Preprint
UCRL-JC-135741

Optimal Segmentation Strategy for Compact Representation of Hyperspectral Image Cubes

D.W. Paglieroni, R.S. Roberts

This article was submitted to American Society of Photogrammetry and Remote Sensing Annual Conference, Washington, D.C., May 22-26, 2000

U.S. Department of Energy
Lawrence Livermore National Laboratory

February 8, 2000

OPTIMAL SEGMENTATION STRATEGY FOR COMPACT REPRESENTATION OF HYPERSPECTRAL IMAGE CUBES

David W. Paglieroni and Randy S. Roberts

Lawrence Livermore National Laboratory
P.O. Box 808, Livermore, CA 94550

Abstract

By producing compact representations of hyperspectral image cubes (*hypercubes*), image storage requirements and the amount of time it takes to extract essential elements of information can both be dramatically reduced. However, these compact representations must preserve the important spectral features within hypercube pixels and the spatial structure associated with background and objects or phenomena of interest.

This paper describes a novel approach for automatically and efficiently generating a particular type of compact hypercube representation, referred to as a *supercube*. The hypercube is segmented into regions that contain pixels with similar spectral shapes that are spatially connected, and the pixel connectivity constraint can be relaxed. Thresholds of similarity in spectral shape between pairs of pixels are derived directly from the hypercube data. One *superpixel* is generated for each region as some linear combination of pixels belonging to that region. The superpixels are optimal in the sense that the linear combination coefficients are computed so as to minimize the level of noise. Each hypercube pixel is represented in the supercube by applying a gain and bias to the superpixel assigned to the region containing that pixel. Examples are provided.

1. Introduction

The amount of data contained in hyperspectral image cubes (hypercubes), which typically contain hundreds of spectral bands, can be very large, particularly if the number of rows and columns of pixels is large. Consequently, algorithms for analyzing and interpreting hypercubes can require a substantial amount of computational effort. It is therefore no surprise that a great deal of research effort has been put into developing algorithms that segment hypercubes into regions. Not only can segmented hypercubes be represented much more compactly than raw hypercubes, but they can also be analyzed and interpreted much more efficiently.

Hypercube segmentation has its origins in early work on segmentation of single band images. One of the largest families of image segmentation techniques consists of region growing algorithms [8]. Region growers group pixels into regions based either on both proximity and pixel value (in the case of local techniques [6]) or solely on pixel value (in the case of global techniques). In principle, the notions of local and global techniques can be unified by applying the concept of *connectivity relaxation*, which specifies the maximum distance allowed between pixels that belong to the same region. A region grower becomes more global in nature as connectivity relaxation increases, but its computational complexity also increases. This increase in complexity can be mitigated by specifying a small degree of connectivity relaxation and then allowing regions with similar pixels to be subsequently merged [4].

When generalized from single band images to hypercubes, pixels represent sampled spectra (or vectors) that contain one element per band, and image segmentation can be viewed as a pixel classification problem. In the supervised approach, there are pre-conceived notions of what the pixel classes are, and these notions are driven by the application (e.g., classification of land usage as for one of two crops or for neither crop). In the unsupervised approach, the pixel classes that exist within a hypercube are explicitly derived from the data in the hypercube. An unsupervised approach is indicated when the objective is to represent a hypercube more compactly (as in this paper) so that it can be stored and analyzed more efficiently.

In both supervised and unsupervised pixel classification, pixel class assignment is often based on linear decision functions, whose weights are obtained by training a neural network (see, e.g., [7]). Other approaches are based on maximum likelihood estimation (MLE) (see, e.g., [2], [5]) or maximum a posteriori (MAP) estimation (see, e.g., [3]). MLE and MAP techniques require assumptions to be made about the probability densities of the various classes, the density parameters and, in some cases, the prior class probabilities. In any case, the inputs to the

classifier are either hypercube pixels, or feature vectors of lower dimensionality (such as high-energy principal component transform coefficients) derived from hypercube pixels.

The hypercube segmentation method presented in this paper can be viewed either in the context of classical image segmentation techniques, or as an unsupervised pixel classification method. Unsupervised pixel classification methods are based on some form of pixel clustering. Pixel clustering is normally based on a variant of some standard cluster seeking algorithm (such as Isodata) that implements some test of pixel homogeneity. Hypercube segmentation algorithms use a test of pixel homogeneity that is based on a decision function which accounts for statistical assumptions made about the pixel data (if any). For example, the decision function could be the norm of the difference between a hypercube pixel and the mean of pixels that belong to a specific region or cluster of pixels.

