Reprinted from

Tenth International Symposium

Machine Processing of

Remotely Sensed Data

with special emphasis on

Thematic Mapper Data and

Geographic Information Systems

June 12 - 14, 1984

Proceedings

Purdue University
The Laboratory for Applications of Remote Sensing
West Lafayette, Indiana 47907 USA

ANALYZING REMOTELY SENSED DATA ON THE MASSIVELY PARALLEL PROCESSOR

J.C. TILTON

Science Applications Research
Riverdale, Maryland

J.P. STRONG III

National Aeronautics and Space
Administration/Goddard Space Flight Center
Greenbelt, Maryland

ABSTRACT

The new earth resources observation satellites of the 1980's will provide imagery with much higher information content than provided by the Landsat Multispectral Scanners (MSSs) of the 1970's. Concurrent with the increase of information content in data from earth resources observation satellites has been an increase in (ground based) computer processing capability. Of particular relevance to image processing and classification is the development of several large parallel processors, the largest of which currently is the Massively Parallel Processor (MPP). Large parallel processors such as the MPP will foster the rethinking of applications algorithms in parallel to increase computational speeds several orders of magnitude and allow the development of algorithms which previously have been avoided because of their computationally intensive nature. The development of image processing software for the MPP is being greatly facilitated by the earlier development of Parallel Pascal, a high level language for the MPP, which we discuss briefly. We will discuss results from the recoding for the MPP of several existing per pixel image analysis schemes, including ISODATA, the Maximum Likelihood Classifier and the Fast Fourier Transform. However, the most exciting prospects for processing remotely sensed data with the MPP is the development of algorithms which exploit spatial information through utilizing neighborhoods of pixels. In this vein, we will describe the implementation on the MPP of a Spatially Constrained Clustering (Image Segmentation) scheme and a Contextual Classifier.

1. INTRODUCTION

The new earth resources observation satellites of the 1980's will provide imagery with much higher information content than provided by the Landsat Multispectral Scanners (MSSs) of the 1970's. Concurrent with the increase of information content in data from earth resources observation satellites has been an increase in (ground based) computer processing capability. Of particular relevance to image processing and classification is the development of several large parallel processors. The largest such operational processor is the Massively Parallel Processor (MPP), which was built by Goodyear Aerospace for the NASA Goddard Space Flight Center. (Batcher, 1980 and Schaefer et al, 1982) The MPP is a Single Instruction, Multiple Data stream (SIMD) computer containing 16,384 bit serial microprocessors logically connected in a 128-by-128 array with each element having direct data transfer interconnections with its four nearest neighbors. With of this novel architecture, the MPP is capable of billions of operations per second. The MPP is being used at NASA's Goddard Space Flight Center for developing applications software for solving problems in image analysis, synthetic aperture radar processing, weather modeling and other applications in which the parallel architecture can result in greatly increased computational speeds. (Burkley and Mickelson, 1983)

The availability of the MPP and other large parallel processors will foster the rethinking of applications algorithms in parallel to increase computational speeds several orders of magnitude and allow the development of algorithms which previously have been avoided because of their computationally intensive nature. Development of applications software for the MPP has

followed closely the construction and testing of the MPP hardware. This software development has been facilitated greatly by the earlier development of Parallel Pascal, a high level language for the MPP (and potentially other SIMD computers) which we discuss briefly below.

The higher information content of imagery provided by the newer earth resources observation satellites implies substantially more massive data volumes that could easily overwhelm analysis algorithms currently implemented on serial computers. The throughput of these current operational image processing and classification algorithms can be greatly increased by rethinking them in parallel and implementing them on the MPP. We will discuss results from the recoding for the MPP of several existing per pixel image analysis schemes, including ISODATA, the Maximum Likelihood Classifier and the Fourier Transform, and compare the serial and parallel versions in terms of computation speed.

