JPL PUBLICATION 77-43

Cluster Compression Algorithm A Joint Clustering/Data Compression Concept

National Aeronautics and Space Administration

Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California 91103

JPL PUBLICATION 77-43

Cluster Compression Algorithm A Joint Clustering/Data Compression Concept

Edward E. Hilbert

December 1, 1977

National Aeronautics and
Space Administration

Jet Propulsion Laboratory
California Institute of Technology
Pasadena,California 91103

Prepared Under Contract No NAS 7-100
National Aeronautics and Space Administration

PREFACE

The work described in this report was performed by the Information Systems Division of the Jet Propulsion Laboratory. This work was also performed in the course of completing a Dissertation presented to the University of Southern California as partial fulfillment of the requirements for a Doctor of Philsosophy Degree.

ACKNOWLEDGMENT

I should like to sincerely acknowledge the following people who have supported the effort in this report. Thanks go first to my colleagues at the Jet Propulsion Laboratory, S. S. Salisbury, H. S. Hendeles, and J. E. Getz for providing computer simulation support, and R. F. Rice, Dr. C. L. Wu and R. G. Piereson for their technical suggestions. I am also indebted to the members of my dissertation committee, Dr. L. D. Davisson, my committee chairman, and Drs. H. C. Andrews and R. E. Bruck. I also gratefully acknowledge the considerations of Arlene LaVertu and Carol George for their typing efforts.

TABLE OF CONTENTS

TABLE OF CONTENTS (contd)

TABLE OF CONTENTS (contd)

TABLE OF CONTENTS (contd)

LIST OF FIGURES

LIST OF FIGURES (contd)

LIST OF FIGURES (contd)

LIST OF TABLES

ABSTRACT

This report describes the Cluster Compression Algorithm (CCA), which was developed to reduce costs associated with transmitting, storing, distributing, and interpreting Landsat multispectral image data. The CCA is a preprocessing algorithm that uses feature extraction and data compression to more efficiently represent the information in the image data. The format of the preprocessed data enables simple look-up table decoding and direct use of the extracted features to reduce user computation for either image reconstruction, or computer interpretation of the image data. Basically, the CCA uses spatially local clustering to extract features from the image data to describe spectral characteristics of the data set. In addition, the features may be used to form a sequence of scalar numbers that define each picture element in terms of the cluster features. This sequence, called the feature map, is then efficiently represented by using source encoding concepts. Various forms of the CCA are defined and experimental results are presented to show trade-offs and characteristics of the various implementations. Examples are provided that demonstrate the application of the cluster compression concept to multispectral images from Landsat and other sources.

CHAPTER I

INTRODUCTION

A. Objectives of Data Redundancy Reduction

Early removal of redundant data is important in reducing costs and time delays in the link between the sensor data and the users' results, or alternately early removal of redundancy can be used to obtain higher performance. Data redundancy reduction, or information extraction is especially important in data systems involving high rate remote data sensors which require related storage, communication, archiving, distribution and interpretation subsystems. Imaging data systems are important examples for application of early information extraction. Data rate reductions, or compression ratios in image approximation applications often approach and may exceed an order of magnitude. In applications of computer classification of the image data, the rate reductions may be many orders of magnitude. An ever increasing trend toward automatic interpretation emphasizes the need for data redundancy reduction jointly suited to computer classification and approximation of images.

The objective of this work is to develop a joint feature extraction/data compression technique for removal of redundant data in image approximation and computer classification applications. A technique is described which jointly applies clustering and source encoding concepts to obtain data compression. An algorithm, called the Cluster Compression Algorithm, for implementing this concept is investigated with emphasis given to practical data system considerations.

B. Practical Data System Considerations

The emphasis in this work is on the development of a practical information extraction technique which benefits the entire data system. The performance of a data compression technique must be judged by more than its compression ratio. Other system aspects for which the impact and interaction of data compression should be assessed are data storage, transmission and channel errors, distribution and archiving, and user interpretation.

A block diagram of a practical data system is shown in Fig. 1.1. All of the subsystem elements shown in Fig. 1.1 are not present in every application. For example, in many systems the sensor-located storage, channel coding/decoding, archiving and distribution are unnecessary. The sensor design has an important impact on data redundancy. The sensor acquires data which may range from inadequate to more than adequate to provide the information desired, but in either situation there is usually considerable extraneous measurement data. Flexible instrument control strategies can be used to reduce data redundancy, for example, through control of measurement rate and resolution, dynamic range scaling, and simple on-off functions.

Channel errors have a more serious impact on data after it has been compressed. Ideally one would like to jointly solve the problems of source encoding to remove redundancy and channel encoding for error protection. From a practical standpoint, however, the present approaches usually involve using increased transmitter power to lower the error rate, and the use of periodic references in the compressed data for source decoding restarts to limit the source decoding error

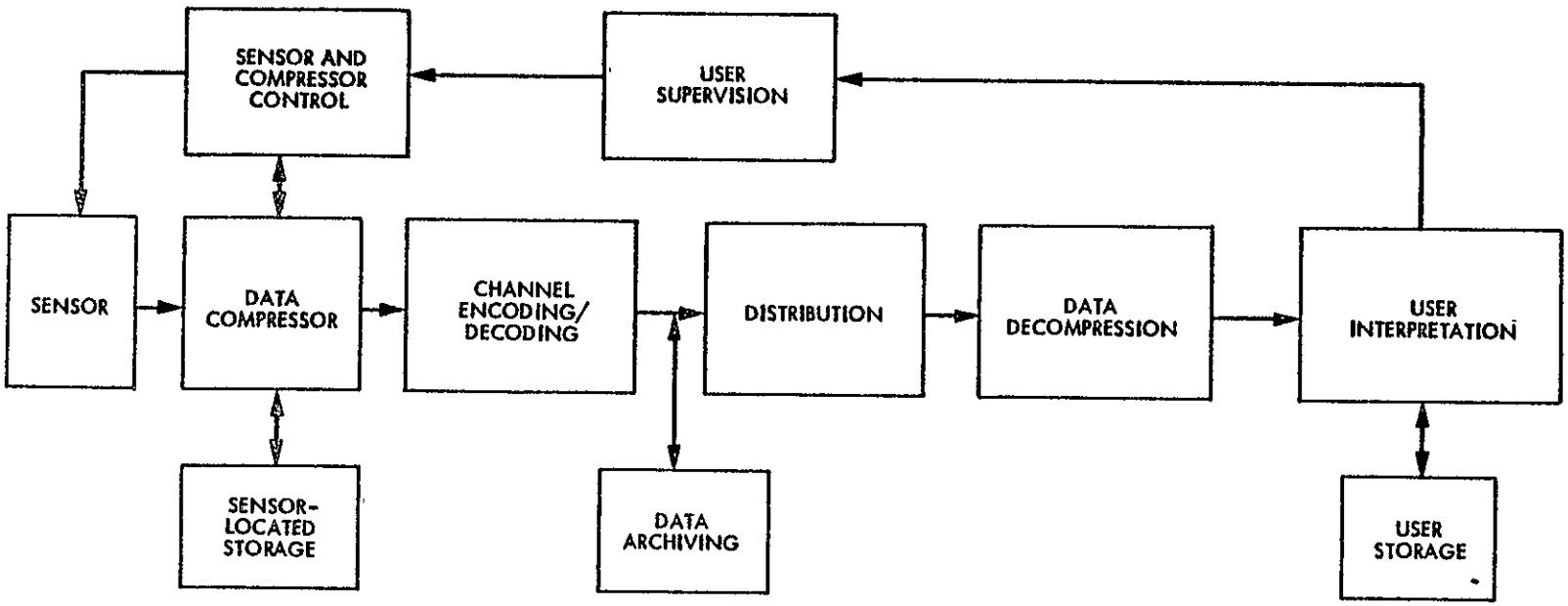


Fig. 1.1. A Block Diagram of a Practical Data System.

propagation due to a channel error. An alternate approach derived in [1] uses channel encoding to group channel errors into long bursts. Since compressed data is degraded nearly equally by a single error and a burst of errors, the use of channel coding to group errors permits using a higher average error rate.

In high rate data systems involving large scale archiving, distribution and interpretation, redundant data can cause serious cost and time delay problems. However, data compression may not benefit these subsystem functions if complicated decoding is required before selective access of the data for distribution and interpretation is possible. Benefits would result if the data compression could be obtained through extraction of only essential features which can be selectively accessed and interpreted directly without extraneous decoding.

The preceding discussed the interactions of practical data system functions which need to be considered when incorporating data compression into a system. Now consider the desirable characteristics of the data compressor itself. In general, the process of redundancy removal might occur in various stages, at the sensor and later in the communication link. In many cases a data system requires data compression for widely different applications. Even for a dedicated application, the user definition of adequate quality is an uncertainty which suggests a need for capability to change. Therefore, a good data compressor emphasizes flexibility, allowing a high performance rate vs. quality trade-off over wide ranges of compression ratios and applications. In order to aid the user in supervising the redundancy removal, a data compressor should use features which provide easy understanding of the

relationship between the feature quality and the resulting quality of image approximation, or computer classification. Basically, the data compressor should allow the user to attempt reduction of the data directly to the desired information.

C. Landsat As a Specific Practical Example

In view of the emphasis in this work on practical data compression an existing data system will be used throughout to exemplify the results of investigating the Cluster Compression Algorithm. A particularly demanding, but practical data system for applying data redundancy reduction is NASA's Landsat, formerly known as the Earth Resources Technology Satellite (ERTS). Landsat I is in a sun synchronous polar orbit at an altitude of 915 km and orbits the Earth every 103 minutes. A block diagram of the major elements in the Landsat data system is shown in Fig. 1.2, and additional details of these elements can be found in [2].

There are other sensors on-board Landsat I, but attention here is focused on data-management for the high rate Multispectral Scanner (MSS) imaging sensor. The MSS sensor continuously scans 6 lines in 4 spectral bands between 0.5 to 1.1 μm. The instantaneous field of view (IFV) is 79 m on a side and the cross track motion of the IFV is 56 m per sample. Every picture element is a vector with four spectral components, each digitized to one of 64 levels (6 bits) of radiation intensity. A new picture element is scanned approximately every 1.6 μs producing a 15 Mbps data rate. The resulting PCM data is temporarily stored on a Wide

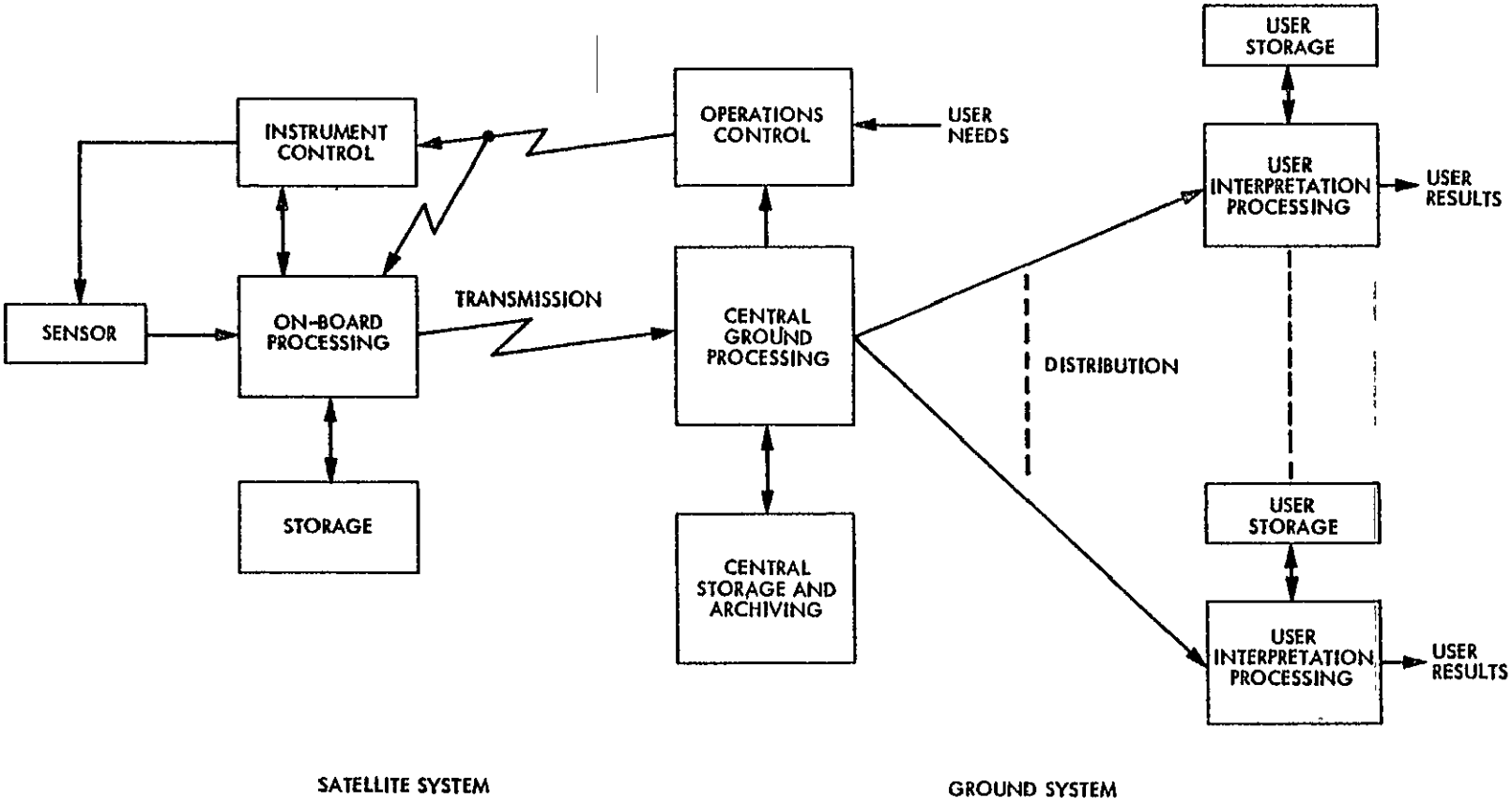


Fig. 1.2. A Block Diagram of the Major Elements in the Landsat Data System.

Band Video Tape Recorder and/or used to FSK modulate a carrier for transmission to Earth with an average error rate less than 10^{-5}.

The NASA Data Processing Facility (NDPF) at Goddard Space Flight Center in Greenbelt, Maryland segments the continuous MSS data into 185 Km square scenes and provides standard output products of black and white, and color negatives, positives and prints in 70 mm and 241.3 mm enlargements as well as computer compatible digital tapes. The NDPF also provides radiometric, geometric and other corrections to the MSS data, but it does not perform any redundancy reduction, since the satellite is presently a research vehicle. The NDPF products are archived and distributed to many users over the entire world.

User analysis techniques include film interpretation, computer classification, predictive modeling and combinations of these methods. By analyzing the sampled electromagnetic energy in four spectral windows the investigators can extract information from the image data which is useful in water and crop management, urban analysis, pollution detection, weather and flood prediction, mapping, and many other applications. Discussions of applications of Landsat MSS data are given in [3] and [4].

Presently there is no data compression on-board the satellite, because Landsat has been used principally as a research vehicle to determine the beneficial applications of remote sensing. The future outlook is for operational satellites which gather data for specific purposes determined to be useful through previous research. These operational satellites will increase their capabilities by incorporating higher

spatial and spectral resolution. A reasonable improvement to 10 m spatial resolution and 10 band spectral resolution, however, results in an increase in data rate to over 2×10^9 bps, justifying the concern over the often mentioned data explosion.

For the Landsat data system one must consider more than just the use of on-board data compression to reduce the channel transmission rate requirements. An important part of the Landsat data management problem is the ground data handling (e.g., data storage, retrieval, distribution and intepretation). Furthermore, the Landsat data system is an example which involves both user needs of visual image interpretation and computer classification. Thus this data system serves as a good example application for a joint feature extraction/ data compression technique.

D. Prior Efforts in Feature Extraction and Data Compression

Historically, the emphasis in data compression has been in developing techniques to approximate the raw data with fewer bits required for transmission. Techniques such as DPCM [5], Transform methods [6] - [8], and combinations of these techniques combined also with variable-length entropy encoding [9] have received much attention in the literature. Such approaches are suitable, for example, in applications concerned with subjective image quality, but they are less suitable in applications involving feature extraction or classification of the data. For example, a feature of interest in the image may be difficult to emphasize in terms of differences used in DPCM, Hadamard coefficients in Hadamard Transform techniques, or in terms of the principal

components resulting from the Karhunen-Loève Transform. Even more so for classification applications, it would generally be difficult to tailor the type of data degradation introduced so as to tend to preserve classification accuracy. This decreases the efficiency in redundancy reduction and complicates the user requirement input interface. Furthermore, these approaches generally require decompression of the compressed data to its expanded form before selective access and interpretation is possible.

In those applications where features or classification are the desired end product, it is reasonable to consider the use of pattern recognition theory. Pattern recognition and the subsets of feature extraction and classification are discussed with extensive references in the surveys of [10] and [11].

A very common classification approach is supervised statistical classification with a parametric model of multivariate Gaussian [12]. Classification for Landsat data is usually done on a per picture element basis because of the large volume of data and extra computation required when texture is used in classification. A discussion and analysis of using texture in classification is given in [13].

Much work in feature extraction has centered around finding measures of distance between distribution functions, such that the distance measures are well correlated with the probability of error associated with the distribution functions. These measures of distance are easier to calculate than the probability of error and, therefore, reduce the computation required to determine the performance of selected features in the sense of classification accuracy. A

discussion of such distance measures and their uses are given in references [13, pp. 27-47] and [14].

Other often used feature extraction and classification is based on the nonsupervised techniques of clustering [15], [16]. Clustering is a technique for extracting statistical features from a data set by grouping data vectors of similar characteristics. The clustering of data into groups can be used to directly provide nonsupervised classification, or to provide statistical features for groups of data from which further classification can be conducted. The application of clustering for feature extraction and nonsupervised classification to remote sensing applications such as Landsat is discussed in [13], [17], and [18].

Feature extraction and classification as described in the above references are primarily considered interpretation techniques to be performed by the user. However, if the feature extraction or classification is performed at the source it is clearly providing data compression. For example, if every picture element was classified to one of eight classes at the data source, then a total of three bits could be used instead of the six bits per each band originally used to define the spectral signature of each picture element. This type of data compression also directly provides the user with desired results, eliminating the step of first decoding the compressed data to obtain an approximation of the raw data before interpreting. One obvious problem with using feature extraction and classification at the data source is the large computational complexity associated with these techniques. Another practical problem, often overlooked, is the difficulty in providing supervision to the classifier located with a remote sensor.

The prior research basically consists of the following: 1) data compression techniques suitable for image approximation, but not well suited for feature extraction or classification; and 2) feature extraction and classification techniques not well suited for image approximation, or for practical implementation at remote sensors. This dissertation develops a technique which uses feature extraction to obtain data compression. Modifications to the feature extraction technique of clustering are investigated which make it useful for image approximation as well as practical to implement. In addition, the data compression technique of entropy coding is combined with feature extraction to further increase performance. This approach of joint feature extraction/data compression is called the Cluster Compression Algorithm and some of the initial investigation results are contained in [19] and [20].

E. A Joint Feature Extraction/Data Compression Model

Assume the data source is the multispectral image source modeled by the continuous random process $s(y_1,y_2,w)$ which is the electromagnetic energy at wavelength w for spatial coordinates y_1 and y_2. The measurement vector elements of a digitized d-band multispectral image are then represented by the vector $\underline{X}(Y_1,Y_2)$ obtained from $s(y_1,y_2,w)$ by discretizing the variables y_1, y_2 and w to give

$$\underline{X}(Y_1,Y_2)=[s(Y_1,Y_2,W_1),s(Y_1,Y_2,W_2),\ldots,s(Y_1,Y_2,W_d)]. \qquad (1.1)$$

Figure 1.3 shows a model for the general concept of joint feature extraction/data compression. The Cluster Compression Algorithm defined

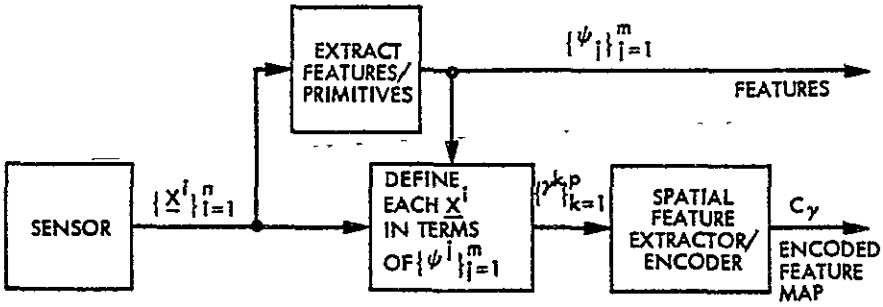


Fig. 1.3. A Joint Feature Extraction/Data Compression Model.

in the next chapter is a specific form of this more general model. Let $\{\underline{X}^i\}_{i=1}^n$ be a sequence of n measurement vectors obtained from a subset of multispectral image data. The entire sequence $\{\underline{X}^i\}_{i=1}^n$ is analyzed to extract features $\{\psi^j\}_{j=1}^m$ for the sequence $\{\underline{X}^i\}_{i=1}^n$. These features could also be considered as primitives, or as a basis for approximating the measurement vectors. The sequence of m features $\{\psi^j\}_{j=1}^m$ provides a description of the characteristics pertaining jointly to the entire sequence of measurement vectors $\{\underline{X}^i\}_{i=1}^n$ and in some applications may be the only output required. In other applications each measurement vector needs to be approximated in terms of $\{\psi^j\}_{j=1}^m$ by the sequence of scalar numbers $\{\gamma^k\}_{k=1}^p$ which assigns to each measurement vector an approximation in terms of one of the primitives. Alternately, the concept of Fuzzy Set Theory [21] might be used to allow each measurement vector to be described by a weighted mixture of more than one of the features in the sequence $\{\psi^j\}_{j=1}^m$. The scalar sequence $\{\gamma^k\}_{k=1}^p$ constitutes a spatial map of the measurement vectors in terms of the primitives, and is called the feature map. Spatial features can be extracted from $\{\gamma^k\}_{k=1}^p$, and source encoding can be used to efficiently represent the spatial characteristics of $\{\underline{X}^i\}_{i=1}^n$ through the encoded feature map denoted by C_γ. For each sequence $\{\underline{X}^i\}_{i=1}^n$ of measurement

vectors the model basically uses feature extraction concepts to determine a set of features or primitives, and then the model uses data compression techniques to efficiently represent the spatial features in terms of the primitives.

The Cluster Compression Algorithm uses clustering to extract multidimensional primitives from subsets of spatially contiguous measurements in the image data and then applies entropy encoding to the feature map. Another independently developed data compression technique related to the model in Fig. 1.3 is the Blob algorithm [22]. This algorithm is basically a boundary finding algorithm that guarantees closed boundaries. A blob is defined as a connected set of picture elements all of which have some common characteristic. In the Blob algorithm the primitives are defined by the boundary finding algorithm and consist of statistical descriptions of the measurement vectors contained within the various blobs. Spatial definition can then be provided by encoding a sequence which defines the spatial boundaries of the blobs. This algorithm is most useful as a data compressor when the image consists of well defined boundaries enclosing picture elements of nearly uniform characteristics, (e.g., agricultural fields). The Cluster Compression Algorithm is different in that it extracts primitives and obtains significant compression independent of the spatial characteristics of the data. Thus the cluster compression technique can efficiently compress images whether or not they contain uniform areas with well defined boundaries. In addition, if the image does consist of uniform areas with well defined boundaries, such as fields,

the entropy encoding of the feature map results in efficient representation of these spatial boundaries.

F. Organization of the Dissertation

In Chapter II the use of clustering as a feature extractor is discussed and several clustering algorithms are defined for the CCA. In Chapter III the basic Uncoded CCA is defined and examples are included of its use in image approximation and automatic image classification.

Chapter IV investigates entropy bounds for a class of entropy coders and then defines an entropy coder for use in a Coded CCA configuration. Simulations are used to compare the performance of the entropy coder with the performance bounds.

Alternate clustering forms of the CCA are defined in Chapter V. These include the Adaptive CCA and the Cascaded CCA. The performance of the Uncoded, Coded, Adaptive and Cascaded CCA are investigated in Chapter VI. Computer simulation results are used to compare the performance of various CCA options in terms of percent mean square error, subjective image appearance, and classification accuracy as a function of data rate. Conclusions and recommendations for further research are contained in Chapter VII.

CHAPTER II

CLUSTERING AS A FEATURE EXTRACTOR IN DATA COMPRESSION

A. Clustering for Spectral Intensity Feature Extraction

Now consider the selection of features for use in the Cluster Compression Algorithm (CCA). Let $\{\underline{X}^i\}_{i=1}^n$ be a sequence of n d-dimensional spectral intensity measurements for a spatially local region of a multispectral image. These vectors generally tend to form groups in multispectral intensity space. A typical plot of $\{\underline{X}^i\}_{i=1}^n$ for two bands in spectral intensity space is shown in Fig. 2.1(a). An intuitively natural set of features to use for representing $\{\underline{X}^i\}_{i=1}^n$ in Fig. 2.1(a) are the descriptions for groups of measurement vectors, for example, the groups depicted in Figs. 2.1(b) - 2.1(d). Each feature then consists of whatever set of parameters are used to describe the source distribution for the group of measurement vectors. The mean and covariance are an obvious example of a group description, or feature. The number of features extracted from any sequence of measurement vectors would depend on how accurately $\{\underline{X}^i\}_{i=1}^n$ must be represented, and correspondingly what data rate is acceptable. An example of how two, four, or eight features might be chosen is shown in Figs. 2.1(b) - 2.1(d).

Clustering is a means for automatically grouping multidimensional data [11], [15], [16]. The features resulting from clustering a set of measurement vectors are typically a subset of the following: cluster means, cluster variances per band, cluster covariances, number of vectors in each cluster, set of intercluster distances, etc. Anything

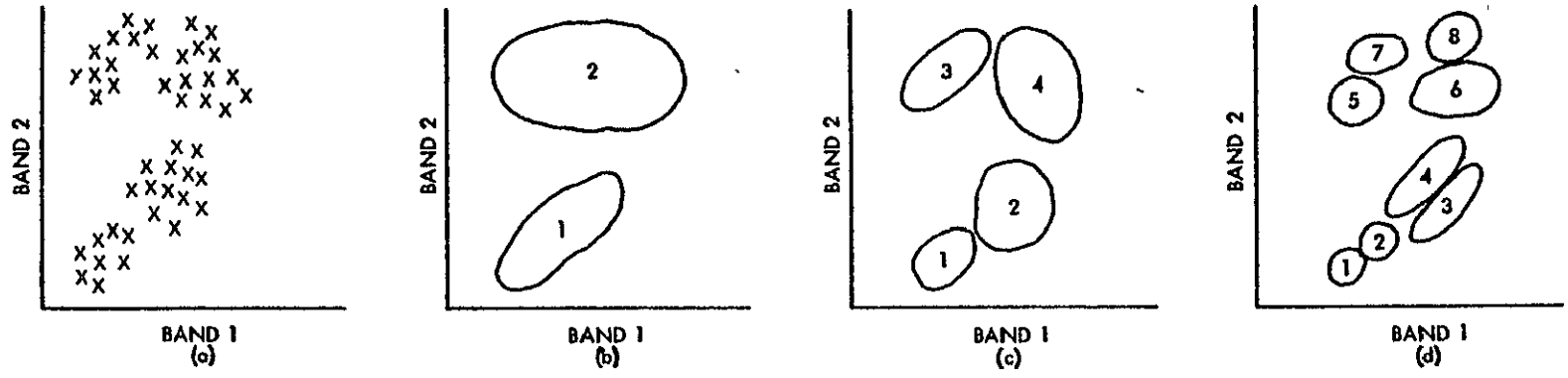


Fig. 2.1. Typical display in two band spectral intensity space for measurement vectors and for groups of measurement vectors used as features. (a) Measurement vectors $\{\underline{x}^i\}_{i=1}^n$. (b) Two groups. (c) Four groups. (d) Eight groups.

contributing to the description of the cluster can be considered a feature, even the set of vectors themselves. Clustering is used at the Laboratory for Applications of Remote Sensing (LARS) at Purdue University to aid ground analysis of Landsat MSS image data [12]. LARS mainly uses clustering to aid in selecting training sets from the image data for use in supervised classification, and also to verify the unimodal assumptions for the data. Clustering has also been used as a nonsupervised classifier of Landsat data in studies by Johnson Space Flight Center [17]. The self-scaling and adaptive characteristics of clustering together with the convenient supervision requirements suggest it might also be useful as an on-board feature extractor for data compression purposes.

First consider the efficient manner in which cluster features used by a data compressor can preserve classification accuracy. Let Fig. 2.2(a) represent sample conditional distributions for class A and class B which are very separable, and let Fig. 2.2(b) represent two classes which are less separable. The dashed line represents a good classification boundary for the sampled data sets. The user generally

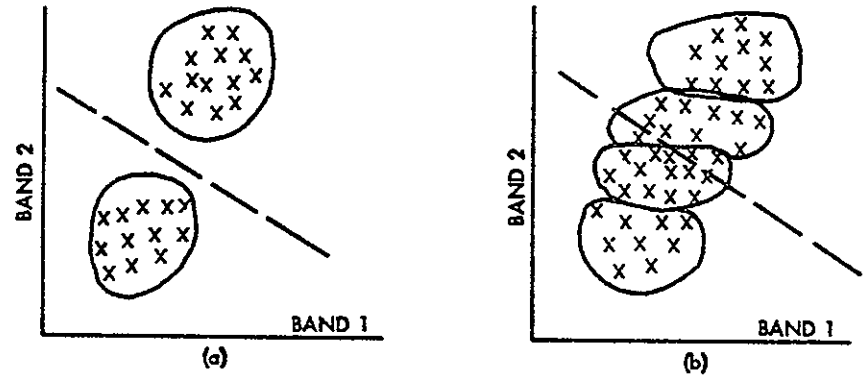


Fig. 2.2 Example of Two Classes in Two Dimensional Spectral Space. (a) Separable classes (b) Nonseparable classes requiring more clusters.

knows the separability of the classes, but the distributions and boundaries move around in spectral intensity space due to disturbance effects on the measurement process. Clustering will be relatively insensitive to this shifting since it inherently tracks the data. The user can supervise the clustering so as to conserve classification accuracy by simply specifying control parameters such as how many clusters, or maximum variance per band allowed for a cluster, etc. For example, assume the data in Fig. 2.2 is compressed by approximating each data vector by one of the cluster means. Then in Fig. 2.2(a) two clusters are adequate while in Fig. 2.2(b) four or more might be required for preserving classification accuracy. If other than Class A and B data is present, then a specification of maximum cluster variance per band could be used in determining the number of clusters to use, and the allowed cluster variance would be larger in Fig. 2.2(a) than in Fig. 2.2(b). There exists a relatively close relationship between cluster features and data features typically used in supervised classification.

Clustering can also be used by a data compressor to provide a very sophisticated adaptive multidimensional quantizer for obtaining image approximation. Image reconstruction can consist simply of using the cluster mean feature for every vector in that cluster. Again simple user inputs such as the number of clusters, variance per cluster, etc., are easily related to the quality of image approximation. Thus, cluster features appear to comprise an efficient and easily interpreted set of features for data compression in both classification and image approximation applications. However, we must yet consider the modifications

necessary to the usual clustering approaches in order to make more practical the use of clustering for data compression.