The hypercube segmentation method presented in this paper is novel in that the test of homogeneity is specifically designed so that the pixels in a region are all guaranteed to have spectral shapes that are closely similar. The test of homogeneity is based on a threshold spectrum that is easily derived from the data in the hypercube, without making any assumptions about the probability distributions of the data. The shape of the threshold spectrum reflects the shape of the noise spectrum associated with the hypercube. Pixel clustering is accomplished by applying region growing to a normalized hypercube obtained from the original hypercube by removing the bias from each pixel and scaling each pixel to unit energy. Specifically, a pixel is added to a region only when the magnitude of the difference spectrum between the normalized version of the pixel and the aggregated pixel representing the region, lies completely beneath the threshold spectrum.

The segmentation method is optimal in two ways. First, the aggregated pixel representing a given region has the smallest possible amount of noise, under the assumption that the random noise vectors corresponding to the different pixels are additive, zero-mean and independent identically distributed (iid). Second, the sampled spectrum associated with a pixel in a given region is modeled by an optimal linear fit of the aggregated pixel spectrum associated with that region. In addition, when a capability to merge regions is added to the segmenter, the number of regions in the segmented hypercube can be nearly reduced to the minimum possible number of regions that one can associate with a given hypercube and a given threshold spectrum.

2. Hypercube Segmentation

The hypercube segmentation process described in this paper can be divided into three broad stages (see Fig.2.1). First, a similarity threshold spectrum is derived directly from the data in the hypercube. Then, pixels with similar spectral shapes that are close to one another are aggregated into regions using a region growing technique. Finally, *superpixels* representative of entire regions are formed as linear combinations of pixels aggregated into those regions. The linear combination weights are optimal in the sense that they minimize the level of noise in the superpixel. A *supercube* is formed by replacing each hypercube pixel with an optimally scaled and biased version of the superpixel associated with the region containing that pixel.

2.1 Threshold Spectrum

When segmenting hypercubes into regions containing pixels with similar spectral shapes, one must be able to set limits, appropriate for the hypercube in question, on the maximum allowed discrepancy in spectral shape between pixels that are deemed similar. These limits are captured in a similarity *threshold spectrum*, which provides an upper bound on the discrepancy between spectra associated with pairs of pixels that are deemed similar.

The maximum allowed discrepancy in spectral shape between pixels that are deemed similar is based on the noise spectrum associated with the hypercube. The shape of the noise spectrum depends on the hyperspectral imaging instrument, and is indicative of the relative degree of certainty in the values of spectral samples as a function of spectral band. The similarity threshold spectrum is thus appropriately envisioned to be a scaled version of the noise spectrum.

Although the shape of the noise spectrum can be most easily estimated from hypercubes of blackbody radiators, it can also be estimated directly from the data in the hypercube of a real scene. The approach taken in this section is to derive a similarity threshold spectrum, whose shape closely resembles the shape of the noise spectrum, directly from a specific subset of all possible pairs of hypercube pixels. In most hypercubes, pixels are more likely to have similar spectral shapes when they are adjacent. It thus makes sense to derive the threshold spectrum from the magnitude of difference spectra associated with a large number of pairs of adjacent pixels. In this section, all pairs of adjacent pixels along all rows and along the first column are used.

