Is_a_plane)
        action( (assign featureid plane) )
)

(goal Assign_plane_smc
        text "planes have surface material category metal"
        prior 0.01
        antecedents (and Is_a_plane)
        action( (assign smc metal) )
)

(hypothesis Is_a_plane
        text "region is a plane"
        prior 0.001
        antecedents (and Plane_class Good_length Good_width)
)

(evidence Plane_class
        text "region has been classified as a plane"
        prior 0.1
        action ( (test (current) with (class = plane) ) )
)

(evidence Good_length
        text "check for region length in database"
        prior 0.01
        action( (test (current) with (length > 0) ))
)
```

```
(evidence Good_width
        text "check for region width in database"
        prior 0.01
        action( (test (current) with (width > 0) ))
)

(goal Assign_airport_featuretype
        text "region is an airport areal feature"
        prior 0.01
        antecedents (and Is_an_airport)
        action( (assign featuretype areal) )
)

(goal Assign_airport_featureid
        text "region has feature type airport"
        prior 0.01
        antecedents (and Is_an_airport)
        action( (assign featureid airport) )
)

(goal Assign_airport_smc
        text "airports have surface material category concrete"
        prior 0.01
        antecedents (and Is_an_airport)
        action( (assign smc concrete) )
)

(hypothesis Is_an_airport
        text "region is an airport"
        prior 0.001
        antecedents (and Has_planes Good_area Good_length Good_width)
)
```

```
(evidence Has_planes
        text "region has planes nearby"
        prior 0.02
        action ( (test ( >= 2 neighbor ) with (class = plane) ) )
)

(evidence Good_area
        text "region has sufficient area to be an airport"
        prior .03
        action ( (test (current) with (area > 10000) ) )
)
```

C.3 URBAN SEGMENTATION RULES

The rulebase which follows separates an image into urban and nonurban region and, then, performs a second segmentation on the urban regions to identify individual buildings and other features. It uses two methods. The first, called wuv, is a pixel method which performs an initial segmentation into classes water, urban, and vegetation. The second is an edge method called urban edges; it detects edges of buildings, roads, etc. in urban regions. These methods are listed below:

```
wuv
method_type:
        pixel
measurements:
        band1/ei_ranger/ep_smooth 1
        band1/ep_smooth 3/median 3
classifier:
        mahal
(training set)
class:
water
```

regions:

52w1

52w2

52w3

class:

vegetation

regions:

52v1

class:

urban

regions:

52u1

edit:

mdf 5 5

comments:

experimental method to segregate Water, Urban, Vegetation

urban edges

method_type:

edge

measurements:

avg 3/ep_smooth 2/sobel/athres 1 1 15

classifier:

none needed

(training set)

edit:

edge_thin 2

comments:

Experimental method to detect urban edges

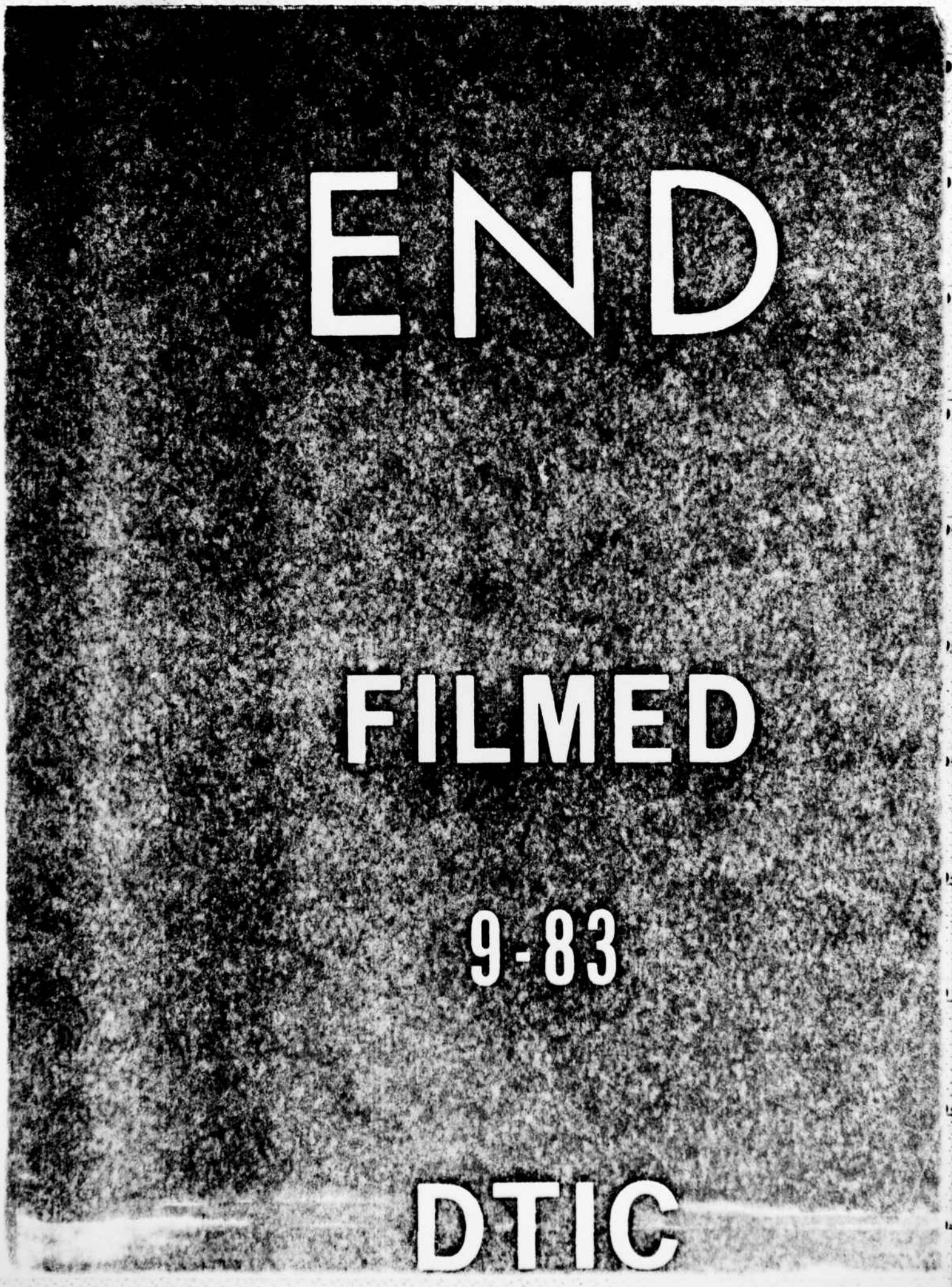
Rule Base

