IMAGE SEGMENTATION BY ITERATIVE PARALLEL REGION GROWING WITH APPLICATIONS TO DATA COMPRESSION AND IMAGE ANALYSIS

James C. Tilton

Mail Code 636
NASA Goddard Space Flight Center
Greenbelt, MD 20771

ABSTRACT

Image segmentation can be a key step in data compression and image analysis. However, the segmentation results produced by most previous approaches to region growing are suspect because they depend on the order in which portions of the image are processed. An iterative parallel segmentation algorithm avoids this problem by performing the globally best merges first. After a background section, this paper describes such a segmentation approach, and two implementations of the approach on NASA's Massively Parallel Processor (MPP). Application of the segmentation approach to data compression and image analysis is then described, and results of such application are given for a Landsat Thematic Mapper image.

Keywords: Image segmentation, Image Analysis, Data compression, Data parallel analysis.

BACKGROUND

Segmentation is the process of partitioning images into constituent parts called regions using image attributes such as pixel intensity, spectral values, and textural properties. Image segmentation produces an image representation in terms edges and regions of various shapes and interrelationships.

Image segmentation is a key step in many approaches to data compression and image analysis. An optimal coding of an image segmentation, such as through a region label map and region feature file, can be used to effect data compression (see Ref. 3). Image analysis can be performed on an image segmentation by using the shape, texture, spectrum, etc. of the regions found by the image segmentation and interrelationships between the regions. This region based analysis of imagery is potentially more effective than pixel based analysis, because region based analysis exploits spatial information whereas pixel based analysis does not.

Most image segmentation approaches can be placed in one of three classes: (i) characteristic feature thresholding or clustering, (ii) boundary detection, and (iii) region extraction. Characteristic feature thresholding or clustering is often ineffective because it does not exploit spatial information. Boundary detection does exploit spatial information through examining local edges found throughout the image. For simple noise-free images, detection of edges results in straightforward boundary delineation. However, edge detection on noisy, complex images often produces missing edges and extra edges which cause the detected boundaries to not necessarily form a set of closed connected curves that surround connected regions. One way to overcome this problem is to combine region extraction and boundary detection. Ref. 2, reports on some experiments in combining boundary detection approaches with the iterative parallel region growing approach discuss here.

Early approaches to region extraction (usually by region growing) had the disadvantage that the regions produced depended on the order in which portions of the image are processed. But Schachter, *et al* (Ref. 1) suggest that implementing region growing as "an iterative parallel process" would overcome the order dependent problem. This is the approach taken by the iterative parallel image segmentation algorithm presented here.

ITERATIVE PARALLEL REGION GROWING

The basic concept behind our iterative parallel segmentation approach is to perform the globally best merges first. With this approach, the whole image is processed in parallel, eliminating the order dependence problem that troubled earlier approaches to region extraction by region growing.

The globally best merge is defined as follows. A similarity criterion is calculated for all pairs of spatially adjacent regions in the image. The globally best merge is the merge of the pair of spatially adjacent regions with the best similarity criterion value over the entire image (*i. e.*, the most similar pair of regions). (NOTE: For convenience, we assume from this point that the best similarity criterion value is the *minimum* similarity criterion value.)

Since only spatially adjacent regions can group together in this approach, we call our approach the Spatially Constrained Clustering (SCC) algorithm. The basic SCC algorithm is as follows:

i. Initialize the segmentation process by labeling each pixel as a separate region.
ii. Calculate a similarity criterion between each pair of spatially adjacent regions.
iii. Find the minimum similarity criterion measure value for the entire image.
iv. Check for convergence by projecting if the proposed merge would produce an error larger than the error threshold. If converged, stop. Otherwise continue on to step v.
v. Merge pairs of regions with the minimum similarity criterion measure value.
iv. If the number of regions remaining in the image is less than the preset minimum, stop. Otherwise return to step ii.

Two different versions of the SCC algorithm have been implemented (see next section) that differ only in how step v is handled. The serial merge version is:

v^s Merge a single pair of regions with the minimum similarity criterion measure value (break ties arbitrarily).

The parallel merge version of the SCC algorithm implements step v as:

v^p Merge all pairs of regions with similarity criterion equal to or less than $1 + \delta$ times the minimum similarity criterion.