B. Clustering Approaches

1. Basic Clustering Algorithm

The CCA could use any of the many approaches for grouping data [11], [15], [16], [23] - [25]. The clustering algorithms investigated in this dissertation are all derived from the basic iterative approach to clustering referred to in this work as the Basic Clustering Algorithm (BCA) and shown in Fig. 2.3. This approach is chosen because it requires simple repetitive computations, and can be structured in a highly parallel manner for very high data rates.

Consider the assignment step shown in Fig. 2.3. Let $\underline{X}^i$, i=1, 2, . . . , n be the ith d-dimensional data vector which is to be assigned to one of m cluster centers, or means, $\underline{C}^j$, j=1, 2, . . . , m. The components of these vectors are defined by $\underline{X}^i = \left(x_1^i, x_2^i, \ldots, x_d^i\right)$, and $\underline{C}^j = \left(c_1^j, c_2^j, \ldots, c_d^j\right)$. The assignment of data vectors to clusters depends on the definition of distance between a data vector and a cluster center. In general, the assignment of vectors to clusters could involve a distance measure dependent on more than the cluster center [24], [25]. For example, the distance measure might involve the sample covariance of the cluster to account for the possibly nonsymmetric distribution of its members. Furthermore, the assignment of vectors to clusters could be probabilistic as in Fuzzy Set Theory [21], where the degree of assignment of a vector to each cluster would depend on the relative distance between the vector and each cluster. However,

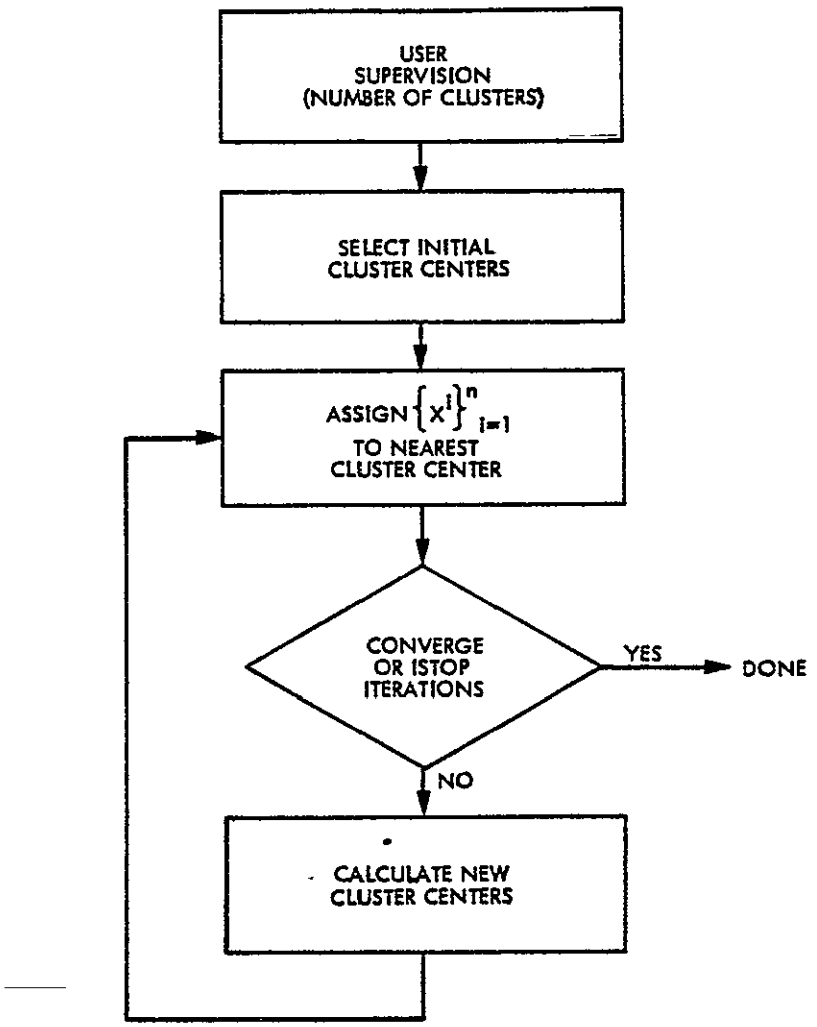


Figure 2.3. The Basic Clustering Algorithm.

since practical considerations are being emphasized only cluster means and Fuzzyless assignments are considered here. In addition, only the most often used and easier to calculate distance measures are investigated. The most common distance measure is the Euclidean distance. Let the Euclidean distance between data vector $\underline{x}^i$ and cluster center $\underline{c}^j$ be defined by

$$D_e(\underline{X}^i,\underline{C}^j) = \left(\sum_{k=1}^{d} \left(x_k^i - c_k^j \right)^2 \right)^{1/2}. \tag{2.1}$$

Another distance measure which requires even less calculation is the absolute value distance, which is defined by

$$D_a(\underline{X}^i,\underline{C}^j) = \sum_{k=1}^{d} \left| x_k^i - c_k^j \right|. \tag{2.2}$$

Thus assume the assignment task of the BCA consists of assigning each data vector to the closest cluster, where the distance is measured by (2.1) or (2.2).

After assigning every data vector to a cluster, there is a check to see if the iterating procedure should be terminated. In computer interpretation applications this check consists of terminating when the clustering has converged, or equivalently, when no data vectors have changed cluster assignment. For practical data compression the iterating should terminate if either the clustering converges, or if a chosen maximum number ISTOP of iterations are reached. The impact on clustering quality due to limiting the number of iterations is investigated in Chapter VI.

A change in cluster assignment for one or more data vectors usually results in a change in the cluster centers. Thus as shown in Fig. 2.3, the cluster centers need to be recalculated. Let C^j represent the set of all data vectors belonging to the jth cluster, and n^j the number of data vectors in C^j. Then the cluster centers are simply recalculated according to

$$c_k^j = \frac{1}{n^j} \sum_{\{i:\underline{X}^i \in C^j\}} x_k^i \tag{2.3}$$

where $j = 1, 2, \ldots, m$ and $k = 1, 2, \ldots, d$.

The only user supervision required for the BCA is the number of clusters desired (ISTOP is assumed to be specified once at time of implementation only). The number of clusters obtained is generally the same as the number of initial cluster centers upon entering the algorithm. However, in some cases a cluster may die during the iteration process due to no data vectors being assigned to it. The BCA is somewhat insensitive to the choice of intitial cluster centers. Basically, the initial cluster centers should be well scattered throughout the set of all data vectors. Let $\underline{C} = (c_1, c_2, \ldots, c_d)$ and $\underline{V} = (v_1, v_2, \ldots, v_d)$ be the mean and variance respectively of all the data vectors to be clustered. Then one method of choosing m initial cluster centers, $\underline{C}_i^j = (c_{i1}^j, c_{i2}^j, \ldots, c_{id}^j)$ is defined by

$$c_{ik}^j = c_k + (v_k)^{1/2}\left(\frac{2(j-1)}{(m-1)} - 1\right) \tag{2.4}$$

where $j = 1, 2, \ldots, m$ and $m > 1$. Equation (2.4) places the m initial cluster centers evenly spaced on the diagonal of positive correlation through the hyperrectangle enclosing plus and minus one standard deviation about the mean in each band.

The BCA of Fig. 2.3 heavily emphasizes simplicity for practical implementation. This is particularly true if an absolute value distance measure is used and if there is a limit placed on the number of iterations allowed. The BCA is only the core of more sophisticated

clustering algorithms used in data interpretation. However, it is shown in Chapter III that very simple clustering is useful for data compression purposes. Observe that although only the cluster centers were used in the above algorithm, other cluster features such as variance, covariance, etc., could be obtained if desired by calculating these features once upon terminating the BCA.

2. Adaptive Clustering Algorithm

The BCA generates a predetermined number of features per data set, which would correspond to a rate controlled mode for a data compressor. However, for some applications it is desirable to have the number of clusters used per data set be variable and adaptively determined to meet certain quality requirements. There are many ways to modify the BCA to make it adaptive. For example, clusters might be split, combined, or deleted during the iterative clustering based on intracluster and intercluster distance measures. A widely used algorithm, ISODATA, with these adaptive traits was originated by Ball and Hall [26]. Modifications of ISODATA for ground data evaluation of Landsat multispectral data produced ISOCLS [17], [18], and NSCLAS [27]. An Adaptive Clustering Algorithm (ACA) is defined in Fig. 2.4 for use in investigating adaptive clustering for data compression. The ACA results from simplifications of the ISOCLS approach. Observe that the ACA consists of the BCA as a core with the added capability of modifying the number of clusters between iterations. However, changes in the number of clusters is inhibited in the last two iterations, where only the BCA core is used,

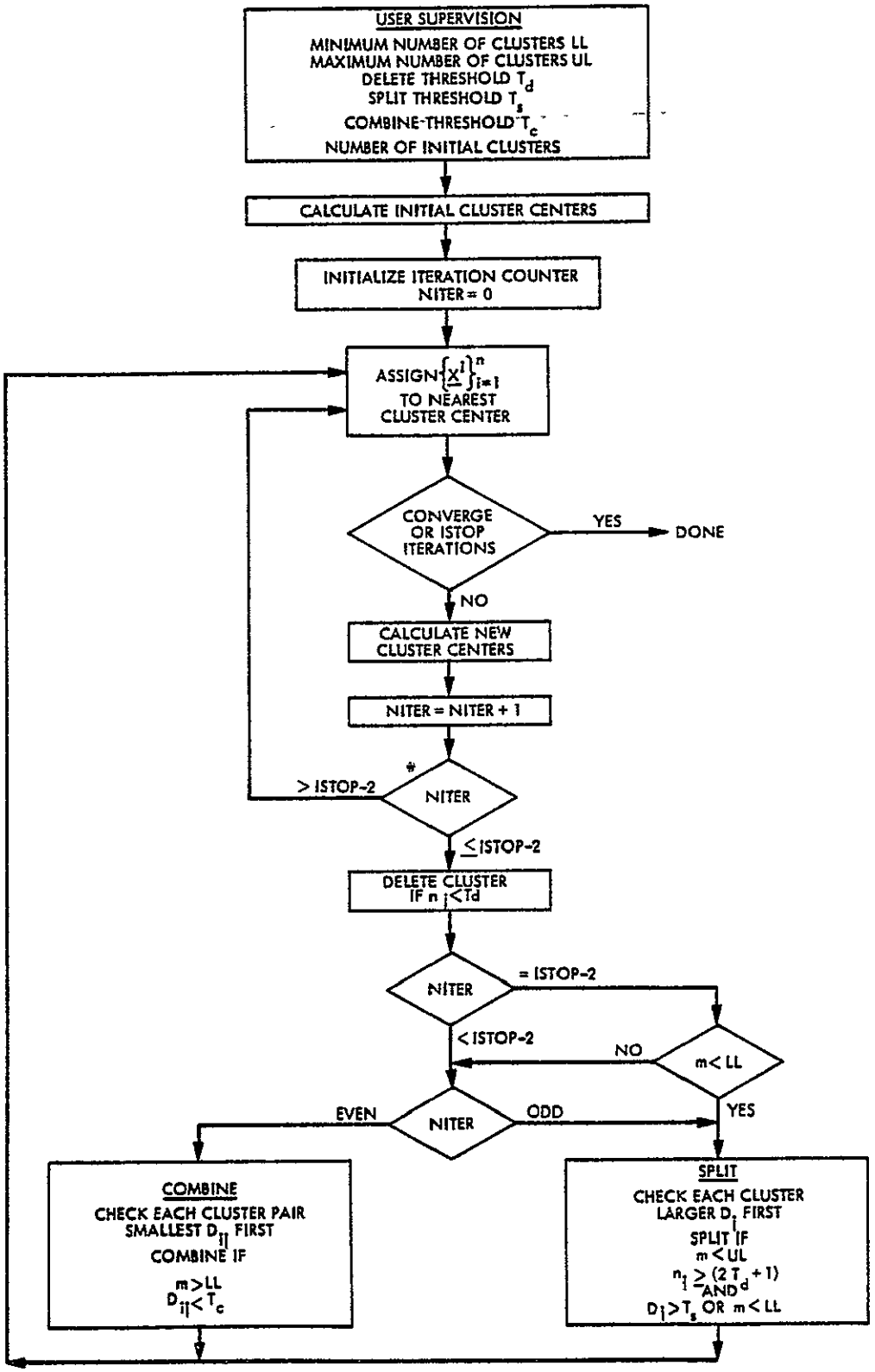


Fig. 2.4. The Adaptive Clustering Algorithm.

in order to tend toward clustering convergence before reaching the maximum number of iterations.

The user supervision of the ACA is still simple with only the additional inputs of limits and thresholds. The limits LL and UL specify the lower limit and upper limit respectively for the number of clusters. From a practical standpoint, these limits provide important control in trading-off the degree of adaptability vs. the system complexity. The thresholds for deleting, splitting, and combining are defined respectively by T_d, T_s, and T_c. Deleting consists of just eliminating a cluster center for which the number of vectors assigned to it are less than T_d. This corresponds to a minimum density requirement in order for a type of data to be important. Defining a split threshold corresponds to specifying an intracluster distance value above which the cluster is likely to consist of more than one class of interest. Each cluster can be split at most once per splitting iteration. A necessary condition for splitting the j-th cluster is that the intracluster distance D_j be greater than the splitting threshold T_s. However, if the number of elements in the j-th cluster is not much more than T_d, then splitting would likely result in two clusters which would both be deleted. Therefore, an additional requirement for splitting the j-th cluster is that the number of elements of the cluster be $\geq(2T_d + 1)$. The final additional condition necessary for splitting is that the present number of clusters be less than the maximum number allowed, UL. In summary, if the j-th cluster has not been split in this iteration, the necessary conditions for splitting the j-th cluster are

$$D_j > T_s$$

$$n_j \geq (2T_d + 1) \tag{2.5}$$

$$m < UL$$

where n_j is the number of elements in $\underline{C}^j$ and m is the present number of clusters. An exception to these conditions occurs if $m < LL$, in which case the splitting occurs even if $D_j \leq T_s$. The number of clusters can be reduced below LL by the deleting step.

A discussion of various intracluster distance measures can be found in [24] and [25]. A very simple distance measure based on the variance per band is used in the ACA. Let $\underline{\sigma}^j = \left(\sigma_1^j, \sigma_2^j, \ldots, \sigma_d^j\right)$ be the vector of standard deviations for each band of the j-th cluster. Then the intracluster distance for the j-th cluster is defined by

$$D_j = \underset{k}{\text{Max}}\ \sigma_k^j \tag{2.6}$$

where $k = 1, 2, \ldots, d$ and $j = 1, 2, \ldots, m$.

Let k' be the value of k which maximizes σ_k^j. Then if splitting conditions in (2.5) are satisfied, the j-th cluster center $\underline{C}^j$ is split into two new cluster centers, $\underline{C}^{j1}$ and $\underline{C}^{j2}$ defined by

$$\underline{C}^{j1} = \left(c_1^j, c_2^j, \ldots, c_{k'}^j - \sigma_{k'}^j, \ldots, c_d^j\right)$$

and (2.7)

$$\underline{C}^{j2} = \left(c_1^j, c_2^j, \ldots, c_{k'}^j + \sigma_{k'}^j, \ldots, c_d^j\right).$$

This simply corresponds to splitting $\underline{C}^j$ into two cluster centers placed a plus and minus standard deviation from the original mean and along the axis of greatest variance.

The combining of clusters in the ACA involves first finding the intercluster distance measure D_{ij} for all cluster pairs. Then cluster pairs are combined in the order of minimum distance first, provided the distance is less than T_c and there are presently more than LL clusters. However, clusters already combined on this algorithm pass are not combined further. Assume the i-th and j-th cluster are to be combined into an i'-th cluster. Then the i'-th cluster center is simply defined by a weighted average, or

$$\underline{C}^{i'} = \frac{n_i \underline{C}^i + n_j \underline{C}^j}{n_i + n_j} \quad . \tag{2.8}$$

Usual characteristics of D_{ij} are

$$\begin{aligned} D_{ij} &\geq 0 \\ D_{ii} &= 0 \\ D_{ij} &= D_{ji} \end{aligned} \tag{2.9}$$

for i and j any integer from 1 through m. Thus the number of unique intercluster distances, N(m), to calculate for m clusters is given by

$$N(m) = \frac{m(m-1)}{2} \quad . \tag{2.10}$$

The intercluster distance measure used in the ACA is defined by

$$D_{ij} = \sum_{k=1}^{d} \frac{\left(c_k^i - c_k^j\right)^2}{\sigma_k^i \, \sigma_k^j} \quad . \tag{2.11}$$

A more simple measure of distance between clusters would result from using the cluster means, for example $D_{ij} = D_e(\underline{C}^i, \underline{C}^j)$. The distance measure in (2.11) is better in that it weights the distance between cluster means in each band relative to the spread of the clusters in that band. Typically the intercluster distance measures used in data interpretation are more sophisticated than the one in (2.11). For example, the intercluster distance measure used in NSCLAS is the Swain-Fu distance, which uses all the covariance characteristics of both clusters in computing the distance [12]. Many of these general inter-cluster distances are discussed in [13]. These other measures could often give a more accurate measure of intercluster distance, but they would also require considerably more computation. The emphasis in this preliminary study of adaptive clustering for data compression use is on clustering approaches amenable to high data rate implementation. From this standpoint it is desirable to simplify rather than increase complexity relative to the measure in (2.11). Furthermore, initial investigations suggest that more sophisticated clustering results in only slight benefits for data compression purposes. The BCA and ACA defined in this section provide examples of practical clustering approaches which can be used for data compression. The next section discusses other considerations in making clustering more practical for data compression.

C. Clustering of Spatially Local Sources

Clustering for data interpretation generally involves a clustering of an image into twenty or fewer groups. However, if a Cluster

Compression Algorithm is to be useful for image approximation, it must be able to cluster an image (e.g., 256 × 256 picture elements) into perhaps thousands of clusters to adequately preserve image quality.

One cannot simply obtain thousands of clusters per image as might be needed in image approximation, by performing the clustering jointly for thousands of clusters. The large number of clusters would increase the computation time beyond practical limits. However, one can cluster the vectors in very small array sizes (e.g., 8 × 8). Then even though there is a small average number of clusters per area, say 4 per 8 × 8, there would still be a large number (4096) of clusters per 256 × 256. Other advantages of clustering on such a small scale are discussed below.

Define $P_g(\underline{X})$ as a global sample distribution function in spectral intensity space obtained by sampling measurement vectors over some spatially large source g, for example, 2340 × 3300 picture elements as in a Landsat image. $P_g(\underline{X})$ can be expressed as a mixture of local sample distribution functions $P_\ell(\underline{X})$ obtained from sampling measurement vectors within small spatially local sources, (e.g., 8 × 8) giving

$$P_g(\underline{X}) = \frac{N_\ell}{N_g} \sum_{\ell_i \in g} P_{\ell_i}(\underline{X}), \tag{2.12}$$

where N_ℓ and N_g are the number of picture elements in ℓ and g respectively. Sources g and ℓ are depicted in Fig. 2.5(a) and typical distributions for $P_g(\underline{X})$ and $P_\ell(\underline{X})$ are shown in Fig. 2.5(b). Observe that it is reasonable to expect a low average number of classes of interest per local source relative to the total number of classes within the global

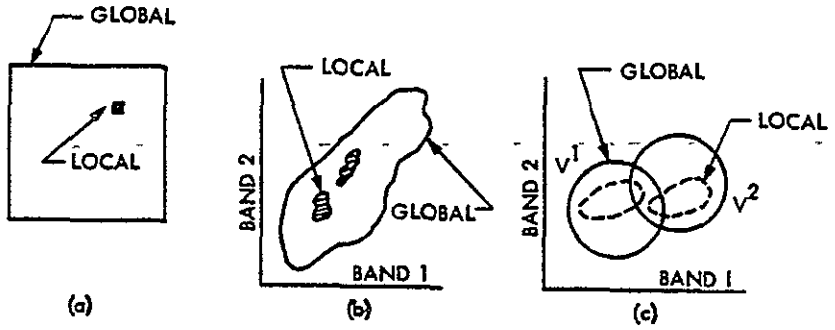


Fig. 2.5. Global vs. Local Sources. (a) Spatially, (b) Distributions, (c) Conditional Distributions.

source. Thus, clustering at the local source level is initially easier to implement because there are fewer classes and fewer vectors to cluster simultaneously.

Consider some local source ℓ^* with classes V^1 and V^2. Typical $P_g(\underline{X}/V^1)$, $P_g(\underline{X}/V^2)$, $P_{\ell^*}(\underline{X}/V^1)$ and $P_{\ell^*}(\underline{X}/V^2)$ are shown in Fig. 2.5(c). The larger spread in each global conditional distribution is due to averaging over many local conditional distributions, each with corresponding various perturbations due to class variance, moisture, atmosphere, etc. Therefore, at the local scale the vectors of a given class tend to form tighter clusters which are more separable.

The greater separability of classes within a local source would tend to permit the use of simpler clustering algorithms while still preserving separability. Thus clustering at a spatially local source level is not only important for practical implementation reasons, but it is also an advantage from the standpoint of clustering quality. Experimental results relating local source size and clustering quality are contained in Chapter VI.

CHAPTER III

UNCODED CLUSTER COMPRESSION ALGORITHM

A. Spectral Features and Spectral Data Rate

The most basic form of the Cluster Compression Algorithm is the Uncoded CCA as shown in Fig. 3.1. Let $\{\underline{X}^i\}_{i=1}^n$ be a set of n measurement vectors from one local source out of a sequence of spatially local sources. The local sources might consist of sequences of multispectral arrays obtained most conveniently from a single scan swath from the image sensor. Each local source is then clustered into m clusters to extract spectral intensity features, where m is assumed constant for an image and represents the only user supervision input. In this study of the Uncoded CCA the clustering used is the Basic Clustering Algorithm defined in Chapter II. The features which can be extracted from $\{\underline{X}^i\}_{i=1}^n$ by clustering would typically be a subset of the following: cluster means, cluster variances per band,

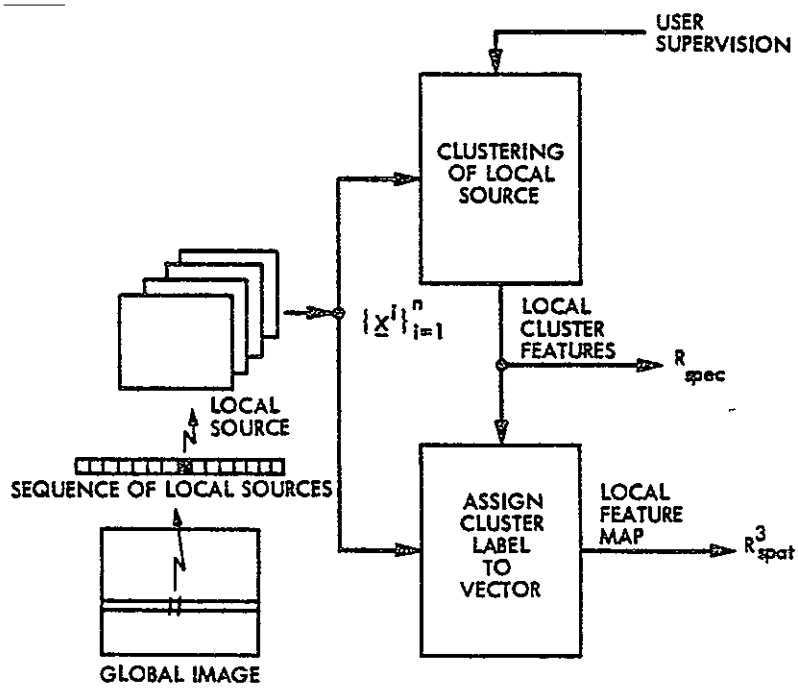


Fig. 3.1. A Block Diagram of the Uncoded CCA.

cluster covariances, number of vectors in each cluster, set of inter-cluster distances, etc. These types of cluster parameters comprise a compact grouping description of $\{\underline{X}^i\}_{i=1}^n$. Let $\{\psi^j\}_{j=1}^m$ be the set of m cluster features which describe the sequence of n d-dimensional measurement vectors. Assume each ψ^j requires f bits of quantization. For example, in image approximation ψ^{j*} might be the j*-th cluster mean with f equal to 6•d bits, or in classification uses ψ^{j*} might be the j*-th cluster mean and variance with f equal to 9•d bits if the variance per band is defined to 3 bit resolution. The spectral rate R_{spec} is defined as the bits per picture element per band (bpppb) needed to define the spectral characteristic of $\{\underline{X}^i\}_{i=1}^n$, or

$$R_{spec} = \frac{m \cdot f}{n \cdot d} \quad . \tag{3.1}$$

A graph of R_{spec} vs. m for various values of n is shown in Fig. 3.2 with f = 24 and d = 4, which corresponds to using only cluster means as features. In some cases this description of $\{\underline{X}^i\}_{i=1}^n$ may be all that is needed, and R_{spec} then represents the total rate in bpppb.

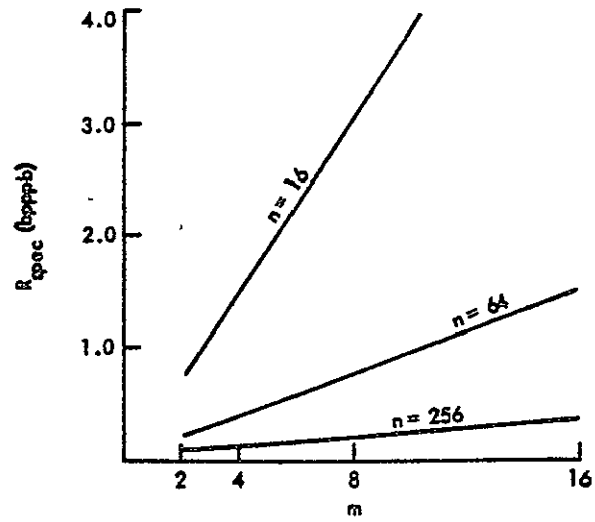


Fig. 3.2 Graph of Spectral Rate per Band vs. Number of Features for Various n and With f = 24 and d = 4.

B. Spatial Features and Spatial Data Rate

In many applications it is desired to further approximate $\{\underline{X}^i\}_{i=1}^n$ by approximating each vector individually in terms of the cluster features. For example, if vector $\underline{X}^{i*}$ belongs to cluster j*, then $\underline{X}^{i*}$ could be approximated by ψ^{j*} which might be simply the mean of cluster j*. Alternately, $\underline{X}^{i*}$ could be approximated by a combination of $\{\psi^j\}_{j=1}^m$. Refer again to Fig. 3.1. Assume that after clustering the local source, each cluster is uniquely labeled by an integer from 1 through m. Then map the sequence of data vectors $\{\underline{X}^i\}_{i=1}^n$ into a sequence of integers $\{\gamma^i\}_{i=1}^n$ by replacing each vector with the integer label of the cluster to which it belongs. This sequence of integers is called the local feature map and an example of a feature map is shown in Fig. 3.3.

The feature map sequence $\{\gamma^i\}_{i=1}^n$ defines the approximation of each measurement vector and thus gives spatial definition for the sequence $\{\underline{X}^i\}_{i=1}^n$. Let R^1_{spat} represent the rate in bpppb for representing each element of the m-ary feature map sequence with natural coding. Then

$$R^1_{spat} = \frac{1}{d} \left(\lceil \log_2 m \rceil\right), \tag{3.2}$$

where

$$\lceil x \rceil \equiv (\text{smallest integer} \geq x). \tag{3.3}$$

This representation of the feature map is wasteful if m in (3.2) is not a power of two (e.g., the use of 3 bits to specify one of 5 integers for m=5). The representation can be generally improved by using the simple fixed rate natural coding on the ℓ-th extension (ℓ-tuples) of the source. The spatial rate for natural coding of the ℓ-th source extension is given by

$$R^{\ell}_{spat} = \frac{1}{d\ell} \left(\lceil \log_2 m^{\ell} \rceil\right) . \tag{3.4}$$

From (3.3) we obtain

$$\log_2 m^{\ell} \leq \lceil \log_2 m^{\ell} \rceil < \log_2 m^{\ell} + 1 , \tag{3.5}$$

and from (3.4) and (3.5) there results

$$\frac{1}{d} \log_2 m \leq R^{\ell}_{spat} < \frac{1}{d} \log_2 m + \frac{1}{d\ell} . \tag{3.6}$$

Therefore, from (3.6) we have

$$R^{\ell}_{spat} \geq \frac{1}{d} \log_2 m \tag{3.7}$$

for all ℓ, and

$$\lim_{\ell \to \infty} R^{\ell}_{spat} = \frac{1}{d} \log_2 m . \tag{3.8}$$

Now define R_{spat} as

$$R_{spat} = \frac{1}{d} \log_2 m . \tag{3.9}$$

From (3.7) it is observed that the best we can do with simple natural coding is R_{spat}, and from (3.6) and (3.8) it is observed that for any m, R_{spat} can be approached by coding higher source extensions. Of course, even natural coding of high source extensions is impractical, but consider now the use of the 3rd source extension which is easy to implement for the small values of m of interest. Let R^3_{spat} be the rate achieved when groups of three of the original m alphabet source are represented by natural coding. Figure 3.4 shows R_{spat} and R^3_{spat} vs. m for two values of d and shows that R^3_{spat} is never significantly greater than R_{spat} for $m \leq 16$. Thus the fixed spatial data rate for the Uncoded CCA is R^3_{spat}, where

$$R^3_{spat} = \frac{1}{3d} \left(\lceil \log_2 m^3 \rceil\right) . \qquad (3.10)$$

Further reductions in the spatial data rate can be obtained through variable-length encoding techniques which will be considered in Chapter IV.

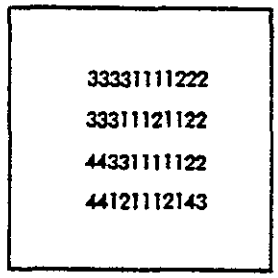


Fig. 3.3 Figure of a Typical Feature map.

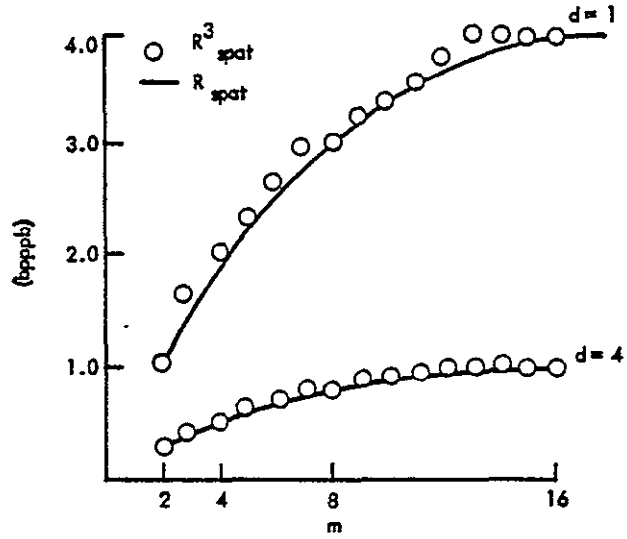


Fig. 3.4. Graph of R_{spat} and R^3_{spat} vs. m for d=1 and d=4.