The most exciting prospects for processing remotely sensed data with the MPP is the development of algorithms which exploit spatial information by utilizing neighborhoods of pixels. The new generation of earth resources observation satellites will produce imagery with much higher spatial information content, as well as higher spectral and radiometric information content. The MPP will be a key research tool in developing analysis algorithms that, in particular, effectively extract spatial information from earth resources imagery.

Present operational image processing and classification algorithms for remote sensing (earth resources) applications primarily use spectral information and generally make little use of spatial information (an exception is ECHO (Kettig and Landgrebe, 1976)). These algorithms may have been adequate for relatively low resolution Landsat MSS imagery, but they are not adequate for the higher resolution data provided by the Landsat Thematic Mapper (TM). Using these current serial techniques, several studies have shown that the 30-meter resolution of the TM data tends to confuse these per pixel algorithms, producing poorer classification results with TM data than with 80-meter resolution MSS data. (Markham and Townshend, 1981, Williams et al, 1983, and Alexander et al, 1983) Researchers theorize that TM data would produce much better classification results if the classification algorithms used would exploit spatial information as effectively as current techniques exploit spectral information.

In the past it has proved difficult to implement algorithms exploiting spatial information because of the constraints imposed by the available serial computers. Fortunately, parallel computers such as the MPP are well suited for algorithms that exploit spatial information through utilizing neighborhoods of pixels. In this vein, we will describe the implementation on the MPP of a Spatially Constrained Clustering (Image Segmentation) scheme and a Contextual Classifier. As we will see, when implemented serially these algorithms are impractical (in terms of processing times) for all but the smallest image sizes, but when implemented in parallel (e.g. on the MPP) they become practical for common image sizes.

II. PARALLEL PASCAL

In order to fully exploit the capabilities of the MPP and make it a useful tool to not only the image processing researcher, but also the remote sensing applications scientist, a high priority was given to the development of a high level language to be used on the MPP. The first high level language developed, known as Parallel Pascal, is an extended version of the popular Pascal programming language. Developed by Reeves and Bruner* at Purdue University, Parallel Pascal was designed to be easy to use, portable, efficient, and was designed to provide good error detection. (Reeves and Bruner, 1980) The parallel extensions are a small number of carefully chosen features with an eye toward making Parallel Pascal portable even among the wide variety of parallel architectures. These extensions to Pascal are few and straightforward to allow the declaraton of parallel array, a few parallel manipulation and indexing functions and extended control structures.

Parallel arrays are explicitly declared in Parallel Pascal by affixing the word PARALLEL to the standard Pascal array declaration, viz.:

---

* A. P. Reeves is now with the School of Electrical Engineering, Cornell University, and J. D. Bruner is now with the Lawrence Livermore Laboratory.

name: PARALLEL ARRAY[index range] OF type

This explicit declaration of parallel arrays provides the programmer direct control over which arrays are operated on in parallel. The last two dimensions of parallel arrays are restricted in Parallel Pascal to 128-by-128 (or the last dimension to 16,384) to conform with the architecture of the MPP.

Operationally, parallel arrays are different from standard Pascal arrays in that parallel arrays can be added, divided, compared, etc. as aggregate units rather than element-by-element. Special indexing mechanisms are provided to selectively operate on particular portions of a parallel array. Standard elemental functions, listed in Table 1, perform the same operation on each element of a parallel array, independently and in parallel. Special transformational functions, listed in Table 2, perform transformations upon the entire parallel array.