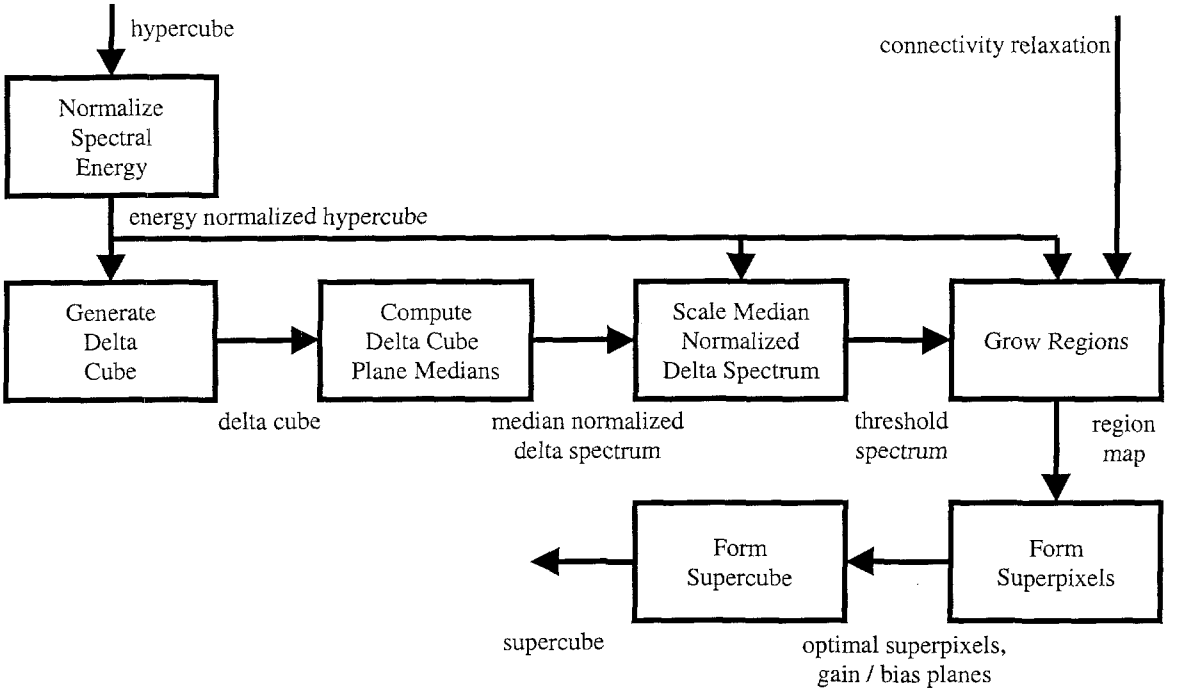

Fig.2.1 Overall flow of hypercube segmentation.

Since spectral gain and bias have nothing to do with spectral shape, the spectra associated with all pixels in the hypercube are first normalized to zero bias and unit energy. Let $\boldsymbol{y} = [y(0),...,y(K-1)]^T$ be the sampled spectrum associated with some arbitrary pixel in a hypercube with K spectral bands. The normalized spectrum $\boldsymbol{x}$ associated with $\boldsymbol{y}$ has a bias (sample mean) of $\bar{x} = 0$ and an energy (inner product) of $\boldsymbol{x}^T\boldsymbol{x} = 1$. Therefore,

(2.1) $$\boldsymbol{x} = a(\boldsymbol{y} - \bar{\boldsymbol{y}}) = a\boldsymbol{y} + \boldsymbol{b}$$

(2.2) $$a = 1/\|\boldsymbol{y} - \bar{\boldsymbol{y}}\| \qquad (\bar{\boldsymbol{y}} \neq \boldsymbol{y})$$

(2.3) $$b = -a\bar{y}$$

In equations (2.1)-(2.3), $\bar{y}$ is the sample mean of $\boldsymbol{y}$, $\bar{\boldsymbol{y}}$ is the $Kx1$ constant vector with elements equal to $\bar{y}$, a is the normalizing gain, b is the normalizing bias and $\boldsymbol{b}$ is the $Kx1$ constant vector with elements equal to b.

Consider a hypercube with I rows and J columns of pixels $\boldsymbol{y}_{i,j}$. The degree of similarity in spectral shape between hypercube pixel $\boldsymbol{y}_{i,j}$ and its adjacent neighbor will be based on the magnitude of the difference spectrum between their normalized versions. Let us define a *delta cube* $\{\Delta_{i,j}(k)\ \ i = 0,...,I-1\ \ j = 0,...,J-1,\ \ k = 0,...,K-1\}$ to be a cube of difference magnitude spectra formed from all pairs of adjacent pixels on all rows and the first column (as in delta modulation). Then

(2.4) $$\Delta_{i,j}(k) = \begin{cases} |x_{i,j}(k) - x_{i,j-1}(k)| & j = 1,...,J-1 \\ |x_{i,j}(k) - x_{i-1,j}(k)| & j = 0, \quad i = 1,...,I-1 \\ 0 & i = j = 0 \end{cases}$$

The threshold spectrum is derived directly from the data in the delta cube. If the hypercube represents an image of a blackbody radiator, then the outputs of the imaging detectors should ideally all be fairly consistent. In this case, the normalization gains and biases associated with the hypercube pixels should all be close to one and zero respectively (see equations (2.1)-(2.3)), so the delta cube can be estimated as in equation (2.4) with the normalized pixels x replaced by the original pixels y. Furthermore, the noise spectrum can be estimated by computing the sample standard deviation of hypercube values within each band, which is closely related to the mean of delta cube values within each band. However, if the hypercube represents the image of a real scene, one must derive the delta cube from normalized pixels, and instead of computing the sample standard deviation of each spectral band (as for a blackbody cube), the median delta magnitude within each spectral band should be computed instead. Mathematically, the threshold spectrum $\boldsymbol{T} \triangleq [T(0),...,T(K-1)]^T$ is derived from the delta cube as

$$T(k) = \alpha \cdot \Delta_{median}(k) \tag{2.5}$$

$$\Delta_{median}(k) \triangleq \underset{\substack{i = 0,...,I-1 \\ j = 0,..,J-1}}{\text{median}} \Delta_{i,j}(k) \tag{2.6}$$

where α is a scale factor relating threshold spectrum to median normalized delta spectrum.

