Reprinted from

Symposium on

Machine Processing of

Remotely Sensed Data

and

Soil Information Systems

and

Remote Sensing and Soil Survey

June 3-6, 1980

Proceedings

The Laboratory for Applications of Remote Sensing

Purdue University
West Lafayette
Indiana 47907 USA

IEEE Catalog No.
80CH1533-9 MPRSD

INVENTORY ESTIMATION ON THE MASSIVELY PARALLEL PROCESSOR

PETER D. ARGENTIERO, JAMES P. STRONG, DAVID W. KOCH

NASA/Goddard Space Flight Center

ABSTRACT

This paper describes algorithms for efficiently computing inventory estimates from satellite based images. The algorithms incorporate a one dimensional feature extraction which optimizes the pairwise sum of Fisher distances. Biases are eliminated with a premultiplication by the inverse of the analytically derived error matrix. The technique is demonstrated with a numerical example using statistics obtained from an actual LANDSAT scene.

Attention was given to implementation ofi the Massively Parallel processor (MPP). A timing analysis demonstrates that the inventory estimation can be performed an order of magnitude faster on the MPP than on a conventional serial machine.

1. INTRODUCTION

Satellite borne remote sensing instruments of the future will be characterized by a substantially increased

- data volume
- data rate
- number of spectral bands
- spatial resolution

To accommodate the 1980-1990 mission requirements, a large increase in data processing efficiency is required. The NASA End-to-End Data Systems' (NEEDS) program[1] is an attempt to satisfy this need. Part of the NEEDS program is the development of specialized information extraction algorithms and matched computational architectures which can be used to reduce data to information as soon as possible in the end-to-end data processing operation. In support of this effort, attention was focused on appropriate inventory estimation procedures. The technique discussed in this paper relies on a one dimensional feature extraction and classification. Resulting inventory estimates are corrected for bias by premultiplying by the inverse of the analytically derived error matrix.

The procedure is highly parallel and attention is given to the problems and potential of implementation on a large scale parallel processing machine.

Section 2 gives a mathematical description of the inventory estimation technique. Section 3 provides a demonstration using class statistics obtained from an observed LANDSAT scene. Section 4 describes potential implementation on the Massively Parallel Processor[2,3] in development under NASA sponsorship and scheduled for delivery in 1982. Section 5 summarizes the results. Further details are given in appendices A and B.

2. MATHEMATICAL DEVELOPMENT

Assume that a decision rule with error matrix C has been implemented on a K class classification problem. Let $\hat{P}$ be a K vector of correct proportions of the observation set accounted for by each of the K classes. Let P be the K dimensional vector of proportions estimated by applying the decision rule and counting the number of samples allocated to each class. One can show that

$$C\,P = E(\hat{P}) \tag{1}$$

One can also show that the diagonal terms of the covariance matrix $\hat{P}$ are bounded by 1/N where N is the total number of samples. Hence, for the large sample size typical of satellite-based remote sensing scenes the variances of the individual terms of $\hat{P}$ are negligible. It follows that if the error matrix C is invertible, a very accurate estimate of P can be obtained by premultiplying $\hat{P}$ by C^{-1}.

The above results imply that appropriate criteria for the selection of decision rule for inventory estimation are

(A) computational efficiency
(B) the computability of the associated error matrix
(C) the invertibility of the associated error matrix

With regard to criterion B it is worth mentioning that there is no practical algorithm for computing the error matrix of a multidimensional Bayes classifier. For this reason, as well as for reasons of computational efficiency, the conventional Bayes or maximum likelihood decision rule is not an optimal choice for use in inventory estimation. However, it is possible to analytically compute the error matrix associated with a one dimensional Bayes decision rule. Details are in Appendix A. Hence, in order to satisfy criteria A and B, the search for an appropriate decision rule was limited to one dimensional feature extraction and Bayes classification procedures. With regard to satisfying criterion C, there is no way of choosing a decision rule which insures that the associated error matrix is invertible. To enhance the likelihood that the associated error matrix be invertible, it is important to notice that there is a close connection between the accuracy of a decision rule and the conditioning of its error matrix. Hence, it is logical to choose the coefficients of our one dimensional feature extraction and Bayes classification which optimize a class separability measure. For instance, Guseman and Walton[4] suggest that the global probability of correct classification be used as a class separability measure. We have found that a simpler and computationally more tractable separability measure is adequate.