Table 1: Elemental Functions (from (Reeves and Bruner, 1982))

syntax	meaning
	type conversions
trunc(x)	truncate real to integer
round(x)	round real to integer
ord(x)	ordinal value of x (for scalar types)
chr(x)	character with ordinal value x
	arithmetic functions
abs(x)	absolute value
sqr(x)	square
exp(x)	exponential
ln(x)	natural logarithm
sin(x)	sine function
cos(x)	cosine function
arctan(x)	arctangent function
	miscellaneous
odd(x)	boolean: true if x is odd
eof(f)	boolean: true if at end-of-file on file f
eoln(f)	boolean: true if at end-of-line on file f
succ(x)	successor of x (if defined)
pred(x)	predecessor of x (if defined)

Table 2: Transformational Functions (from (Reeves and Bruner, 1982))

syntax	meaning
shift(array,D1,D2,...,Dn)	end-off shift data within A
rotate(array,D1,D2,...,Sn)	circularly rotate data within A
expand(array,dim,size)	expand array along specified dimension
transpose(array,D1,D2)	transpose 2 dimensions of A
sum(array,D1,D2,...,Dn)	reduce* array with arithmetic sum
prod(array,D1,D2,...,Dn)	reduce* array with arithmetic product
all(array,D1,D2,...,Dn)	reduce* array with boolean AND
any(array,D1,D2,...,Dn)	reduce* array with boolean OR
max(array,D1,D2,...,Dn)	reduce* array with arithmetic maximum
min(array,D1,D2,...,Dn)	reduce* array with arithmetic minimum

* Do the indicated operation over all specified dimensions of the array reducing the dimensionality by the specified number of dimensions. E.g., for two-dimensional array, "A", sum(A,1,2) takes sum over both dimensions of array, resulting in a scalar value.

III. ISODATA

The development of the MPP was driven by the need for ultra high speed image processing of data form satellite sensors. One of the first image processing algorithms to be coded for running on the MPP, the ISODATA algorithm was chosen as a benchmark of the MPP's processing power. (Slotnik, 1977-1980) The MPP implementation of ISODATA works in the following way:

1. Given a set of class mean values, determine the N-dimensional Euclidean distance between the class mean and all pixels in the image.

2. Assign image pixels to the class with the shortest distance.

3. Re-compute class means based on the arithmetic means of pixels in the class.

4. Repeat steps 2 through 4 until the class means stabilize.

This early MPP version of ISODATA is a simple implementation, there is no combining of close clusters and no cluster splitting. Furthermore, there is no point weighting with MPP ISODATA. (A later MPP version of ISODATA is complete, including point weighting and cluster combining and splitting.)

Speed and efficient use of the MPP architecture were primary concerns, therefore the ISODATA algorithm was coded in Main Control Language (MCL) embedded within the structure of a "C" program. (Reeves and Bruner, 1980) As a result a 512-by-512 image, with four bands, 8 bit pixels with 16 classes required 20 seconds on the MPP. A comparable time on a VAX-11/780 is 7 hours.

IV. MAXIMUM LIKELIHOOD CLASSIFIER

The conventional maximum likelihood classifier is a spectral per point classifier. The algorithm classifies a multi-band input image into spectral classes according to spectral class statistics constructed from training site data, using a maximum likelihood decision rule. When each class is assumed to have equal prior probability, a pixel X is classified into the class i if:

$$gi(X) >= gj(X) \text{ for all } j=i$$

where gi(X), the discriminant function for class i, is given as:

$$gi(X) = \ln(Di) - (X-Mi)^T * Ci * (X-Mi)$$

where

Ci = Covariance matrix for class i
Di = Determinant of covariance matrix for class i
Mi = Mean vector for class i

In the parallel implementation of this algorithm, written in Parallel Pascal, all image pixels within a 128-by-128 array are classified simultaneously. The final implementation of the maximum likelihood classifier is not yet complete, so computation time can only be estimates. Ramapriyan and Strong estimate that the MPP can classify a 512-by-512 4 band image in 0.5 seconds, a full (Landsat) Multispectrial Scanner (MSS) scene in 0.27 minutes, and a full (Landsat) Thematic Mapper scene in 2.5 minutes. Similar processing times utilizing a Floating Point Sytems AP180V array processor are 107 seconds for a 512-by-512 4 band image, 72 minutes for a full MSS scene, and 734 minutes for a full TM scene. (Ramapriyan and Strong, 1983)