The scale factor α is obtained by recognizing that the threshold spectrum is intended to be an upper bound on delta spectra derived from pairs of hypercube pixels with similar spectral shapes. This suggests that α can be derived by computing, for each delta cube pixel, the minimum scale factor for which all samples of the scaled median normalized delta spectrum are at least as large as the corresponding delta cube samples. Mathematically,

$$\alpha_{i,j} = \max_{k = 0,...,K-1} [\Delta_{i,j}(k) / \Delta_{median}(k)] \tag{2.7}$$

α is then estimated from the resulting sample density of scale values $\alpha_{i,j}$. In particular, if a designated percentage of the adjacent pairs of pixels in the set are presumed similar, then the scale factor associated with the corresponding percentile from the sample density will result in a threshold spectrum whose values are typical of delta values between corresponding samples from similar pairs of pixels. By choosing the median scale factor, only half of the adjacent pairs of pixels in the set are presumed similar. This assumption is weak in the sense that many more than half of adjacent pixels from the set may be similar. Therefore, one may wish to use a larger percentile. For example, in this paper, the 75th percentile scale factor is used. This is consistent with the belief that at least three out of every four pairs of adjacent pixels are similar. It is also possible for less than half of the adjacent pixels to be similar. However, this situation is unusual (even pathological) since rarely do significant pixel variations occur more often than at every other pixel on average in an image of a scene. This type of behavior would be expected in random fields, such as images of noise.

2.2 Pixel Aggregation

In order to ensure that pixels assigned to a region all have similar spectral shapes, one can apply a test of homogeneity to pixels that one wishes to add to the region, and the seed pixel that represents the region as a whole (in its current form). The test of homogeneity must make use of some measure of spectral similarity. *Pixel aggregation* is the process of grouping pixels that are spatially proximate and spectrally similar into regions. Pixel aggregation can thus be viewed as a pixel clustering procedure. The approach to pixel aggregation taken in this section is based on region growing applied in conjunction with a test of pixel homogeneity based on the threshold spectrum from the previous section.

In its most basic form, the test of pixel homogeneity can be expressed as follows:

$$| x_i(k) - x_j(k) | < T(k) \text{ for } k = 0,...,K-1 \quad \Rightarrow \quad \text{pixels } \boldsymbol{y}_i \text{ and } \boldsymbol{y}_j \text{ are similar} \tag{2.8}$$

This basic version of the test can be extended in two ways (see Fig.2.2). First, one may wish to apply narrowband smoothing with a two or three point moving average filter to the difference magnitude spectrum $| x_i(k) - x_j(k) |$, particularly if the spectra are noisy. This handles spurious single-band samples in the difference magnitude spectrum that do not exceed the threshold spectrum by much. Such samples occur more frequently if $x_i(k)$ and $x_j(k)$ are noisy, in which case $| x_i(k) - x_j(k) |$ will be even more noisy, because spectral differencing tends to accentuate spectral noise. Second, one may wish to limit the comparison between $| x_i(k) - x_j(k) |$ and $T(k)$ to spectral subbands that corrrespond to the locations of spectral features of interest.

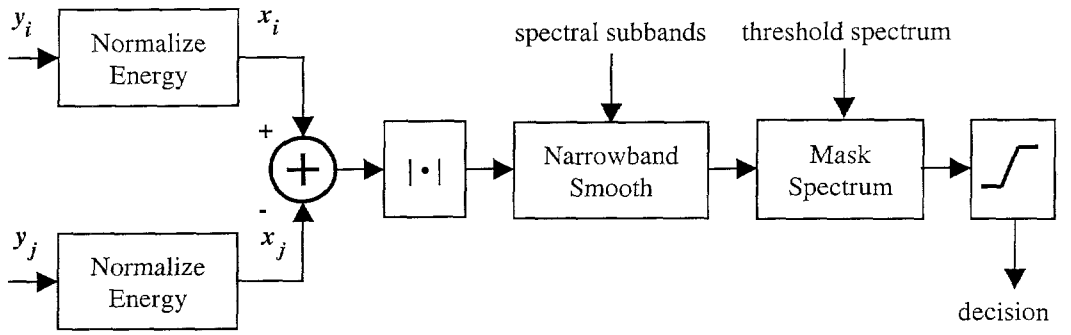

Fig.2.2 Test of pixel homogeneity.

The test of pixel homogeneity in expression (2.8) is applied in the context of region growing. Region growing begins by specifying an initial seed pixel for the first region as pixel *(0,0)*. The initial seed pixel for a subsequent region is specified as the first pixel that does not already belong to some other region encountered on row scan from the location of the initial seed pixel for the previous region. A neighborhood of width $2w+1$ centered on the initial seed pixel is searched for pixels that pass the test of homogeneity with the seed pixel. The parameter $w > 0$ is the integer-valued *connectivity relaxation parameter*. It specifies the maximum distance allowed between pixels that belong to the same region. Neighborhood pixels that pass the test of homogeneity are pushed onto a stack, and the number of pixels in the region is incremented. If the number of pixels in the region is less than the maximum number of allowed updates to the seed pixel, the seed pixel is updated with the pixel that was pushed onto the stack. Finally, each time the stack is popped, the neighborhood of width $2w+1$ centered on the popped pixel is searched. Region growing is complete when the stack is empty.