Assume that our classification problem involves K classes and that the observation set is N dimensional. Let V be the N dimensional vector which defines the transformation to one dimension. Each N dimensional sample X is mapped onto a scalar Y by

$$Y = V^T X \tag{2}$$

The classification problem can be defined as follows: Assume K one dimensional normal random variables $N(U_j,\sigma_i)$, i - 1,2, . . . K. The required and variances are defined as

$$U_i = V^T M_i,\ i = 1,2,\ldots K \tag{3}$$

$$\sigma_i^2 = V^T \Sigma_i\, V,\ i = 1,2,\ldots K$$

Where M_i and Σ_i are respectively the mean and covariance matrix associated with the i th multidimensional class. The task is to assign the scalars defined by the left side of eq. 2 to one of the populations whose a priori statistics are given by the left side of eq. 3. To obtain a class separability measure, define the Fisher distance[5] between one dimensional populations i and j as

$$f_{i,j} = \frac{(U_j - U_j)^2}{\sigma_i^2 + \sigma_j^2} \quad (4)$$

The class separability measure for the set of one dimensional classes is the pairwise sum of Fisher distances given by

$$f = \sum_{i=2}^{k} \sum_{j=1}^{i-1} f_{i,j} \quad (5)$$

The left side of equation 5 is a homogeneous function (independent of scale) of V. Hence, to perform a Newton-Raphson iteration procedure to locate a V which optimizes f, it is necessary to constrain the solution to the unit sphere by means of the usual Lagrangian multiplier technique. We also require first and second variations of f with respect to V. They can be obtained by means of compact and easily computed matrix expressions. Hence, attention can be restricted to linear one dimensional feature extraction and Bayes classification procedures. This approach to inventory estimation can be outlined as follows:

(A) Choose the coefficients of a one dimensional feature extraction transformation which maximizes the sum of pairwise Fisher distances and use the coefficients to linearly map each sample onto one dimension.

(B) Obtain one dimensional a priori class statistics from training sample statistics by applying the usual laws for the behavior of first and second order moments under linear transformations.

(C) Employ a normality assumption and allocate the one dimensional samples among classes using the conventional Bayes or maximum likelihood decision rule.

(D) Construct a vector of inventory estimates by determining the proportion of the total sample set allocated to each class.

(E) Compute the error matrix of the one dimensional Bayes decision rule.

(F) Premultiply the vector of inventory estimates by the inverse of the error matrix to obtain unbiased estimates.

3. A NUMERICAL SIMULATION

To insure that the computational scheme described in the previous section is compatible with small computing machines, the procedure was incorporated into a computer program on the HP9852A programmable desk calculator. The program is in the HP basic language and occupies about 4000 bytes of the available memory. The input to the program consists of N dimensional mean vector and covariance matrices, and convergence control parameters for the constrained Newton-Raphson optimization procedure. The output of the program consists of an N dimensional vector V which defines the one dimensional feature extraction, the means and variances of the resulting one dimensional classes, and the error matrix.

For this numerical simulation class statistics were obtained from a Landsat 2 scene obtained over Finney County, Kansas during May of 1975.[6] The five classes consisted of two types of winter wheat and three confuser crops. The class statistics were obtained from well known sites in Finney County. The four channels are those of the Multispectral Scanner on board the Landsat 2. The sizes of the training sample sets range from about one hundred to about three hundred. The class statistics are shown in Table 1. Among the several factors which limit the accuracy of a Bayes decision rule in classifying remote sensing data are

(A) Significant deviations of class populations from normality

(B) Errors introduced by small sample sizes

(C) Training samples are not randomly selected from the populations in question and, thus, are not representative.

Because this simulation ignores such factors, misclassification probabilities will be somewhat optimistic compared to what might be obtained in an actual application. To measure the quality of our feature extraction and classification procedure it was decided to compare its performance to that of a Bayes decision rule operating on all four channels of the Multispectral Scanner data. This decision rule is optimal in the sense that it minimizes global misclassification probability. The performance of the four dimensional Bayes classifier was determined by means of a monte carlo program written for the HP9852A programmable desk calculator. The assumptions of the monte carlo simulation were

A. The five class populations are normally distributed

B. The class statistics are those of Table 1

C. A priori class probabilities are equal

D. Monte carlo sample sizes are sufficiently large that sampling error is insignificant.

Table 1
Statistics of LANDSAT-2 MSS Signatures Acquired May 1975
Over Finney County, Kansas