V. 2-D FAST FOURIER TRANSFORM

A 2-D Fast Fourier Transform (FFT) using 10-bit integer data was implemented earlier. (Slotnik, 1977-1980, Slotnik, 1981) This version can compute transforms on images of size 2 -by-2 where $1<n<7$. We are currently designing an efficient 2-D FFT implementation which can use 32-bit real (64-bit complex) data and which will work for image sizes larger than n=7 (larger than 128-by-128). The final details of this implementation were still being worked out at the writing of this paper, but hopefully more complete description of the MPP implementation of this algorithm will be available for discussion at the symposium.

The Fourier transform uses spatial information in the sense that the transform result can be interpreted as "spatial frequencies." In the following sections we discuss the implementation on the MPP of algorithms which exploit spatial information through utilizing neighborhoods of pixels. The first spatially oriented algorithm we discuss is spatially constrained clustering.

VI. SPATIALLY CONSTRAINED CLUSTERING

ISOCLAS and most other clustering algorithms completely ignore the spatial location of each pixel in an image or transformed image* in the clustering process. In contrast, the relative spatial location of image pixels is directly used to constrain the clustering process in our spatially constrained clustering (SCC) algorithm. This use of image pixels' relative locations is supported in Schacter et al's study of image segmentation through clustering of local feature values. (Schacter et al, 1979) After examining several approaches to clustering images and transformed images, they were dissatisfied with their results and suggested that "if we want to obtain better segmentation performance, we must make use not only of similarities among the image points, but also of their relative positions." They suggest doing a preliminary segmentation based on feature space clusters and then taking "the points belonging to the clusters as 'core points' of image regions and ...[completing] the segmentation of the image by a region growing process starting from these points."

Schachter et al and other researchers (e.g. (Fu and Mui, 1981)) point out that region growing approaches have had the disadvantage that the regions produced depend on the order in which portions of the image are processed. But Schachter et al suggest that implementing region growing as "an iterative parallel process" would overcome the order dependent problem. The core of our SCC algorithm consists of alternate iterations of parallel region growing and parallel region switching.

Prior to region growing and region switching, the SCC algorithm is initialized with some initial segmentation of the data. One possible initial segmentation could be based on feature space clustering, as suggested by Schachter et al above. Another very simple initialization is a segmentation of the image into n-by-n regions, where n>=2. Whatever the source of the initial segmentation, the initial segmentation is represented in terms of n-by-n subregions of pixels from the original data. An m-by-m array of subregions (each sized n-by-n) is formed from the n*m-by-n*m data set. If we are so fortunate to have m=128, we can then load the initial region feature values directly into the MPP's 128-by-128 array of microprocessors, one region per microprocessor. If m != 128 we must do some bookkeeping, and/or cluster the image in n*m-by=m*m sections and piece together the results after all the data is processed. The feature values we use are mean, standard deviation and degrees of freedom. In addition, a region label is given to each initial region.

We now have initial region mean, standard deviation, degree of freedom and label values loaded into the 128-by-128 MPP microprocessor array. Next, all neighboring subregions are compared in terms of a predefined similarity criterion. For each neighbor ("east," "southeast," etc.), the similarity criterion is calculated in parallel, and the result is stored in another parallel array: testval. The similarity criterion we use is the geometric mean (or, equivalently, the product) of the probabilities of random occurrence of the t- and F-statistics calculated from a modified Student's t-test and modified F-ratio test, respectively. (Tilton and Cox, 1983, p.132)

Some of the microprocessors at the edge of the image will contained invalid values for testval. The value of testval is set to zero for these microprocessors. The value of testval is also set to zero for neighboring subregions that are already in the same region. As we are cycling through the "east," "southeast," "south," and "southwest" neighbors, we record the best comparison value encountered so far and the corresponding merge direction in the parallel arrays bestval and bestmdir, respectively. (We don't have to look "west," "northwest," "north," or "northeast" because these directions would be redundant.) The best merge for the entire image section of the image contained in the MPP array is then found by noting the pair of regions associated with the overall best comparison value in the bestval array. This globally best pair of regions is now merged to complete one iteration of region growing.