```
(comment Segmentation Rules)

(goal Initial_Segmentation
        text "segment region 1 with wuv"
        prior %0.00001
        antecedents (and First_Region)
        action (segment using wuv)
)

(evidence First_Region
        text "this is the first region"
        prior %0.001
        action ( (test (current) with (firstregion) ) )
)

(goal Second_Segmentation
        text "segment regions with urban_edges"
        prior %0.00001
        antecedents (and Not_First_Region Wuv_Region Urban Biggie)
        action (segment using urban_edges)
)

(hypothesis Not_First_Region
        text "this is not the first region"
        prior %0.999
        antecedents (not First_Region)
)

(evidence Wuv_Region
        text "region has been derived by wuv"
        prior %0.1
        action ( (test (current) with (method eq wuv)) )
```

```
)

(evidence Urban
        text "region has class urban"
        prior %0.1
        action ( (test (current) with (class eq urban)) )
)

(evidence Biggie
        text "region is large enough to resegment"
        prior %0.3
        action ( (test (current) with (area gt 10000)) )
)

(comment Region Deletion Rules)

(goal Delete_Region
        text "region should be deleted from further consideration"
        prior %0.05
        antecedents ( or Delete1 Delete2)
        action ( deleteregion)
)

(hypothesis Delete1
        text "region is misclassified from method wuv and should be deleted"
        prior %0.01
        antecedents ( and Offspring_of_Firstregion Not_Too_Big Not_Urban
        Surrounded_by_Urban)
)

(evidence Offspring_of_Firstregion
        text "current region is an offspring of region 1"
        prior %0.01
```

```
        action ( (test (eq 1 parent) with (firstregion) ) )
)

(hypothesis Not_Urban
        text "region is not urban"
        prior %0.25
        antecedents (not Urban)
)

(hypothesis Surrounded_by_Urban
        text "region is surrounded by class urban"
        prior %0.2
        antecedents (and Urban_Neighbor No_Nonurban_Neighbor)
)

(evidence Urban_Neighbor
        text "region has 1 urban neighbor"
        prior %0.2
        action ( (test (eq 1 neighbor) with (class eq urban) ) )
)

(evidence No_Nonurban_Neighbor
        text "region has 0 nonurban neighbors"
        prior %0.2
        action ( (test (eq 0 neighbor) with (class ne urban) ) )
)

(hypothesis Delete2
        test "region is misclassified from method urban_edge and should be
        deleted"
        prior %0.1
        antecedents ( and Second_Generation Very_Small One_Edge)
)
```

```
(evidence Second_Generation
        text "region is an offspring of an offspring of region 1"
        prior %0.8
        action ( (test (eq 1 parent) with (parent eq 1) ) )
)

(evidence Very_Small
        text "region is very small"
        prior %0.2
        action ( (test (current) with (area le 20) ) )
)

(evidence One_Edge
        text "region has a single edge"
        prior %0.15
        action ( (test (eq 1 edge) with (length gt 0) ) )
)
```


END
FILMED
9-83
DTIC