Channel	Mean	Std. Dev.	Covariance Matrix 1	2	3	4
(1) 184 Pixels of Non-Wheat						
1	27.7	3.6	12.7			
2	24.5	8.0	25.0	63.4	SYMMETRIC	
3	75.1	20.4	-51.4	-140.7	415.5	
4	37.4	12.0	-30.8	-84.2	242.1	143.4
(2) 333 Pixels of Non-Wheat						
1	34.7	3.6	12.7			
2	40.4	5.5	17.2	30.0		
3	47.0	5.2	8.8	9.9	27.3	
4	19.7	2.5	0.6	-1.2	10.4	6.0
(3) 324 Pixels of Non-Wheat						
1	33.3	1.6	2.6			
2	38.5	2.7	2.6	7.2		
3	44.1	6.4	4.3	2.5	41.2	
4	18.7	3.3	1.9	0.3	19.9	11.1
(4) 106 Pixels of Winter Wheat						
1	28.5	2.4	5.8			
2	27.5	4.0	7.4	16.2		
3	51.2	5.2	-6.0	-14.4	26.7	
4	24.0	3.0	-4.3	-8.9	14.1	9.0
(5) 127 Pixels of Winter Wheat						
1	21.5	2.7	7.3			
2	16.7	4.2	10.3	18.0		
3	54.9	5.1	4.1	4.9	26.0	
4	29.1	2.8	-1.0	-2.8	11.4	8.1

One thousand samples were obtained from each class and classified into one of the five classes according to a four dimensional Bayes decision rule. The element in the ith row and the ith column of the confusion matrix was estimated as the proportion of samples chosen from the jth class which were assigned to the ith class. The estimated error matrix is shown in Table 2. The table also shows the expected values of inventory estimates as obtained from the four dimensional Bayes decision rule. In this case, expected values of inventory estimates can be obtained by averaging the rows of the error matrix. From Table 2 it is also seen that the combined misclassification probability for classes 4 and 5 which are associated with winter wheat is 0.1.

The statistics shown in Table 1 were used as input to the one dimensional feature extraction and classification program. For this simulation, a priori class probabilities were assumed to be equal. Hence, the correct value for each class inventory estimate is 0.2. The resulting error matrix along with expected values of inventory estimates are shown in Table 3. The error matrix was computed analytically. The combined misclassification probability for classes 4 and 5 is 0.14. However, expected values of inventory estimates for winter wheat classes 4 and 5 are seen to be considerably worse

Table 2
Error Matrix for Four Dimensional Bayes Classifier

Class	1	2	3	4	5
1	0.81	0.01	0.01	0.01	0.02
2	0.02	0.58	0.12	0.03	0
3	0.08	0.34	0.84	0.03	0
4	0.07	0.07	0.03	0.88	0.06
5	0.02	0	0	0.05	0.92
Expected Values of Inventory Estimate	0.17	0.15	0.26	0.22	0.20

Table 3
Error Matrix for One Dimensional Feature Extraction and Bayes Classifier

Class	1	2	3	4	5
1	0.22	0	0	0	0.04
2	0.07	0.39	0.20	0.03	0
3	0.04	0.54	0.78	0.04	0
4	0.34	0.07	0.02	0.84	0.08
5	0.34	0	0	0.09	0.88
Expected Value of Inventory Estimate	0.05	0.14	0.28	0.27	0.26

than corresponding estimates for the optimal four dimensional Bayes classifier. The primary reason is errors of commission introduced by misclassification of samples from class 1 into classes 4 and 5. But the error matrix can be used to correct inventory estimates to yield unbiased estimates. Let P be a five dimensional vector representing inventory estimates for the five classes. An unbiased inventory estimate P_0 for these classes can be obtained as

$$\hat{P}_0 = C^{-1}\hat{P}$$

where C is the error matrix given in Table 3.

An explicit form for C^{-1} is

$$C^{-1} = \begin{bmatrix} 4.85 & -0.01 & 0 & 0.02 & -0.22 \\ -1.02 & 4.00 & -1.02 & -0.10 & 0.06 \\ 0.54 & -2.75 & 1.99 & 0.01 & -0.03 \\ -1.73 & -0.27 & 0.04 & 1.20 & -0.03 \\ -1.70 & -0.03 & 0 & -1.13 & 1.23 \end{bmatrix}$$

From Table 3 we have

$$E(\hat{P}) = \begin{bmatrix} 0.05 \\ 0.14 \\ 0.28 \\ 0.27 \\ 0.26 \end{bmatrix}$$

Also $$E(\hat{P}_0) = C^{-1}\, E(\hat{P})$$

The above equations yield

$$E(\hat{P}_0) = \begin{bmatrix} 0.2 \\ 0.2 \\ 0.2 \\ 0.2 \\ 0.2 \end{bmatrix}$$

This result is a logical consequence of the unbiased property of P_0 as an estimator. But the results of this simulation serve as a useful numerical check on the validity of our development.

