A C-Extension for Rule-Based Image Classification Systems

Gerhard Mehldau
Department of Computer Science, University of Arizona, Tucson, AZ 85721
Robert A. Schowengerdt
Department of Electrical and Computer Engineering, University of Arizona, Tucson, AZ 85721

ABSTRACT: Most current systems for prototyping rule-based systems use applicative, interpreted languages such as LISP or its derivatives. This makes it easy to prototype small applications in a flexible way. However, such systems are less useful in production applications involving large amounts of input data and specialized, numerically oriented libraries. An augmentation of C is presented containing the control structures, predicates, and data types that make possible the convenient specification of a rule-based system. The extended C language is used to build a rule-based system for labeling features in a digitized aerial photograph. Integration of the rule-based system with conventional image preprocessing and the performance of the rule-based system development tool are discussed.

INTRODUCTION

MOST OF THE SYSTEMS that have been developed for automatically generating rule-based systems employ an applicative approach. They use AI techniques such as heuristic search and backtracking and are generally written in LISP or LISP dialects. These design choices make it easy to quickly produce a working system and thus encourage experimenting with different approaches to the problem at hand. However, such systems are less useful for problems where large amounts of input data are involved. The code they produce is in most cases interpreted, not compiled, and therefore very slow. Furthermore, many conventional production environments for numerical computation are based on languages such as FORTRAN or C, and in general do not have compilers or interpreters available for LISP. If libraries of application-dependent software are to be used, the interface to the rule-based system creates an additional problem. Last, but not least, programmers in production environments are often unfamiliar with the concepts behind applicative languages and their use.

For these reasons we initiated the development of a system that potentially overcomes these problems. The standard C language has been augmented in a general way, so as to permit the convenient specification of a rule-based system. A system written in this extension of C is translated into standard C by a preprocessor and subsequently compiled. The advantages of this method are threefold:

- The compiled code is faster than interpreted systems,
- Existing application software written in C can easily be integrated, and
- Programmers who currently use C need only become familiar with a few new concepts.

We present in this paper the extensions to the C language, containing the control structures, predicates, and data types that make possible the specification of a rule-based system that classifies a sequence of input records into categories specified by rules. The extended C language is subsequently used to create an experimental rule-based system for classification of a multispectral aerial photograph. A general paradigm for image classification using conventional image processing techniques and a rule-based system is presented. A brief description of this paradigm has been given elsewhere (Schowengerdt and Mehldau, 1987). The image is first segmented into small regions which are then input to the rule-based classifier, where their spectral signatures, geometric properties, and context relative to neighboring regions are used to label each segment. The incorporation of region shape and spectral features in the classification of high resolution remote sensing images was pioneered by Nagao *et al.* (1979). A C language integrated production system (CLIPS) has also been recently developed that is written in standard C code and incorporates a set of tools for building rule-based systems (Giarratano, 1987). CLIPS uses a LISP-like syntax and is primarily an interpreted system, even though a compilation option is available.

METHODOLOGY

The extensions to C correspond to the basic constituents of any rule based system; the most important extensions are "rules," which include a number of new predicates to help in rule specification, and "phases," which implement the search procedure. A new data type, "list," is added for convenient handling of variable length data sequences, such as the vector description of image regions. Features are also provided to facilitate the input and output of data into and from the knowledge base.

The new features in C are used to build a rule-based system that operates in the following manner. Images are pre-processed (segmented) into a list of spatial segments. The rule-based system assigns values for a number of labels to each segment. The rules used in the classification process are grouped into three phases, each of which determines the values for the segments' label in that phase. It is important at this point to distinguish between labels and their values. More than one value may be assigned to any label, allowing multiple interpretations or classifications. A value is assigned to a label when a rule in a given phase evaluates to true for a given segment. If more than one rules "fires," a tree is created, where each node of the tree represents a value assigned to a label. This concept is shown in Figure 1, where the label assigned in the first phase has the two values "soil" and "asphalt," and the label assigned in the second phase, which is also transferred to the third phase, has five values.