After a preset number of region growing iterations, region growing is invoked. In region growing all original n-by-n subregions are compared to the region to which they currently belong and

* Some clustering approaches first transform an image by calculating such features as a gradient magnitude, Laplacian, standard deviation or a texture transform before performing clustering. While these transforms may incorporate spatial information over a local window, the clustering process itself completely ignores the spatial location of the transformed pixels.

to any neighboring regions (subregions internal to a region have no neighboring region, and thus cannot participate in region switching). The comparison with the current region is done with the subregion temporarily removed from the region. The region switching criterion is the difference between the subregion comparison with the neighboring region and the subregion comparison with the current region. The subregion with the largest region switching criterion value is then switched from its current region to the indicated neighboring region. The region switching criterion is recalculated for all subregions, and the region switching process is repeated iteratively until no more region switches occur.

Region switching is necessary because as a region grows, its feature values may gradually become very different from its original feature values. When this occurs, the region feature values may become sufficiently different from the feature values of certain subregions making up the region that these subregions may become more similar to some adjacent region. The region switching process switches these subregions over to the appropriate neighboring region.

Once the region switching process completes, another iteration of region growing is performed followed by another group of region switching iterations. These alternating iterations of region growing and region switching are repeated over and over again until no pair of neighboring regions a similar enough to be merged, according to a preset threshold on the comparison test value, or until a specified number of iterations are completed.

We currently have the region growing and region switching portions of the SCC algorithm implemented in Parallel Pascal and are currently at work implementing the region splitting process (it is be similar to the region switching process). We plan to embed the basic SCC algorithm in a coarse-to-fine resolution schema to produced a Multiresolution SCC algorithm. (Tilton, 1984)

The SCC algorithm uses the full power of the MPP in the portions of the algorithm that calculate comparison tests between neighboring regions. The actual merging of a pair of regions, or the switching of region assignment is substantially a serial operation, however. Nevertheless, the SCC algorithm run significantly faster on the MPP than on any serial processor. (Firm estimates of comparable execution times were not available at this writing.)

VII. CONTEXTUAL CLASSIFIER

In the contextual approach to classification, the probable classifications of neighboring pixels influence the classification of each pixel. Classification accuracies can be improved through this approach since certain ground-cover classes naturally tend to occur more frequently in some contexts than in others.

We are investigating a contextual classification algorithm which was formulated by Swain et al and further developed by Tilton et al. (Swain et al, 1981, Tilton et al, 1982) Here compound decision theory is invoked to develop a classification method which exploits spectral/spatial information. A serial implementation of the Swain-Vardeman-Tilton (SVT) contextual classifier takes an impractical amount of computer time even for moderately sized imagery (over 64-by-64 pixels). In this section we describe an implementation on the Massively Parallel Processor (MPP), which makes it possible to process reasonably sized images (e. g. 1024-by-1024) in a reasonable amount of time.

The decision rule for the SVT contextual classifier was derived in (Swain et al, 1981). We will not repeat that derivation here. We just present the decision rule itself. Let p-1 equal the number of neighbors to be used as context and let $\underline{X}ij = (X1,X2,\ldots,Xp)^T$ be a vector of observations from the "context array." (Xp=Xij, the pixel being classified.) Let $\underline{C}^p = (C1,C2,\ldots,Cp)^T$ be a vector of possible classifications for the elements of the p-context array. (Cp=Cij, a possible classification of the pixel being classifier.) The decision rule, which defines the set of discriminant functions for the classification problem, is then

$$d(\underline{X}ij) = \text{action (classification) } 'a'$$

which maximizes

$$\sum_{\substack{\text{all } \underline{C}^p, \\ Cp='a'}} G(\underline{C}^p) \prod_{k=1}^{p} f(Xk|Ck) \qquad (1)$$

where $G(\underline{C}^p)$, the "context function," is the relative frequency with which $\underline{C}^p$ occurs in the scene being analyzed, and $f(Xk|Ck)$ is the set of class-conditional

probabilities developed from training. This training is similar to the training required for a per pixel maximum-likelihood classifier.