The signal-to-noise ratio (SNR) associated with the seed pixel initially assigned to a new region may be low if the data is noisy. It is more difficult to establish similarity between two noisy spectra than between one noisy spectrum and one relatively clean spectrum. If the noise is additive, zero-mean and independent identically distributed (iid) from pixel to pixel (so that the noise within a spectral band is assumed white and spatially uncorrelated, though not spectrally uncorrelated), then the SNR of a seed pixel can be increased by computing the average over similar spectra. Therefore, each time a new pixel $\boldsymbol{x}$ is added to a region that initially contains m pixels to form a region with $m+1$ pixels, the seed pixel $\boldsymbol{x}_0$ should be updated as

$$\boldsymbol{x}_0 \leftarrow (m\,\boldsymbol{x}_0 + \boldsymbol{x})/(m+1) \tag{2.9}$$

However, if too many updates are allowed, then although the SNR of $\boldsymbol{x}_0$ will tend to increase, $\boldsymbol{x}_0$ may drift. When $\boldsymbol{x}_0$ drifts, its spectral shape may begin to deviate from the shapes of some of the pixel spectra contained in the region. But since the rate of drift is low, this issue can be resolved by limiting the number of updates. If, for example, the number of updates is limited to 25, and at least 25 pixels have been added to the region, then the SNR of the final seed pixel will have increased by a factor of as much as $\sqrt{25} = 5$ over the SNR of the initial seed pixel, without appreciable drift.

The output of pixel aggregation is a *region map*. The value of region map element (i,j) is the number of the region (an integer starting from zero) that hypercube pixel (i,j) belongs to. Region map elements thus range in value from 0 to $n-1$, where n is the number of segmented regions.

2.3 Optimal Superpixels and Supercubes

A *superpixel* is a linear combination of all of the pixels aggregated into the same region. Its spectral shape is representative of the spectra corresponding to all of the pixels in the region. Consider a region consisting of m pixels $\mathbf{y}_i$ $i = 0,...,m-1$. The superpixel $\mathbf{y}$ associated with this region has the form

$$\mathbf{y} = \sum_{i=0}^{m-1} w_i \mathbf{y}_i \tag{2.10}$$

where the coefficients w_i are the superpixel integration weights. Assume the random noise vectors corresponding to the different hypercube pixels are additive, zero-mean and iid (in which case, the noise is assumed spatially uncorrelated, but not necessarily spectrally uncorrelated). Even though there is likely to be some amount of spatial correlation in the noise, the foregoing assumptions about the noise, though not strictly valid, provide a useful model for noise in hypercubes. It can be shown that the weights w_i which yield the superpixel that is optimal in the sense that it contains the smallest possible amount of noise, subject to the constraint that the weights sum to one, are the uniform weights

$$w_i = 1/m \qquad i = 0,...,m-1 \tag{2.11}$$

If all the pixels being aggregated are nearly identical except for noise, then the SNR of the superpixel will exceed the SNR of any of its component pixels by approximately a factor of $\sqrt{m}$ (this assumes that the SNR is defined to be the square root of the ratio of signal energy to expected energy in the noise).

A *supercube* is a hypercube in which each pixel is replaced by an optimally scaled and biased version of its superpixel. A supercube is thus a model or representation of a hypercube. The relationship between hypercube pixel $\mathbf{y}_i$ and its associated superpixel $\mathbf{y}$ is

$$\mathbf{y}_i \approx \hat{\mathbf{y}}_i \stackrel{\Delta}{=} c_i \mathbf{y} + d_i \tag{2.12}$$

where $\hat{\mathbf{y}}_i$ is an optimal *linear superpixel model* of $\mathbf{y}_i$. The upper bound on elements of $|\mathbf{y}_i - \hat{\mathbf{y}}_i|$ can be approximated as $\boldsymbol{T} / |a_i|$, where $\boldsymbol{T}$ is the threshold spectrum and a_i is the energy normalization gain associated with $\mathbf{y}_i$.