The label tree in Figure 1 is built in a breadth-first manner, one level per phase. Phase I starts with the root (the image segment) and creates the first level of the tree using spectral rules. Parallel with the assignment of a label, a probability value is assigned that represents the (heuristic) confidence in the particular label. Geometric rules are evaluated in Phase II for each node at the lowest level in the tree, and a new level is appended to the tree. Phase III alters each value's probability based on neighboring segments' labels and probabilities. In AI terminol-

PHOTOGRAMMETRIC ENGINEERING AND REMOTE SENSING,
Vol. 56, No. 6, June 1990, pp. 887–892.

0099-1112/90/5606–887$02.25/0

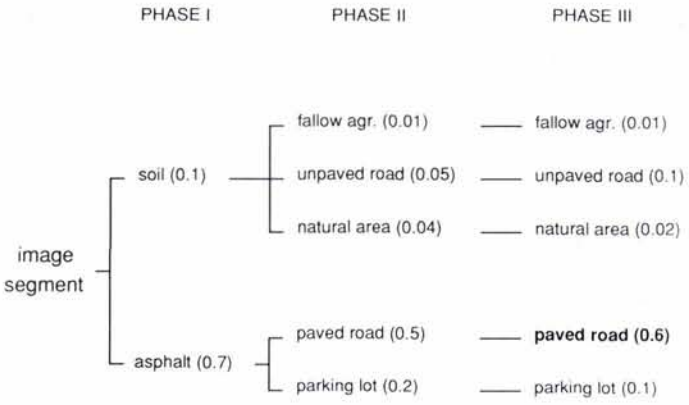


FIG. 1. Hypothetical label tree for one image segment. The associated probability of each label is indicated in parentheses. The label with the highest probability used in the final map is shown in **bold**.

ogy, this system uses forward-chaining with the goal that, in the end, the rules will result in some classification that is clearly most likely.

The final value for the label of a segment is taken as that with the highest probability after Phase III. However, all possible classifications are enumerated in the output. Rules can refer to labels assigned in earlier phases, in both their conditional and consequential parts, and thus refine earlier classifications. It is also possible to assign the same label in more than one phase. This is useful if new information has been computed and old labels (or their probabilities) are no longer appropriate.

EXTENSIONS TO C

In this section, the extensions to C are introduced by example. For a formal definition in the sense of programming languages, and for a description of their implementation, see Mehldau (1986).

List Data Type. Lists have been added to C as a new data type to make possible the convenient handling of data sequences of variable length. An example of a function that makes use of lists is given in Figure 2. The argument to the function is a list of characters. The function builds and returns a new list that contains all lowercase characters from the argument list.

Lists are declared with the "@"-sign in a syntax similar to the "*" or "pointer-to" notation in standard C. List constants are specified as expressions or range expressions within square brackets; the example in Figure 2 shows the degenerate case of an empty list. Elements of a list are addressed like elements of an array — by following the list with an expression (or a range expression) in square brackets. Finally, lists can be concatenated using the "*"-operator. A function `len()` is provided which returns the length of its list argument.

Input/Output. It is assumed that the input for classification consists of a list of records of uniform format, albeit of varying lengths. An example for an input declaration is given in Figure 3, where an individual record consists of an integer, `id` , two real numbers, `a` and `b` , and a list of characters, `string`.

The system generates an input function that reads a list of records according to the input specification and stores them in an implicit system variable. The system also generates an output function which writes out the input as well as the results of the classification process in a standardized form. Users familiar with the system have the option of writing their own output function.