When $\delta = 0$, the parallel merge version is still an exact implementation of the basic SCC algorithm. It is only different from the serial merge implementation in that ties are not broken when more than one pair of regions have the minimum similarity criterion value. All such regions are merged (in parallel). For $\delta > 0$, the parallel merge version becomes an approximation of the basic SCC algorithm. Using $\delta > 0$ speeds convergence with the cost of finding a less optimal segmentation.

For either the serial or parallel merge version, the algorithm is considered to have converged when either a desired number of regions remain, or when no pair of adjacent regions is similar enough to be merged according to a predefined bound on the similarity criterion.

A key aspect of any region growing approach is the similarity criterion employed. The optimum similarity criterion depends upon the application. To fully explore the utility of the general SCC approach, we will need to devise and test several different similarity criteria for different types of image data and for various analysis procedures performed on each type of image data. In the experiments reported here, the similarity criterion used is based on minimizing variance normalized mean squared error.

The Mean Square Error (MSE) of band "i" of a multiband image is defined as

$$MSE_i = E[(D_i - D_i^r)^2] \cong \frac{1}{N-1} \sum_{p=1}^{N} (D_{ip} - D_{ip}^r)^2 \quad (1)$$

where D_i and D_i^r are the data values of the i^{th} band of the original and reconstructed images, respectively; D_{ip} and D_{ip}^r are the values of the p^{th} pixel of the i^{th} band of the original and reconstructed images, respectively; E denotes the expected value; and N is the total number of pixels in the image.

The variance normalized mean squared error for band "i" ($NMSE_i$) is defined as

$$NMSE_i = \frac{MSE_i}{VAR_i} \quad (2)$$

where VAR_i is the variance of band "i". The similarity criterion used in our tests is the $\underset{i}{MAX}(\Delta NMSE_i)$ for each pair of spatially adjacent regions, where the maximum is taken over all bands ($1 \le i \le m$). (Optionally, the similarity criterion can be taken as $\sum_{i=1}^{m} (\Delta NMSE_i)$.) For a particular pair of spatially adjacent regions, $\Delta NMSE_i$ is the change in $NMSE_i$ when the pair of regions is merged and the reconstructed image is formed by substituting the mean vector of each region for the multispectral radiance values of each pixel in the region.

The change in $NMSE_i$, or $\Delta NMSE_i$, is calculated as follows:

$$\Delta NMSE_i = \frac{MSE_i^c - MSE_i}{VAR_i} \quad (3)$$

where MSE_i^c is the mean squared error when regions j and k are merged, while MSE_i is the mean squared error before regions j and k are merged. Using the definitions of MSE_i and the region mean, it is easy to derive a more fundamental version of equation (3), *viz*

$$\Delta NMSE_i = \frac{n_j (\bar{D}_{ij} - \bar{D}_{ijk})^2 + n_k (\bar{D}_{ik} - \bar{D}_{ijk})^2}{(N-1)VAR_i} \quad (4)$$

where n_j and n_k are the number of points in regions j and k, respectively, before combining, and N is the number of points in the image. $\bar{D}_{ij}$ and $\bar{D}_{ik}$ are the mean values of band i for regions j and k, respectively, before combining, and $\bar{D}_{ijk}$ is the mean value of band i for the region that would result from combining regions j and k.

IMPLEMENTATION ON THE MPP

We have implemented the serial and parallel merge versions of the SCC algorithm on the Massively Parallel Processor (MPP) at the NASA Goddard Space Flight Center. For a description of the MPP see Ref. 4. Both implementations use the staging memory extensively to allow the processing of multispectral images of up to 512-by-512 pixels and up to 12 bands. Without the staging memory, either implementation would be restricted to a 128-by-128 4-band image, or a 128-by-256 2-band image or a 128-by-384 single band image because of the local

array memory limitations of the MPP. While the use of the staging memory makes possible the processing of reasonably large multispectral images, this use does extract a penalty in the terms of processing time for the data transfers between the staging memory and array memory. We estimate that for a 7-band, 256-by-256 pixel image, the parallel merge version of the SCC algorithm would execute 10 times faster on an MPP with sufficient local array memory to eliminate the need for extensive stager-array data movements.