C. Uncoded CCA Examples and Total Data Rates

The Uncoded CCA of Fig. 3.1 compresses all local sources of an image to a constant data rate, dependent on the user supervision inputs of m, n, f, and d for the image. The output is in two parts: 1) the cluster definitions defining the features extracted in spectral intensity space, and 2) an array of scalar numbers giving spatial definition to the occurrence of features in spectral intensity space. The total output data rate, R_{tot}, consists in general of both spectral and spatial definitions defined in (3.1) and (3.10). Thus

$$R_{tot} = R_{spec} + R^3_{spat} = \frac{m \cdot f}{n \cdot d} + \frac{1}{3 \cdot d} \quad (\lceil \log_2 m^3 \rceil). \tag{3.11}$$

In some situations it is only necessary to know the group characteristics (e.g., what is there, and how much of each) for each local source, with no need to define each individual picture element. Then the feature map is not sent and $R^3_{spat} = 0$ and $R_{tot} = R_{spec}$. Similarly, there might be situations in which only spatial characteristics in the feature map are of interest and R_{spec} could be zero. There is a great degree of flexibility in choosing the relative division of R_{tot} between R_{spec} and R^3_{spat}.

1. Example for Image Approximation Use

Consider an example of applying the Uncoded CCA to obtain compression in image approximation. Assume that 8 clusters are obtained for each local source of 16 × 16 picture elements of 4 bands each. Also assume that for each cluster the user is sent 24 bits for the cluster mean. This corresponds to m=8, f=24, n=256, and from (3.1), R_{spec} = .1875 bpppb. The .1875 bpppb spectral definition defines the centroids of 8 groups of data vectors within each local source of 16 × 16 picture elements. Now assume the user is also sent the feature map sequence consisting of 16 × 16 three bit codewords defining a spatial definition of the local source in terms of the 8 cluster mean features. From (3.10), R^3_{spat} = .75 bpppb. The total data rate for sending the spectral and spatial definition (compressed image data) is R_{tot} = .9375 bpppb.

As shown in Fig. 3.5(a) the user reconstructs the approximate image in an extremely simple manner as follows: 1) for the given local source

the 8 cluster centroids are placed in a small look up table; 2) the feature map codewords are used as addresses to access the look up table and provide the proper cluster mean for approximating each picture element in the local source. Thus the compressed image data is very easily decoded into an image by the user. Furthermore, the Uncoded CCA should provide excellent image compression performance due to the sophisticated multidimensional quantization of spectral space provided by clustering. Reconstructed image examples resulting from simulations of the Uncoded CCA are presented in Chapter VI.

2. Example for Supervised Classification Use

Now consider an example where the user desires a supervised classification result for each picture element in the image. Again assume that 8 clusters are obtained for each local source of 16 × 16 picture elements in 4 bands, and that the same m, f, and n parameters of the previous example are used. Thus as before $R_{spec} = .1875$, $R_{spat}^{3} = .75$, and $R_{tot} = .9375$. In fact, the exact same compressed data is sent by the compressor in both examples. However, this example differs in that the compressed data is to be interpreted into classified picture elements rather than an image approximation. As shown in Fig. 3.5(b) the user obtains the classified picture elements as follows: 1) for a given local source, the 8 cluster means are placed in a small look-up table; 2) each of the 8 cluster means are classified according to the users' desired classification method, and the resulting user class label is also stored in a small look-up table; 3) the feature map codewords are used as addresses to access the look-up table and provide for each picture element the user class label of the cluster to which the picture element belongs.

Thus the compressed data is also very easily decoded for classification use. In fact for each 16 × 16 local source only 8 representative vectors needed to be classified instead of 256, thereby reducing by a factor of 32 the user classification computation. Observe also that the user could elect to use feature maps only for those local sources in which a class of interest occurs. The Uncoded CCA should perform well in terms of rate vs. classification accuracy since the clustering tends to preserve separability of the data. Basically, for whatever number of user classes are desired in the entire image, if m and n are such that more than m of these user classes seldom occur within a single n picture element subset, then classification performance from the compressed data is expected to be good. Examples of classification accuracy vs. total data rate resulting from simulation studies are presented in Chapter VI.

3. Example for Feature Extraction Use

Next consider an example where it is not necessary to define each individual picture element and where R_{spat} can be zero. For example, in agricultural classification it may be adequate to know what crops are present and in what percentages in each local source. Again assume 8 clusters are used per 16 × 16 picture elements of 4 bands each. As in the previous examples, for each cluster there is sent to the user 24 bits for the cluster mean, but in addition 8 bits are sent to identify the number of vectors in the cluster. This corresponds to m=8, f=32, n=256, and R_{spec} = .25 bpppb. The per picture element definition is not needed, so R_{spat} = 0. Thus for a total rate of .25 bpppb

the user receives the identification of centroids and the number of associated data vectors for 8 groups of vectors in spectral space for each local source of 16 × 16 picture elements in the image. Figure 3.5(c) shows how the user can then convert this compressed data directly to crop information by classifying the received cluster means and calculating percentages from the received data identifying the number of data vectors assigned to each cluster. This example demonstrates the potential for the CCA to obtain very high compression ratios and simple decoding for interpretation, through the extraction of group features.

4. Example for a Combination of Uses

The previous three examples demonstrated the relative simplicity in the Uncoded CCA for user distribution, access, and interpretation for three different applications. Now consider an example in which all three of the above data uses are simultaneously desired from the same image data, but by three different users. As in the immediately preceding example an R_{spec} of .25 bpppb is used to define 8 cluster means and the associated number of vectors in each for every 16 × 16 local source. Also an R^3_{spat} of .75 bpppb is used to again define each picture element in terms of the 8 cluster means. Thus the total compressed data rate is R_{tot} = 1.0 bpppb. The operations of the Uncoded CCA for this example are depicted in Fig. 3.6. Efficient distribution of the compressed data is accomplished by 1) sending only the spectral data at .25 bpppb to the user desiring crop percentage information, and 2) sending the total data at 1.0 bpppb to the other two users. Each user

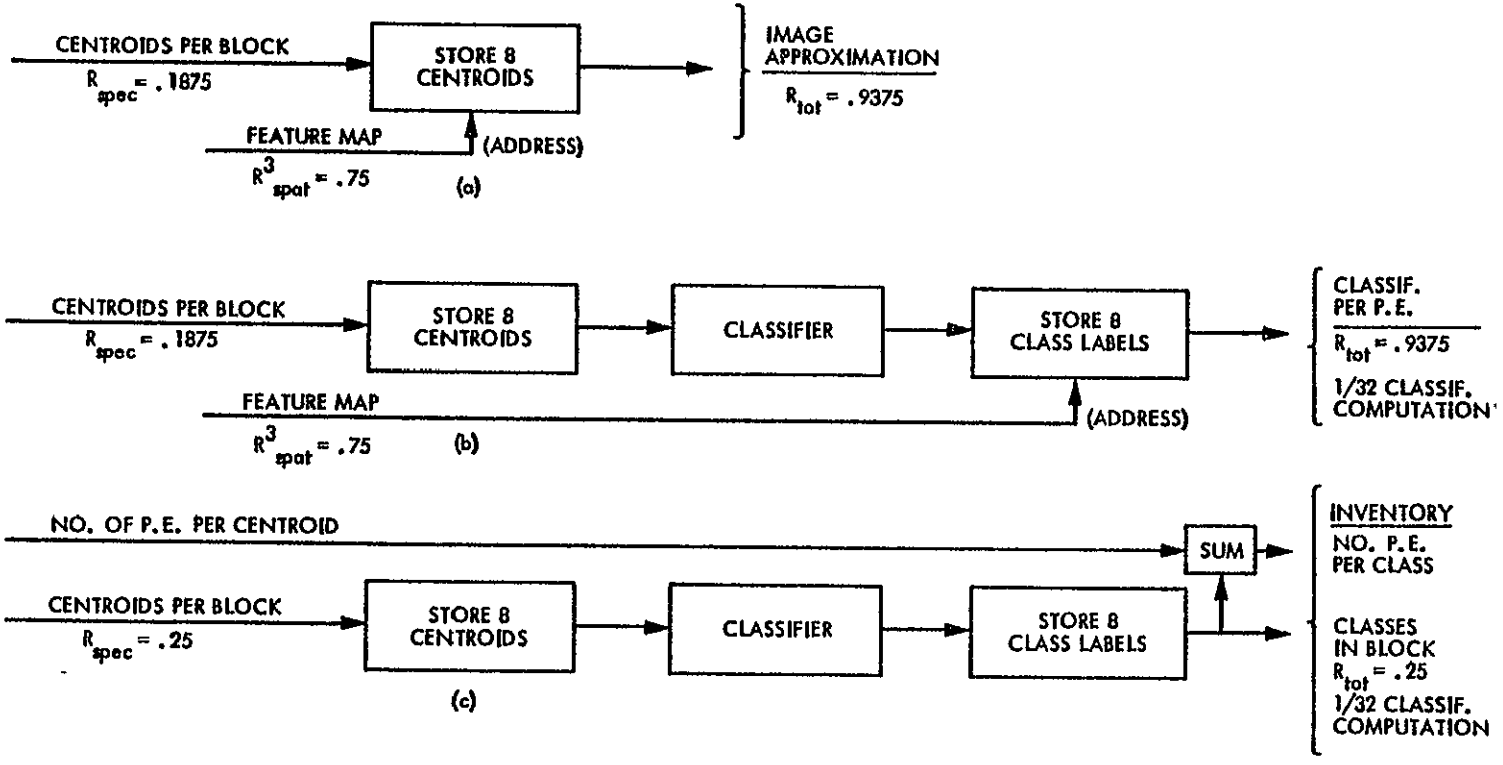


Fig. 3.5. Structure for Decoding and Interpreting Uncoded CCA Data (a) Image Approximation Use, (b) Supervised Classification Use, (c) Feature Extraction Use.

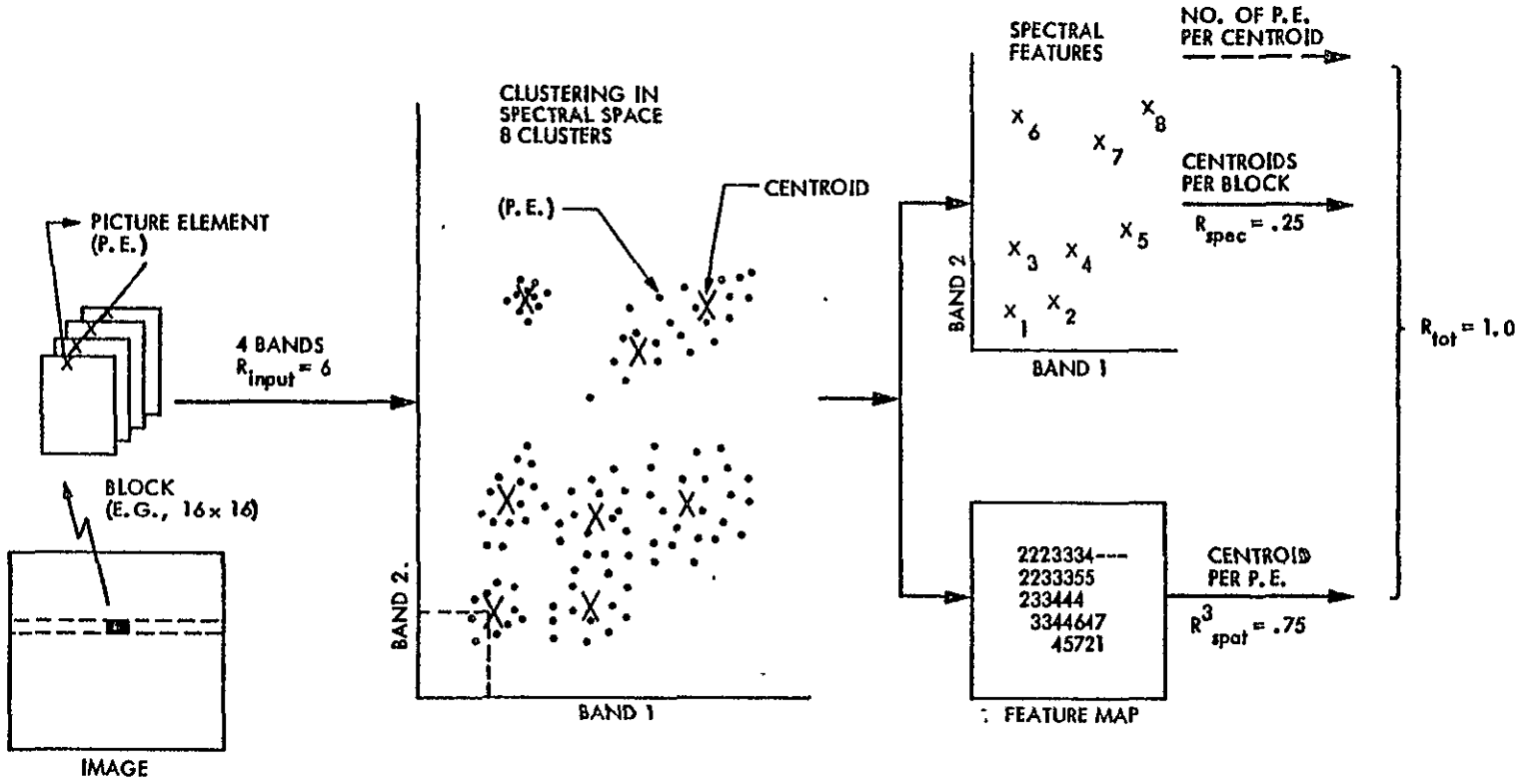


Fig. 3.6. Example of Uncoded CCA Operation.

then accesses and interprets his data in the same direct fashion as in the corresponding previous examples.

Figure 3.5 shows the structure for decoding and interpreting the compressed data for any of the above example applications. The basic look-up table operations could of course be done at very high data rates by a minicomputer. These examples demonstrated the following important attributes of the CCA approach: 1) the same data compressor is used for image approximation and classification applications, 2) group features and/or per picture element definition can be selectively accessed for distribution and interpretation directly from the compressed data without first reconstructing an approximation of the original image, 3) the compressed data decoding operations required for classification and image approximation are very simple and can be done at high data rates with readily available equipment, 4) the computation required for user classification is substantially reduced due to the CCA compression (e.g., 1/32 as many classifications). Actual simulation results of image approximation and classification performance are presented in Chapter VI.

CHAPTER IV

CODED CLUSTER COMPRESSION ALGORITHM

A. Feature Map Source Models and Performance Bounds

A typical feature map sequence for $\{\underline{X}^i\}_{i=1}^n$ described by four cluster features is shown in Fig. 3.3. In image data there is significant spectral correlation between spatially close picture elements, and correspondingly between adjacent elements in the feature map. Entropy encoding techniques can use this spatial correlation to represent feature map sequences with a lower average spatial data rate than R_{spat} given in (3.9). Furthermore, this reduction in spatial data rate is obtained without any degradation to the feature map sequence.

The performance of an entropy encoder is usually compared to the best performance possible for encoding a data source which statistically models the actual sequences to be encoded. Basically, a source model is defined such that the observed feature map sequences are typical sample sequences from the source model. Then the entropy of the source model is used to bound the lowest possible average data rate for encoding sequences from the source model [9], [29]. An entropy encoder attempts to achieve performance close to the entropy of the source model for the feature map sequences. Two key elements to entropy encoding are 1) the determination of a model for the source which has low entropy, and 2) the determination of a practical encoder which can encode sequences from the source model at an average rate near the entropy. The source model will have lower entropy if the model incorporates more memory and nonstationary statistics. However, the encoder also tends to become impractical when it attempts to make

use of source memory, or adapt to nonstationary statistics. In this section some source models and associated entropies are defined for the feature map sequences. Then a practical entropy encoding technique is defined, and simulation results used to demonstrate the actual coding performance relative to the entropy bounds for several source models.

1. Bound for Nonadaptive, Zero-Memory Encoder of Feature Map Differences

The first source model is similar to the very common and simple approach of representing the image data in terms of differences between adjacent picture elements. Of course, the feature map can be represented in terms of differences with no loss in information. The differences between spatially adjacent elements of a feature map are typically distributed as shown in Fig. 4.1. For image data, there is little correlation between the differences of adjacent elements, thus these differences are often modeled as samples from a stationary, zero-memory source. We now similarly define the analogous source model

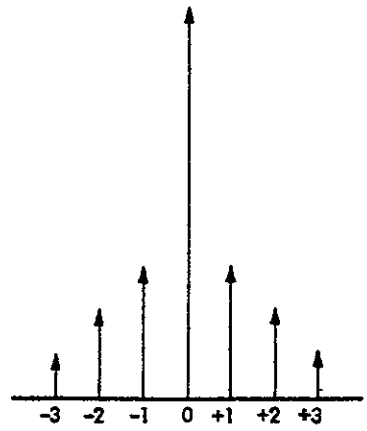


Fig. 4.1. Typical Distribution of Differences Within a Feature Map.

and entropy for the differences between adjacent elements of a feature map.

A difference in a feature map is an integer δ, where $\delta = -(m-1)$ through $(m-1)$. Let P_δ represent the relative frequency of occurrence of the integer δ over all feature maps of the image. Define S as a source which produces integer outputs δ with probability P_δ. Then S is a stationary, zero-memory source model for producing sample sequences of feature map differences. The entropy in bits for the source S is defined by

$$H(S) = \sum_{\delta=-(m-1)}^{(m-1)} - P_\delta \log_2 P_\delta \quad . \tag{4.1}$$

H(S) represents a performance bound (lowest possible rate) for encoding differences from the feature maps with any nonadaptive, zero-memory encoder.

---

2. Bound for Adaptive, Zero-Memory Encoder of Feature Map Differences

Image data and the corresponding feature map data are typically nonstationary. For example, the set of frequency of occurrences $\{P_\delta^{\ell}\}_{\delta=-(m-1)}^{(m-1)}$ obtained from differences in the ℓ-th feature map of an image may be quite different than the frequency of occurrences $\{P_\delta\}_{\delta=-(m-1)}^{(m-1)}$ of differences obtained from averaging over all feature maps in the image. A source model for the feature maps will have less entropy if it also models the nonstationarity in the source. Similarly, for nonstationary sources a better performance can be obtained

from an entropy encoder which adapts to the statistical characteristics on a local basis. Therefore, we next define a nonstationary source model for the feature map sequences. The entropy of this nonstationary model serves as a useful bound on the performance of an adaptive entropy encoder.

Define S^{ℓ} as a stationary zero-memory source model for the sample sequence of differences from the ℓ-th feature map in the image, where $\ell = 1,2,\cdots,N_L$ and N_L is the number of feature maps in the image. For each ℓ, S^{ℓ} is defined to ouptut an integer δ with probability P_{δ}^{ℓ}, where P_{δ}^{ℓ} is the frequency of occurrence of the difference between adjacent feature map elements being δ within the ℓ-th feature map, and δ is an integer from $-(m-1)$ through $(m-1)$. Then let S_L be the nonstationary zero-memory source which is constructed from the many different stationary sources. For each local source in the image, the output of S_L is a sample sequence from a different stationary source. The entropy in bits for each S^{ℓ} is given by

$$H(S^{\ell}) = \sum_{\delta=-(m-1)}^{(m-1)} - P_{\delta}^{\ell} \log_2 P_{\delta}^{\ell} \quad . \tag{4.2}$$

Averaging the entropies in (4.2) over all local sources in the image results in an average local entropy, or the entropy for S_L,

$$H(S_L) = \frac{1}{N_L} \sum_{\ell=1}^{N_L} H(S^{\ell}) \quad . \tag{4.3}$$

$H(S_L)$ is a performance bound on the average rate obtained by a zero-memory encoder which adaptively encodes each feature map sequence of differences. Actually, $H(S_L)$ represents an unachievable performance bound for practical encoding, since we ignored the overhead rate required to inform the decoder of the new statistics relative to each local sequence. It can be shown that $H(S_L) \leq H(S)$ [9, pp. 27-28].

3. Bound for Nonadaptive and Adaptive Zero-Memory Encoder of Feature Map Distance Rank Symbols

Recall that the feature map resulted from replacing each picture element with the integer label of the cluster to which it belonged. In the above modeling of the feature map differences and the corresponding entropy $H(S^{\ell})$ in (4.2), it was implicitly assumed that the integer labels assigned to the feature map clusters resulted from the establishment of labeled initial mode centers in the clustering algorithm defined by (2.4). However, the clusters corresponding to any feature map can certainly be assigned any set of distinct labels, and the entropy of differences in the feature map is not necessarily minimized by using the cluster labels from the initial mode center definition. Intuitively, the entropy of differences is reduced when the set of frequency of occurrences for differences become less uniform in value, for example, the small magnitude differences occur more frequently and the large magnitude differences less frequently. In monochromic image data the differences consistently tend to lower magnitude, based on the underlying observation that the next image element tends more probably to be similar to the preceding. Similarly, for multispectral images, an image element tends more probably to be similar (in a

multidimensional sense) to the preceding. However, for multispectral data, it is often impossible to define a set of cluster labels such that the magnitude of the difference between each pair of cluster labels is monotonically related to the multidimensional intercluster distance (e.g., Euclidean distance between cluster means) between each pair of clusters. For example, consider the four clusters in two bands depicted in Fig. 4.2. The difference symbol resulting for each pair of adjacent elements of the feature map is shown in Table I for the arbitrarily chosen cluster labels. Table II is constructed by assigning an ordering to the multidimensional intercluster distances between each pair of clusters. For cluster 2, for example, the clusters when ordered as closest first are 2, 3, 4, 1, and the transitions from cluster 2 to these clusters are labeled in the table respectively as 1, 2, 3, 4. Observe that the magnitude of the differences in Table I are not consistently monotonically related to the distance ordering in Table II. Furthermore, no assignment of cluster labels can achieve such a relationship. Observe also that the difference representation of the source required a source alphabet of seven

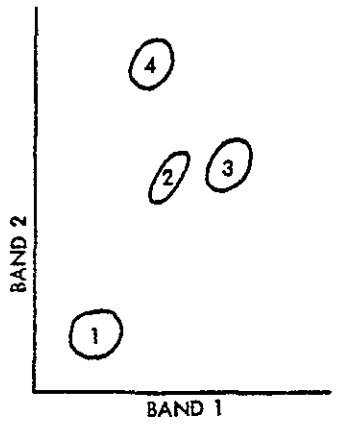


Fig. 4.2. Four Integer Labeled Clusters.

TABLE I
DIFFERENCE OUTPUTS FOR PAIRS OF SOURCE SYMBOLS

	(i+1)-th Source Symbol			
i-th Source Symbol	1	2	3	4
1	0	1	2	3
2	-1	0	1	2
3	-2	-1	0	1
4	-3	-2	-1	0

TABLE II
DISTANCE RANK OUTPUTS FOR PAIRS OF SOURCE SYMBOLS

	(i+1)-th Source Symbol			
i-th Source Symbol	1	2	3	4
1	1	2	3	4
2	4	1	2	3
3	4	2	1	3
4	4	2	3	1

symbols as opposed to four symbols used in Table II. Therefore, the representation of the source in terms of differences becomes less appropriate for multispectral imaging. We next define a new representation of the feature map which incorporates the multidimensional nature of the source. This representation will be generally lower in entropy and independent of the cluster labeling. The difference representation of the source is still worthy of note because of its greater simplicity, and also because its entropy may often be not significantly higher.

Define SS^{ℓ} as a new source model for the ℓ-th feature map. Assume the ℓ-th feature map has m clusters with any arbitrary but distinct set of cluster labels. Then the output symbol from SS^{ℓ} for each feature map element is an integer r, $r = 1, \cdots, m$ with r determined as follows: if the cluster corresponding to the present feature map element is the i-th closest of all clusters to the cluster corresponding

to the previous feature map element, then r = i. Closeness is measured here in terms of the simple intercluster distance, the Euclidean distance between cluster centers. Any predetermined ordering can be used for equally close clusters. Refer again to the example of four clusters in Fig. 4.2. Table II defines the mapping between feature map transitions and output symbols for SS^{ℓ}. For example, when the previous feature map element is a 2, the output symbol from SS^{ℓ} is either 4, 1, 2, or 3 dependent on whether the next feature map element is a 1, 2, 3, or 4 respectively. Note that a symbol of SS^{ℓ} is consistently smaller in value when the adjacent feature map elements are more similar. This generally results in a more nonuniform distribution of symbol values and a lower entropy. The distance rank representation SS^{ℓ} is more generally applicable than the difference representation S^{ℓ} in that SS^{ℓ} possesses the above desirable traits for any cluster labeling and any dimensionality of the cluster space. The conversion between feature map elements and SS^{ℓ} symbols is specified by an $m \times m$ distance rank matrix. Therefore, this source representation requires the additional computation of the distance rank matrix at the source and at the destination for each feature map. However, no overhead rate is required to inform the destination of the distance rank matrix, since this matrix can already be calculated at the destination from the available spectral data which defined the clusters of the feature map.

Let SS^{ℓ} be a source model for the sequence of distance rank symbols from the ℓ-th feature map. Every symbol in a sequence from SS^{ℓ} is defined to have probability P_r^{ℓ} of being r, where P_r^{ℓ} is the frequency of

occurrence of the value r in the distance rank representation of the ℓ-th feature map. Then the entropy for SS^{ℓ} is

$$H(SS^{\ell}) = \sum_{r=1}^{m} - p_r^{\ell} \log_2 P_r^{\ell} , \tag{4.4}$$

and the average local entropy for the nonstationary model of feature map distance rank symbols SS_L is

$$H(SS_L) = \frac{1}{N_L} \sum_{\ell=1}^{N_L} H(SS^{\ell}) \quad . \tag{4.5}$$

Now let P_r be the frequency of occurrence for the distance rank symbol r averaged over all feature maps in the image. The stationary model of feature map distance rank symbols is defined as SS and it produces sequences for which the probability of each symbol being r is P_r. The entropy of SS is

$$H(SS) = \sum_{r=1}^{m} - P_r \log_2 P_r \, . \tag{4.6}$$

H(SS) and $H(SS_L)$ represent performance bounds for a nonadaptive and adaptive encoder respectively which encodes sequences of distance rank symbols from the stationary and nonstationary models of the feature maps.

4. Bound for Adaptive, First-Order Memory Encoder of Feature Map Symbols

Before comparing entropies of the above practical source models, we consider a nonstationary first order Markov source model SSS_L for the feature maps. This source model is of less practical value than the preceding source models for the following reasons: use of the additional memory in the model would complicate the encoder, and the overhead rate and complexity associated with informing the decoder of the new statistics for each feature map becomes substantial. Therefore, SSS_L is used here only to provide a performance bound which specifies the limits to the performance improvements which could be obtained from the next level of modeling complexity.

In a first order Markov source the probability of emitting a given source symbol depends only on the previous symbol. If there are m source symbols then there are m states for the source. A state diagram is used to depict a first order Markov source for three symbols in Fig. 4.3. In general, a first order Markov source is completely specified by giving the source alphabet $\{a^j\}_{j=1}^m$ and the conditional probabilities $p(a^j/a^k)$ of getting symbol a^j after a^k for $j = 1, 2, \ldots, m$

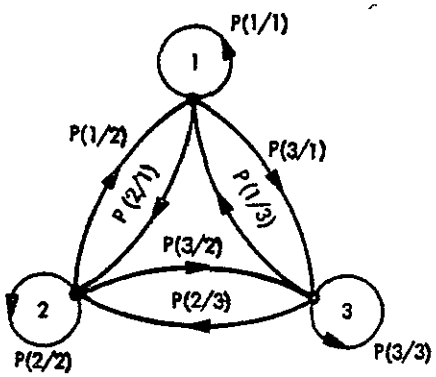


Fig. 4.3. State Diagram for a First Order Markov Source.

and k = 1, 2, . . . , m. The state probabilities $p(a^j)$ for j = 1, 2, . . . , m can be calculated from the conditional source probabilities [30, pp. 338-376]. We can model the feature maps as sequences from a nonstationary first order Markov source as follows. Define the source alphabet as the integers which are used to label the clusters,

$$a^j = j \qquad \text{for } j = 1, 2, \ldots, m . \tag{4.7}$$

Also define the conditional probabilities of the ℓ-th feature map by

$$p^{\ell}(a^j/a^k) = p^{\ell}(j/k) \tag{4.8}$$

where $p^{\ell}(j/k)$ is the frequency of occurrence of an integer j following an integer k in the ℓ-th feature map. Let SSS^{ℓ} denote the stationary first order Markov source model for the ℓ-th feature map sequence as defined by (4.7) and (4.8). Then the entropy of SSS^{ℓ} is defined as

$$H(SSS^{\ell}) = \sum_{k=1}^{m} p^{\ell}(k) H(SSS^{\ell}/k) , \tag{4.9}$$

where $H(SSS^{\ell}/k)$ is the conditional entropy of SSS^{ℓ} while in the k-th state,

$$H(SSS^{\ell}/k) = \sum_{j=1}^{m} - p^{\ell}(j/k) \log p^{\ell}(j/k) \tag{4.10}$$

for k = 1, 2, . . . , m. Averaging the entropies in (4.9) over all feature maps in the image gives the average entropy for the nonstationary source model SSS_L,

$$H(SSS_L) = \frac{1}{N_L} \sum_{\ell=1}^{N_L} H(SSS^{\ell}) \quad . \tag{4.11}$$

$H(SSS_L)$ represents a performance bound for an entropy encoder which adapts to each sample sequence from the nonstationary first order Markov source model of feature maps. Actually, $H(SSS_L)$ represents an unachievable performance bound, since $H(SSS_L)$ does not include the rate required to inform the decoder of the relevant source statistics for each local source, which is required for decoding. However, this bound is useful since it does place a limit on possible performance improvement from using the full first order memory of the source.

B. Simulation Results Comparing the Performance Bounds

In Chapter III the Uncoded Cluster Compression Algorithm was shown to have a spatial data rate, (bpppb) of R^3_{spat} as defined in (3.10). It was also shown that R^3_{spat} was approximately the same as $R_{spat} = (1/d) \log_2 m$, where m is the number of clusters in the d dimensional measurement space. The lower spatial data rate due to entropy coding is bounded by the entropy per band,

$$H_d(\mathscr{S}) = \frac{H(\mathscr{S})}{d} \quad , \tag{4.12}$$

where $\mathscr{S}$ is one of the source models, S, S_L, SS_L, SS, SSS_L, and where $H(\mathscr{S})$ is defined by (4.1), (4.3), (4.5), (4.6) and (4.11) respectively. The entropy per band depends directly on m, the number of clusters per local source, and also upon the test image used which determines the frequency of occurrences for symbols of the source model. Entropy per band also depends indirectly upon the size of each feature map source n, which impacts the nonstationarity of the source, and the set of frequency of occurrences. The performance bounds for each of the source models is plotted in Fig. 4.4 for various values of m and n. The bounds were obtained in each case from using the appropriate feature map source models for all the spatially disjoint square subsets in the test image with n picture elements. For example, $H(S_L)$ for m = 8 and n = 256 is determined as follows: Each 16 × 16 subset of picture elements of the

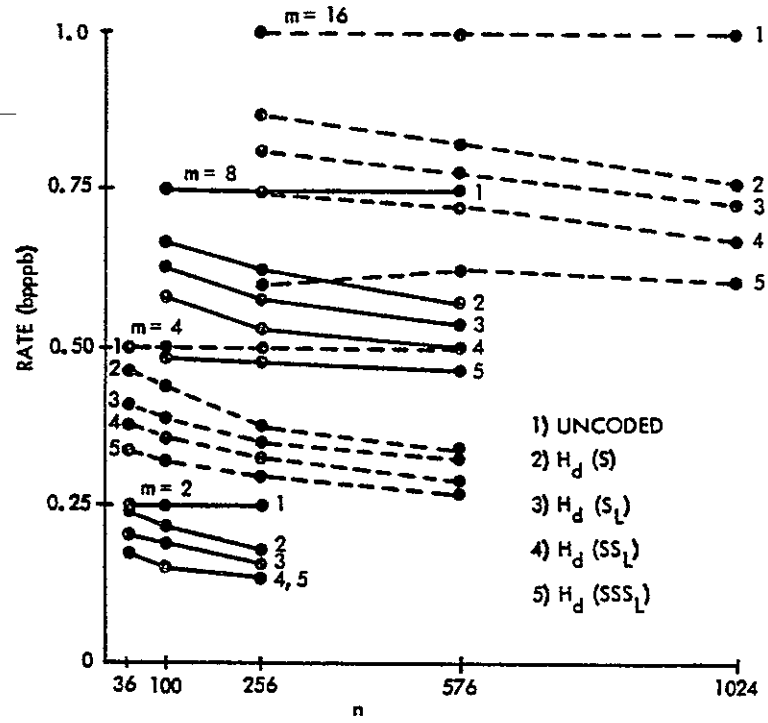


Fig. 4.4 Performance Bounds from the Various Source Model Entropies as a Function of m and n.

test image is clustered to 8 clusters, then the frequency of occurrences for differences within each 16 × 16 feature map is calculated, and the corresponding $H(S^{\ell})$ and $H(S_{\bar{L}})$ result from (4.2) and (4.3). The test image is shown in Fig. 6.8(a) and is a 256 × 256 picture element subset of a four band multispectral image of the Verde Valley, Arizona, taken by the Landsat I satellite. $H(\mathscr{S})$ was also determined for a wide range of other test images and only slight differences from the $H(\mathscr{S})$ for the test image in Fig. 6.8(a) were observed.