Control Structures. The two control structures added to C are called phases and rules. A phase contains a set of rules which produce one level of the label tree. The sample phase shown in Figure 4 assumes the input configuration from the example in Figure 3. It obtains two additional values, and assigns a label

```
char @example (list) char @list; {
    int i;
    char @newlist;

    newlist = [];
    for (i = 0; i < len(list); i++) {
        if ('a' <= list[i]  &&  list[i] <= 'z') {
            newlist *= list[i..i];
            /* ... or it could be written as ... */
            /* newlist = newlist * [list[i]];    */
        }
    }
    return (newlist);
}
```

FIG. 2. Sample list function.

```
input {
    int id;
    double a, b;
    char @string;
}   example;
```

FIG. 3. Sample input declaration.

```
input { int id; double a, b; char @string; };

phase example (label) char *label; {
    int value_1, value_2;

    obtain_values(&value_1,&value_2);

    rule one
    value_1 < ?->a  &&  ?->a < ?->b  &&  ?->b > value_2   =>
        assign (1.0) "increasing sequence";

    rule two
    value_1 > ?->a  &&  ?->a > ?->b  &&  ?->b > value_2   =>
        assign (1.0) "decreasing sequence";

    printf("Phase finished\n");
}
```

FIG. 4. Sample phase declaration.

to the current record if the sequence value_1, a, b, value_2 is either increasing or decreasing. The question mark in the code stands for a pointer to the record being processed.

Phase declarations are very similar to function declarations in standard C — they consist of a name, a label to be assigned in parentheses, and a body. The body of a phase, in turn, contains declarations, statements, and rules. Rules are name and consist of a conditional part, followed by a "=>" and a consequential part. The consequential part only needs to contain a label assignment (as in the example above), but it can also be a block consisting of declarations, statements, and label assignments.

Expressions. To help the user of the system access the input data and express properties of objects, the syntax of standard C for expressions has been enhanced. Some of these enhancements are shown in the example in Figure 5, which again assumes the input configuration from Figure 3. Both rules scan the character string of the current record to determine which label to assign. The question mark ("?") denotes a pointer to the current input record.

Three new expressions help simplify the formulation of rule-predicates: the forall- and exist- quantifiers, and the in-expression. All three expressions yield boolean (integer) values with the obvious meaning. In addition, the probability associated with an already assigned label can be obtained with the function `prob()` .

RESEARCH DESIGN

We used the extended C language to write an experimental rule-based system for classification of a digitized aerial color infrared image of Tucson, Arizona (Plate 1a). This was a three-band image, 512 by 512 pixels, with a ground resolution of approximately 4 m. The image was processed on a VAX-11/750 minicomputer, and the image processing software used was the "System at Arizona for Digital Image Experimentation" (SADIE, 1988) and International Imaging System's interactive image processing package (I2S, 1984).

```
input { int id; double a, b; char @string; };

phase example (label) char *label; {
    char c;

    rule one
    for_all c in ?->string: c in ['a'..'z']    =>
        assign (1.0) "lower case";

    rule two
    (for_all c in ?->string: c in ['a'..'z','A'..'Z'])  &&
    (there_exists c in ?->string: c in ['a'..'z'])  &&
    (there_exists c in ?->string: c in ['A'..'Z'])    =>
        assign (1.0) "mixed case";
}
```

FIG. 5. Sample phase declaration using new expressions.