The gain c_i and bias d_i in equation (2.12) are chosen such that $c_i \mathbf{y} + d_i$ is normalized to $\mathbf{y}_i$ in some sense. *Energy normalization criteria* are more robust than the least-squares normalization criterion because the least-squares criterion tends to produce unusually small gains when the spectra $\mathbf{y}$ and $\mathbf{y}_i$ are similar but noisy, whereas energy normalization criteria tend to select a gain value that is relatively insensitive to noise. There are several possible ways to define an energy normalization criterion. One way is to choose c_i to be the gain which, when applied to $\mathbf{y}$ with its bias removed, produces a spectrum whose energy equals the energy of $\mathbf{y}_i$ with its bias removed. d_i is then obtained by adding the bias of $\mathbf{y}_i$ onto the product of c_i and the unbiased version of $\mathbf{y}$. Mathematically,

$$c_i = \|\mathbf{y}_i - \bar{\mathbf{y}}_i\| / \|\mathbf{y} - \bar{\mathbf{y}}\| \qquad (\mathbf{y} \neq \bar{\mathbf{y}}) \tag{2.13}$$

$$d_i = \bar{\mathbf{y}}_i - c_i \bar{\mathbf{y}} \tag{2.14}$$

A supercube can be compactly represented by a region map, a list of superpixels, a *gain plane* of c_i values (one per pixel) and a *bias plane* of d_i values (one per pixel). When the number of segmented regions is much less than the number of pixels, supercubes can be stored and processed much more efficiently than hypercubes.

3. Hypercube Segmentation Results

Fig.3.1 shows 5 bands of an AVIRIS hypercube of Moffett Field in Sunnyvale CA, with 256 rows and 256 columns of 16 bit pixels. The associated broadband image obtained by averaging all 224 bands is also shown. AVIRIS hypercubes contain visible and near-infrared (VNIR) data with a wavelength sensitivity range of approximately 400-2500 nm [1].

(a) (b)

Fig.3.1 (a) Bands 20, 70, 120, 170 and 220 (from left to right) of an AVIRIS hypercube of Moffett Field in Sunnyvale CA. (b) AVIRIS broadband image.

Fig.3.2 shows the threshold spectrum for the hypercube in Fig.3.1. Recall from section 2.4 that the threshold spectrum is derived from *normalized* hypercube data, so its units are *not* the same as the units of the original data. The scale factor α in equation (2.5) was chosen to be the 75th percentile of the scale factor sample distribution derived from 65536 pairs of adjacent hypercube pixels. Notice that the lower bands tend to have more noise.

Fig.3.3 shows the region maps produced when the segmentation algorithm described in section 2 was applied to the AVIRIS hypercube in Fig.3.1. All segmentations were based on the threshold spectrum shown in Fig.3.2. However, the degree of connectivity relaxation applied to region growing was allowed to vary. Specifically, connectivity relaxations of 1, 4, 8 and 16 pixels were applied, each of which resulted in a different region map. Fig.3.3 depicts only the 64 regions with the largest numbers of pixels in different colors. In each case, the number of regions is much greater than 64, but all pixels that do not belong to one of the 64 largest regions are depicted as black. It is interesting to note that some of the larger regions appear to contain solely pixels of water, whereas others appear to contain solely pixels of pavement.

Notice that the amount of black in the region maps of Fig.3.3 decreases as the degree of connectivity relaxation increases. This indicates that as the connectivity constraint is relaxed, the number of regions tends to decrease. Table 3.1 quantifies how the number of regions decreases as the degree of connectivity relaxation increases. Unfortunately, the time it takes to segment a hypercube tends to increase with the degree of connectivity relaxation. The run-times recorded in Table 3.1 are for IDL code (which is highly sub-optimal in terms of speed) running on a PC with one 450 MHz Pentium II processor.

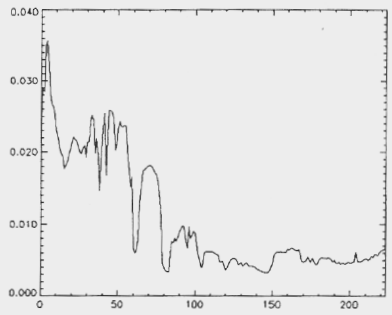

Fig.3.2 Threshold spectrum vs. spectral band for AVIRIS hypercube.