The implementation of the serial merge version of the SCC algorithm (using step v^s) on the MPP is extremely straightforward. The initialization is trivial, and local neighborhood data movements are used in step ii to calculate in parallel the similarity criterion for spatially adjacent regions. (For images larger than 128-by-128 pixels, a virtual MPP of up to 512-by-512 processors is emulated by data rotates across the edges of the 128-by-128 array and masked assignments.) In step v^s, a single pair of regions is identified for merging. (When more than one pair of regions has similarity function value equal to the minimum, the pair of regions with a minimum region label value is chosen.) The feature values (number of pixels and mean vector) for this pair of regions is extracted from the array, and new feature values are calculated in scalar mode for the new region. The merged region is given a new region label equal to the minimum of the two region labels, and the feature values are assigned to the merged region using a masked assignment.

The implementation of the parallel merge version of the SCC algorithm (using step vP) on the MPP is more complicated than the serial version. In order to merge more than one pair of regions in parallel in step vP, we need to resort to more than just local neighborhood data movements and masked assignments. The method we chose is as follows. First perform all the merging on the region label level. This is done through parallel region label propagation keyed on the similarity criterion function values. Once the new region label map is established, the new region feature values (number of pixels and mean vector) need to be calculated. In order to do this in parallel we grow a tree from a single pixel (seed pixel) in each region until it covers every region completely. (A unique seed pixel can be identified in region by comparing the current region label map with the initial region label map.) Then the number of pixels and sum of the data values at each pixel in each region are accumulated by tracing back up each tree. All region means are then calculated at each seed pixel, and the feature values for each region are broadcast out to each pixel in each region by traveling back down each tree, and depositing the feature values at each node of each tree.

APPLICATION TO DATA COMPRESSION AND IMAGE ANALYSIS

An image segmentation can be a key step in a lossy data compression process. This type of data compression is a variant upon an image data compression process often referred to as vector quantization. In this form of data compression, each region in an image segmentation is given a unique label, and a list is generated of feature values corresponding to each region. This region label map and feature list is then encoded by a lossless compression scheme. For a more detailed discussion of this process, see Ref. 3.

The amount of information lost by this lossy data compression process is determined by how well the segmented image represents the original image. If the key region feature is taken to be the multispectral mean vector for each region, the effect of this data compression an image can measured by calculating the Root Normalized Mean Squared Error (RNMSE), which we define as follows:

$$\text{RNMSE} = \frac{1}{m} \sum_{i=1}^{m} \sqrt{\text{NMSE}_i} \,. \qquad (5)$$

The Normalized Mean Squared Error of band "i", NMSE_i, was defined in equation (2). The RNMSE carries the following intuitive interpretation: The RNMSE is the band average of the single-band RNMSE, which can be regarded as the mean deviation of a reconstructed image pixel value from the corresponding original image pixel value per standard deviation of the band.

An image segmentation can also be used as a first step in an image analysis scheme. As mentioned before, image analysis can be performed on an image segmentation by using the shape, texture, spectrum, etc. of the regions found by the image segmentation, and by the interrelationships between the regions. Whereas the more complicated shape, texture and interrelationship analysis have the greatest analysis potential, we will demonstrate here how even a simple analysis approach using spectral information alone - the Maximum Likelihood Classifier - can be improved by proceeding it with an image segmentation step.

EXPERIMENTAL RESULTS

A 256-by-256, 7-band subset of a Landsat Thematic Mapper (TM) image over Ridgely, Maryland was used as a test data set for this study. For this test, we processed the TM image with the parallel merge SCC algorithm. We first used a value 0.5 for δ and stopped the segmentation process when the total remaining number of regions was $\leq$ 2.5% of the number of pixels in the original image (1486 regions). Then we restarted the algorithm and processed from that point with a δ value of 0.1 until the number of regions was $\leq$ 2.0% of the number of pixels in the original image (1299 regions). (This produced better results than processing all the way down to 2.0% with a δ of 0.5.)

Figure 1 (color plate VII, p. 699) shows the original and segmented images, along with the difference image (plus a bias) between the original and segmented images (bands 2, 4 and 5 of the 7-band image are displayed). A subjective evaluation of the segmented image reveals that areas in the original image that are relatively homogeneous, but not neces-

sarily uniform, become completely uniform in the segmented image. Low contrast spatial features are often lost in the segmented image, but higher contrast spatial features, such as edges of regions, are retained very precisely. Even very small spatial features are retained if they have sufficient contrast relative to the surrounding area.

The RNMSE image quality measure for segmented image in figure 1 is 0.33. That is, the mean deviation of an image pixel value in the segmented image from the corresponding original image pixel value per standard deviation of each band is 0.33.