For any source $\mathscr{S}$ let $e_h(\mathscr{S})$ be the ratio of entropy per band to the uncoded rate per band,

$$e_h(\mathscr{S}) = \frac{H_d(\mathscr{S})}{R^3_{spat}(\mathscr{S})} . \tag{4.13}$$

The following information concerning feature map entropy coding can be observed from the typical performance bounds shown in Fig. 4.4. The ratio $e_h(\mathscr{S})$ ranges mainly between 0.6 to 0.9 with a typical value of 0.75. For a given n, $e_h(\mathscr{S})$ reduces as n increases. Similarly, for a given n, $e_h(\mathscr{S})$ reduces as m decreases. In each case the decrease in $e_h(\mathscr{S})$ is due to the decrease in the ratio of clusters to picture elements. A decrease in this ratio corresponds to a coarser approximation of the image, which in turn causes a smoother contouring in the feature map and lower entropy. Incorporating the nonstationary characteristics of the source into the source model results in reducing $e_h(\mathscr{S})$ by about 5 to 15 percent. The difference between the simplest source model S and the most complicated practical source model SS_L is typically 15 percent. Furthermore, the use of the clearly unachievable bound

$H_d(SSS_L)$ results in less than 10 percent improvement in $e_h(\mathscr{S})$ relative to the use of $H_d(SS_L)$. In summary, the performance bounds for the practical source models indicate that entropy encoding could possibly reduce the spatial data by factors approaching 0.8 to 0.65 dependent on source modeling complexity. However, a reduction in spatial data rate by a factor less than 0.55 is not possible even with a considerably more complicated and less practical source model. Therefore, the practical encoder defined next will be applied only to the source models S_L and SS_L.

C. Definition of Practical Entropy Encoder

The chosen practical entropy encoder is an adaptive variable length encoder described in [31] and briefly defined below. An adaptive encoder is chosen to enable encoding of widely different types of image data. Once one can adapt to statistics of different images, it is a simple extension to be adaptive at a more local scale within the image. The encoder also includes the ability to represent 3-tuples of the source, thereby enabling average encoded data rates below one bit per element of the feature map. Of course, an adaptive Huffman encoder for the third extension of the source could be used, but such an encoder would be much less practical to implement. The adaptive variable length encoder defined below is shown to be easy to implement and performs near the previously discussed entropy bounds over a large range of source model entropies. Basically, the procedure of the adaptive encoder is to encode each code block with four different methods and use the method which results in fewest bits. Each encoding method is capable of good performance on typical sample sequences from

a source with an entropy within a specific and limited entropy range. By using the best of all four methods for each code block, good encoding performance is obtained for sources with entropies over the full range of interest.

The functions of the practical entropy encoder are diagramed in Fig. 4.5. The first element of each feature map sequence $\{\gamma^i\}_{i=1}^n$ is left uncoded and used as a decoding start reference. The rest of the feature map elements are segmented into code blocks of length J, where J is typically fixed between 16 to 32, except for the last code block which may be smaller. Each of these code blocks is then encoded with one of four methods and the method used is identified for each code block by using two overhead bits. A J between 16 to 32 represents a good performance compromise between adaptability and overhead costs. The first coding method is simply the natural coding of 3-tuples. This method serves as a backup coding procedure which limits the coded data rate to a rate only slightly higher than the uncoded data rate, because of the overhead bits. Furthermore, this encoding method is efficient for code blocks for which the symbols are nearly uniformly distributed. The results of this coding for the code block is stored in Buffer 1.

The other three coding methods require that the code block sequence be represented in terms of either differences between adjacent elements, or as a sequence of distance rank symbols as previously discussed in the source representations for the S^{ℓ} and SS^{ℓ} source

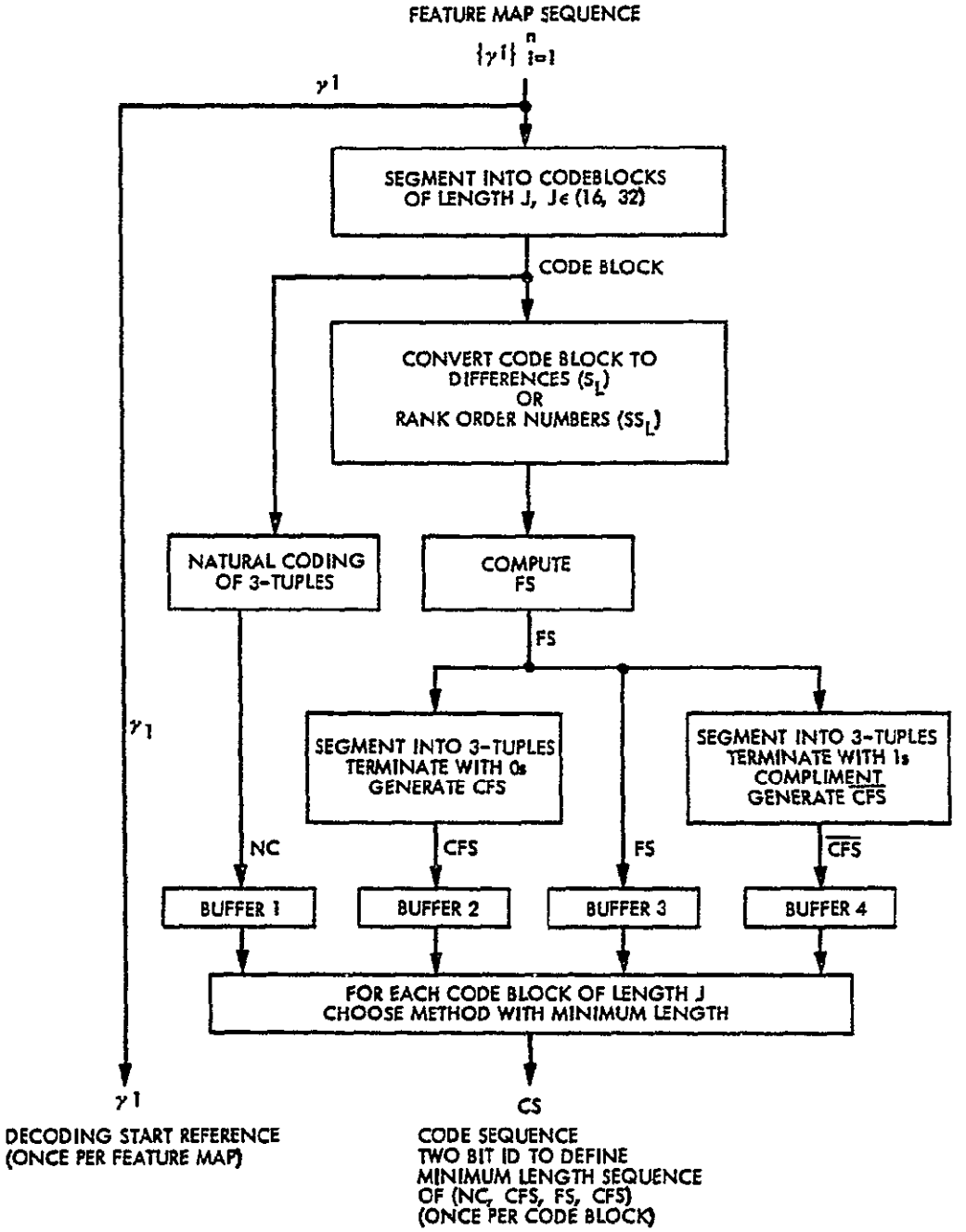


Fig. 4.5. Block Diagram of a Practical Entropy Encoder.

models respectively. If the source is modeled by S^{ℓ}, then a Fundamental Sequence FS is generated for the code block according to Table III. If the source is modeled by SS^{ℓ}, then the FS is generated according to Table IV. The FS for the code block is collected into Buffer 3 and constitutes the method 3 encoding for the code block. As can be seen from Tables III and IV, this encoding is efficient when the first few symbols of Tables III and IV occur with high frequency in the code block. However, this method is obviously not efficient for code block sequences which result in many 0s within FS. Neither is method 3 efficient coding when FS is nearly all 1s, since the average rate per symbol of the FS cannot be below 1 bit per symbol. Method 3 provides efficient coding for source models with entropies in the range of 1.5 to 3.0 bits per symbol [31].

At the same time that FS is stored into Buffer 3, it is also segmented into 3-tuples and terminated with extra zeros as needed. This sequence of 3-tuples is then used to generate a Coded Fundamental Sequence CFS according to Table V. The CFS is collected into Buffer 2 and constitutes the method 2 encoding for the code block. From Table V

TABLE III
CONSTRUCTION OF FS FROM DIFFERENCES

Difference	FS Contribution
0	1
1	01
-1	001
2	0001
-2	00001
.	.
.	.
.	.

TABLE IV
CONSTRUCTION OF FS FROM DISTANCE RANK SYMBOLS

Distance Rank Symbol	FS Contribution
1	1
2	01
3	001
4	0001
5	00001
.	.
.	.
.	.

it is clear that the method 2 encoding is more applicable to an FS with considerable runs of 0s. Method 2 provides efficient coding for source models with entropies in the range of 3.0 bits per symbol up to an entropy equal to the rate of natural coding of 3-tuples in method 1.

Method 4 is implemented in parallel by again segmenting FS into 3-tuples, but terminating with 1s. Then the sequence of 3-tuples are complemented and used to generate a Coded Fundamental Sequence Bar $\overline{CFS}$ by using the same Table V as used for generating CFS. The $\overline{CFS}$ sequence resulting from coding method 4 is stored into Buffer 4. Method 4 provides efficient coding for long sequences of 1s in the FS. This

TABLE V
CONSTRUCTION OF CFS FROM 3-TUPLES

3-Tuple Input	CFS Output
000	0
001	100
010	101
100	110
101	11100
011	11101
110	11110
111	11111

corresponds to efficient encoding for source models with entropies below 1.5 bits per symbol and down to entropies approaching 0.5 bit per symbol. The details concerning which range of entropies each coding method is most efficient for are discussed in [31].

For the b-th code block the coder selects the coded sequence which has minimum length and outputs the following: 1) a two bit sequence ID which identifies for the decoder the method used, and 2) the corresponding minimum length coded sequence, NC^b, CFS^b, FS^b, or $\overline{CFS}^b$. This procedure is repeated for each code block in the feature map. In this way the entropy encoder is noted to adapt to the data source for every J elements. The entropy encoder output for each feature map consists of the decoder start reference, and a multiple number of adaptively coded sequences. This encoded data stream includes all the necessary overhead to enable similarly simple decoding back to the original feature map elements.

Let the coded output sequence for code block b be represented by CS^b. Also let $L[\eta]$ represent the length in bits of any binary sequence η. Then for the b-th code block the coded output has length

$$L[CS^b] = \min\left[L[NC^b], L[CFS^b], L[FS^b], L[\overline{CFS}^b]\right] + 2. \quad (4.14)$$

The total length of all the coded output sequences for B code blocks in the ℓ-th feature map is given by

$$TL^{\ell} = \sum_{b=1}^{B} L[CS^b] \ . \quad (4.15)$$

The decoder start reference for each feature map requires $\lceil \log_2 m \rceil$ bits, where m is the number of source symbols in the feature map.

The averaged coded spatial data rate in bpppb for the ℓ-th feature map with source representation $\mathscr{S}^{\ell}$ is

$$R^{c}_{spat}(\mathscr{S}^{\ell}) = \frac{\lceil \log_2 m \rceil}{d \cdot n} + \frac{TL^{\ell}}{d \cdot n} , \tag{4.16}$$

where $\mathscr{S}^{\ell}$ is either S^{ℓ}, or SS^{ℓ}. The entropy coded spatial data rate averaged over the N_L feature maps of the image is

$$R^{c}_{spat}(\mathscr{S}_L) = \frac{1}{N_L} \sum_{\ell=1}^{N_L} R^{c}_{spat}(\mathscr{S}^{\ell}) . \tag{4.17}$$

In summary, $R^{c}_{spat}(S_L)$ is the average encoded spatial data rate (bpppb) resulting from entropy encoding the difference between adjacent feature map elements from the entire image. Similarly, $R^{c}_{spat}(SS_L)$ represents the average spatial data rate resulting from entropy encoding the distance rank symbols representing the feature maps.

D. Computer Simulation of the Practical Entropy Encoder

A computer simulation of the entropy encoder was developed to demonstrate its performance on the test image in Fig. 6.8(a). The performance of the simulated entropy encoder is shown in Fig. 4.6 in terms of $R^{c}_{spat}(S_L)$ and $R^{c}_{spat}(SS_L)$ as a function of m and n. The uncoded spatial data rate and the entropy per band for source models S_L and SS_L are also plotted for comparison. Recall that $H_d(S_L)$ and $H_d(SS_L)$ in Fig. 4.6 represent an unachievable bound to the performance obtainable by an entropy encoder which adapts to each feature map while encoding the nonstationary source models S_L and SS_L for the image in Fig. 6.8(a). The results in Fig. 4.6 are typical of results similarly obtained from many other multispectral images. It can be observed from Fig. 4.6 that

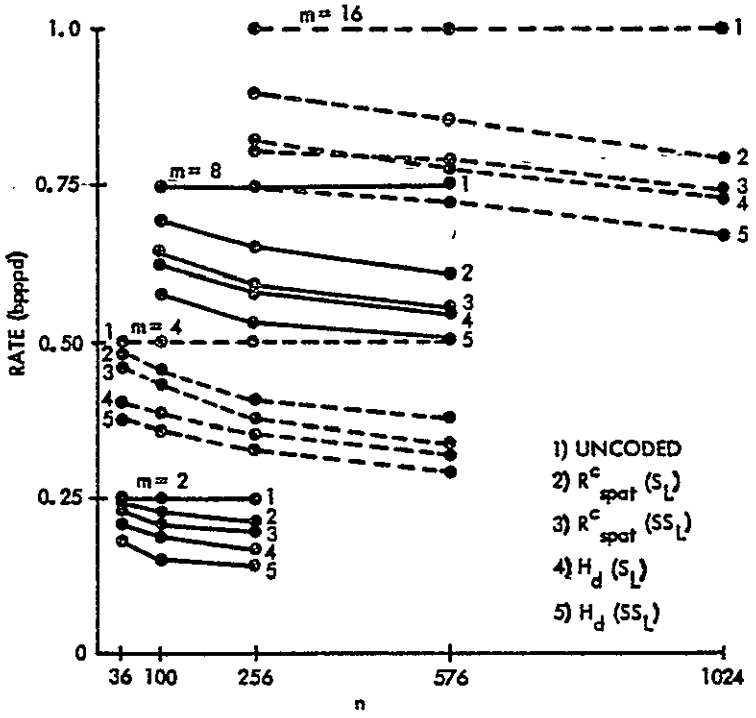


Fig. 4.6. Performance of the Practical Entropy Encoder.

the entropy encoder performs reasonably close to the corresponding unachievable performance bounds. Therefore, further refinement in the entropy encoder can result in only small improvements in performance. Furthermore, it was also shown earlier in Fig. 4.4 that even nonstationary 1st-order Markov source modeling of the data would not result in substantial improvement in the performance bound for entropy encoding. These observations indicate that additional substantial encoding performance gains are very unlikely to be obtained from any other practical encoder which is similarly constrained to permit exact reconstruction of the feature map. The above practical entropy encoder is also relatively easy to implement, since it requires simple logic and little memory. Note, for example, that Buffer 2 through Buffer 4 in Fig. 4.5 need be no larger than Buffer 1. Therefore, Buffer 1 through

Buffer 4 are all at most 128 bits. A serial implementation of the four coding methods can be used in lower data rate applications.

In summary, the entropy encoder should provide a typical average data rate reduction of about 20 percent. This data reduction is significant, especially since no degradation of the feature map occurs. The entropy encoder itself was also shown to be relatively easy to implement at high data rates. However, the entropy encoder is a variable length encoder, and variable data rates often seriously impact the complexity of the rest of the data system. For example, in applications on-board a satellite it is usually necessary to interface with a constant data rate transmitter. Thus the use of a variable data rate encoder introduces a need for considerable data buffering and data rate control measures between the encoder and the transmitter. The implementation complexity of these additional data system elements must be considered before choosing to use any variable-length encoder. In the next section, we consider other feature map coding methods which are not constrained to exact reconstruction of the feature map.

E. Other Feature Map Encoding

The above entropy encoding did not introduce any degradation in representing the feature map. However, in some applications only certain spatial characteristics of the feature map need to be preserved. For example, only closed boundaries in the feature map may be desired for field interpretation. The use of spatial texture in this application would allow the elimination of most irregular single elements within fields. When only specific spatial characteristics are of

interest, the feature map can in effect be represented by a sequence typical of a source with lower entropy. Then the following entropy encoding may obtain significantly lower data rates. However, the texture to be preserved and the associated fidelity criterion for its preservation are usually very difficult to define. Correspondingly, encoding algorithms which incorporate spatial texture are typically less amenable to high data rate implementations. Perhaps the task of extracting spatial texture from d-dimensional image data (e.g., boundary finding algorithms) may be aided by first having local regions of the image converted to scalar numbers which represent significantly different spectral features. The incorporation of spatial texture into the feature map encoding is not pursued at this point, but appears to be a topic worthy of further investigation.

An example of a very simple method for reducing the spatial data rate is the deletion of subsets of the feature map with reconstruction by linear interpolation. This approach is a crude but simple method which might be used if spatial resolution is higher than necessary for observing texture in a particular application. The spatial data rate in this case is simply reduced by the feature map subsampling factor.

F. Definition of the Coded CCA

The block diagram for the Coded CCA is shown in Fig. 4.7. The Coded CCA is identical to the Uncoded CCA shown in Fig. 3.1 except for the addition of the feature map encoder. The feature map encoder may be a spatial feature extractor, an entropy encoder, or both. In this work, primary consideration is given to entropy encoding, and the Coded

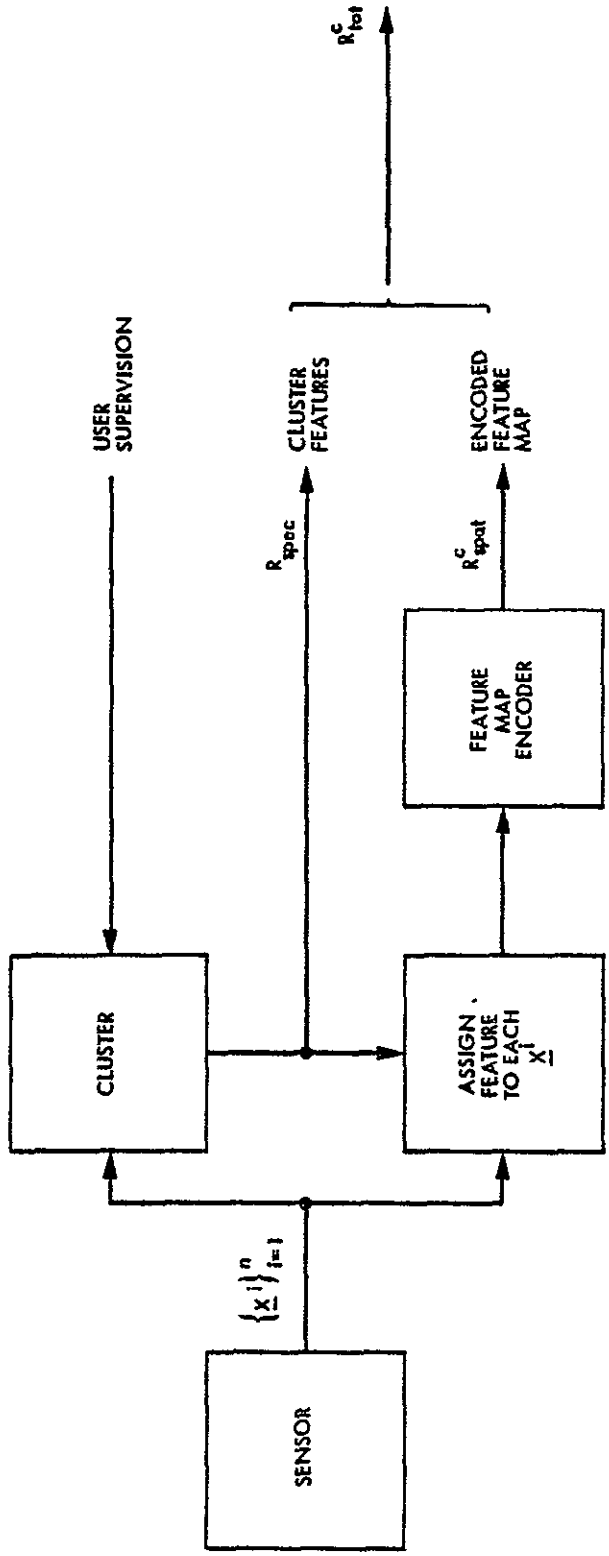


Fig. 4.7 The Coded Cluster Compression Algorithm.

CCA should be assumed to use this encoding unless otherwise specifically stated. Let R^c_{spat} represent the average rate in bpppb for the coded feature map sequence. Define a compression ratio CR for the feature map encoder by

$$CR = \frac{R^3_{spat}}{R^c_{spat}} = \frac{\frac{1}{3 \cdot d}\left\lceil \log_2 m^3 \right\rceil}{R^c_{spat}} \quad . \tag{4.18}$$

In general, the rate for the coded feature map sequence is then given by

$$R^c_{spat} = \frac{\frac{1}{3}\left\lceil \log_2 m^3 \right\rceil}{d \cdot CR} \quad , \tag{4.19}$$

and the total coded data rate for the Coded CCA is represented by R^c_{tot}, where

$$R^c_{tot} = R_{spec} + R^c_{spat} = \frac{m \cdot f}{n \cdot d} + \frac{1/3\left\lceil \log_2 m^3 \right\rceil}{d \cdot CR} \quad . \tag{4.20}$$

Equation (4.20) also applies to the Uncoded CCA when CR = 1.

As noted for the Uncoded CCA, the total data rate R^c_{tot} for the Coded CCA also consists of a mixture of spectral and spatial definitions. However, for the Coded CCA the division of R^c_{tot} between spectral and spatial definition is dependent on CR in addition to m and n. Let R^c_{tot} in (4.20) be a constant R^{c*}_{tot}, and express n as a function of m to get

$$n = \frac{m \cdot f}{d \cdot \left[R^{c*}_t - \frac{\frac{1}{3}\left\lceil \log_2 m^3 \right\rceil}{d \cdot CR} \right]} \tag{4.21}$$

where

$$R_{tot}^{c*} > \frac{\frac{1}{3}\left\lceil \log_2 m^3 \right\rceil}{d \cdot CR} .$$

A plot of n vs. m in (4.21) for a constant $R_{tot}^{c*} = 1$ is given in Fig. 4.8 with f = 24, d = 4 and CR = 1 and 2. This plot demonstrates that a given R_{tot}^{c} may be approximately obtained by many combinations of m, n and CR. Chapter VI will compare the performance of various (m, n, CR) combinations which have the same R_{tot}^{c}.

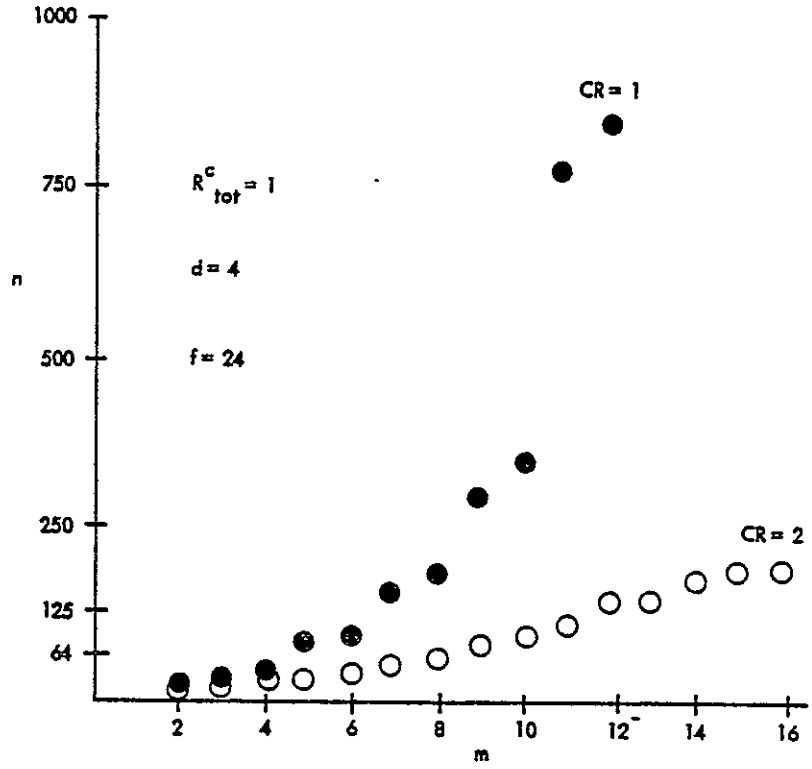


Fig. 4.8. Plot of n vs. m for $R_{tot}^{c} = 1$ with f = 24, d = 4, and CR = 1 and 2.

CHAPTER V

OTHER CLUSTERING FORMS OF THE CCA

A. The Adaptive CCA

In both the Uncoded CCA and the Coded CCA, it was assumed that the number of clusters m per local source was a constant throughout the entire image. However, since most image data is nonstationary, it is desirable to have the number of clusters used per local source be variable and adaptively determined to meet the quality requirements for the data of each specific local source. For example, one local source may consist of data for which only two clusters are necessary to meet the quality requirements, while another local source may require ten clusters. Rather than use ten clusters for each local source, it is preferable to use only as many clusters as necessary for each local source. The Adaptive CCA is defined to incorporate the use of adaptive clustering. The advantage of the Adaptive CCA is that a lower average value of m can be used and thereby the spectral and spatial data rates are both reduced.

The form of the Adaptive CCA is the same as for the Coded CCA shown in Fig. 4.7 except the clustering is adaptive. The Adaptive Clustering Algorithm (ACA) described in Chapter II is used to demonstrate the concept of the Adaptive CCA. The ACA is adaptive by deleting, splitting, or combining clusters according to corresponding user supervision theresholds. The user supervision also specifies the lower and upper limits, on the number of clusters used per local source. These thresholds and limits ideally allow the user to define the quality requirements in terms of 1) minimum number of elements

(relative spatial density) required for distinct representation, 2) maximum intracluster variance, 3) minimum intercluster distance, and 4) the minimum and maximum number of clusters per local source (minimum, and maximum number of meaningfully different spectral reflectances present per local source). These user supervision inputs are constants and need to be specified only once for the entire image.

For each local source, the adaptive clustering results in a variable number of clusters. The number of clusters for each local source is dependent on the user supervision inputs and the data of the local sources. The spectral data definition still defines the cluster features present for the local source, only now the number of features is variable. Thus, the spectral data definition of each local source must now also include a short lead feature which indicates the number of clusters present. The feature map encoder and the coded spatial data rate are unchanged from the nonadaptive CCA, except that the number of symbols per feature map element now can change from one feature map to the next. The ground data selection, reconstruction, or interpretation proceeds as before except that the lead spectral feature is used to inform the decoder of the number of spectral features being transmitted for each local source.

The extra complexity introduced by adaptive clustering comes from two sources, 1) the added complexity of the clustering algorithm, and 2) the increased data system complexity resulting from the variable data rate. Refer to the block diagram of the ACA shown in Fig. 2.4, and the BCA shown in Fig. 2.3. Observe that both algorithms have the same Assign function which requires a very high computation rate of a simple

distance measure between vectors, for example the distance measure in (2.1) or (2.2). The ACA in addition requires more difficult computation during the cluster modification phase between Assign iterations, for example the intercluster distance measure of (2.12) for combining clusters. However, this more involved computation can be performed at a lower rate than the simple Assign computation, since it is required only once per each complete Assign function. Thus the structure of the ACA is such that the more complicated computation is required at a greatly reduced frequency of repetition.

The adaptive clustering results in a variable data rate and the same associated data system problems discussed in Chapter IV for the variable-rate feature map encoder. The range of the data rate is limited by the specification of the upper and lower bounds for the number of clusters. In a data system application requiring a fixed data rate, one should consider augmenting the adaptive clustering with a control structure to achieve a fixed average data rate over a volume of data which is compatible with practical data buffer limitations. This control structure would likely consist of an estimator for determining the percentage rate allocation to be made between the local sources of a fixed rate block. Then the data rate allocations could be made through corresponding modifications of the thresholds and limits used for each local source. At this time the investigation is focused on the Adaptive CCA without consideration of rate control.

Let m^{ℓ} and CR^{ℓ} represent the number of clusters and the feature map encoding compression ratio respectively for the ℓ-th local source of the image. Let M_{ub} be the maximum number of clusters allowed in any

local source as specified by the user supervision. Assume the number of clusters per local source is identified by simply using $\lceil \log_2 M_{ub} \rceil$ bits at the beginning of the spectral definition data for every local source. Then the average adaptive spectral data rate $R(A)_{spec}$ and the average adaptive spatial data rate $R^c(A)_{spat}$ for the Adaptive CCA are given by

$$R(A)_{spec} = \frac{\lceil \log_2 M_{ub} \rceil}{d \cdot n} + \frac{1}{N_L} \sum_{\ell=1}^{N_L} \frac{m^\ell \cdot f}{d \cdot n} \tag{5.1}$$

and

$$R^c(A)_{spat} = \frac{1}{N_L} \sum_{\ell=1}^{N_L} \frac{\lceil 3 \log_2 m^\ell \rceil}{3\, d \cdot CR^\ell} \tag{5.2}$$

where N_L is the number of local sources in the image. The average adaptive total data rate $R^c(A)_{tot}$ is given by

$$R^c(A)_{tot} = R(A)_{spec} + R^c(A)_{spat} . \tag{5.3}$$

In the next chapter computer simulations are used to measure performance improvements due to using adaptive clustering.