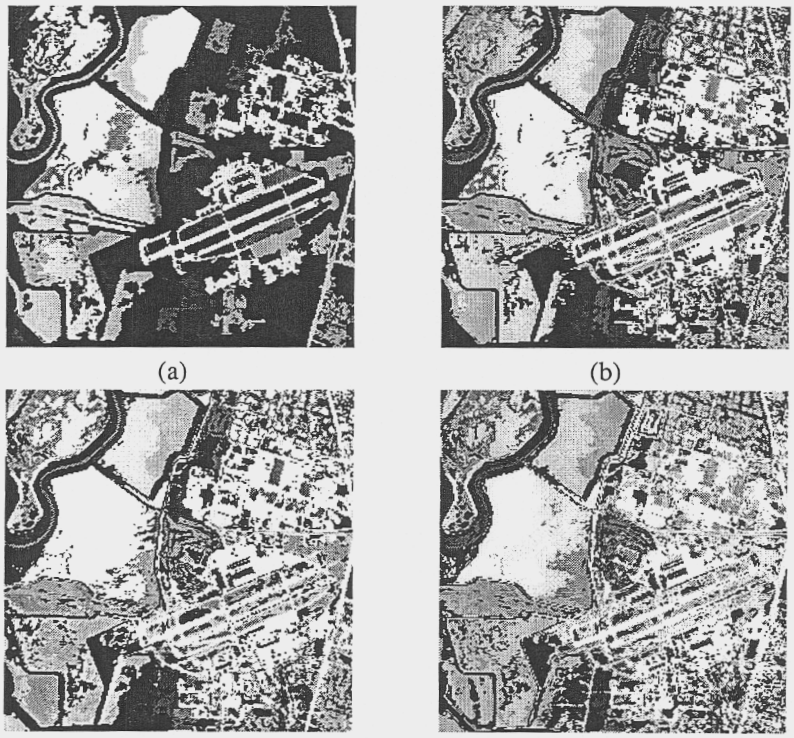

Fig.3.3 Region maps of the largest 64 regions that the AVIRIS hypercube was segmented into for connectivty relaxations of (a) 1 (b) 4 (c) 8 and (d) 16 pixels. The black pixels do not belong to any of the largest 64 regions.

Table 3.1
AVIRIS Hypercube Segmentation Results

Connectivity Relaxation (Pixels)	*Number of Regions*	*Number of Pixels / Number of Regions*	*Segmentation Run Time (s)*	*MSE*
1	6618	9.9	88	64.9
4	2587	25.3	115	74.6
8	1627	40.3	180	76.5
16	1020	64.3	385	76.0

The effect of connectivity relaxation on the number of segmented regions is further emphasized in Fig.3.4, which displays log plots of region size (for regions sorted in order of decreasing size) as a function of varying degrees of connectivity relaxation. Notice that many regions contain only a single anomalous pixel. However, as the degree of connectivity relaxation increases, more small regions get absorbed into larger regions.

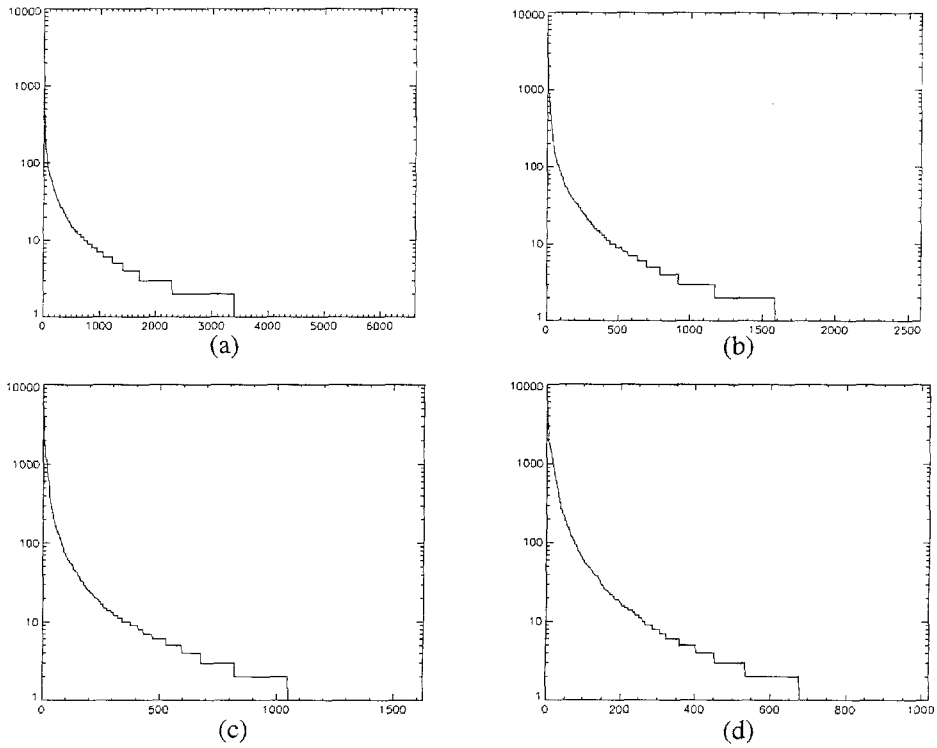

Fig.3.4 Log plots of region sizes (in pixels) sorted in descending order for connectivity relaxations of (a) 1 (b) 4 (c) 8 and (d) 16 pixels.

Finally, Fig.3.5 shows the discrepancy between a single pixel from the original hypercube in Fig.3.1 and the corresponding pixel from its supercube. Recall that supercube pixels are aggregations of hypercube pixels (i.e., superpixels) to which an optimal gain and bias to target pixels from the original hypercube have been applied. In general, there will be a discrepancy between a hypercube pixel and its superpixel model. The discrepancy will tend to be somewhat larger in bands that have more noise (see Fig.3.5(b) for an example).