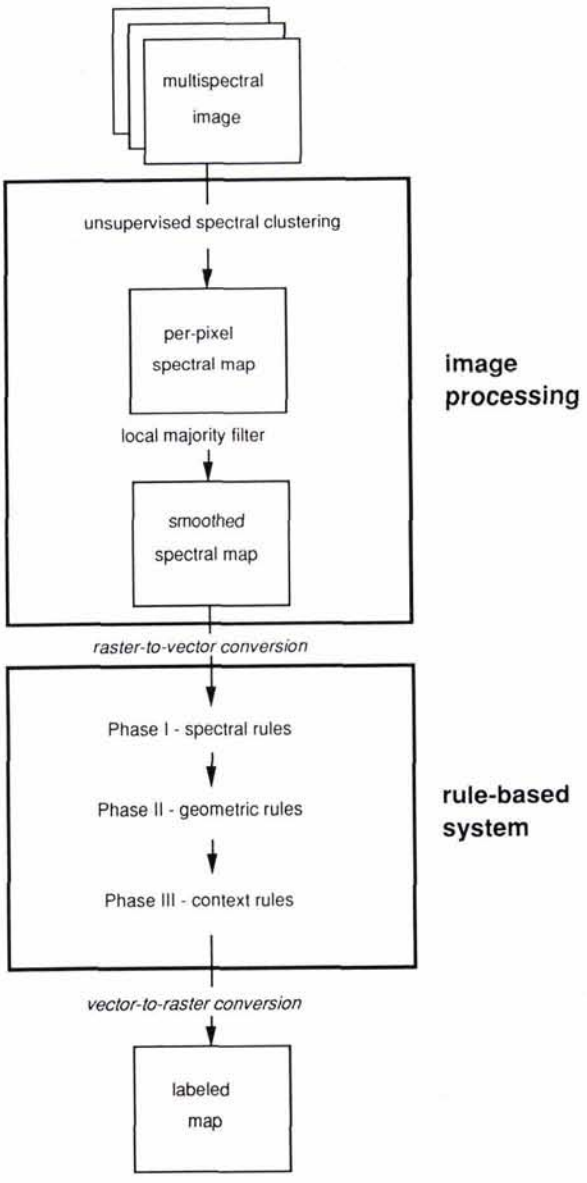


FIG. 6. Paradigm for combining image and rule-based processing.

IMAGE PREPROCESSING

The image was first submitted to an unsupervised statistical multispectral classifier (an implementation of the k-means clustering algorithm with a nearest-mean decision rule) that assigned each pixel independently to one of eight spectral clusters. Eight clusters were chosen to yield a compromise between spectral resolution and the average size of spatial segments in the cluster map. The classifier also provided the global statistics (mean and variance of the pixels in the three spectral bands) for each of the clusters. Pixel noise in the cluster map was reduced with a local majority filter (Schowengerdt, 1983), thus introducing some spatial consistency. The resulting smoothed, spectral cluster map constituted a segmented image. Two 128 by 128 windows (Plates 1b and 1c) were further processed. Spatially connected segments were consecutively numbered with a "blob-coloring" algorithm (Ballard and Brown, 1982), and the boundaries of the segments were obtained with a modified "turtle" algorithm (Papert, 1973) to yield a vector-format. The windows in Plates 1b and 1c contained approximately 1700 and 3200 spatially connected segments, respectively.

The input to the rule-based system was a list of segment descriptions, each consisting of a unique ID (the segment number), the mean gray levels in all three bands for the pixels in that segment, a list of integer (pixel) coordinates that describe the boundary of the segment, and, finally, a list of the IDs of all neighboring segments. The latter information provided the basis for contextual rules applied in the third phase. The overall flow of data in this system is illustrated in Figure 6.

IMPLEMENTATION OF THE RULE-BASED SYSTEM

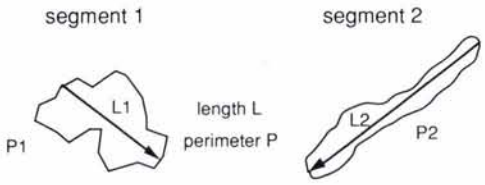
The classification of segments was done in three phases, as described below. The parameters and content of the rules in each phase were set heuristically, with a few iterations performed for adjustments.

Phase I: Spectral Information. The first phase consisted of eleven rules and attempted to classify the segments into basic categories according to their spectral characteristics. The six categories used in this phase were "vegetation," "water," "soil," "asphalt," "sand," and "gravel," i.e., rather general labels that could be assigned from spectral signatures. The following is an example of a rule for this phase:

```
rule spect_3
dist [0] <= MAXDIST[0]   =>  assign (mindist/dist[0])
  SOIL;
```

The three spectral bands define a three-dimensional space. The rule compares the spectral distance between the segment signature and one of the eight clusters (`dist [0]`, determined during the preprocessing of the image), to some threshold (`MAXDIST[0]`), and conditionally assigns the label together with a probability which depends on some other threshold (`mindist`). Thresholds were determined manually from the global spectral clustering statistics. Because the gray levels in the spectral bands had not been calibrated, the thresholds would not be valid for a different image. Note that the statistical classifier in the preprocessing stage assigned a cluster to each pixel, whereas the rules in Phase I made this decision for a segment as a whole, based on the mean spectral values for the segment. In Phase I the cluster classes were also given names.