B. The Cascaded CCA

When a sequence of local sources is individually clustered, there is likely to be considerable similarity between many of the clusters. For example, assume a sequence of 16 local sources are clustered to 8 clusters each. This results in 128 different clusters for the

sequence of 16 local sources. However, many of these 128 clusters resulted from image elements in different local sources, but with similar spectral reflectance properties. Thus many of these 128 clusters are very similar. Usually similar clusters come from different local sources. For example, cluster 1 of local source 1, cluster 1 of local source 2, cluster 2 of local source 3, etc., could all be similar clusters because they might each represent similar image data in different local sources. However, similar clusters can also be from the same local source. Although the clusters within a local source are by design dissimilar from one another, the clusters might still be very similar on a relative basis when compared with all the other clusters resulting from a sequence of local sources. For example, assume that the above 128 clusters represent 16 distinctly dissimilar types of image data. However, assume local source 3 possesses only 2 of the 16 dissimilar types of image data. Then many of the 8 clusters from local source 3 will be quite similar relative to the dissimilarities occurring between the other clusters.

One method of taking advantage of these similar clusters is to use cascaded clustering within the CCA. Cascaded clustering basically collects the cluster descriptions for a sequence of local sources and then clusters these local clusters into a smaller set of cluster descriptions to serve as the cluster features for the sequence of local sources. For example, a sequence of 16 local sources are initially clustered to 8 clusters each, and the cascaded clustering combines the resulting 128 local clusters into 16 sequence clusters to represent the

sequence. Each local source is still represented in terms of local cluster features, however, the local cluster features for each local source are now a subset of the 16 sequence clusters which represent the sequence of local sources. In addition, the cascaded clustering combines those clusters within a local source which are similar relative to the clusters present in the sequence. Thus the average number of clusters per local source might for example be reduced from 8 to 5. Cascaded clustering reduces the spectral data rate by eliminating the need to separately define the many similar clusters resulting from independent clustering of local sources. Cascaded clustering also reduces the spatial data rate by reducing the average number of clusters per local source through the combining of clusters in a local source which are similar compared to all the clusters present in the sequence of local sources. An inherent advantage in cascaded clustering is the ability to modify the local source clustering based on knowledge of clustering results for surrounding local sources.

1. Sequence Spectral Definition

The form of the Cascaded CCA is shown in Fig. 5.1, and a flow chart of the functions is given in Fig. 5.2. The input data consists of a sequence of N_s local sources, each consisting of n image elements. The partial supervision as before specifies the constant number of clusters m_ℓ to be obtained for each local source. Each local source is then clustered as in the previous CCA forms to provide a set of m_ℓ cluster features and a feature map for each local source. However, now the m_ℓ cluster definitions and the feature map for each of the N_s local

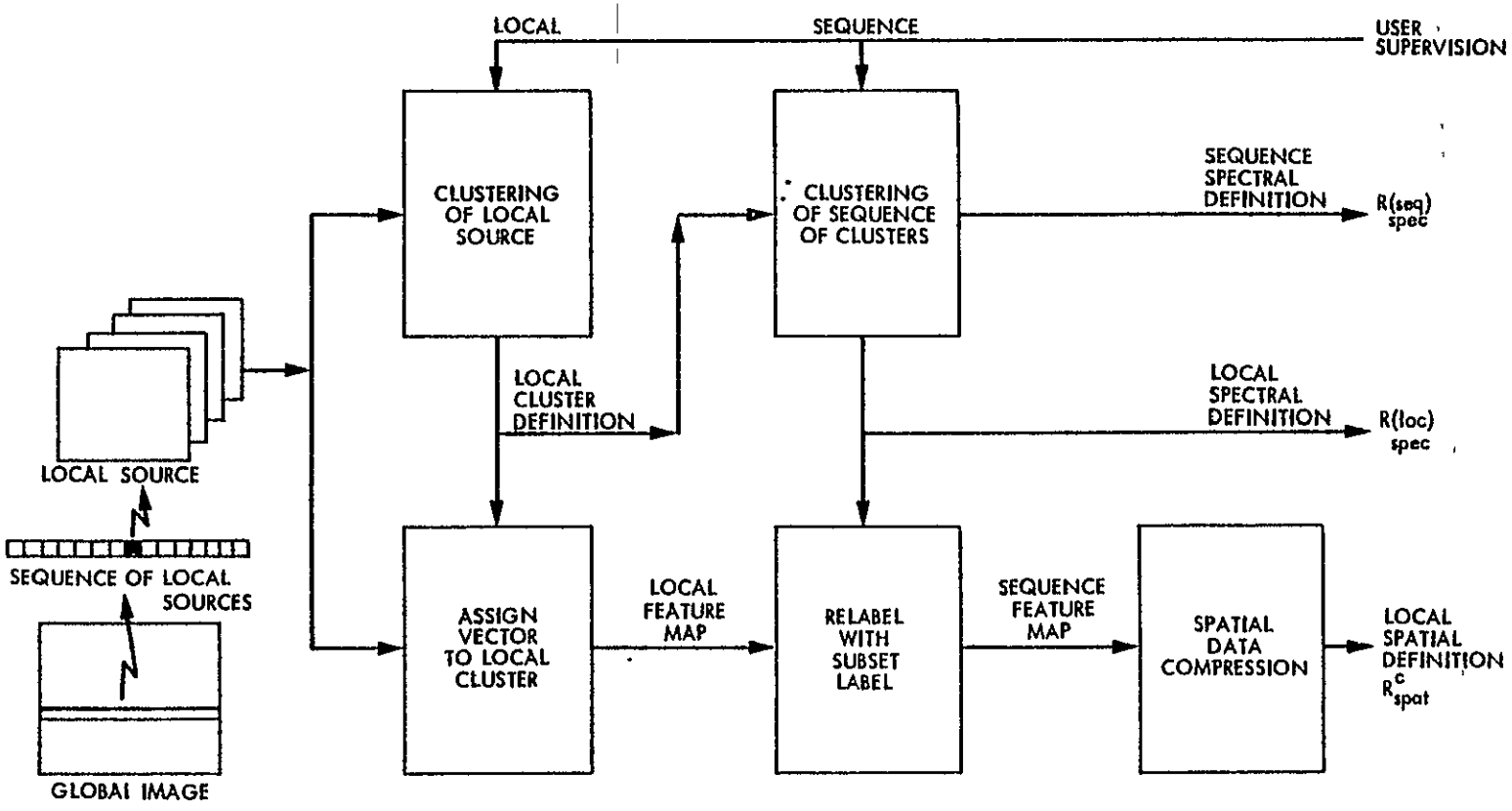


Fig. 5.1. A Block Diagram of the Cascaded CCA.

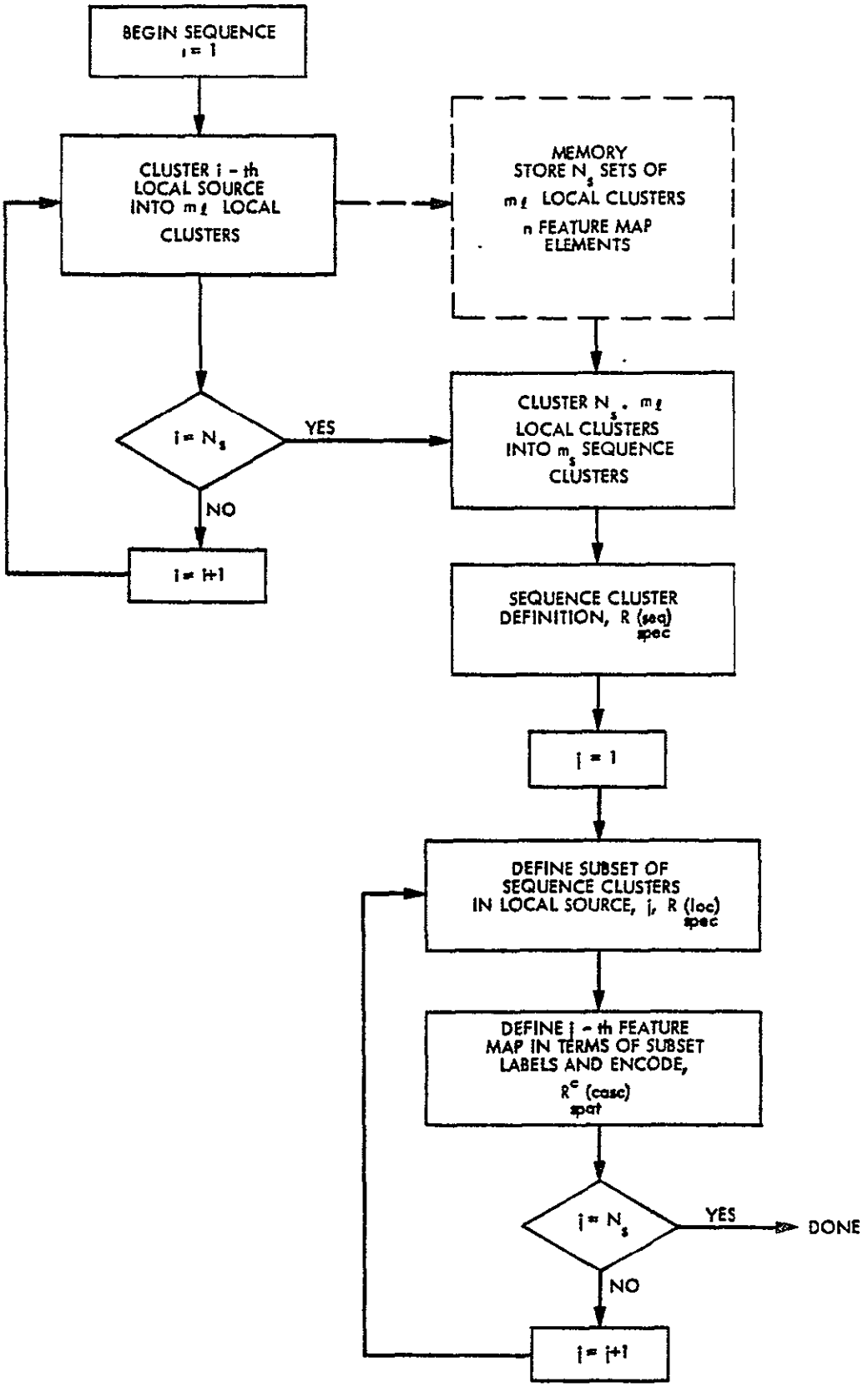


Fig. 5.2. Flow Chart of Cascaded CCA Functions.

sources are collected in memory for further processing. The user supervision for the Cascaded CCA must in addition specify the user quality requirements on a sequence basis, as, for example, the number and type of cluster features desired to represent the sequence of local sources. Once the $m_\ell \cdot N_s$ local clusters have been collected, they are clustered to obtain a set of m_s sequence clusters to be used as the spectral features for the sequence of local sources. Assume the Cascaded CCA and the ground decoder agree to assign the integer labels $1,2,\ldots,m_s$ to sequence clusters, according to the order of transmission and reception. The cascaded clustering results in a mapping between the integer labels for the $m_\ell \cdot N_s$ local clusters and the integer labels for the m_s sequence clusters, as demonstrated in Fig. 5.3. The m_s sequence clusters are output to provide the cluster definition per sequence at a sequence spectral data rate defined as R_{spec} (seq).

2. Local Spectral and Spatial Definition

Figure 5.4(a) shows sections of typical feature maps for the first and second local sources in terms of the local labels. Figure 5.4(b) shows the same feature map sections represented in terms of the sequence labels. The feature map for the Cascaded CCA is in terms of the sequence labels as in Fig. 5.4(b). However, since the total number of sequence clusters is usually larger than the number of sequence clusters used in a given local source the feature map representation in Fig. 5.4(b) is inefficient. A more efficient method of transmitting each feature map is to send $\lceil \log_2 m_\ell \rceil$ bits to identify the number of sequence clusters m_s^b occurring within the b-th local

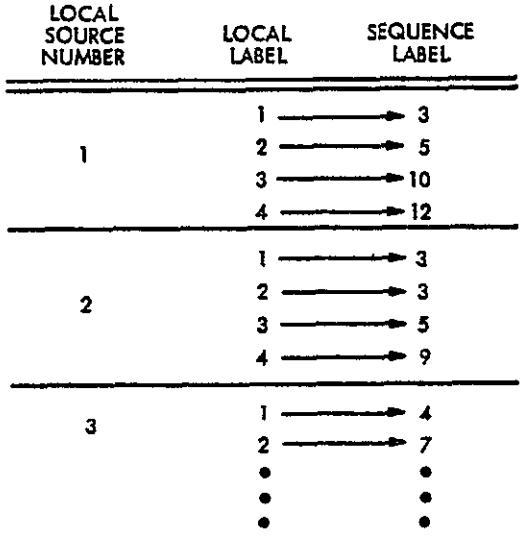


Fig. 5.3. Mapping Between Local Cluster Labels and Sequence Cluster Labels.

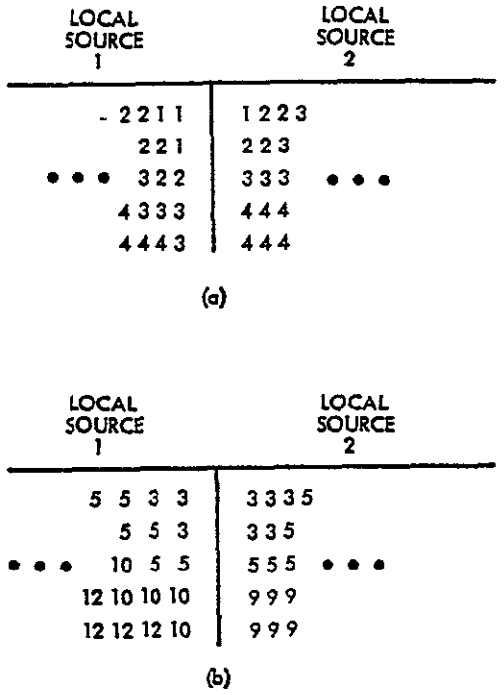


Fig. 5.4. Typical Sections of Feature Maps. (a) In local labels. (b) In sequence labels.

source. Then send $\lceil \log_2 m_s \rceil$ bits to identify each of the m_s^b sequence clusters present in the b-th local source. This data is referred to as the local spectral data with a data rate of $R(loc)_{spec}$, and an example of this data is given in Fig. 5.5 for the local source examples shown in Fig. 5.3 and Fig. 5.4. Assume the Cascaded CCA and the ground decoder agree to assign in the order of transmission integer labels 1,2,... to the subset of sequence clusters present in a given source. These integer labels are defined as the subset labels, and they are assigned to the subset of sequence labels in the order of their transmission. An example of the mapping between the subset of sequence labels in a local source and the subset labels is shown in Fig. 5.6.

LOCAL SOURCE 1	LOCAL SOURCE 2
4 - 3, 5, 10, 12	3 - 3, 5, 9

Fig. 5.5. Local Spectral Data for Local Sources 1 and 2.

LOCAL SOURCE 1		LOCAL SOURCE 2	
SEQUENCE LABEL	SUBSET LABEL	SEQUENCE LABEL	SUBSET LABEL
3	1	3	1
5	2	5	2
10	3	9	3
12	4		

Fig. 5.6. Mapping of Sequence Labels into Subset Labels for Local Sources 1 and 2.

Observe that the mapping in Fig. 5.6 simply takes the sequence labels back into the original local labels of Fig. 5.3 except for the deletions of local clusters which were combined in the cascaded clustering. The feature map for the Cascaded CCA can then be represented in terms of the subset labels as shown in Fig. 5.7. The decoding of the subset labels is defined by the transmitted spectral data shown in Fig. 5.5 which specifies for the decoder the mapping in Fig. 5.6. The transmission of the uncoded feature map in terms of subset labels (Fig. 5.7) requires $\lceil \log_2 m_s^b \rceil$ bits per element, which is usually considerably less than $\lceil \log_2 m_s \rceil$ bits per element required to directly transmit

LOCAL SOURCE 1	LOCAL SOURCE 2
2211	1112
221	112
• • • 322	222 • • •
4333	333
4443	333

Fig. 5.7. Cascaded CCA Feature Map in Terms of Subset Labels.

the feature map in terms of sequence labels (Fig. 5.4(b)). Also the overhead bits required to enable decoding of the subset labels is very small when averaged over all the samples in the local source. Consider local source 1 in the example of Figs. 5.3-5.7, where m_s=16, m_s^1=4, m_ℓ=4, and assume the number of elements n is 256. Direct transmission of the sequence labels in Fig. 5.4(b) requires 4 bits/element. Alternately, transmission of the overhead in Fig. 5.5 requires an average of $(2 + 4 \times 4)/256 = .07$ bit/element, and the transmission of the subset labels in Fig. 5.7 requires 2 bits/element, or a total of 2.07 bits/element. Representation of the Cascaded CCA feature map in terms of subset labels significantly reduces the spatial data rate, while preserving a simple look-up table architecture for the decoding implementation.

3. Implementation Considerations

The primary new function in the Cascaded CCA of Fig. 5.1 is the cascaded clustering. Cascaded clustering impacts the implementation complexity in terms of memory and computation requirements. Additional memory is required to collect the local cluster definitions and the local feature maps which make up a sequence. The number of cluster definitions to be stored per sequence is $m_\ell \cdot N_s$, where m_ℓ is the number of clusters per local source and N_s the number of local sources per sequence. For example, assume m_ℓ=8 and N_s=16, and that the mean and variance of each of 4 bands are stored for each cluster. Then the storage required is 1024 8-bit words. Each local feature map requires storage for $n \cdot N_s$ local labels, where n is the number of elements per feature map. A reasonable maximum value of m_ℓ is 16, so each label

takes 4 bits. Thus a sequence of 16 local sources, each of 16 × 16 elements requires 4096 4-bit words of storage. These numbers indicate a relatively small amount of memory is adequate for the Cascaded CCA.

The additional computational requirements in the Cascaded CCA arise mainly due to the cascaded clustering. The relabeling of any feature map from local labels to subset labels requires a simple mapping as demonstrated in Figs. 5.3 and 5.5. However, the clustering of clusters requires the use of intercluster distance measures which are generally more complicated than the distance measures used in clustering vectors. A discussion of many intercluster distance measures is contained in [13], and an intercluster distance measure used for the Adaptive Clustering Algorithm (ACA) is discussed in Chapter II. An important attribute of the Cascaded CCA structure is that the more difficult computations require a correspondingly lower frequency of repetition. The more difficult computation of the cascaded clustering must be accomplished only once per the clustering of N_s local sources, and similarly only once per the $N_s \cdot n$ vectors present in an entire sequence. Therefore, the impact of cascaded clustering on the average computation per vector is greatly reduced.

The basic concept of this cascaded structure is to use fast special purpose hardware for simple computation at the data vector level for extracting local features, and then to use slower general purpose microprocessors for more complicated computation at the local source level for extracting sequence features. Of course, this cascading could be expanded with additional higher levels of even slower rate and more sophisticated computation. In addition, one could incorporate adaptive clustering within the Cascaded CCA at the local level, sequence level,

and higher. The generalized concepts of this cascaded architecture appear to be useful for applications in artificial intelligence and other sophisticated pattern recognition problems dealing with high data rate sources. However, the scope of this work is limited to formulation of the concept and investigation of a very simple form of the Cascaded CCA.

4. A Specific Cascaded CCA Form

A simple form of the Cascaded CCA in Fig. 5.1 is defined below. The user supervision consists of specifying a constant number of local clusters to be obtained for each local source, a fixed number of local sources to be used for each sequence, and a constant number of sequence clusters to be obtained for each sequence. This user supervision is specified only once per entire image. The clustering of vectors is accomplished using the Basic Clustering Algorithm discussed in Chapter II and shown in Fig. 2.3. The cascaded clustering is obtained from using the BCA again to cluster the local cluster means. This cascaded clustering is nonadaptive and uses the Euclidean distance between cluster means for a simple and crude measure of intercluster distance. Feature map encoding consists of the entropy encoder defined in Chapter IV. Let n be the number of elements per local source, N_s the number of local sources per sequence, m_s the number of sequence clusters, and f the number of bits used to define each sequence cluster. The sequence spectral data rate $R(\text{seq})_{\text{spec}}$ in bpppb for transmitting the sequence cluster definition is given by

$$R(\text{seq})_{\text{spec}} = \frac{m_s \cdot f}{N_s \cdot n \cdot d} . \tag{5.4}$$

Let m_ℓ be the number of local clusters per local source, and m_s^b the number of sequence clusters present in the b-th local source. The average local spectral data rate $R(loc)_{spec}$ for defining how many and which sequence clusters are present per local source is given by

$$R(loc)_{spec} = \frac{\lceil \log_2 m_\ell \rceil}{n \cdot d} + \frac{\lceil \log_2 m_s \rceil}{n \cdot d \cdot N_L} \sum_{b=1}^{N_L} m_s^b \,, \tag{5.5}$$

where

$$\overline{m_s} = \frac{1}{N_L} \sum_{b=1}^{N_L} m_s^b \tag{5.6}$$

is the average number of sequence clusters per local source for the N_L local sources in the image. Let d be the number of spectral bands and CR^b be the feature map encoding compression ratio for the b-th local source. Then the average encoded spatial data rate for the feature map of subset labels is

$$\overline{R^c_{spat}} = \frac{1}{d \cdot N_L} \sum_{b=1}^{N_L} \frac{1}{3} \frac{\lceil 3 \log_2 m_s^b \rceil}{CR^b} \tag{5.7}$$

The total rate in bpppb for the Cascaded CCA is

$$R(casc)_{tot} = R(seq)_{spec} + R(loc)_{spec} + R^c_{spat} \,, \tag{5.8}$$

or

$$R(casc)_{tot} = \frac{m_s \cdot f}{N_s \cdot n \cdot d} + \frac{\lceil \log_2 m_\ell \rceil}{n \cdot d} + \frac{\lceil \log_2 m_s \rceil \cdot \overline{m_s}}{n \cdot d} + \frac{1}{d \cdot N_L} \sum_{b=1}^{N_L} \frac{\frac{1}{3} \lceil 3 \log_2 m_s^b \rceil}{CR^b} \,. \tag{5.9}$$

5. Impact on User Data Processing

The following is an example of use of the Cascaded CCA. Assume 16 local sources (N_s=16) of 16 × 16 elements (n=256) are clustered to 8 local clusters (m_ℓ=8) each and then the clusters are clustered to 16 sequence clusters (m_s=16). Assume each sequence cluster definition consists of 7 bits to define within 1 percent the number of elements per cluster, 32 bits to define the cluster mean in 4 bands (d=4), and 16 bits to define the cluster variance in 4 bands. Each cluster definition thus requires 55 bits (f=55). The sequence spectral definition data rate from (5.4) is $R_{spec}(seq)$ = .0537 bpppb. A typical average number of sequence clusters present per local source would be about 5.5 ($\bar{m}_s$=5.5). Then the local spectral definition ($R_{spec}(loc)$) in (5.5) is .0244 bpppb. There are 4 spectral bands and assume the average coded spatial data rate per band in (5.7) for the average of 5.5 subset labels is R^c_{spat}=.5 bpppb. This corresponds to an average entropy encoder spatial data rate reduction of .81. Thus the total Cascaded CCA data rate is .578 bpppb. If the entropy coder was not used the total data rate would be about .7 bpppb.

In some data systems (e.g., Landsat) it is not sufficient to provide data compression as a means for obtaining an approximate image with the transmission of fewer bits. In fact it can be more important to make the data compression provide a form of preprocessing which facilitates user data selection and user data interpretation. An accurate measure of how much any algorithm benefits user data processing can only come through interactive assessment with specific data users.

However, here we attempt to exemplify potential user processing benefits of the Cascaded CCA concept through continuation of the above example.

Figure 5.8 (a) - (d) shows block diagrams of how the outputs of the Cascaded CCA might be used to aid user data selection and user data interpretation. Assume first that the user needs to know what of interest is there, but does not need to determine accurately where. Such information is useful for user data selection to quickly eliminate large sequences of data with nothing of interest. Also this information may be all that is needed, as in inventory assessments (e.g., determination of the percentage of crop types). The user can obtain a quick overview of the data by observing only the sequence spectral definition of the Cascaded CCA. In the above example, the sequence spectral definition provides a representation of what and how much is present in a sequence of 4096 picture elements in terms of 16 sequence cluster definitions. As shown in Fig. 5.8 (a), the user provides a classifier to identify which clusters represent data belonging to a class of interest. The use of $R_{spec}(seq)$ to identify which classes of interest are present per sequence of 4096 picture elements required a data rate of only .0537 bpppb, and the user classification of only 16 features. This reduces the data volume the user must handle by a factor of 112, and also reduces the number of user classifications to be made by a factor of 256.

Assume the user has selected a sequence of 4096 picture elements which has one or more classes of interest, and the user now desires to obtain a better spatial definition for these classes. Let the class

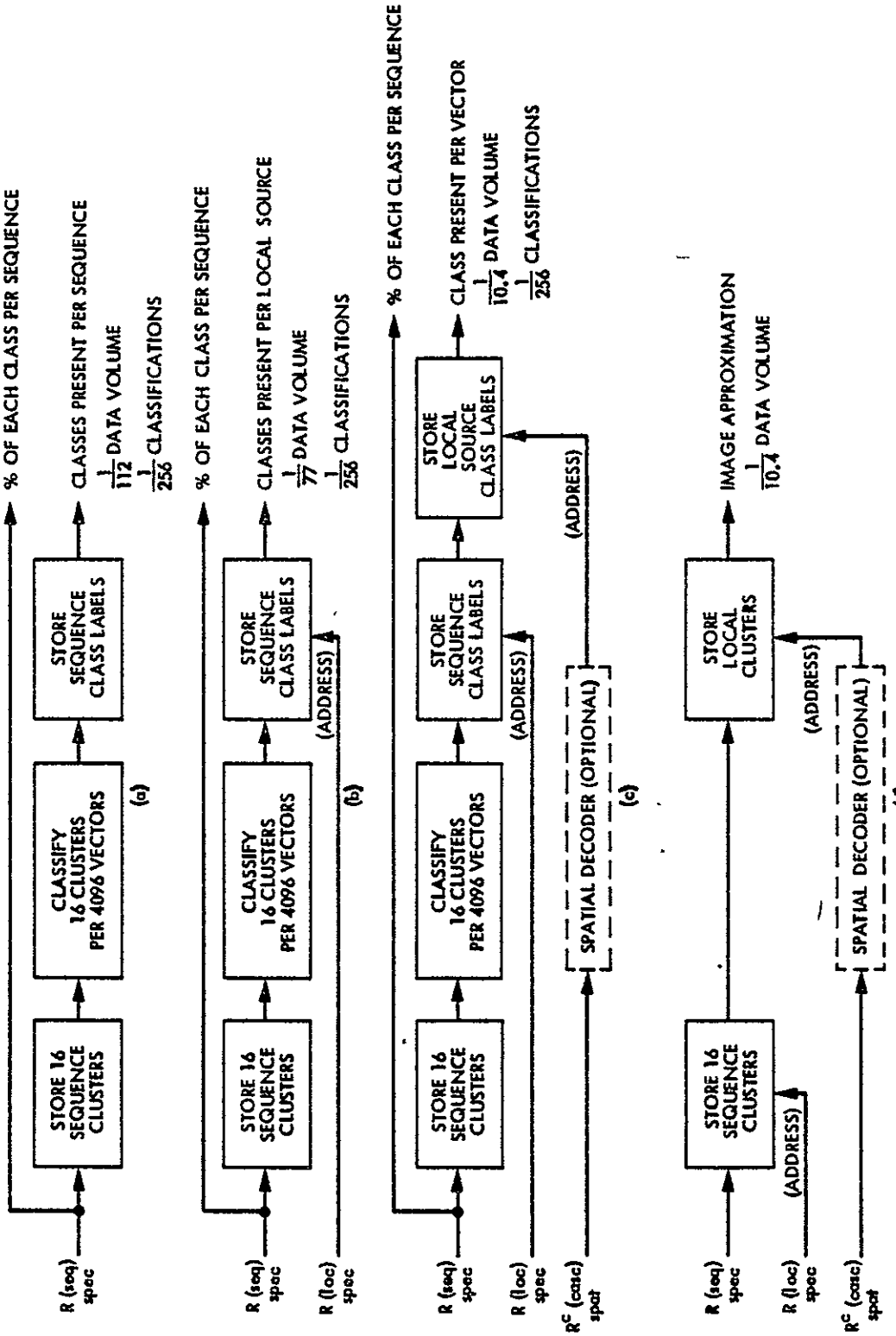


Fig. 5.8. Block Diagram of User Data Selection and Interpretation for the Cascaded CCA. (a) Classes per sequence. (b) Classes per local source. (c) Class per vector. (d) Image approximation.

labels assigned to the 16 sequence clusters be stored in a 16 word look-up table. The local cluster definition is a list of which sequence clusters are present per 16 × 16 local source in the sequence. Figure 5.8 (b) shows the use of the local spectral definition as an address to the look-up table with the class labels. The output is the identification of classes present per each 16 × 16 image elements, thus permitting a finer level of data selection within a larger sequence of image elements known to have classes of interest. Even when this finer level (16 × 16 image elements) of data selection is used, the data rate is only .078 bpppb. Thus the data volume is reduced by 77, and the number of classifications still reduced by 256.

In some applications the identification of what is present within 4096, or 256 image element sets would be adequate. In other applications the user may desire to know where the classes of interest occur on an image element by image element basis. Refer to Fig. 5.8(c). The subset of class labels for each 16 × 16 local source are stored in an 8-word look-up table. The spatial data definition given by the feature map is used as an address to define the class label appropriate for each image element. If feature map encoding is used then a decoding is required before obtaining the address. This additional detailed spatial information increases the data rate to .578 bppb. The data volume is reduced by 10.4, but still the number of classifications is reduced by 256.

Finally, assume the user desires instead a reconstructed image for manual interpretation. This application of the data is shown in Fig. 5.8(d). The 16 sequence clusters defined by the sequence cluster

definition data are stored in a 16-word look-up table. The local cluster definition data is used as an address to output the subset of sequence clusters which are present for each local source. The subset of sequence clusters are stored in an 8-word look-up table. Then the spatial data definition is used as addresses to output an approximate reconstruction for each picture element. The approximation typically would be the cluster mean, although the cluster variance per band could also be used in reconstructing an image. The data rate required for this image reconstruction is also .578 bpppb, representing a data reduction of 10.4.

Observe that a combination of the functional elements in Figs. 5.8(c) and 5.8(d) give the capability for acquiring all or any subset of the user outputs simultaneously. Figure 5.8 also demonstrates that a simple look-up table architecture can be used for decoding. The above examples conceptually demonstrate how the Cascaded CCA benefits ground data handling by providing for data selection and reduced interpretive computation. Some computer simulation results of the Cascaded CCA are given in the next chapter. Further research is needed to better assess the practical benefits and to investigate other forms of this concept.

CHAPTER VI

PERFORMANCE OF VARIOUS CCA CONFIGURATIONS

The performance of a data compression algorithm is measured in terms of the data quality vs. the data rate. Data quality is most commonly measured in terms of percent mean square error (%MSE), classification accuracy, and subjective appearance.

Subjective appearance is an appropriate criteria when manual photographic interpretation is used. In such cases it is often too difficult to define a mathematical expression for adequately quantifying data quality. The obvious disadvantage of this criteria is that it is subjective rather than quantitative.

Classification accuracy is an important quantitative measure of data quality in applications where the compressed data is automatically interpreted (e.g., Landsat data applications). However, classification accuracy is dependent on the classification technique, as well as the data compression technique, and often significant accuracy improvements can be obtained by tailoring the classification technique to the specific data compression technique. Therefore, it is important to jointly investigate compression and classification when the data quality is based on classification accuracy.