4. IMPLEMENTATION ON THE MASSIVELY PARALLEL PROCESSOR

The Massively Parallel Processor (MPP) as shown in figure 1 is a 128 X 128 element array processor with its associated control unit, buffers, host computer, and disk storage. The array processor is of 16384 identical processing elements which perform identical operations simultaneously under control of the Array Control Unit. The processing elements have the following characteristics:

(A) Each performs bit serial arithmetic
(B) Each has 1024 bits of memory
(C) Each is connected to its neighboring processing element to the left, right, above, and below.

Sixteen of the processing elements have data connections directly to the Array Control Unit, a feature which is used when summing the values in an array. While bit serial operation tends to slow processing compared to byte or word parallel operation, the processing of 16384 elements in the array simultaneously gives a net increase in speed for arithmetic operations of over two orders of magnitude or more. The connections between each processing element and its neighbors makes the MPP ideal for processing image data where identical operations are performed at each picture element. For image processing, each processing element in the array processor is assigned to a picture element. Several processing tasks of the algorithm described in the previous section lend themselves easily to parallel processing on the MPP. These tasks are:

(A) Reducing the dimensionality of the data by performing the operation $Y = V^T X$.
(B) Evaluation of the maximum likelihood function.
(C) Classification of the picture elements in the image based on the value of the maximum likelihood function.
(D) Performing the inventory by summing the picture elements in each class.

In the following subsections, an estimate is given of the processing time required by the MPP for each of these tasks.

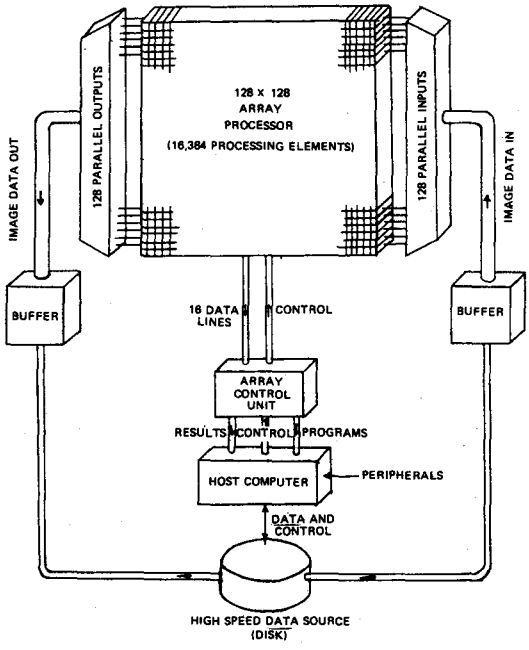

Figure 1. Massively Parallel Processor

4.1 Multiplication by the Reduction Vector

If the reduction vector is defined as $V = v_1, v_2, v_3, \ldots v_n$ and the multiband image data is defined as $X = x_1, x_2, x_3, \ldots x_n$, where x_i is the image in band i, then $Y = V^T X$ of the previous sections is given by:

$$Y = \sum_{i=1}^{n} v_i x_i$$

In the MPP, each x_i is a 128 X 128 array of integer data. Each v_i is to be a scalar constant. Based on bit serial arithmetic and the characteristics of the MPP processing element, the calculation of Y would require:

$$T_Y = \frac{n \cdot s \cdot t}{2} + 3n\,(s + t + \log_2 n) \quad \times 10^{-1} \mu\text{sec} \qquad (6)$$

where n is the number of bands, s is the number of bits in the data of each band, and t is the number of bits in the scalar constant. This equation, as well as all other equations for processing times on the MPP, is derived in Appendix B. Note that the bit serial nature of the MPP's arithmetic operations makes all equations dependent on the bit length of the data involved. Letting n represent 4 bands of data and using typical values of s = 8 image bits, t = 20 scalar bits, the processing time required to perform the multiplication and summing on the MPP would be 68 microseconds. This is the amount of time required for processing a 128 X 128 array of data. Consider a satellite based image approximately 4000 X 4000 picture elements. Performing the inventory on such an image would require processing about one thousand 128 X 128 arrays. Thus the total time required for the MPP to multiply the 4 bands of a satellite based image by a reduction vector would be about 68 milliseconds.