Phase II: Geometric Information. The second phase used the geometric characteristics of the segments (derived from the



area	A1 = A2
curvilinearity $C = 4\sqrt{A}/P$	C1 > C2
aspect ratio AR = L**2/A	AR1 < AR2
raggedness $R = \sum_{P}$ |angle between adjacent vectors|	R1 > R2

FIG. 7. Geometric features used in Phase II. Two hypothetical segments with equal areas but different shapes are shown. Length is defined as the longest diameter of the segment.

description of their boundaries) to refine the classification of the first phase. The labels here were "active agriculture," "urban vegetation," "natural vegetation," "river," "lake," "fallow agriculture," "unpaved road," "natural area," "paved road," "parking lot," "gravel road," "commercial building," and "residential building." The incorporation of geometric information in Phase II, and context information in Phase III, made this relatively high level of land-use discrimination feasible. In this phase, we used rules like the following:

```
rule geom_4
?->spl == WATER && curvilinear < 0.3 && aspect_ratio
  > 10.0 => assign (prob(?->spl)) RIVER;
```

Based on the classification result from the first phase (?->spl) and on the values of the (precomputed) geometric features curvilinear and aspect_ratio, this rule assigns the label "river." The four geometric features used in Phase II are illustrated in Figure 7.

Phase III: Context Information. The third and potentially most powerful phase reassessed the probabilities assigned to the labels in the second phase. This was done by looking at the neighboring segments and increasing or decreasing the confidence values of the labels assigned in the second phase according to the overall consistency. An example of a rule in this phase is

```
rule context_14
?->label == PARKING_LOT && for_all element in
  ?->neighbors: element->label in [PAVED_ROAD,
  GRAVEL_ROAD,COMMERCIAL_BUILDING] => assign
  (min(prob(?->label)*(1+0.05*len(?->neighbors)),1))
  ?->label;
```

This rule represents knowledge about spatial relationships between adjacent features in a scene, in the sense that certain types of objects are usually found next to certain other types. If the classification of an object is consistent with its neighbors, the probability for the particular label is increased. If the classification of an object is not consistent with the labels of its neighbors, the probability of that classification is decreased. This technique of improving the overall consistency of the classification, when applied iteratively, is called "constraint relaxation" (Levine, 1985). The results presented here represent only one pass through Phase III. Other approaches to the use of context in image classification are described by Wharton (1982) and Tilton *et al.* (1982).

RESULTS AND DISCUSSION

Computational Performance

The preprocessor and the experimental rule-based system have been implemented on a VAX-11/785 under the UNIX operating system. The rule-based system was specified with approximately 300 lines of extended C. The preprocessor expanded this into a standard C program of almost 900 lines, which was then compiled under the UNIX C-compiler with full optimization. The whole process took under one minute of real time, and the C-compiler alone accounted for approximately 80 percent of that amount.

The resulting C program was linked with the code that implements the diverse list functions (approximately 230 lines), and run on the test data described earlier. During the first and second phase, between two and three labels per segment and phase were assigned on the average. The third phase only reassessed the probabilities of the labels in the second phase, so it did not continue this expansion. The first window (1662 individual segments) was processed in 3.0 minutes of real time, and the second window (3224 segments) took 4.2 minutes. Again, these times are quite reasonable, especially if compared to some of the image processing routines (the "turtle"-algorithm, for example, took about 10 minutes of real time).