Methods for estimating the context function G(CP) were discussed in (Tilton et al, 1982). The most flexible and successful of the methods discussed is the "unbiased estimator." This estimator requires as its only inputs the data being classified and the set of class-conditional probability functions being used to model the ground-cover classes.

The unbiased estimator is best implemented as an adaptive estimator of the context function. (Tilton et al, 1982) The local context function estimate for a particular nl*n2 block of image data (nl<<128 and n2<<128) is made from a ml*m2 block centered on the nl*n2 block(nl<=ml and n2<=m2). The central nl*n2 block of image data is then classified using this local estimate of the context function. This process is repeated until the entire data set is classified. Better results are generally obtained when ml>nl and m2>n2 (e. g. ml = nl+10 and m2 = n2+10). This is because when ml=nl and m2=n2, the context function estimate is not accurate for the pixels at the edges of the image data block being classified.

The context function is estimated in parallel for each ms*ms section and the classification is done for each ns*ns section. The classification result for the (128-ms+ns)-by-(128-ms+ns) section of image data is then read out, and the next block of data is processed. This implementation gives an ideal (128-ms+ns)*(128-ms+ns) times speed-up in execution time. For ns=16 and ms=32, the ideal speed-up would be by a factor of 12,544. We expect that the actual speed-up will be by a factor of about 500. We estimate that the MPP can perform a contextual classification of a 512-by-512 4 band image in 18 seconds, a full (Landsat) Multispectral Scanner (MSS) scene in 9 minutes, and a full (Landsat) Thematic Mapper (TM) scene in 40 minutes. Corresponding processing times on a floating point systems AP180V array processor are 7625 seconds for a 512-by-512 4 band image, 5000 minutes for a full MSS scene, and 20000 minutes for a full TM scene. (Ramapriyan and Strong, 1983)

VII. CLOSING REMARKS

The economies in image processing speed to be realized by large parallel processors like the Massively Parallel Processor (MPP) will prevent the increasingly large data volumes of earth resources imagery data from overwhelming our computer analysis capability. Moreover, computationally intensive algorithms which are difficult to implement on serial machines can be more easily implemented in Parallel Pascal on the MPP and run in "reasonable" times. Perhaps the most significant result of this work is that it is now practical to design image analysis approaches using a parallel perspective; a perspective that is probably much akin to the way human beings naturally processes imagery with the eye-brain image processing system. Such parallelism may make it possible for a computer to come much closer to being able to match a human being's image analysis capabilities. Whatever the case may be with the human analogies, we hope that this new tool, the MPP, will stimulate new thinking about approaches to analyzing image data.

IX. REFERENCES

Alexander, D., et. al., "Spectral, Spatial and Radiometric Factors in Cover Type Discrimination," Proceedings of the 1983 International Geoscience and Remote Sensing Symposium, San Francisco, CA (1983).

Batcher, K.E., "Design of a Massively Parallel Processor," IEEE Transactions on Computers, Vol. C-29, pp. 836-840 (1980).

Burkley, J.T., C.T. Mickelson, "MPP: A Case Study of a Highly Parallel System," Proceedings of the AIAA Conference on Computers in Aerospace #4, (1983).

Kettig, R.L. and D.A. Landgrebe, "Classification of Multispectral Image Data by Extraction and Classification of Homogeneous Objects," IEEE Transactions on Geoscience Electronics, Vol. GE-10, No. 1, pp. 19-26 (1976).