4.2 Evaluation of the Maximum Likelihood Function

The equation for the maximum likelihood function is

$$G_i = (Y - U_i)^2 \times \frac{1}{\sigma_i^2} + \log c_i.$$

In this equation, G_i and Y are 128 X 128 element integer arrays and U_i, $1/\sigma_i^2$, and $\log c_i$ are all scalar constants.

The time required for the MPP to evaluate the maximum likelihood function array G_i for K classes has been calculated as:

$$T_E = K\left[3s + \frac{s \cdot s}{2} + \frac{2s \cdot t}{2} + 3 \times \text{larger of } 2s + t \text{ or } m\right] \times 10^{-1}\,\mu\text{sec} \qquad (7)$$

where s is the bit length of Y and U_i, and t and m are the bit lengths of $1/\sigma_i^2$ and $\log c_i$ respectively. For ten classes and typical data bit lengths of s = 8, t = 16, and m = 16, the calculation of 10 maximum likelihood function arrays would require 232 microseconds. Processing a 4000 X 4000 satellite based image would require 1000 X T_E or 232 milliseconds.

4.3 Classification of the Elements in the Array

A parallel processor algorithm developed for maximum likelihood classification of images with the MPP generates a binary array corresponding to each class. Such a binary array B_K will have a logical 1 in each element that belongs to class k and a logical zero in all other elements. That is:

$$B_k = 1 \text{ where } G_k \geq \text{all other } G_j, k \neq j$$

$$B_k = 0 \text{ where } G_k < \text{any other } G_j, k \neq j$$

To generate each binary array, the MPP algorithm compares the maximum likelihood function array for each class to the maximum likelihood function array for all other classes. The above conditions are applied to form the binary arrays. The algorithm is described in more detail in Appendix B. The time required for the MPP to generate the binary 128 X 128 arrays for K classes is:

$$T_C = K(5s + 1) + \sum_{i=2}^{K} 2(i - 1) \quad \times 10^{-1}\,\mu\text{seconds} \qquad (8)$$

where S is the bit length of the maximum likelihood function array. For typical values of K = 10 and s = 8, T_C = 50 μseconds. To perform the classification procedure on a 4000 X 4000 satellite based image would require 1000 X T_C = 50 milliseconds.

4.4 Performing the Inventory

A parallel algorithm which performs the inventory task uses a "partial sum" array Ps(i) for each class.

As the binary arrays for each class are calculated for each 128 X 128 sub array of the 4000 X 4000 image, they are added to the corresponding partial sum array. After n sub arrays have been processed, the partial sum for each class is given by

$$PS(i) = \sum_{j=1}^{n} B_{(i)j}$$

where $B_{(i)j}$ is the binary array for class i evaluated at the jth 128 X 128 sub array. This operation on the MPP has been calculated to require:

$$T_{PS} = [2n + 2\log_2 e \times ((\log_e n) + 1 - n)] \times 10^{-1}\,\mu\text{sec} \qquad (9)$$

where n is the number of 128 X 128 sub arrays. If n = 1000, T_{PS} = 1905 μseconds or about 2 milliseconds for a 4000 X 4000 image.

After all sub arrays of the 4000 X 4000 image have been classified, the partial arrays PS(i) for each class must be integrated or "summed over" in order to get the total number of picture elements in each class. In the MPP, this summing procedure makes use of the connection paths between the array processing elements. The time required for this summation is:

$$T_S = \sum_{i=1}^{5} s + 10 + (2s + 2i + 3)\,(3 + 2^{i-1})\ 10^{-1}\,\mu\text{seconds} \qquad (10)$$

where s is the bit length of the partial sum array. The maximum value s can have for a 4000 X 4000 image is 10 assuming a particular picture element is in the same class in each of the 1000 sub-arrays. Using this value, the maximum value for T_s is 138 μs. Performing the summation for 10 classes could take 1.4 milliseconds.

4.5 Timing Estimates

The total amount of time required for the 4 tasks is approximately $(68 + 232 + 50 + (2 + 1.4)) \doteq 353$ milliseconds. Thus, excluding I/O time, a 4000 X 4000 satellite based image could be inventoried in less than one half second using the Massively Parallel Processor.