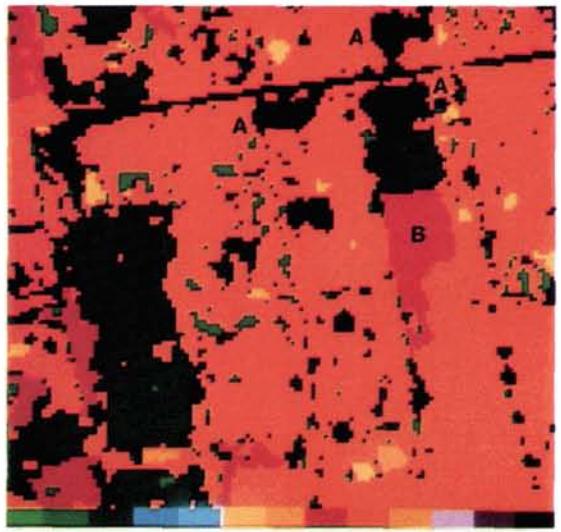
To show the superiority of our system over an interpreted system, the rules were implemented in LISP and run on the same test data. The C-based system proved to be faster by a factor of 260! Due to problems with the available LISP compiler (notably with respect to trigonometric functions), no comparison could be made between our system and compiled LISP. To determine the efficiency of the system, a profile of the running program was obtained with UNIX profiling software. Only about 10 percent of the execution time was required for each phase of the rule-based system; the remainder was used for data I/O and list processing.

Classification Results

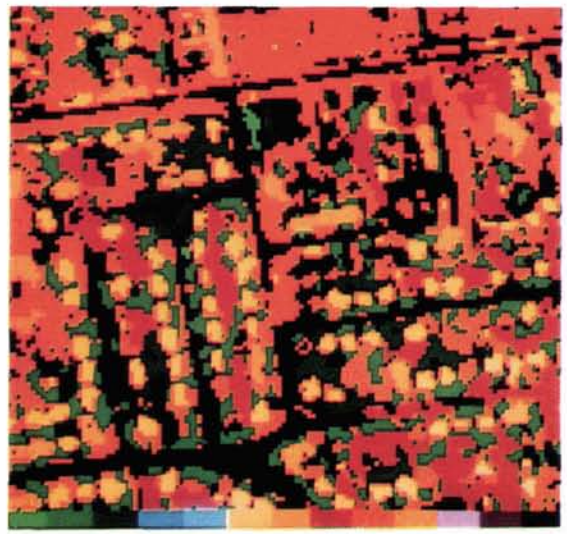
The final classifications of the two windows are shown in Plates 1b and 1c. In the non-urban window (Plate 1b), the results are generally reasonable, with some anomalies, such as the inclusion of three dark fields (A) in the paved road object because they are connected by a few pixels to the actual road. Also note the classification of the field (B) as a parking lot because of its dark spectral signature and large size. This indicates weaknesses in the Phase III context rules for "parking lot."

The classification of the urban window (Plate 1c) is worth examining in some detail. In Plates 1d, 1e, and 1f we show an enlargement of part of the window containing houses, lawns and trees, and streets. The per-pixel spectral cluster map contains considerable classification "noise," resulting in many spatial segments that are not clearly related. The final map from the rule-based system shows the benefits gained by incorporating segment geometry and context. Many neighboring, but spectrally different segments are merged because of the context rules. This occurs even though we have doubled the number of possible class labels from Phase I to Phase III and applied only one cycle of probability modification based on spatial context. Such merging of spectral categories is common practice in manually labeled cluster maps. However, it is being achieved here in a combined space of spectral, geometric, and context features in an "automated" fashion (given the pre-defined rules and their parameters). A simple count of houses (an example residential building is labeled R in Plates 1d, 1e, and 1f) from these images

(a)



(b)



(c)

1 urban vegetation
2 natural vegation
3 active agriculture
4 lake
5 river
6 residential building
7 commercial building
8 parking lot
9 natural area
10 fallow agriculture
11 gravel road
12 unpaved road
13 paved road

(d)

(e)

(f)

PLATE 1. Classification results: (a) color infrared aerial photograph with study areas outlined, (b) non-urban map, (c) urban map, (d) aerial photograph enlarged, (e) spectral cluster map, (f) final map (same color code as in (c)).

yields 16 from the center block of Plate 1d and 11 from the final map, Plate 1f, or an accuracy of about 70 percent for the "residential building" class. Some categories, notably "parking lot," remain troublesome for dark regions of the image, again pointing out the need for improvement in Phase III rules for this category.