The segmented image was encoded into region label map and a region feature files, and the region label map was losslessly compressed using run-length encoding. This segmentation/run-length encoding combination produced a data compression ratio of 13.1 to 1. (A optimal lossless compression technique may produce an even higher compression ratio). Optimal lossless encoding of the original TM image data typically produces a compression ratio of 3 to 1 or less (see Ref. 5).

We tested an image analysis approach where the segmented image was classified by a simple Maximum Likelihood Classifier. This analysis result was compared with the result obtained by using the same classifier on the original image. (For a more detailed description of the test setup see Ref. 3.)

The classification results for the original and segmented image are given in figure 2 (color plate VII, p. 699) and Table 1. The classification accuracies are consistently better for the segmented image than they were for the original data! We hypothesize that the segmentations produced by the SCC algorithm encode information from the surrounding regions of the image in each pixel. The MLC classification results are improved because each pixel has knowledge of its spatial surroundings in the segmented image.

Table 1. Accuracy comparison (% correct classification) between classifications of the original and segmented TM images.

	Classification	
Class	Original Image	Segmented Image
Water/Marsh	73.7%	79.3%
Forest	74.8%	75.6%
Residential	54.4%	64.9%
Ag./Dom. Grass	81.9%	83.4%
OVERALL	79.2%	80.9%

The first ten iterations of the parallel merge version took 118 seconds to perform 6192 merges. The serial merge version would need 6192 iterations to perform 6192 merges. In an actual test, the serial merge version took 2913 seconds to perform 6200 merges. This means that the parallel merge version performed the first 6192 merges nearly 25 times faster than the serial merge version. The last ten iterations of the parallel merge version took 2174 seconds to perform 164 merges. We estimate that the serial merge version would take roughly 250 seconds to perform those 164 merges. Thus, the serial merge version would have performed those last 164 merges better than 6 times faster than the parallel merge version did them. For this data set, it would have been most efficient to use step vP for 138 iterations (resulting in 60,037 merges), and switch to step v^s for the remainder of the processing (to do the last 4,013 merges at one merge per iteration).

The parallel merge version took 4.6 hours to produce the segmentation shown in Figure 1. The serial merge version would have taken an estimated 8.4 hours to do the same number of merges. An optimal parallel merge/serial merge combination would have taken an estimated 2.4 hours. Further, such a combined implementation on an MPP-like machine with sufficient local array memory for all data and variables would take roughly 15 minutes (assuming the estimated 10 times speed-up mentioned earlier.) Clearly, the best way to implement this iterative parallel region growing approach is a parallel merge/serial merge combination on an MPP-like machine with significantly more local array memory. Within the coming year, we hope to have made such an implementation on AMT's DAP 610.

An ultimate segmentation goal would be to find the globally best image segmentation for a given similarity criterion and number of regions. Our iterative parallel region growing approach can only approximate this desired result. Fortunately, for many applications an approximate result may be sufficient. Nevertheless, we are seeking improvements to our SCC algorithm. One such improvement would be to allow pixels split out of regions when appropriate. We eventually plan to explore neural network optimization as an approach that could actually produce the globally best image segmentation.

REFERENCES

1. B. J. Schachter, L. S. Davis and A. Rosenfeld, "Some Experiments in Image Segmentation by Clustering of Local Feature Values," Pattern Recognition, Vol. 11, No. 1, pp. 19-28, 1979 .
2. M. Manohar, H. K. Ramapriyan and J. P. Strong, "Parallel Algorithms for Determining Motion Vectors in Ice Floe Images by Matching Edge Features," Proc. of the 2nd Symposium on the Frontiers of Massively Parallel Computation, Fairfax, VA, Oct. 10-12, 1988.
3. J. C. Tilton and H. K. Ramapriyan, "Data Compression Experiments with Landsat Thematic Mapper and Nimbus-7 Coastal Zone Color Scanner Data," Proc. of the Scientific Data Compression Workshop, Snowbird, UT, May 3-5, 1988.
4. K. E. Batcher, "Design of a Massively Parallel Processor," IEEE Trans. on Computers, Vol. C-29, No. 3, pp. 836-840, 1980.
5. T. M. Chen, D. H. Staelin and R. B. Arps, "Information Content Analysis of Landsat Image Data for Compression," IEEE Trans. on Geoscience and Remote Sensing, Vol. GE-25, No. 4, pp. 499-501, 1987.