%MSE also provides a quantitative measure of data quality. This criteria is widely used because of its convenient mathematical form. The %MSE is not generally meaningful for measuring data quality in an absolute sense. For example, it would not usually be useful for comparing data quality across different compression techniques, or across

different test images. However, the %MSE can be very useful for comparing performance from options of the same compression technique and the same test image.

In this chapter computer simulations are used to provide comparative performance results for the various CCA configurations. These performance comparisons can be helpful in assessing the impact of data compression on the data user, and also for defining performance vs. implementation complexity trade-offs. In terms of implementation it is desirable to keep m, n, and the number of clustering iterations small in order to lower memory and computation requirements. It is also easier to implement simpler distance measures and nonadaptive configurations of the CCA. However, the impact on performance of these parameters must also be determined. Below, the impact on performance of various CCA configurations is given in terms of %MSE, subjective appearance, and classification accuracy.

A. %MSE Performance Results

Let $\underline{X}$ represent the original d-dimensional image data, and let $\hat{\underline{X}}$ represent the reconstructed approximation of the same image data after compression. Then the MSE between the original image and the approximate image is the expected value of the square of the Euclidean distance between $\underline{X}$ and $\hat{\underline{X}}$, or

$$MSE = E\left[\left|\left|\, \underline{X} - \hat{\underline{X}} \,\right|\right|^2\right]. \tag{6.1}$$

The average energy, $\mathcal{E}$ in the original image is defined in terms of the variance in the i-th spectral band, σ_i^2, i = 1, 2, ..., d by

$$\varepsilon = \sum_{i=1}^{d} \sigma_i^2 \, . \tag{6.2}$$

Thus the %MSE is defined by

$$\%MSE = \frac{MSE \cdot 100}{\varepsilon} \tag{6.3}$$

1. Uncoded CCA

Various configurations of the CCA were simulated on the test image shown in Fig. 6.8(a). This test image is a 256 × 256 picture element subset from a 4-band Landsat image of the Verde Valley in Arizona. The purpose of these simulations is to obtain a measure of %MSE vs. total data rate as a function of the (n, m, CR) combinations. The first simulations used the Uncoded CCA of Fig. 3.1. This corresponds to determining performance as a function of (n,m) pairs for CR = 1. The performance of this Uncoded CCA in terms of %MSE vs. R_{tot} for various values of n is plotted in Fig. 6.1. For each n shown, all square subsets of n picture elements in the test image were compressed using various values of m. For example, for n = 36 the test image was processed four times as follows: two features (m = 2) per each subset of 6 × 6 (n = 36) picture elements, three features (m = 3) per 6 × 6, four features (m = 4) per 6 × 6, and five features (m = 5) per 6 × 6. These four simulation results were then plotted in Fig. 6.1 and a smooth curve was drawn between them. The cluster feature used for this image approximation application was only the cluster mean, and each cluster mean was defined by either 6 or 8 bits per band (f = 24 or 32). Therefore, the image approximation consisted of substituting the closest cluster mean for each picture element. For every simulation with a

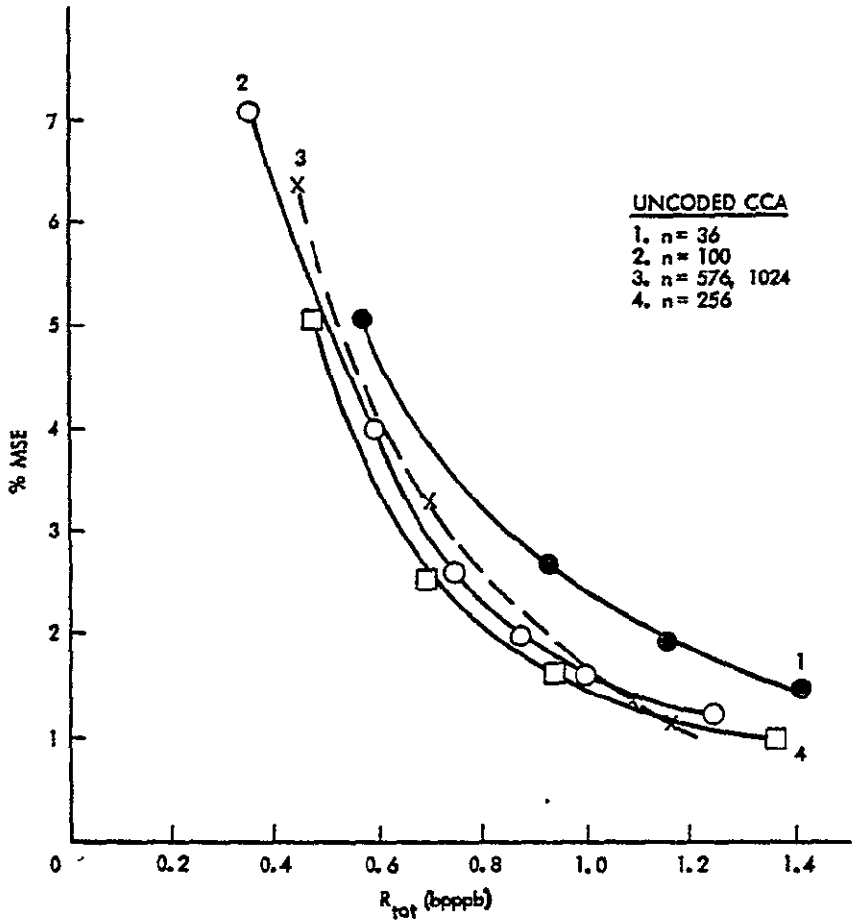


Fig. 6.1. %MSE vs. R_{tot} for Various Values of n for the Uncoded CCA.

given n and m, the %MSE between the original and reconstructed image was calculated from (6.1) through (6.3) and R_{tot} was calculated from (3.11). For n = 576 and n = 1024 the performance curves were essentially identical. The results in Fig. 6.1 show that for the selected source data the Uncoded CCA peaks in performance for all R_{tot} at n approximately equal to 256. Similar results have also been obtained from simulations with other test images. From (3.11) one can see that for constant R_{tot} the spectral data definition decreases as n increases, and the spatial data definition increases. The simulations suggest

that as n increases above 256, the data quality (%MSE) is decreased more due to poorer spectral definition than it is improved due to better spatial definition.

2. Coded CCA

The same simulations that were conducted for the Uncoded CCA (CR=1) were also conducted with entropy coding of the feature map. This simulation configuration is called the Coded CCA, and is shown in Fig. 4.7. The purpose of these simulations was to measure %MSE vs. data rate as a function of (m,n) pairs when CR is determined by the entropy coding of the feature map. The entropy encoding technique used is that which was described in Fig. 4.5, with the feature map representation of SS_L as described in Chapter IV. The simpler feature map representation of S_L could also be used with only slightly decreased performance. Entropy encoding basically reduces the spatial data rate without increasing the %MSE. The reduction in spatial data rate for the feature map representations S_L and SS_L are plotted in Fig. 4.6. The total coded data rate R^c_{tot} is defined in (4.20). All overhead data rate costs have been included in the entropy coding results.

The Coded CCA simulations were conducted in the same manner as the Uncoded CCA simulations, resulting in the performance curves shown in Fig. 6.2. The spatial rate reduction, or CR due to entropy coding was shown in Fig. 4.6 to be greater for larger n. Thus the performance curves for the Coded CCA will shift more to the left of the Uncoded CCA curves when n is larger. This effect is observed by comparing the Coded CCA curves in Fig. 6.2 with the Uncoded CCA curves in Fig. 6.1. The

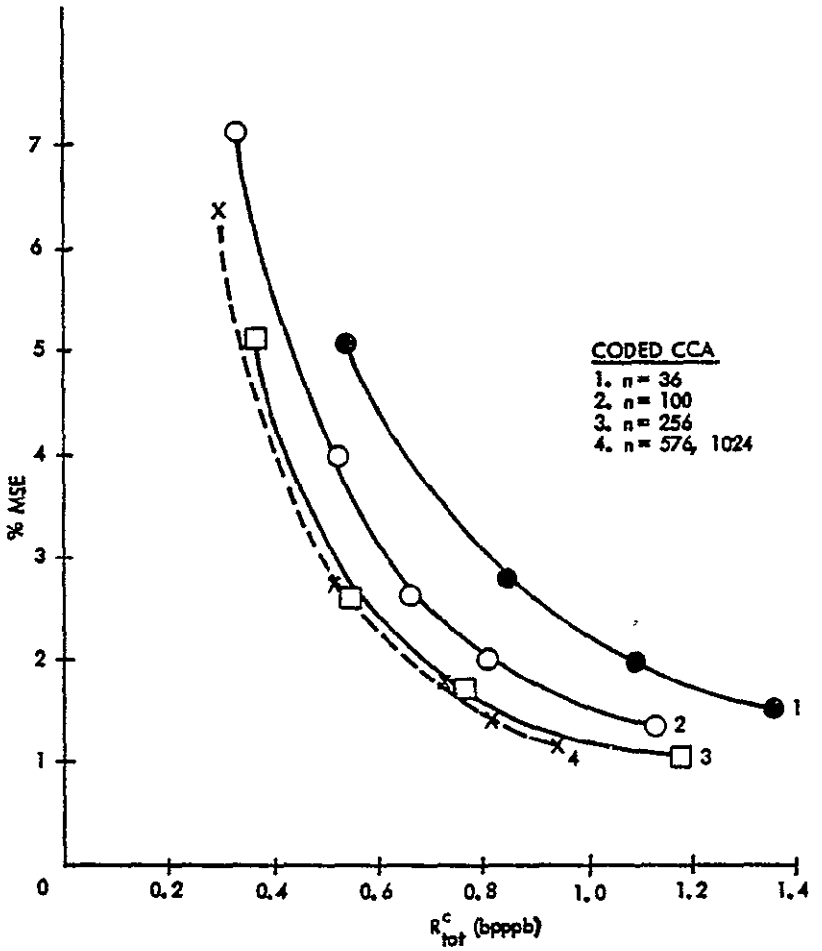


Fig. 6.2. %MSE vs. R^c_{tot} for Various Values of n for the Coded CCA.

Coded CCA continues to increase in performance as n increases, but it is near peak performance for n = 256.

3. Adaptive CCA

In the next simulations, the Adaptive Clustering Algorithm (ACA) which was discussed in Chapter II was used to adaptively determine the number of clusters to use for each sequence of n picture elements.

The Adaptive CCA configuration is the same as the Coded CCA in Fig. 4.7, except the clustering is adaptive. The total data rate $R^{c(A)}_{tot}$

is defined in (5.3). The advantage of the Adaptive CCA is that a lower average value of m can typically be used to obtain similar data quality. The performance and rate of the Adaptive CCA are now dependent on the criteria for increasing or decreasing the number of clusters, as well as the combination of (n, m, CR). The performance of the Adaptive CCA was observed to be less dependent on the size of n, with all results for n of 100 or greater nearly on the same performance curve. Although $R_{tot}^{c}(A)$ was sensitive to changes in the criteria for determining the number of clusters used per sequence, the %MSE vs. rate performance was very insensitive to changes in that criteria. There is a significant performance gain in the Adaptive CCA relative to the Coded and Uncoded CCA. This gain is shown in curves 3, 4, and 5 of Fig. 6.3. However, the performance gain due to adaptability is dependent on both the image data and user application.

4. Performance Comparisons

Tests were performed to compare the %MSE performance of the CCA with some other compression techniques which are well known. The two techniques used for comparison are the well-known Hadamard and Fourier transform techniques. A discussion of Hadamard and Fourier transforms in image coding is given in [6] through [8]. Both techniques were used in an adaptive form, and the compression was done on each band of the test image independently. Some improvement in performance could be obtained by modifying the technique to work jointly on all four bands. The Hadamard transform technique involved taking the two dimensional

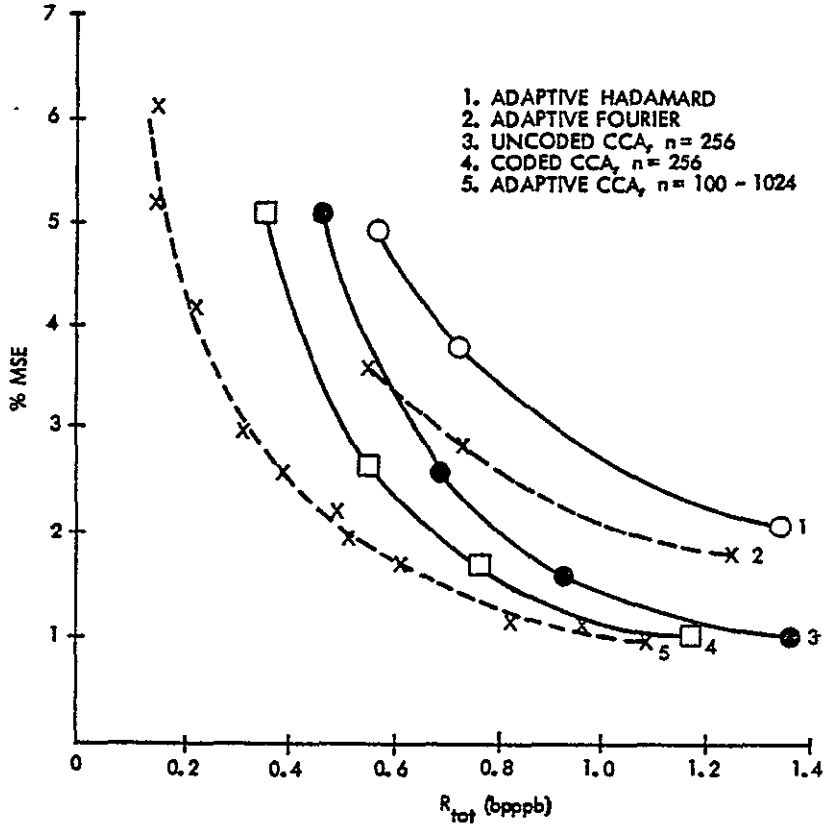


Fig. 6.3 %MSE vs. R_{tot} for Various CCA Configurations and for Adaptive Hadamard and Fourier Techniques.

Hadamard transform of all 8 × 8 subsets of the image. Then the 64 coefficients were placed in 9 zones, each of which had the quantization and corresponding bit rate per coefficient based on the variance, or energy of the coefficients in the zone. The Fourier transform technique was similarly adaptive, but in addition a symmetrical transform approach was used which doubly folded each data subset to provide horizontal and vertical symmetry. This symmetry reduces the intensity discontinuities at the boundaries due to the low pass filtering. The %MSE vs. total data rate for these two adaptive transform techniques is shown in curves 1 and 2 of Fig. 6.3 along with the CCA performance curves.

In Fig. 6.4 a summary of the above simulation results is given by plotting R_{tot} required for 2% MSE vs. n for the various compression configurations. The data points plotted in Fig. 6.4 were interpreted from the smooth curves in the previous figures, as well as from other simulation results. These curves point out a substantial %MSE penalty for n<100 and that a good value for n would be 256. They also show a performance gain of about 20 percent due to entropy coding, and about another 20 percent due to being adaptive. The %MSE vs. R_{tot} performance results agree well with the subjective measure of image appearance for

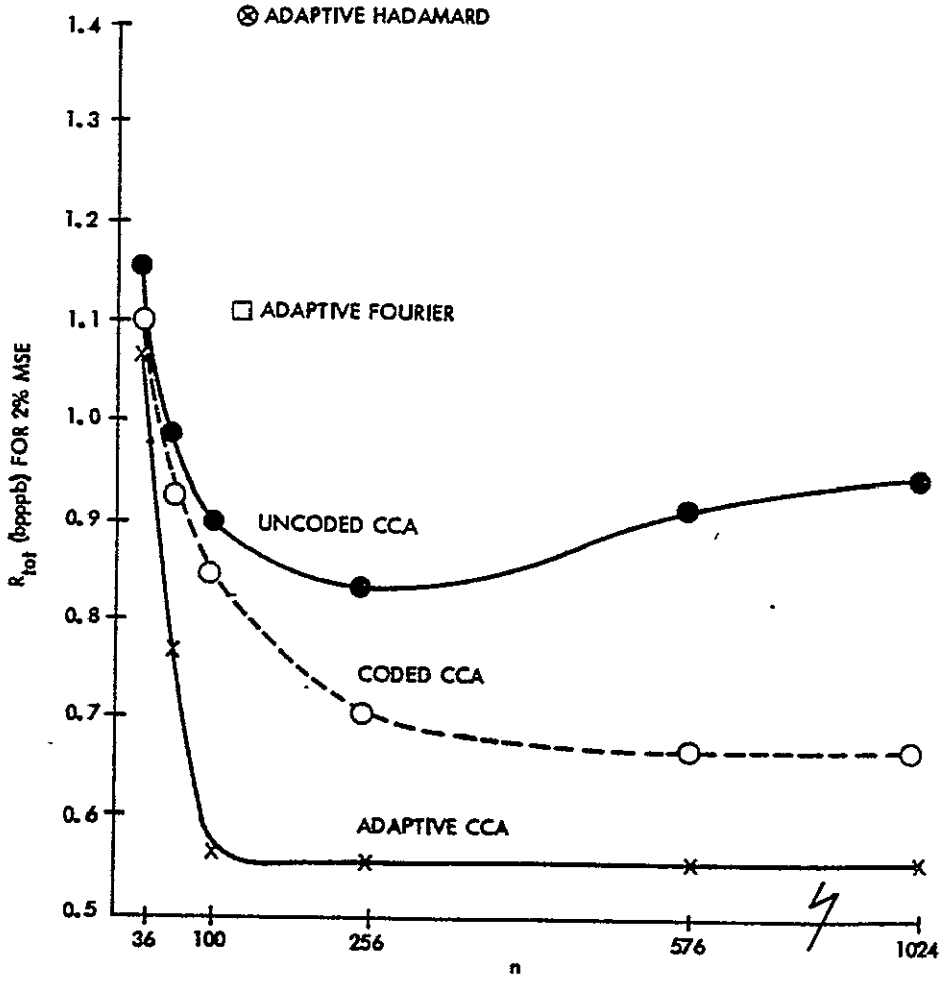


Fig. 6.4. R_{tot} for 2% MSE vs. n for Various CCA Configurations and for Adaptive Hadamard and Fourier Techniques.

the three CCA forms simulated above. This would be expected since the Uncoded, Coded and Adaptive CCA all introduce the same type of image degradation. In particular the coding used in the Coded CCA changes the spatial data rate, but does not change the image quality.

5. Cascaded CCA and Feature Map Subsampling

A simple form of the Cascaded CCA was defined in Chapter V, and a diagram is shown in Fig. 5.1. Points A and B in Fig. 6.5 represent performance results from two simulations of the Cascaded CCA on the test image of Fig. 6.8(a). Point A results from clustering a sequence of 16 16 × 16 local sources into 8 clusters each, and then clustering the 128 local clusters into 16 clusters per sequence and an average of 5.44 subset clusters per 16 × 16 local source. Point B results from clustering a sequence of 16 16 × 16 local sources into 4 clusters each, and then clustering the 64 local clusters into 16 clusters per sequence and an average of 3.37 subset clusters per 16 × 16 local source. $R_{tot}^{c}(casc)$ is determined from (5.8) with CR=1 to account for not using entropy encoding in the following comparisons. Curve 2 is a performance curve for the Cascaded CCA (Uncoded) implied by the simulation results of points A and B. Curve 1 is the performance curve for the Uncoded CCA with n=256 which was presented previously in Fig. 6.1. The Cascaded CCA is compared with the Uncoded CCA such that both CCA configurations are uncoded and nonadaptive. These limited simulations showed the uncoded Cascaded CCA as having a %MSE performance somewhat less than the Uncoded CCA. This unexpected result might be due to the crude intercluster distance measure used for this specific cascaded clustering, or perhaps due to a poor choice of user supervision parameters.

For point A, for example, the spectral data definition requires a data rate of only .047 bpppb out of the total data rate of .665 bpppb. Further investigation and simulations of the Cascaded CCA are needed to adequately define its performance characteristics.

At this time the results of a simulation are presented which incorporates the use of feature map subsampling in the above Cascaded CCA examples. Feature map subsampling was discussed in Chapter IV as a simple means of reducing the spatial data rate by subsampling feature map elements for transmission, and then using linear interpolation for approximate reconstruction at the destination. The above Cascaded CCA examples are used to demonstrate this alternate feature map data reduction, because of the very high ratio of spatial data definition to spectral data definition. The spectral data definition is unaffected by the feature map subsampling. Curve 3 is a performance curve drawn between simulation results obtained from augmenting the Cascaded CCA simulation of point A with subsampling by 2, and subsampling by 4. Performance Curve 4 is obtained by similarly augmenting the Cascaded CCA simulation of point B. The total data rate in each case is obtained directly from the cascaded total data rate by simply reducing the spatial data rate by the subsampling factor. Observe that the slopes of these performance curves are more favorable, even though subsampling is a very crude means of data reduction. This result appears to indicate the importance of properly distributing the total data rate between the spectral and spatial portions. The meaningfulness of using %MSE as a data quality measure is less clear in the example of Fig. 6.5, because of the different types of data degradations being compared. However,

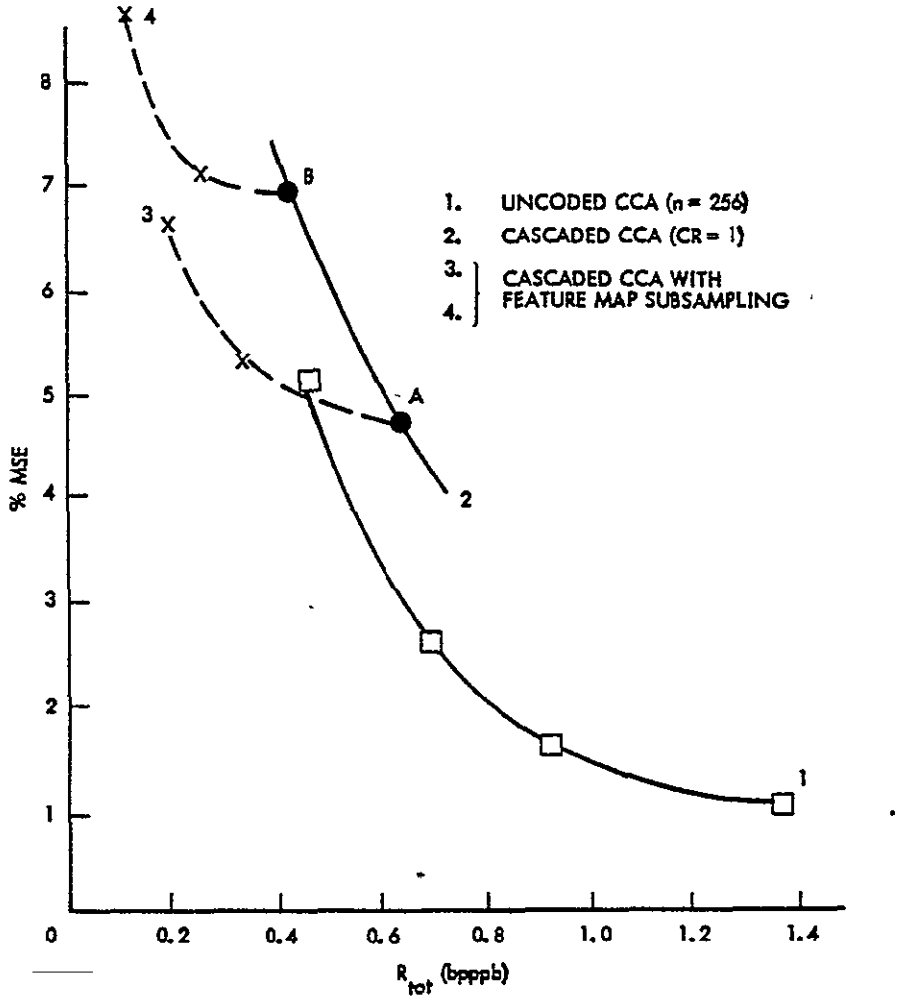


Fig. 6.5. %MSE vs. R_{tot} for the Cascaded CCA With CR = 1 and With Feature Map Subsampling.

%MSE appears to provide a meaningful performance criteria for comparing similar data degradations resulting from the various CCA simulations shown in Figs. 6.1 through 6.4.

6. Convergence Rate of Iterative Clustering

The key element of the clustering approaches used in this work is the iterative clustering which makes up the Basic Clustering Algorithm (BCA) shown in Fig. 2.3. Each iteration of the BCA consists

of an assignment of all the vectors to the nearest mode center, followed by the recalculation of the mode center as the average of all vectors assigned to it. Convergence of the BCA occurs when none of the vectors change assignment, since then the mode centers also do not change. In the above %MSE performance simulations the clustering was allowed to iterate until convergence. However, since the clustering computation is linearly related to the number of iterations, it is necessary to investigate the relationship between clustering performance and the number of iterations.

Figure 6.6 shows the average number of iterations required for convergence as a function of m and n from simulations on the test image of Fig. 6.8(a). As expected, Fig. 6.6 shows the number of iterations increase with increasing m and n. Of course the number of iterations

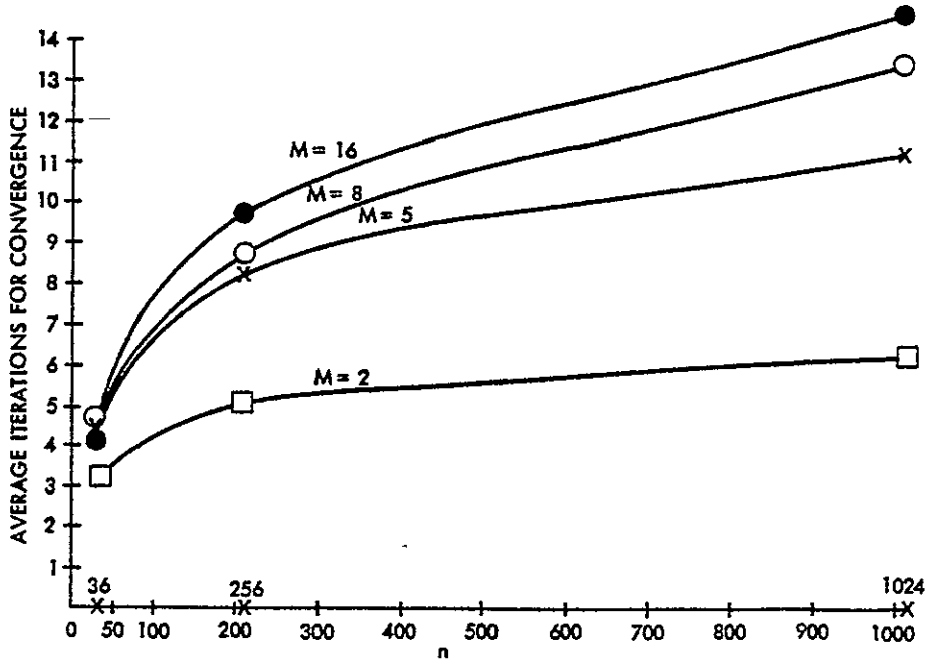


Fig. 6.6. Average Number of Iterations Required for BCA Convergence as a Function of n.

for convergence will fluctuate about the average dependent on the specific data set. Since it is much more difficult to design hardware to handle a varying number of clustering iterations, it is desirable to constrain the number of iterations to some maximum number. Intuitively one would anticipate that many of the last iterations before convergence are just reassigning a small number of vectors which are located nearly equidistant from more then one mode center. The final assignment of such vectors will likely have little impact on the clustering performance. Simulations were conducted to determine the impact on %MSE performance due to constraining the number of clustering iterations. The results of these simulations are shown in Fig. 6.7 where the %MSE is plotted as a function of the maximum number of iterations allowed for various values of m and for n=256. These results indicate that with respect to %MSE performance the number of cluster iterations could be limited to a maximum of about 4 iterations with little performance loss. This is a significant result in terms of practical implementation considerations.

7. Distance Measure Impact on Performance

The primary function of the Basic Clustering Algorithm (BCA) is the assignment of vectors to the nearest mode center. As discussed in Chapter II, there are many different measures of the distance between a vector and a cluster. A more general measure might use the sample covariance of the cluster in addition to the sample mean to account for a possibly nonsymmetric distribution [24], [25]. In order to simplify implementation, the BCA was assumed to use either Euclidean distance

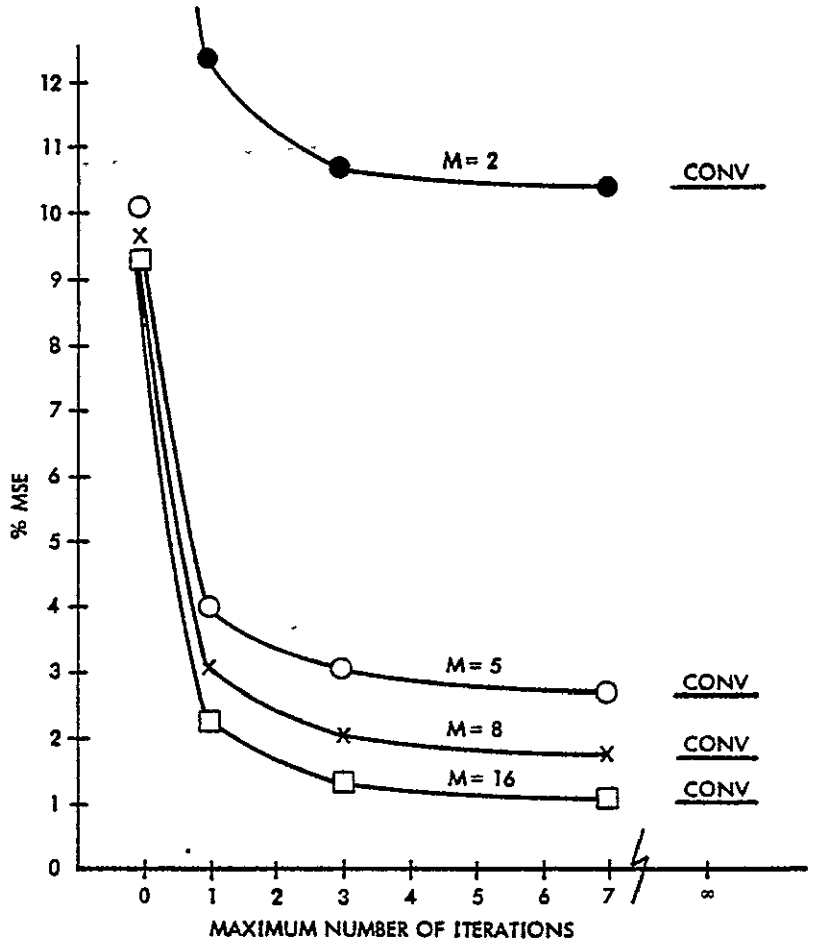


Fig. 6.7. %MSE vs. Maximum Number of Iterations Allowed in BCA for Various Values of m and n = 256.

measure as in (2.1), or the absolute value distance measure as in (2.2). The absolute value is a very crude distance measure compared to the Euclidean distance in higher dimensional space (e.g., four dimensional spectral intensity space). Thus an indication of the sensitivity of the clustering performance to the choice of distance measure can be obtained by comparing results from the use of these two distance measures.