To get an indication of the degree of fit between hypercube pixels and supercube pixels, one can compute the mean-squared error (MSE) between each element in the hypercube and its counterpart in the supercube. The MSE is recorded in Table 3.1 for 4 different degrees of connectivity relaxation. To put the MSE values into context, the elements of the hypercube in Fig.3.1 range in value from -21 to 16886, with a mean of 1138. The MSE values in Table 3.1 represent roughly 6% of the mean value. Alternately, on a 16 bit basis, the loss in information is confined to roughly the 6 least significant bits. The key point to be made here is that the important spectral features within the hypercube pixels are preserved in the supercube pixels.

Notice that the MSE does not tend to vary that much with the degree of connectivity relaxation. This is not surprising, since the threshold spectrum does not change. However, as the degree of connectivity relaxation increases, the number of regions tends to decrease, which increases the compactness of the hypercube representation.

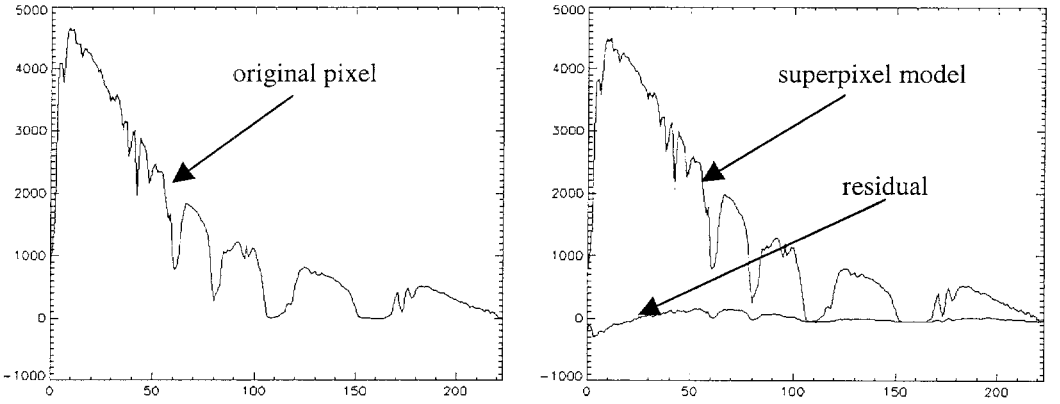

Fig. 3.5 (a) Pixel at (column,row) = (170,173) from original AVIRIS hypercube.
(b) Superpixel model of pixel in (a) and discrepancy between model and original.

4. Conclusions

A novel computationally feasible algorithm that segments hypercubes into regions has been demonstrated to produce compact hypercube representations while preserving the spectral features of hypercube pixels required for successful analysis. One can further reduce the number of regions (and hence produce more compact representations) by increasing connectivity relaxation (which also increases computational cost) or by merging regions resulting from modest connectivity relaxations (which is much faster and an item for future development).

5. References

[1] "Airborne Visible / Infrared Imaging Spectrometer (AVIRIS)", JPL Pub. 87-38, Gregg Vane, Ed., November 15, 1987.

[2] Bolstad, P. V. and Lillesand, T. V., "Rapid Maximum Likelihood Classification", Photogram. Eng. Remote Sensing, Vol.57, 1991, pp.64-74.

[3] Bouman, C. and Shapiro, M., "A Multiscale Random Field Model for Bayesian Image Segmentation", IEEE Trans. Image Processing, Vol.3, No.2, March 1994, pp.162-177.

[4] Horowitz, S. L. and Pavlidis, T., "Picture Segmentation by a Directed Split-and-Merge Procedure", Proc. 2nd IJCPR, August 1974, pp.424-433.

[5] Maselli, F., Conese, C., Petkov, L. and Resti, R., "Inclusion of Prior Probabilities Derived from a Nonparametric Process into the Maximum Likelihood Classifier", Photogram. Eng. Remote Sensing, Vol.58, 1992, pp.201-207.

[6] Muerle, J. L. and Allen, D. C., "Experimental Evaluation Techniques for Automatic Segmentation of Objects in a Complex Scene", in Pictorial Pattern Recognition (G. C. Cheng et al., Eds.), Thompson, Washington, 1968.

[7] Serpico, Sebastiano B. and Roli, Fabio, "Classification of Multisensor Remote Sensing Images by Structured Neural Networks", IEEE Trans. Geosci. Remote Sensing, Vol.33, no.3, May 1995, pp.562-578.

[8] Zucker, S. W. "Region Growing: Childhood and Adolescence", CGIP, Vol.5, No.3, September 1976, pp.382-399.