This compares to a CPU time of 13 minutes for the same task using a 360-95 serial machine programmed in Fortran. Assuming a 10 to 1 reduction in processing time for a serial machine by resorting to an efficient assembly language code, one half second is still two orders of magnitude less time than 1.3 minutes. However, I/O times must also be considered. At the time of delivery of the MPP, disk storage devices are expected to be available with on the order of 40 megabytes per second transfer rates. Making this assumption, a 4 band 4000 X 4000 satellite image could be transfered into the MPP in less than 4 seconds. Including I/O time one has a ratio of about 1.3 minutes to 4.4 seconds. The processing speed factor

between the MPP and a general purpose serial machine is still greater than 10 to 1. This ratio will have a significant impact on any operational inventory estimation requirement.

5. SUMMARY

This paper has described special purpose algorithms and matching computational architectures for efficiently computing inventory estimates from satellite based images. The algorithms incorporate a one dimensional feature extraction which optimizes the pairwise sum of Fisher distances. Estimation biases are connected with a premultiplication by the inverse of the analytically derived error matrix. The technique is demonstrated with a numerical example using statistics obtained from an actual LANDSAT scene.

Attention was given to implementation of the inventory estimation algorithms on the Massively Parallel Processor. Timing computations demonstrate that a 4 band 4000 by 4000 image can be processed to estimate inventory for 10 classes in less than .5 sec when input output considerations are neglected. When input output time is included the speed of the MPP is still greater by an order of magnitude when compared to a conventional high speed serial machine. These results demonstrate the value of large scale parallel processing machines in the performance of standard image processing operations in an operational environment.

References

1. Robert D. Price, " The NEEDS Program," AIAA/NASA, Smart Sensors Conference, Langley Research Center, Hampton, Virginia, November 14-16, 1978.

2. Schaefer, D. H., "Massively Parallel Information System," Proceedings of AIAA Second Computers and Aerospace Conference, Oct. 1979.

3. Batcher, K. E., "Massively Parallel Processing System," Proceedings of AIAA Second Computers and Aerospace Conference, Oct. 1979.

4. Guseman, L., Walton, J., "An Application of Linear Feature Selection to Estimation of Proportions," Commun. Statist. - Theor. Meth., A6(7), 611-617, (1977).

5. Fukunaga, K., "Introduction to Statistical Pattern Recognition," Academic Press, 1972.

6. Σ^2 Study, Capability Demonstration for Landsat-D System, Final Report, by General Electric, contract NAS5-23412, Mod. 30.

APPENDIX A
AN ALGORITHM FOR COMPUTING THE ERROR MATRIX OF A ONE DIMENSIONAL BAYES DECISION RULE

Assume K normal random variables $\eta(U_i, \sigma_i)$, i = 1, 2, . . . K each of dimension one. Samples are chosen from these populations according to a probability law defined by a priori class probabilities P_i, i = 1, 2, . . . K. The conventional classification problem is to classify each sample by assigning it to the class from which it was chosen. In this case the Bayes classifier can be obtained by defining K functions, each with one dimensional argument

$$\Im_i(x) = \frac{(x - u_i)^2}{\sigma_i^2} + \ell n\, \sigma_i^2 - 2\, \ell n\, P_i, \; i = 1, 2, \ldots K \tag{1}$$

The scalar x is assigned to the class indexed by i if and only if

$$f_i(x) \leqslant f_j(x), j = 1, 2, \ldots K \tag{2}$$

The decision rule defined by equation 2 can be shown to minimize risk under a zero-one loss matrix. It also maximizes liklihood. Hence it minimizes the global probability of misclassification.

With any decision rule one can associate an error matrix D defined as

$$D(i,j) \rightarrow \text{ conditional probability that a sample chosen from class i is assigned to class j under a given decision rule} \tag{3}$$

Our task is to provide an algorithm for computing the error matrix associated with the Bayes decision rule. For simplicity of exposition we assume that if $i \neq j$, $\sigma_i \neq \sigma_j$. It is not difficult to modify results to account for duplications.