Table VI compares the simulation results from using Euclidean and absolute value distance measures in the Coded CCA. The simulations for Table VI used 16 × 16 element local sources of the four dimensional

TABLE VI
COMPARISON OF EUCLIDEAN AND ABSOLUTE VALUE DISTANCE MEASURES

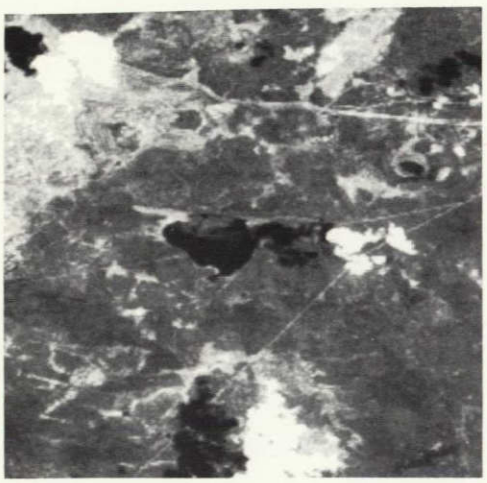
Coded CCA
n = 256
d = 4
f = 24

	Euclidean		Absolute Value	
m	R^c_{tot}	%MSE	R^c_{tot}	%MSE
2	.236	10.43	.238	10.45
5	.562	2.63	.567	2.70
8	.777	1.69	.779	1.77
16	1.19	1.036	1.19	1.134

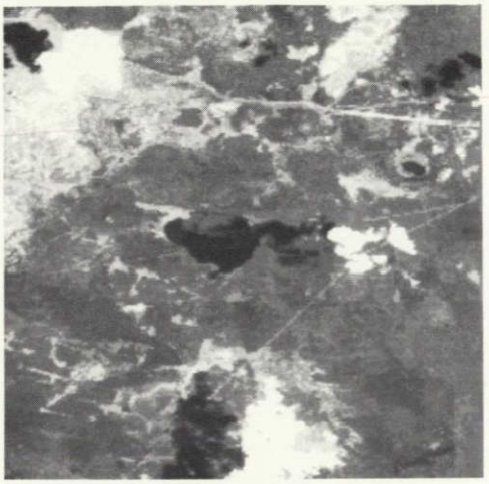
test image in Fig. 6.8(a). Simulations were conducted to obtain 2, 5, 8, and 16 clusters per local source, and each cluster centroid was defined by 24 bits. The results show that the absolute value distance measure degrades the %MSE performance by about .02 %MSE at low total data rates, and the degradation increases to about .1 %MSE at high total data rates. One would expect the quality of the distance measure to be more important at higher total data rates, since the increased number of clusters make the assignment function more dependent on the distance measure. However, even for the higher data rates, the %MSE degradation due to using the absolute value distance measure is very small. This indicates that the clustering performance is quite insensitive to the distance measure used in the assignment process. The results in Table VI are also typical of results obtained from simulations with other CCA configurations and other test images.

B. Subjective Image Appearance Performance

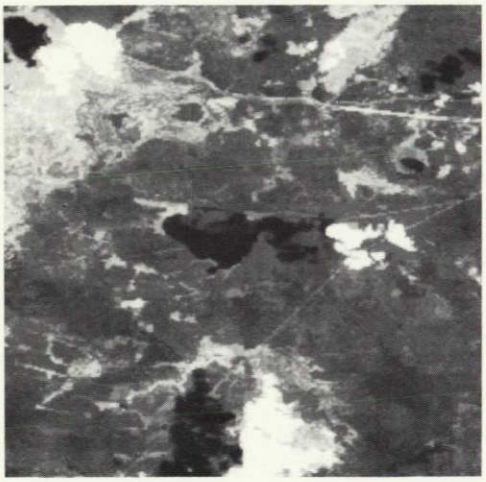
Figures 6.8 through 6.13 provide examples of reconstructed images from simulations of the various CCA forms. The image reconstructions



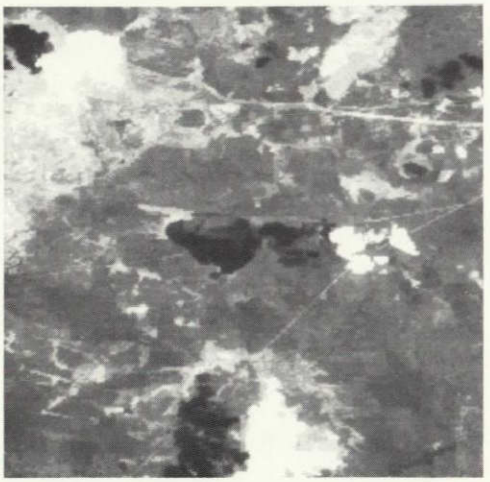
(a) 6 bpppb



(b) .937 (.776) bpppb



(c) .88 (.78) bpppb



(d) .917 (.885) bpppb

(a) Original test image of 256 x 256 picture elements quantized to 6 bpppb. A subset of a Landsat 1 image of the Verde Valley, Arizona; band 2 selected for display from a total of 4 bands.

(b) 16 x 16 local sources with 8 clusters each, n=256, m=8, f=24, d=4;
R_{spec} = .187, R_{spat} = .75(.589), R_{tot} = .937(.776) bpppb.

(c) 10 x 10 local sources with 5 clusters each, n=100, m=5, f=24, d=4;
R_{spec} = .3, R_{spat} = .58(.48), R_{tot} = .88(.78) bpppb.

(d) 6 x 6 local sources with 3 clusters each, n=36, m=3, f=24, d=4;
R_{spec} = .5, R_{spat} = .417(.385), R_{tot} = .917(.885) bpppb.

Fig. 6.8. Examples of the Uncoded (Coded) CCA for Different Values of m and n.

(a) 1.03 (.921) bpppb

(b) .74 (.67) bpppb

(c) .597 (.516) bpppb

(d) .37 (.327) bpppb

(a) 6 clusters per local source,
n=100, m=6, f=24, d=4;
R_{spec} = .36, R_{spat} = .667(.561), R_{tot} = 1.03(.921) bpppb.

(b) 4 clusters per local source,
n=100, m=4, f=24, d=4;
R_{spec} = .24, R_{spat} = .5(.43), R_{tot} = .74(.67) bpppb.

(c) 3 clusters per local source,
n=100, m=3, f=24, d=4;
R_{spec} = .18, R_{spat} = .417(.336), R_{tot} = .597(.516) bpppb.

(d) 2 clusters per local source,
n=100, m=2, f=24, d=4;
R_{spec} = .12, R_{spat} = .25(.207), R_{tot} = .37(.327) bpppb.

Fig. 6.9 Examples of the Uncoded (Coded) CCA for a varying number of clusters in a constant local source size of 10 × 10 elements.

(a) 8 bpppb

(b) 1.18 (1.04) bpppb

(c) 1.18 (1.04) bpppb

(d) 1.18 (1.04) bpppb

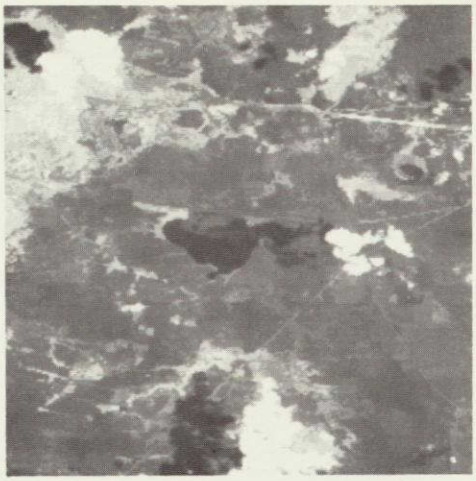
(a) Original girl test image of 256 x 256 picture elements quantized to 8 bpppb. Red band selected for display from a total of 3 bands.

(b) Euclidean distance measure, n=100, m=5, f=24, d=3; R_{spec} = .4, R_{spat} = .777(.639), R_{tot} = 1.18(1.04) bpppb.

(c) Absolute valued distance measure, n=100, m=5, f=24, d=3; R_{spec} = .4, R_{spat} = .777(.639), R_{tot} = 1.18(1.04) bpppb.

(d) Maximum of 4 clustering iterations, n=100, m=5, f=24, d=3; R_{spec} = .4, R_{spat} = .777(.641), R_{tot} = 1.18(1.04) bpppb.

Fig. 6.10. Examples of the Uncoded (Coded) CCA with 10 ×10 element local sources and 5 clusters each, but for different distance measures and for a limitation on the number of clustering iterations.



(a) .735 (.623) bpppb

(b) 1.16 (1.05) bpppb

(c) .896 (.686) bpppb

(d) .536 bpppb

(a) 10 x 10 local sources with an average of 4.0 clusters each, n=100, average m=4.0, f=24, d=4;
R_{spec} = .249, R_{spat} = .486(.374), R_{tot} =.735(.623) bpppb.

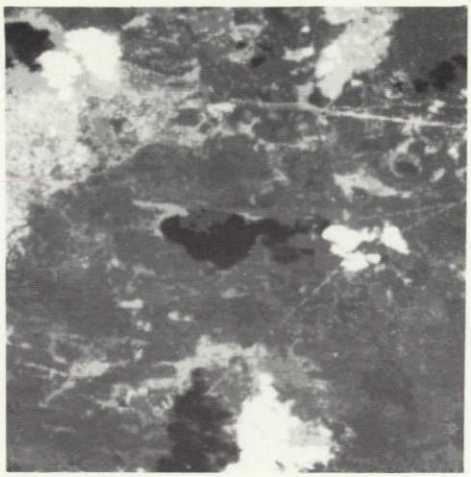
(b) 10 x 10 local sources with an average of 5.6 clusters each, n=100, average m=5.6, f=24, d=3;
R_{spec} = .46, R_{spat} = .70(.59), R_{tot} = 1.16(1.05) bpppb.

(c) 16 x 16 local sources with an average of 5.47 clusters each, n=256, average m=5.47, f=24, d=3;
R_{spec} = .176, R_{spat} = .72(.51), R_{tot} = .896(.686) bpppb.

(d) The same image as in (c), except the feature map is subsampled by 2 and reconstructed through linear interpolation. Entropy coding is not used.
R_{spec} = .176, R_{spat} = .36, R_{tot} = .536 bpppb.

Fig. 6.11. Examples of the Adaptive CCA, including variations of no coding, entropy coding, and subsampling of the feature map.

(a) .664 (.50) bpppb



(b) .356 bpppb

(c) .583 (.403) bpppb

(d) .183 bpppb

(a) 16 x 16 local sources are initially clustered to 8 clusters each. The Cascaded clustering combines 128 local clusters from 16 local sources into 16 sequence clusters, and an average of 5.44 sequence clusters are used per local source.
$n=256$, $m_\ell=8$, $f=24$, $d=4$, $N_s=16$, $m_s=16$, $\bar{m}_s=5.44$;
$R_{spec}(seq)=.023$, $R_{spec}(loc)=.024$, $R_{spat}=.617(.453)$, $R_{tot}=.664(.50)$ bpppb.

(b) The same image as (a), except the feature map is subsampled by 2.
$R_{spec}(seq)=.023$, $R_{spec}(loc)=.024$, $R_{spat}=.309$, $R_{tot}=.356$ bpppb.

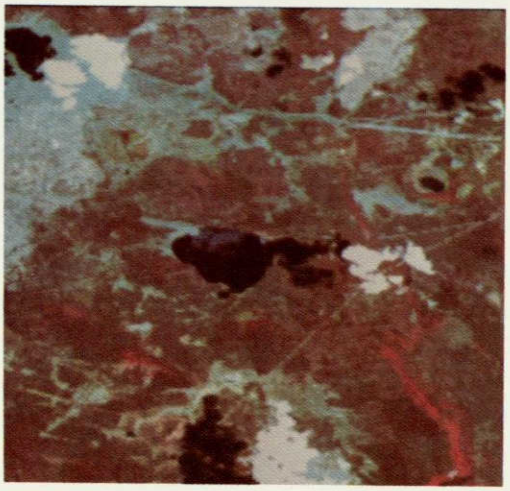
(c) 16 x 16 local sources are initially clustered to 4 clusters each. The cascaded clustering combines the 64 local clusters from 16 local sources into 16 sequence clusters, and an average of 3.16 sequence clusters are used per local source.
$n=256$, $m_\ell=4$, $f=24$, $d=3$, $N_s=16$, $m_s=16$, $\bar{m}_s=3.16$;
$R_{spec}(seq)=.031$, $R_{spec}(loc)=.019$, $R_{spat}=.533(.403)$, $R_{tot}=.583(.403)$ bpppb.

(d) The same image as (c), except the feature map is subsampled by 4.
$R_{spec}(seq)=.031$, $R_{spec}(loc)=.019$, $R_{spat}=.133$, $R_{tot}=.183$ bpppb.

Fig. 6.12. Examples of the Cascaded CCA, including variations of no coding, entropy coding, and subsampling of the feature map.

(a) 6 bpppb

(b) 8 bpppb



(c) .937 (.776) bpppb

(d) 1.25 (1.01) bpppb

REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR

(a) False color composite of Verde Valley test image. First three multispectral scanner bands selected for display in blue, green, and red respectively.

(b) Color composite of girl test image.

(c) 16 x 16 local sources with 8 clusters each, n=256, m=8, f=24, d=4; R_{spec} = .187, R_{spat} = .75(.589), R_{tot} = .937(.776) bpppb.

(d) 16 x 16 local sources with 8 clusters each, n=256, m=8, f=24, d=3; R_{spec} = .25, R_{spat} = 1.0(.764), R_{tot} = 1.25(1.01) bpppb.

Fig. 6.13. Examples of color composites for the Uncoded (Coded) CCA.

consisted simply of substituting for each picture element the centroid value of the cluster to which the picture element was assigned. Other cluster parameters such as the variance per band could also be useful for image reconstruction in special cases. In all the CCA simulations, all the bands of the test image are jointly processed, and the band with the highest signal energy is selected for display in black and white. Two different types of image data shown in Fig. 6.8(a) and Fig. 6.10(a) were selected to demonstrate typical performance in terms of subjective image appearance. One example of color composites of these test images are also included in Figs. 6.13(a) and (b). Simulations were conducted on many other test images, and the CCA performance was found to be very similar for wide variations in the type of test data.

Figures 6.8(b) through (d) demonstrate reconstructed images which have similar total data rates, but widely varying local source sizes and number of clusters per local source. Note that the use of entropy coding on the feature map lowers the spatial data rate, but does not cause any changes to the reconstructed image. Therefore, the Uncoded CCA and the Coded CCA have the same resulting reconstructed image, but different total data rates. Throughout the image examples, the total data rate corresponding to use of entropy coding on the feature map is given in parentheses. Figure 6.8 indicates considerable improvement in using local sources larger than 6 × 6 elements, but the differences between local sources larger than 10 × 10 elements appear small. In addition, the result for a 6 × 6 element local source does not benefit as well from entropy coding.

example in Fig. 6.11(c). The feature map is subsampled by a factor of 2, and then reconstructed by using linear interpolation. Thus the spatial data rate is reduced by a factor of 2, and the image appearance degrades mainly due to smoothing of sharp edges in the image.

Images resulting from simulations of the Cascaded CCA are shown in Fig. 6.12. Figure 6.12(a) resulted from clustering 16 16 × 16 local sources into 8 clusters each, and then clustering the 128 local clusters into 16 sequence clusters. The average number of sequence clusters used per local source is 5.44. In Fig. 6.12(c), a sequence of 16 16 × 16 local sources are clustered to 4 clusters each, and then the 64 local clusters are clustered into 16 sequence clusters. For this image the average number of sequence clusters used per local source is 3.16. In both Fig. 6.12(a) and (c) the spectral data rates are very low. This results in heavy contouring within the image which significantly decreases image appearance performance. However, this CCA configuration is more so intended for automatic interpretation applications. In such applications it is often desirable to sharply define boundaries which separate a relatively limited number of classes with different spectral signatures, but it is not important to preserve the intraclass scatter information within the boundaries. Furthermore, if the spatial boundary information is not important, the spectral signature information could be obtained for only the very low spectral data rates. The loss of intraclass scatter information is roughly equivalent to contouring within the image. Generally, contouring degradation of the data

impacts image appearance performance more so than it would impact automatic interpretation performance. However, the image appearance performance is a subjective measure which greatly depends on the specific use of the image data.

Figure 6.12, (b) and (d), represent the use of subsampling on the feature maps from the Cascaded CCA results of Fig. 6.12, (a) and (c), respectively. In Fig. 6.12(b) the feature map was subsampled by a factor of 2 and the spatial data rate reduced by a factor of 2. In Fig. 6.12(d) the feature map was subsampled by a factor of 4 to reduce the spatial data rate by a factor of 4. The impact of feature map subsampling is mainly the poorer definition of sharp transitions in the image. The overall appearance quality of these two images is quite low. However, the total data rate is also quite low. For the image in Fig. 6.12(d), the total data rate is reduced by a factor of about 44.

In Fig. 6.13(a) and (b) are shown color composites of the two test images. Fig. 6.13(a) is a false color composite which displays the first three bands of the Landsat I multispectral scanner image in blue, green, and red respectively. Figures 6.13(c) and (d) show reconstructed color composites from simulations of the Uncoded and Coded CCA. Each simulation uses 16 × 16 local sources with 8 clusters each. The image appearance performance of the CCA is usually better when judged in terms of color composites instead of black and white images. This apparently results because the clustering algorithms are attempting to obtain a clustering representation which minimizes the average error in a multidimensional sense, rather than trying to minimize the average error in each band of the image independently.

C. Classification Performance

Classification performance is specified in terms of the classification accuracy for a given data rate. Classification accuracy is dependent on both the classification technique and the data compression technique. Furthermore, it is also possible that the performance with a given classification technique may be significantly improved through adaptations which account for the changes in the data caused by a given data compression technique. There are of course many different classification approaches (e.g., parametric, nonparametric, supervised, nonsupervised). Similarly the classification might be on a per vector basis, or alternately include texture. A very common classification approach in Landsat applications as well as elsewhere is the Gaussian parametric classifier on a per vector basis. This classifier is often used because of its relatively simple implementation. In this dissertation, the Gaussian parametric classifier is selected for investigating the classification performance of the CCA. In addition, this example demonstrates the interaction between data classification and data compression.

1. The Gaussian Parametric Classifier

A model for statistical classification is shown in Figure 6.14. Let $\underline{X}$ be the vector in measurement space and assume there are q classes $\{w_j\}$, $j = 1, 2 \ldots, q$ in pattern space each characterized by a conditional density function $p(\underline{X}/w_j)$, and a prior probability $P(w_j)$. If the classifier $G(\cdot)$ is picked to minimize probability of error, then the solution for $G(\cdot)$ is to maximize the a posteriori density function, or equivalently to choose

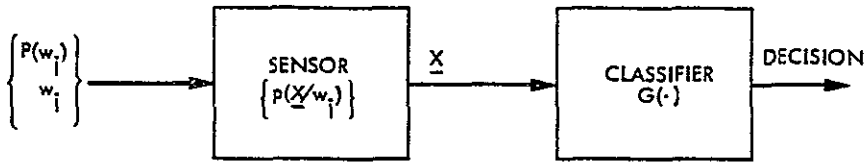


Fig. 6.14. Statistical Classification Model.

$$\underline{X} \varepsilon w_j \text{ iff } p(\underline{X}/w_j) \geq p(\underline{X}/w_k)\ P(w_k)\ \forall\ k \neq j^*. \tag{6.4}$$

The classifier defined in (6.4) is commonly referred to as the Maximum Likelihood Classifier. The classifier solution for minimizing a more generalized average cost criteria is given in [12, p. 9].

The classifier in (6.4) is in terms of unknown density functions which need to be estimated. In parametric classification a functional form for the density functions is assumed, and only a set of parameters need to be estimated in order to completely define the density functions. A functional form which results in a classifier that is relatively easy to implement is the Multivariate Gaussian, or Normal distribution. For this parametric assumption, each conditional density function is entirely defined by a mean vector and a covariance matrix. Assume the measurement vector $\underline{X}$ which is generated by class w_j is normally distributed with mean $\underline{U}_j$ and covariance $[\Phi_j]$. Also, assume the covariance matrix has an inverse $[\Phi_j]^{-1}$. Then the d-variate conditional distribution function of $\underline{X}$ is

$$p(\underline{X}/w_j) = \frac{1}{(2\pi)^{d/2}|\Phi_j|^{1/2}} \exp\left\{-\frac{1}{2}(\underline{X} - \underline{U}_j)^t [\Phi_j]^{-1}(\underline{X} - \underline{U}_j)\right\}, \tag{6.5}$$

where $|\Phi_j|$ is the determinant of $[\Phi_j]$. Note that the classifier in (6.4) is unaffected by taking the log of each side of the inequality

*Ties may be arbitrarily decided.

since the log is a monotonic function. Define a set of discriminant functions $g_j(\underline{X})$ by

$$g_j(\underline{X}) = \log \{p(\underline{X}/w_j)\ P(w_j)\} \quad . \tag{6.6}$$

Then from (6.4) the classifier is also defined by

$$\underline{X} \epsilon w_j \quad \text{iff} \quad g_j(\underline{X}) \geq g_k(\underline{X}) \ \forall\ k \neq j \quad . \tag{6.7}$$

By using the parametric assumption of (6.5), the discriminant function of (6.6) becomes

$$g_j(\underline{X}) = -\frac{d}{2} \log 2\pi - \frac{1}{2} \log |\Phi_j| - \frac{1}{2}(\underline{X} - \underline{U}_j)^t\ [\Phi_j]^{-1}\ (\underline{X} - \underline{U}_j) + \log P(w_j) \quad . \tag{6.8}$$

Now assume the a priori class probabilities are equally likely, or

$$P(w_1) = P(w_2) = \ . \ . \ . = P(w_q) = \frac{1}{q} \quad . \tag{6.9}$$

Since constants can be removed from both sides of (6.7), the discriminant function in (6.8) can be reduced to

$$g_j(\underline{X}) = -\frac{1}{2} \log |\Phi_j| - \frac{1}{2}(\underline{X} - \underline{U}_j)^t\ [\Phi]^{-1}\ (\underline{X} - \underline{U}_j) \quad . \tag{6.10}$$

Thus, the Gaussian parametric classifier is defined by (6.7) and (6.10), where only the mean $\underline{U}_j$ and covariance $[\Phi_j]$ of the conditional distribution function in (6.5) must be estimated.

The mean and covariance of the measurement vector $\underline{X}$ conditioned on class w_j must usually be estimated from samples of $\underline{X}$ which are independently known to have resulted from class w_j. Such a set of measurement vectors are referred to as a training set for class w_j, and classification methods which require such training are referred to as supervised classification methods. Training a classifier can

sometimes be a difficult task, and at the same time the classification accuracy is very dependent on accurate training. Similarly, the use of data compression can greatly impact classification accuracy by affecting the training of the classifier.

The following is an example of typical training of the Gaussian parametric classifier for Landsat applications. For Landsat w_j^* might correspond to a specific earth resource class with a well defined spectral reflectance. However, the class w_j^* must be distinguished from other classes based on the measurement $\underline{X}$ from the multispectral scanner (MSS) instrument. If the distribution of the MSS instrument output for class w_j^* can be reasonably approximated by the normal distribution in (6.5), then determining the discriminant function in (6.10) for w_j^* reduces to estimating the mean and covariance of $\underline{X}$ resulting from w_j^*. In the Landsat example, samples of $\underline{X}$ for w_j^* are obtained by identifying subsets within the image data which are known to represent w_j^* via independent ground truth information. Typically, a total training set for a given class consists of a small number of subsets (e.g., 4) of about 100 picture elements each, which are selected from various locations throughout the image. However, obtaining good ground truth information and registering the MSS image elements with the ground truth is often costly and time consuming. Furthermore, the measurement of spectral radiance $\underline{X}$ for a given class w_j^* is also a function of sun angle, ground slope, ground moisture, atmospheric absorbtion, extraneous radiation, MSS instrument noise, etc. Thus, the mean and covariance of $\underline{X}$ obtained from a training set for class w_j^* is also a function of these unpredictable and time varying disturbance effects. Therefore, it is generally necessary to

repeatably retrain the classifier due to changes in time or location of the image data.

Assume that a total of T_j training set vectors have been obtained from an image for class w_j. Let u_m^j be the m-th element of $\underline{U}_j$, and let ϕ_{mn}^j be the element in the m-th row and n-th column of $[\Phi_j]$. Also, let x_{mk}^j be the m-th element of the k-th measurement vector in the training set for class w_j. Then the usual estimators for $\underline{U}_j$ and $[\Phi_j]$ are given by

$$u_m^j = \frac{1}{T_j} \sum_{k=1}^{T_j} x_{mk}^j \tag{6.11}$$

and

$$\phi_{mn}^j = \frac{1}{T_j} \sum_{k=1}^{T_j} \left(x_{mk}^j - u_m^j \right) \left(x_{nk}^j - u_n^j \right) , \tag{6.12}$$

with

$j = 1, 2, \ldots, q$

$m = 1, 2, \ldots, d$

$n = 1, 2, \ldots, d.$

The classifier used for testing the classification performance of the CCA is the Gaussian parametric classifier defined by (6.7) and (6.10). The mean and covariance parameters for (6.10) are determined from training sets as defined by (6.11) and (6.12).

2. Performance from Training and Classifying Original Data

The following assessment of classification performance for the CCA basically consists of finding the changes in classification accuracy of

the above classifier due to both training and classifying with CCA processed data rather than with original data.

The test image selected for testing classification performance is a 4 spectral band, 256 line by 220 sample subset from the Flight Line C-1 image taken over Indiana on June 28, 1966. The original image consists of 12 spectral bands and 900 lines by 220 samples. The lines selected for the test image are lines 200 through 419. The spectral bands selected for the test image are bands 9, 10, 12, and 1, which correspond respectively to .62-.66μ, .66-.72μ, .80-1.0μ, and .40-.44μ. Band 12 is selected for display in Fig. 6.15(a), and a false color composite of the test image using bands 9, 10, and 12 is shown in Fig. 6.16(a). Further detailed information on this test image can be obtained from [32].

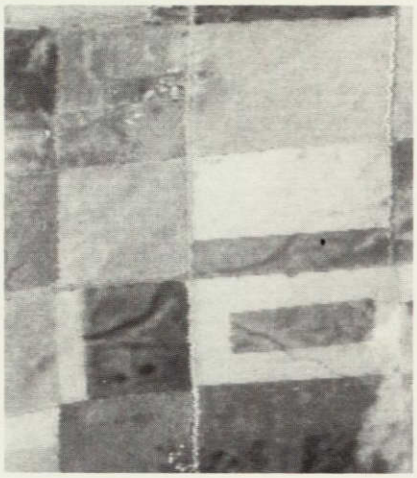
A primary reason for using the Flight Line C-1 data is the availability of reasonably reliable ground truth corresponding to as many as 8 classes. Table VII lists eight classes of interest in the test image and the location and size of training sets for each class. These training sets were carefully selected to avoid suspicious areas within the fields (e.g., apparent ditches). The mean and covariance of the training set samples for each class were determined from (6.11) and (6.12), and all the picture elements of the test image were classified by the Gaussian parametric classifier defined in (6.7) and (6.10). If the maximum discriminant value in (6.10) for a picture element was below -70, the picture element was assumed to belong to none of the 8 classes and was assigned a label of class 9. The results of training

TABLE VII
TRAINING SETS FOR FLIGHT LINE C-1 TEST DATA

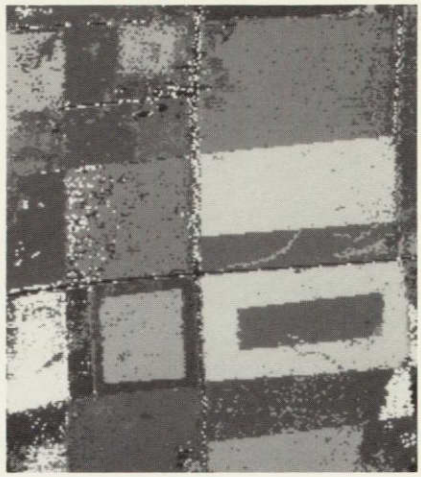
All Training Sets are 10 × 10 Elements

Class Number	Class Name	Starting Line	Starting Sample
1	Soybean I	1	135
2	Soybean II	40	130
		110	60
		240	10
3	Corn I	1	20
		26	40
		120	10
		210	120
4	Corn II	190	10
5	Corn III	180	40
		220	10
6	Red Clover	40	10
		180	70
		240	140
7	Oats	130	130
		170	140
		220	60
8	Wheat	90	190
		100	130
		150	110

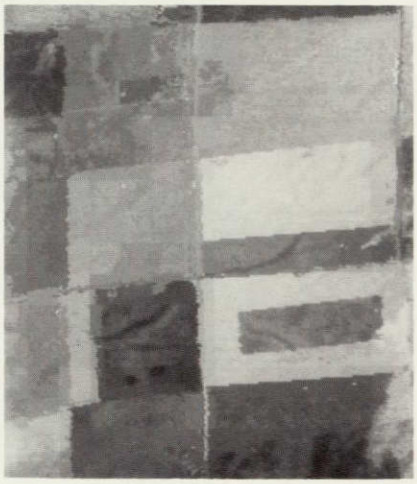
and classifying with the original test image data is shown in Figs. 6.15(b) and 6.16(b), where every picture element is coded to identify its classification to one of 9 classes, and the relationship of class number, grey level, and color are given in Fig. 6.17.



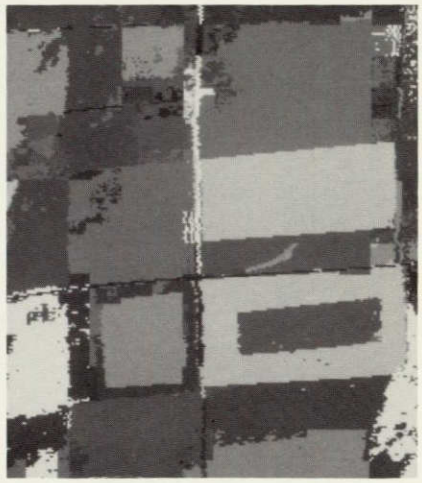
(a) 8 bpppb



(b) 8 bpppb, P_e = 7.8%



(c) .625 (.429) bpppb



(d) .625 (.429) bpppb, P_e = 3.4%

(a) The test image is a 4 band, 256 line by 220 sample subset from the 12 band, 900 line by 220 sample Flight Line C-1 image. The test image consists of bands 9, 10, 12, and 1, and lines 200 through 419. Band 12 is selected for display and the data rate is 8 bpppb.

(b) Grey-level coded classified image obtained from training and classifying on the original test data with the Gaussian parametric classifier. The classification performance is P_e = 7.8% for R_{tot} = 8 bpppb.

(c) Uncoded (coded) CCA compressed image. 16 x 16 local sources with 4 clusters each, n = 256, m = 4, f = 32, d = 4; R_{spec} = .125, R_{spat}=.5(.304), R_{tot}=.625(.429) bpppb.

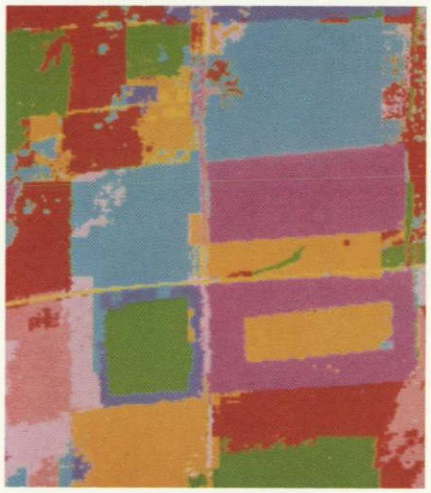
(d) Grey-level coded classified image obtained from training and classifying on the CCA compressed image with the Gaussian parametric classifier (β=2). The classification performance is P_e = 3.4% for R_{tot} = .625 (.429) bpppb.