CONCLUSIONS

An extension of the C language for the generation of rule-based systems has been presented. The implementation of this system has been shown to be feasible and more efficient than comparable interpreted systems.

A multispectral, high resolution image was segmented by multispectral clustering, converted to vector format, and processed by a rule-base system written in the extended C language. The segments were labeled using spectral, geometrical and contextual rules.

The geometric phase generally was not as discriminating as expected. In our paradigm, geometric features were calculated from the per-pixel spectral segmentation. Because only spectral information was used for the initial segmentation, different objects in the image sometimes appeared as a single segment, and single objects sometimes appeared as multiple segments. Thus, the derived geometric parameters were not necessarily representative of the true objects. Spatial-spectral segmentation schemes (Ballard and Brown, 1983; Levine, 1985) should be an improvement. Also, the simple geometric features used here were probably insufficient for the complex shapes that occur in the non-urban areas.

The third phase used the neighboring segments to reinforce or suppress an existing classification for a given segment. The forall- and exist-quantifiers in this phase might not be appropriate for "real world" problems. Quantifiers such as "most" and "few" might be more appropriate and could be achieved by having "forall" and "exist" return probabilities instead of binary values. Any context information, however, appears to be useful in both urban and non-urban areas, even if it only extends to the nearest neighboring segments.

ACKNOWLEDGMENTS

The authors would like to thank Dr. Eugene Myers of the Department of Computer Science at the University of Arizona, who provided the "front end" (lexical analyzer and parser) for our preprocessor as well as an efficient implementation of the list data type. We also wish to thank the reviewers for their valuable comments on the manuscript.

REFERENCES

Ballard D., and C. Brown, 1982. *Computer Vision*. Prentice-Hall, Englewood Cliffs, New Jersey.

Giarratano, J., 1987. *CLIPS User's Guide Version 4.10*. Artificial Intelligence Section, NASA Johnson Space Center.

I2S, 1984. System S570 Digital Image Processing System, *Version 3.0 User's Manual*. International Imaging Systems, Milpitas, California.

Kernighan, B., and D. Ritchie, 1978. *The C Programming Language*. Prentice-Hall, Englewood Cliffs, New Jersey.

Levine, M., 1985. *Vision in Man and Machine*. McGraw-Hill, New York, N.Y.

Mehldau, G., 1986. *A Rule-Based Programming Language and its Application to Image Recognition*. M.S. Thesis, University of Arizona, Department of Computer Science.

Nagao, M., T. Matsuyama, and Y. Ikeda, 1979. Region Extraction and Shape Analysis in Aerial Photographs. *Computer Vision, Graphics and Image Processing*, Vol. 10, No. 3, pp. 195–223.

Papert, S., 1973. *Uses of Technology to Enhance Education*. Massachusetts Institute of Technology, AI Lab, Technical Report No. 298.

SADIE 4.0, 1988. *User's Manual and Subroutine Reference Manual*. University of Arizona, Digital Image Analysis Laboratory.

Schowengerdt, R., 1983. *Techniques for Image Processing and Classification in Remote Sensing*. Academic Press, New York, N.Y.

Schowengerdt, R., and G. Mehldau, 1987. Classification of Multispectral Imagery with a Rule-Based System. *Proceedings of the 40th Annual Conference of the Society of Photographic Scientists and Engineers*, Rochester, N.Y., pp. 186–189.

Tilton, J., S. Vardeman, and P. Swain, 1982. Estimation of Context for Statistical Classification of Multispectral Image Data. *IEEE Transactions on Geoscience and Remote Sensing*, Vol. GE-20, No. 4, pp. 445–452.

Wharton, S., 1982. A Context-Based Land-Use Classification Algorithm for High-Resolution Remotely Sensed Data. *Photographic Engineering*, Vol. 8, No. 1, pp. 46–50.

(Accepted 18 January 1990)