Markham, B.L. and J.R.G. Townshend, "Land Cover Classification Accuracy as a Function of Sensor Spatial Resolution," Proceedings of the Fifteenth International Symposium on Remote Sensing of Environment, Ann Arbor, MI (1981).

Ramapriyan, H.K., and J.P. Strong, "Applications of Array Processors in the Analysis of Remote Sensing Images," Proceedings of the 17th International Symposium on Remote Sensing of Environment, Ann Arbor, MI (1983).

Reeves, A.P., and J.D. Bruner, "High Level Language Specification and Efficient Function Implementation for the Massively Parallel Processor," Interim Report TR-EE 80-32, School of Electrical Engineering, Purdue University, West Lafayette, IN (1980).

Reeves, A.P., and J.D. Bruner, "The Language Parallel Pascal and Other Aspects of the Massively Parallel Processor," Final Report for NASA Grant 5-3, School of Electrical Engineering, Cornell University, Ithaca, NY (1982).

Schaefer, D.H., J.R. Fischer, K.R. Wallgren, "The Massively Parallel Processor," AIAA Journal of Guidance, Control, and Dynamics, Vol. 5, No. 3, pp. 313-315 (1982).

Slotnik, D.L., "Research in the Application and Design Refinement of the Massively Parallel Processing Computer (MPP)," Quarterly Progress Reports for the period between September 1977 and August 1980, NASA Contract NAS5-24275, University of Illinois at Urbana-Champaign, Urbana, IL (1978).

Slotnik, D.L., "Research in the Application and Design Refinement of the Massively Parallel Processing Computer (MPP)," Second Quarterly Progress Report, NASA Contract NAS5-26405, University of Illinois at Urbana-Champaign, Urbana, IL (1981).

Swain, P.H., S. B. Vardeman, and J. C. Tilton, "Contextual Classification of Multispectal Image Data," Pattern Recognition, Vol. 13, No. 6, pp. 429-441 (1981).

Tilton, J.C., and S.C. Cox, "Segmentation of Remotely Sensed Data Using Parallel Region Growing," Proceedings of the Ninth International Symposium on Machine Processing of Remotely Sensed Data, West Lafayette, IN (1983).

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

Tilton, J.C., "Multiresolution Spatially Constrained Clustering of Remotely Sensed Data on the Massively Parallel Processor," Proceedings of the Tenth International Symposium on Machine Processing of Remotely Sensed Data, West Lafayette, IN (1984).

Williams, D.L., et. al., "Impact of Thematic Mapper Sensor Characteristics on Classification Accuracy," Proceedings of the 1983 International Geoscience and Remote Sensing Symposium, San Francisco, CA (1983).

James C. Tilton is a Senior Scientist with Science Applications Research (SAR), Riverdale, MD. He received a B.A. in Electrical Engineering, Environmental Science and Engineering and Anthropology from Rice University in 1976; M.S.E.E. from Rice University also in 1976; a M.S. in Optical Sciences from the University of Arizona in 1978 and a Ph.D. in Electrical Engineering from Purdue University in 1981. Since 1982 Dr. Tilton has been conducting and directing research in remote sensing data analysis techniques as a contractor at the NASA Goddard Space Flight Center. His research interests involve the development of optimal techniques for analyzing remotely sensed data. In particular this includes investigating techniques for incorporating spatial information into the analysis process and applying artificial intelligence techniques to the understanding of remotely sensed imagery.

James P. Strong is with the Interpretive Techniques Section of the Information Extraction Division of the NASA Goddard Space Flight Center, Greenbelt, MD. He received his B.S. in Electrical Engineering in 1958, his M.S. in Electrical Engineering in 1960, and his Ph.D. in Electrical Engineering in 1971, all from the University of Maryland. Dr. Strong was a member of the team overseeing the construction of the Massively Parallel Processor, and is presently involved in developing algorithm that will enable the Massively Parallel Processor to process geoscience data, perform radar signal processing, and to analyse and process images acquired from space.