Fig. 6.15. Original, compressed, and classified images of the Flight Line C-1 test data.

(a) 8 bpppb

(b) 8 bpppb, P_e = 7.8%

(c) .625 (.429) bpppb



(d) .625 (.429) bpppb, P_e = 3.4%

(a) Color composite of original test image with bands 9, 10, and 12 used for blue, green, and red respectively. The data rate is 8 bpppb.

(b) Color coded classified image obtained from training and classifying on the original test data with the Gaussian parametric classifier. The classification performance is P_e = 7.8% for R_{tot} = 8 bpppb

(c) Uncoded (Coded) CCA compressed image. 16 x 16 local sources with 4 clusters each, n = 256, m = 4, f = 32, d = 4; R_{spec} = .125, R_{spat}=.5(.304), R_{tot}=.625(.429) bpppb.

(d) Color coded classified image obtained from training and classifying on the CCA compressed image with the Gaussian parametric classifier (β=2). The classification performance is P_e = 3.4% for R_{tot} = .625 (.429) bpppb.

Fig. 6.16. Color composite original, compressed, and classified images of the Flight Line C-1 test data.

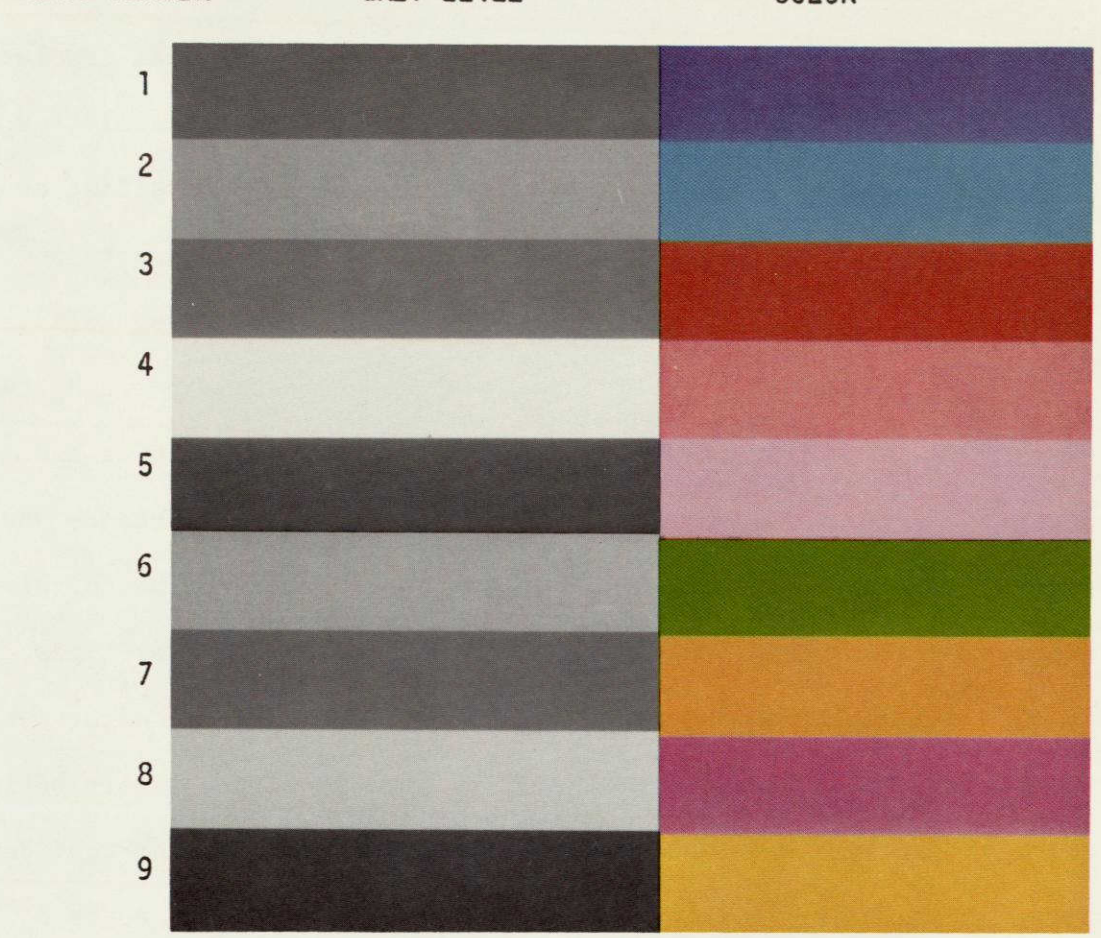


Fig. 6.17. Relationship Between Class Number, Grey-Level, and Color for the Classified Images.

A simple quantitative measure of classifier performance is the percent probability of error P_e in classifying a picture element. An estimate of P_e can be obtained from the ratio of the number of incorrectly classified picture elements to the number of picture elements classified. A test set of picture elements is selected for each class for the purpose of estimating P_e. These test sets basically consist of most of the picture elements for each class minus those elements for which the correct classification is uncertain. Those picture elements near borders of the field, or within regions of apparent ditches and

other anomalies are excluded from the test set. Such picture elements are likely the measured average reflectance over a spatial resolution cell which substantially consists of more than one of the classes (including class 9), and therefore cannot be simply classified as correct or incorrect. The location and size of the test sets for each class are given in Table VIII. The large size of the test sets are chosen to give a higher confidence level to the estimate for P_e, and also to decrease statistical dependence between the training and testing procedure. A large test set is of even greater importance when estimating P_e for classifying compressed data, because the correlation between reconstructed picture elements is often increased. This increased correlation tends to cause error events and no error events to occur in bursts instead of in a more uniform spatial distribution; thereby necessitating a larger test set for the equivalent confidence level in estimating P_e. The classifier accuracy is defined by P_e, where

$$P_e = \frac{100 \times \text{Number of incorrect classifications in test set}}{\text{Number of elements in test set}} . \qquad (6.13)$$

For training and classifying the original test image the classifier accuracy is $P_e = 7.8\%$. The classification performance is given by $P_e = 7.8\%$ for a total data rate of 8 bpppb.

3. Performance from Training on Original Data and Classifying Compressed Data

We now consider the impact on classification accuracy due to first compressing the data with the CCA. Any data compression algorithm can affect the classification accuracy in two ways, 1) by changing the conditional distributions of $\underline{X}$, and 2) by changing the training data

used to estimate the conditional distributions of $\underline{X}$. Consider first the impact of changing the conditional distributions of $\underline{X}$. Usually one thinks of data compression as simply causing a spreading of the conditional distributions, which results in an increased overlap of the distributions and an increased error rate. This is typically the case when the data compression approximates the data based on a criterion not closely related to minimizing the error probability of the classifier. However, better classification performance should be obtainable from a data compressor which specifically provides the classifier with information which aids accurate classification. Clearly, the best obtainable classification accuracy from the original data is also a bound to the classification accuracy which can be obtained from the compressed data. However, the actual classifier is usually not the best classifier due to practical memory and computation limitations in estimating the conditional distribution functions. For example, the Gaussian parametric classifier reduces computation by classifying each vector independently, while it is obvious a more complex classifier using surrounding vector information would have better classification accuracy. When the classifier is suboptimal (a most common occurrence), the use of data compression can in fact result in an improvement in classification accuracy. The accuracy obtained with the original data is no longer a bound to the accuracy obtainable by the same classifier using the compressed data. With the CCA, for example, the data compression modifies each vector based on surrounding vector information, such that the simple vector-by-vector classifier is indirectly making decisions based on more information

than a single vector. Then the classification accuracy may increase, or decrease depending on how much useful new information the data compression provides the classifier, vs. how inaccurate the information is due to the data compression process.

In Fig. 6.18 Curve 1 shows the classification performance of the Uncoded CCA when the Gaussian parametric classifier is able to train on original data. The test image of Fig. 6.15(a) was compressed at three different data rates by using 8, 4, and 2 clusters per each 16×16 element local source. The classifier trained on the original data from the training sets defined in Table VII, and classified the CCA processed data by classifying the mean of each cluster. Each vector was given the class label of the cluster mean for the cluster to which the vector was assigned. The P_e was determined from (6.13) and the test set of Table VIII. The classification performance for the same classifier and the original data is also shown in Fig. 6.18 as the point at $P_e = 7.8\%$ and $R_{tot} = 8$ bpppb. Clearly the classification performance substantially benefits from the grouping information provided by the CCA data compression.

4. Performance from Training and Classifying on Compressed Data

In many situations it may be very difficult for a remote sensor to identify and transmit properly selected subsets of original data from the image for training purposes. Then one must consider the additional classification accuracy degradation which may result because of inaccurate training from the compressed data. Curve 2 in Fig. 6.18 shows the classification performance of the CCA under the same conditions assumed for Curve 1 except the classifier was trained on the

reconstruction from the compressed data. The reconstruction consisted simply of using the appropriate cluster mean to approximate each vector. Curve 2 demonstrates substantial performance degradation relative to Curve 1. The primary reason for this degradation is that the procedure for training with original data is not in general appropriate for training with compressed data. When training with original data it is reasonable to assume that the training sets provide a set of independent and identically distributed samples from which the first and second order statistics can be estimated. However, this assumption is less reasonable when dealing with a compressed representation of the training set, since most data compression introduces increased dependence between vectors in a local region. The conditional distribution of all the samples in the image may be changed very little by the data compression, but the conditional distribution determined by a small training subset may be substantially different.

5. Performance from a Modified Estimate for Training Set Mean and Covariance

There are several approaches to modifying the training procedure to account for the data compression. One could use more training sets, or select the same number of training set vectors from more scattered locations throughout the image. But both of these procedures may considerably complicate an already difficult task of training. Alternately, it may be possible to analytically determine the impact of the data compression on the training for the given classifier. Clearly, a more sophisticated classifier for the CCA compressed data would classify

clusters on the basis of which conditional distribution the cluster was most likely to originate from. However, assume for practical reasons it is desired to restrict the classifier to the more simple Gaussian parametric classifier. Then it is important to obtain a good estimate of the first and second order statistics from the CCA compressed representation of the training sets.

Assume a training set of vectors for class w have been compressed by the CCA. Let the training set have vectors in N different clusters. Let $p_i(\underline{X})$ be a model distribution function for all the vectors in the i-th cluster (whether in the training set, or not), such that the first and second order statistics of $p_i(\underline{X})$ are equal to the sample mean $\underline{U}_i$ and sample covariance $[\Phi_i]$ of the vectors in the i-th cluster, i = 1, 2, . . . , N. Then let P_i define the ratio of the number of vectors from cluster i in the training set to the total number of vectors in the training set. A model distribution $p(\underline{X})$ for the training set can then be expressed as a mixture distribution, or

$$p(\underline{X}) = \sum_{i=1}^{N} P_i \, p_i(\underline{X}) \, . \tag{6.14}$$

The first and second order statistics for $p(\underline{X})$ can be determined in terms of the N cluster means and covariances, and these first and second order statistics for $p(\underline{X})$ can be used as an estimate of the sample mean and sample covariance of the training set.

The mean of $p(\underline{X})$ is given by

$$E(\underline{X}) = \int_{\underline{\lambda}} \underline{\lambda} \, p(\underline{X}) \, d\underline{\lambda} = \sum_{i=1}^{N} P_i \int_{\underline{\lambda}} \underline{\lambda} \, p_i(\underline{\lambda}) \, d\underline{\lambda} \, ,$$

or

$$E(\underline{X}) = \sum_{i=1}^{N} P_i \, \underline{U}_i \, . \tag{6.15}$$

Let $\underline{U} = E(\underline{X})$ and let $[\Phi]$ be the covariance of $p(\underline{X})$. Then $[\Phi]$ can be found in terms of $\underline{U}_i$ and $[\Phi_i]$ by

$$[\Phi] = E\left[(X - \underline{U})(\underline{X} - \underline{U})^t\right] = E(\underline{X}\underline{X}^t) - \underline{U}\,\underline{U}^t$$

$$= \int_{\underline{\lambda}} \underline{\lambda}\,\underline{\lambda}^t \, p(\underline{\lambda}) \, d\underline{\lambda} - \underline{U}\,\underline{U}^t$$

$$= \sum_{i=1}^{N} P_i \int_{\underline{\lambda}} \underline{\lambda}\,\underline{\lambda}^t \, p_i(\underline{\lambda}) \, d\underline{\lambda} - \underline{U}\,\underline{U}^t$$

$$= \sum_{i=1}^{N} P_i \left\{ [\Phi_i] + \underline{U}_i \underline{U}_i^t \right\} - \underline{U}\,\underline{U}^t$$

$$= \sum_{i=1}^{N} P_i [\Phi_i] + \sum_{i=1}^{N} P_i \underline{U}_i \underline{U}_i^t - \left(\sum_{i=1}^{N} P_i \underline{U}_i \right) \left(\sum_{i=1}^{N} P_i \underline{U}_i^t \right). \tag{6.16}$$

Equations (6.15) and (6.16) define an estimate for the mean and covariance respectively for the training set in terms of the means and covariances of the clusters from the CCA processing of the training set. The mean of the training set is of course a simple weighted average of the cluster means. For the covariance in (6.16), observe that the first term on the right is the weighted average of the covariance between the cluster means. This can be observed more clearly by looking at a given

element of the covariance matrix. Let u_j^i be the j-th element of $\underline{U}_i$, and let ϕ_{jk} and ϕ_{jk}^i be the element in the j-th row and k-th column of $[\Phi]$ and $[\Phi_i]$ respectively. Then from (6.16) we have

$$\phi_{jk} = \sum_{i=1}^{N} P_i \phi_{jk}^i + \left\{ \sum_{i=1}^{N} P_i u_j^i u_k^i - \left(\sum_{i=1}^{N} P_i u_j^i \right) \left(\sum_{i=1}^{N} P_i u_k^i \right) \right\}. \tag{6.17}$$

The estimate for the training set covariance consists of the weighted average of intracluster covariance, and the weighted average of intercluster covariance between the cluster means.

Now recall that the usual training procedure simply calculates the mean and covariance of the training set vectors according to (6.11) and (6.12). If the training set is compressed by the CCA, then each vector of the training set is approximated by the mean of the cluster to which the vector was assigned. Thus if one uses the usual training on the CCA compressed training set, the estimate for the mean is the same as the estimate in (6.15), but the estimate for the covariance consists of only the intercluster covariance portion of (6.17). Of course, this is equivalent to assuming a zero value for the intracluster variance of all the clusters. The classification performance corresponding to this training was presented earlier in Curve 2 of Fig. 6.18.

The impact of assuming a zero average intracluster variance can be large, especially with respect to estimating the diagonal terms of the covariance matrix. Experimental observations confirmed that the intracluster covariance in (6.17) was frequently much larger than the intercluster covariance. As discussed in [28, p. 541], clustering basically

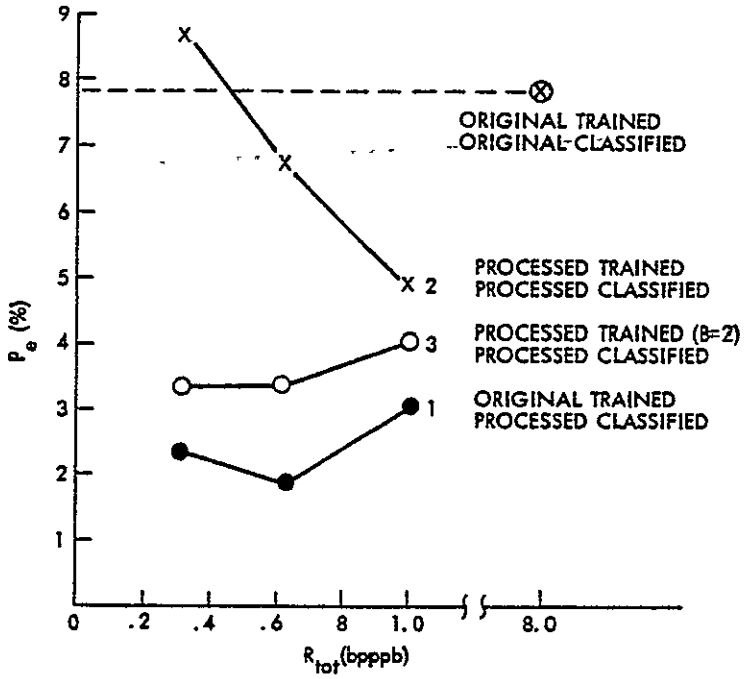


Fig. 6.18. Classification Performance of the Uncoded CCA, n = 256, f = 32, m = 2, 4, and 8.

attempts to minimize the ratio of intracluster to intercluster distances for a given set of elements. However, one must note that a compressed training set may often consist of elements from clusters of more than one local source. Since the vectors from different local sources are not jointly clustered, the intercluster distance between clusters from different local sources may be very small. Therefore, it is not unreasonable to obtain an intracluster covariance greater than the intercluster covariance. The classification performance would likely be most seriously degraded when the intercluster covariance is very small. Then the assumption of zero intracluster variance can substantially change the form of the conditional distribution function.

Now consider how the training procedure can be modified when the training set is compressed by the CCA. Obviously, one procedure could

be to include the cluster covariance as a cluster feature transmitted by the CCA. Then the training set covariance estimate could be calculated directly from (6.17). However, this procedure is likely more than necessary. In particular, one might anticipate that the intracluster covariance term in (6.17) is mainly important for the diagonal terms of the covariance for the training set. Perhaps the crudest modification of the training would be to simply assume a constant value $\beta > 0$ for the diagonal terms of the intracluster covariance for all clusters, and assume a value of zero for all off diagonal elements. This corresponds to assuming a symmetrical distribution of variance β in each band for each cluster. The covariance matrix for the training set could be obtained from the intercluster variance portion of (6.17), or from the usual training procedure, but in either case a value of β is added to the diagonal elements. Curve 3 in Fig. 6.18 shows the classification performance when the covariance matrix for the compressed training set is modified by adding $\beta = 2$ to the diagonal elements. A significant performance increase is observable relative to the usual training procedure which corresponds to using $\beta = 0$, and which is shown in Curve 2. The performance of Curve 3 was quite insensitive to changes in β between 1 through 5. This result demonstrates that even a very simple adjustment in the training procedure can make a substantial difference in the classification performance.

An additional simulation was conducted to determine how much the performance would improve if the CCA did transmit covariance information about the clusters. Only the CCA configuration of 16×16 local sources with 4 clusters each was tested. In addition to the cluster

mean, the CCA also transmitted the cluster variance per band quantized to 3 bits. This extra information increases the data rate of the CCA by only 0.0469 bpppb. The covariance matrix for each training set was calculated from (6.17) by using the transmitted variance per band for the diagonal elements of the intracluster term, and zero for the off diagonal terms. The classification performance of this simulation was not significantly different than the corresponding performance from using $\beta = 2$ for each variance per band. One could continue attempting to improve the classification performance. For example, the classifier could be defined in terms of discriminant functions which result from modeling the conditional distributions as a weighted mixture of Gaussian distributions, one for each cluster. These alternate classifiers are not explored in this work.

6. Discussion of the Data Compression and Data Interpretation Interaction

The relative changes in the performance curves in Fig. 6.18 indicate the importance of considering the interaction between data compression and data classification. If the classification and classifier training are not properly matched to the compressed data, an erroneous measure of classification performance is likely to be obtained. A similar situation exists with regard to assessing the impact of data compression on other data processing techniques, such as clustering, or geometric correction of image data. For example, in assessing the impact of CCA data compression on clustering analysis, one should not simply cluster the reconstructed image elements. Such a procedure again assumes a zero intracluster covariance for every cluster in the

compressed data, which can significantly degrade, or change the nonsupervised classification (clustering results) in much the same manner as occurred for supervised classification. The clustering may first need to be modified in order to reflect the fact that it is clustering clusters of nonzero covariance instead of independent vectors.

7. Performance vs. Class Density per Local Source

From Curve 3 in Fig. 6.18 the classification performance of the CCA is observed to be very good. In particular, the CCA processing enabled the Gaussian parametric classifier to reduce the average error rate while achieving substantial compression of the data. However, the absolute level of performance of the CCA will be dependent on the number of classes occurring within a local source. As the average number of classes per local source increase, the classification performance of the CCA will decrease. Further simulations are required in order to quantify the classification performance as a function of the number of classes per local source. One of the difficulties in conducting such simulations is the unavailability of test images which simultaneously have a higher number of classes per 16 × 16 element regions and reliable ground truth through the class boundaries. Some understanding of the impact of more classes per local source can be obtained from Fig. 6.15.

As previously noted, Fig. 6.15(a) is the original image used for classification performance testing, and Fig. 6.15(b) is the grey-level coded classified image resulting from training and classifying on the

original data. Figure 6.15(c) shows a reconstructed image resulting from using the CCA to compress the original image to 4 clusters per 16 × 16 element local source. In Fig. 6.15(d) is shown the grey-level coded classified image resulting from training and classifying on the CCA compressed image of Fig. 6.15(c). The training on the compressed image data was modified as previously discussed by adding $\beta = 2$ to the diagonal elements of the covariance matrix for each class. Thus this classified image corresponds to one of the simulation results used in Curve 3 of Fig. 6.18. Figure 6.16 is a color composite image representation of the same results shown in Fig. 6.15, and the legend for associating classes and grey-levels, or colors is shown in Fig. 6.17. From Figs. 6.15 and 6.16 one can observe that even though the CCA was limited to only 4 clusters per 16 × 16 elements, the field boundaries and interconnecting regions with the higher number of classes per local source are apparently well classified. Thus the CCA classification performance appears not to be overly sensitive to increases in the number of classes per local source. Of course, the Adaptive CCA could ideally tailor the number of clusters per local source dependent on the number of classes present, which should result in further substantial classification performance gains.

8. Performance in Terms of Inventory Accuracy vs. Data Rate

Another measure of classification performance is inventory accuracy vs. total data rate. Inventory accuracy refers to the accuracy of the classifier in determining the relative density of the classes. In Table IX the true inventory data is given for the test set of Table VIII. Also shown in Table IX is the inventory results obtained

TABLE IX
INVENTORY CLASSIFICATION PERFORMANCE

	True Inventory		Estimate From Original Data		Estimate From CCA Compressed Data	
Class	Number	%	Number	%	Number	%
1	415	2.28	513	2.82	416	2.285
2	6014	33.04	5548	30.48	5712	31.38
3	3440	18.90	3486	19.15	3569	19.06
4	512	2.81	619	3.40	598	3.29
5	604	3.32	856	4.70	785	4.31
6	2398	13.17	2454	13.48	2394	13.15
7	2000	10.99	1917	10.53	2010	11.04
8	2820	15.49	2806	15.42	2819	15.49
9	0		4		0	
			Total = 18,203			

from using the Gaussian parametric classifier with training and classifying on the original image in Fig. 6.15(a). The total data rate for these inventory estimates is 8 bpppb. Table IX also shows the inventory estimates obtained from training and classifying on the CCA compressed data corresponding to the image in Fig. 6.15(c). Again the CCA used 4 clusters per 16 × 16 element local source, and the classifier modified the training covariance matrices by adding $\beta = 2$ to the diagonal elements. The inventory estimates are more accurate for every class, while the total data rate for the Uncoded CCA and the Coded CCA are .625 and .429 bpppb respectively.

9. Summary Discussion of Classification Performance

The classification performance of the CCA, of course, depends on the choice of classification technique. In the proceeding discussions the Gaussian parametric classifier was used to demonstrate the interaction between data compression and data classification. In particular,

the classification accuracy was observed to be sensitive to the training procedure, and it was noted that modifications of the training may be appropriate when training on compressed data. The CCA data compression was shown to be capable of benefiting the classifier by providing group information on the data; however, further investigation is required to determine the conditions for which the group information degrades the classification accuracy. The previous classification performance simulations showed the CCA compression to result in significantly lower average vector classification error and lower inventory error, while substantially reducing the total data rate. It should also be noted that in addition to reducing the data volume, the CCA data compression also results in reducing the number of classifications required. For example, in the CCA simulations where 4 clusters are used per 16 × 16 element local source, the number of classifications is reduced by a factor of 64.

CHAPTER VII

SUMMARY AND RECOMMENDATIONS FOR FURTHER RESEARCH

In this dissertation a joint clustering and data compression concept is defined for efficient transmission of image information which is to be obtained from manual photo interpretation and/or computer classification. Clustering is used to extract features which tend to preserve class separability information within the data. Data compression is used to provide efficient transmission of the information in terms of the clustering features. Various forms of a Cluster Compression Algorithm are defined, discussed, and extensively simulated to investigate application of this concept to multispectral image data.

The following is a list of topics for further research:

1) The Cascaded CAA should be further investigated by incorporating a better intercluster distance measure within the cascaded clustering.

2) An appropriate test set should be obtained, or constructed for testing classification performance as a function of class density. In addition, a determination of typical class densities for intended applications is needed.

3) Other classifiers, such as mixture distribution parametric classifiers, might be assessed for their impact on CCA classification performance.

4) Alternate feature map encoding and spatial feature extraction could be explored. For example, boundary finding algorithms might be implemented more simply on the feature map than on the original data.

5) The use of transformations (e.g., the Karhunen-Loève, or variance normalization transformations) might be investigated for use in preceding the CCA.

6) The concept of joint clustering and data compression could also be researched for applications to television data and artificial intelligence.

REFERENCES

1. R. F. Rice, "Channel Coding and Data Compression System Considerations for Efficient Communications of Planetary Imaging Data," Jet Propulsion Lab., Pasadena, Calif., Tech. Memo 33-695, June 1974.

2. ERTS Project Office, "Data Users Handbook," Goddard Space Flight Center, Greenbelt, Maryland, Doc. No. 71SD4249, Sept. 1971.

3. Symposium of Significant Results Obtained from Earth Resources Technology Satellite-1, Goddard Space Flight Center, Greenbelt, Maryland, S. C. Freden, E. P. Mercanti, Eds., Dec. 1973.

4. G. Nagy, "Digital Image-Processing Activities in Remote Sensing for Earth Resources," *Proc. IEEE*, Vol. 60, pp. 1177-1200, Oct. 1972.

5. J. O. Limb, C. B. Rubinstein, and K. A. Walsh, "Digital Coding of Color Picturephone Signals by Element-Differential Quantization," *IEEE Trans. Commun. Technol.*, Vol. COM-19, pp. 992-1006, Dec. 1971.

6. W. K. Pratt, J. Kane, and H. C. Andrews, "Hadamard Transform Image Coding," *Proc. IEEE*, Vol. 57, No. 1, pp. 58-68, Jan 1969.

7. P. J. Ready and P. A. Wintz, "Multispectral Data Compression Through Transform Coding and Block Quantization," School of Elec. Eng., Purdue Univ., W. Lafayette, Ind., Tech. Rep. TR-EE 72-29, May 1972.

8. A. Habibi, "Hybrid Coding of Pictorial Data," *IEEE Trans. Commun.*, Vol. COM-22, No. 5, pp. 614-624, May 1974.

9. N. Abramson, *Information Theory and Coding*, New York: McGraw-Hill, 1963.

10. G. Nagy, "State of the Art in Pattern Recognition," *Proc. IEEE*, Vol. 56, pp. 836-862, May 1968.

11. L. Kanal, "Patterns in Pattern Recognition: 1968-1974," *IEEE Trans. Inform. Theory*, Vol. IT-20, pp. 697-722, Nov. 1974.

12. P. H. Swain, "Pattern Recognition: A Basis for Remote Sensing Data Analysis," Lab. for Applic. of Remote Sensing, Purdue Univ., W. Lafayette, Ind., LARS Inform. Note 111572, Sept. 1973.

13. A. G. Wacker and D. A. Landgrebe, "Minimum Distance Classification in Remote Sensing," Lab. for Applic. of Remote Sensing, Purdue Univ., W. Lafayeete, Ind., LARS Inform. Note 030772, Feb. 1972.

14. P. H. Swain and R. C. King, "Two Effective Feature Extraction Criteria for Multispectral Remote Sensing," Lab. for Applic. of Remote Sensing, Purdue Univ., W. Lafayette, Ind., LARS Inform. Note 042673, Nov. 1973.

15. E. P. F. Kan, "Data Clustering: An Overview," Lockheed Electronics Co., Inc., HASD, Houston, Texas, Tech. Rep. 640-TR-080, March 1972.

16. M. R. Anderberg, *Cluster Analysis for Applications*, New York: Academic, 1973.

17. E. P. F. Kan, "The JSC Clustering Program ISOCLS and Its Applications," Lockheed Electronics Co., Inc., HASD, Houston, Texas, Tech. Rep. 640-TR, July 1973.

18. E. P. F. Kan, "The Latest Version of ISODATA (A)/ISOCLS," Lockheed Electronics Co., Inc., HASD, Houston, Texas, Tech. Memo TM642-570, Sept, 1972.

19. E. E. Hilbert, "Joint Classification and Data Compression of Multidimensional Information Sources - Application to ERTS," *Internat. Conf. on Commun.*, Vol. II, Session 27, pp. 6-11, June 1975.

20. E. E. Hilbert, "Joint Pattern Recognition/Data Compression Concept for ERTS Multispectral Imaging," SPIE Seminar Proc., Vol. 66, pp. 122-137, Aug. 1975.

21. L. A. Zadeh, "Fuzzy Sets," *Inform. Control*, Vol. 8, pp. 338-353, 1965.

22. J. N. Gupta and P. A. Wintz, "A Boundary Finding Algorithm and Its Applications," *IEEE Trans. on Circuits and Systems*, Vol. CAS-22, No. 4, pp. 351-362, April 1975.

23. G. H. Ball, "A Comparison of Some Cluster-Seeking Techniques," Stanford Research Inst., Menlo Park, Calif., Report No. RADC-TR-66-514, 47 pp., 1966.

24. H. P. Friedman and J. Rubin, "On Some Invariant Criteria for Grouping Data," *J. Amer. Statist Assoc.*, Vol. 62, pp. 1159-1178, 1967.

25. F. J. Rohlf, "Adaptive Hierarchical Clustering Schemes," *Systematic Zoology*, Vol. 19, No. 1, pp. 58-83, 1970.

26. G. H. Ball and D. J. Hall, "ISODATA, A Novel Method of Data Analysis and Pattern Classification," Stanford Res. Inst., Menlo Park, Calif., Tech. Rep. AD699616, Apr. 1965.

27. A. G. Wacker and D. A. Landgrebe, "Boundaries in Multispectral Imagery by Clustering," *IEEE Symposium on Adaptive Processes* (9-th) Decision and Control, pp. X14.1-X14.8, Dec. 1970.

28. S. Wilks, *Mathematical Statistics*, New York: John Wiley and Sons, 1962.

29. R. G. Gallager, *Information Theory and Reliable Communication*, New York; John Wiley and Sons, 1968.

30. W. Feller, *Probability Theory and Its Applications*, New York: John Wiley and Sons, 1950.

31. R. F. Rice and J. R. Plaunt, "Adaptive Variable-Length Coding for Efficient Compression of Spacecraft Television Data," *IEEE Trans. Commun.*, Vol. COM-19, No. 6, pp. 889-897, Dec. 1971.

32. Remote Multispectral Sensing in Agriculture, Laboratory for Agricultural Remote Sensing, Purdue University, Lafayette, Ind., Annual Report, Vol. 3, Research Bulletin 844, Sept. 1968.

jpl → PUBLICATION 77-43