For each j, define the set τ_j as follows:

$$\tau_j = \left\{\text{all x such that } f_j(x) \leqslant f_i(x), i = 1, 2, \ldots K\right\} \tag{4}$$

Then

$$D(i, j) = \int_{T_j} \eta(u_i, \sigma_i)\, dx \tag{5}$$

It remains to provide an explicit description of the sets T_j, j = 1, 2, . . . K. For each $i \leqslant K$ and $j \leqslant K$ define the sets $\beta_{i,j}$ as

$$\beta_{i,j} = \left\{\text{all x such that } f_j(x) \leqslant f_i(x)\right\} \tag{6}$$

Then

$$\tau_j = \bigcap_i \beta_{i,j} \tag{7}$$

From equations 1 and 2 it follows that the boundary points of $\beta_{i,j}$ are the quadratic equation

$$Ax^2 + Bx + C = 0 \tag{8}$$

with

$$A = \frac{1}{\sigma_j^2} - \frac{1}{\sigma_i^2} \tag{9a}$$

$$B = \frac{2u_i}{\sigma_i^2} - \frac{2u_j}{\sigma_i^2} \tag{9b}$$

$$C = \frac{u_j^2}{\sigma_j^2} - \frac{u_i^2}{\sigma_i^2} + \ell n\, \frac{\sigma_j^2}{\sigma_i^2} - 2\, \ell n\, \frac{P_j}{P_i} \tag{9c}$$

Define the boundary points of $\beta_{i,j}$ as

$$T_{1,i,j} = \min\left[\frac{-B + \sqrt{B^2 - 4AC}}{2A}, \frac{-B - \sqrt{B^2 - 4AC}}{2A}\right] \tag{10a}$$

$$T_{2,i,j} = \max\left[\frac{-B + \sqrt{B^2 - 4AC}}{2A}, \frac{-B - \sqrt{B^2 - 4AC}}{2A}\right] \tag{10b}$$

Hence

$$\beta_{i,j} = \begin{cases} [T_{1,i,j}, T_{2,i,j}] \text{ if } \sigma_j < \sigma_i \\ [-\infty, T_{1,i,j}] \cup [T_{2,i,j}, \infty] \text{ if } \sigma_j > \sigma_i \end{cases} \tag{11}$$

It will be convenient to represent the set of boundary points of an arbitrary set S by the symbolism b(S). From equation 11

$$b(\beta_{i,j}) = \left\{T_{1,i,j}\right\} \cup \left\{T_{2,i,j}\right\} \tag{12}$$

From equations 7 and 11 it follows that the set τ_j can be represented as the disjoint union of a finite set of intervals and that

$$b(\tau_j) \subset \bigcup_i b(\beta_{i,j}) \tag{13}$$

When the elements of b (τ_j) are linearly ordered according to size, left sided and right sided boundary points must alternate. Hence, to reconstruct the set τ_j it is sufficient to know which elements of $\bigcup_i b(\beta_{i,j})$ are elements of $b(\tau_j)$ and the classification of the smallest element in $b(\tau_j)$ as either a left sided or a right sided boundary point. For each element $z \in \bigcup_i b(\beta_{i,j})$, z is placed in $b(\tau_j)$ if and only if for each $i \leq K$

$$\begin{aligned} &T_{1,i,j} \leq z \leq T_{2,i,j}, \text{ if } \sigma_j < \sigma_i \\ &z \leq T_{1,i,j} \text{ or } z \geq T_{2,i,j}, \text{ if } \sigma_j > \sigma_i \end{aligned} \tag{14}$$

Order the elements of $b(\tau_j)$ by increasing size to create the indexed set $\{s_{j,m}\}_m$. There exists an $\ell \leq 2$ and an $i \leq K$ such that

$$s_{j,1} = T_{\ell,i,j} \tag{15}$$

Classify $s_{j,1}$ as either a left sided or right sided boundary point according to the following rule

$$\left.\begin{aligned} &\ell = 1 \text{ and } \sigma_j < \sigma_i \\ &\ell = 2 \text{ and } \sigma_j > \delta_i \end{aligned}\right\} \rightarrow \text{left sided}$$
$$\left.\begin{aligned} &\ell = 1 \text{ and } \sigma_i < \sigma_j \\ &\ell = 2 \text{ and } \sigma_i > \sigma_j \end{aligned}\right\} \rightarrow \text{right sided} \tag{16}$$

In order to conveniently compute the integral on the right side of equation 5, transform the indexed set $\{s_{j,m}\}_m$ into the indexed set $\{r_{j,m}\}_m$ by the mapping

$$r_{j,m} = (s_{j,m} - U_i)/\sigma_i \tag{17}$$

Let q be the index of the largest element in $\{r_{j,m}\}_m$ and let d(x) represent the distribution function of the standard normal random variable. Then D(i,j) can be computed as follows when $s_{j,1}$ is a right sided boundary point

$$D(i,j) = 1 - d(r_{j,q}) + d(r_{j,1}) + \sum_{\ell=2}^{q-2} [d(r_{j,\ell+1}) - d(r_{j,\ell})] \tag{18}$$

and when $s_{j,1}$ is a left sided boundary point

$$D(i,j) = \sum_{\ell=1}^{q-1} [d(r_{j,\ell+1}) - d(r_{j,\ell})] \tag{19}$$

APPENDIX B
DERIVATION OF MPP COMPUTING TIMES

Development of the timing formula for Computing $Y = V^T X$

As indicated in the paper $Y = V^T X$ is given by

$$Y = \sum_{i=1}^{n} v_i x_i.$$

In the MPP the number of basic clock cycles required to perform multiplication of an s-bit integer array by a t-bit constant using bit serial operations is st/2. To perform n of these products requires $n \cdot s \cdot t/2$ cycles. Each of the n products has a maximum bit length of s + t bits. The summation of two integer arrays in the MPP requires 3 m cycles where m is the largest bit length in the two arrays. To calculate the number of cycles required to perform the n sums, we note that as several numbers of length ℓ are summed, the result has a maximum bit length of $\ell + \text{Log}_2 i$ where i is the number of summations. Thus, for n sums of numbers s + t bits long, the maximum bit length of the resulting sum is $s + t + \text{Log}_2 n$. To get an upper bound on the number of cycles required to perform we will assume each sum to be of lengths $s + 2 + \text{Log}_2 n$. Thus, n sums requires less than $n \times 3 \times (s + t + \text{Log}_2 n)$ machine cycles.

Thus the total number of cycles for computing $Y = V^T X$ is $Ny = [n \cdot s \cdot t/2 + n \times 3 (s + t + \text{Log}_n]$. In the MPP, a cycle requires 10^{-1} microseconds. Thus T_Y in the paper is given by:

$$T_Y = \left[\frac{n \cdot s \cdot t}{2} + n \times 3 (s + t + \log_2 n)\right] \times 10^{-1} \ \mu\text{seconds}$$

Development of the Timing Formula for Computing the Maximum Likelihood Function

The exponent G_i is given by:

$$G_i = (Y - U_i)^2 \times \frac{1}{\sigma_i^2} + \text{Log } c_i$$

where Y is the array previously computed. The variable U_i is a constant. On the MPP the subtraction of U_i from Y is performed in 3 m cycles where m is the largest bit length among the elements of Y and the constant U_i. Squaring the difference array requires $n^2/2$ cycles where n is the bit length of the difference array. The bit length of $(Y - U_i)^2$ will be twice the bit length of $Y - U_i$. The bit length of $Y - U_i$ will be no more than the maximum for Y or U_i since both the array and the constant are positive. Using the number of cycles required for multiplying by a constant and for addition given in the previous section, the number of cycles N_E required to compute the maximum likelihood function is given by:

$$N_E = 3s + \frac{s \cdot s}{2} + \frac{2s \cdot t}{2} + 3 \times (\text{Larger of } 2s + t \text{ or } m)$$

where S is the bit length of the array and U_i, t is the bit length of $1/\sigma_i^2$ and m is the bitlength of Log c_i. The length of time required to calculate K maximum likelihood functions is therefore given by:

$$T_E = K\left(3s + \frac{s \cdot s}{2} + \frac{2s \cdot t}{2} + 3 \times (\text{Larger or } 2s + t \text{ or } m)\right) \times 10^{-1} \ \mu\text{seconds.}$$

Development of the Timing Formula to Classify Elements in the Array

An algorithm for use on an array processor has been developed for the classification of points in an image. This algorithm creates a binary array corresponding to each class. The ith binary array will have the value 1 at the elements which belong to the ith class and zeros at all other elements. The algorithm assigns all elements of the image to class 1 initially. It then calculates each maximum likelihood function array in order starting from class 1 to class K.

At step i where the maximum likelihood function $G_{(i)}$ for class i is evaluated, an array $\text{Max}_{(i)}$ which contains, at each element, the maximum value at that element calculated fo the i - 1 previous steps is compared to G(i). At the elements where G(i) is greater than $\text{Max}_{(i)}$ a "one" is assigned to the binary array $B_{(i)}$ corresponding to the ith class. "Ones" assigned to these elements in binary arrays up to i - 1 are removed. Finally at the elements where G(i) is greater than $\text{Max}_{(i)}$, the value in G(i) is transfered to $\text{Max}_{(i)}$. At the end of K steps all elements of the image will have been classified.

Table 1A lists these operations and indicates the number of cycles required for the MPP to perform the operation.

Table 1A
Number of Cycles Required for Generating Classification Arrays

Operation	No. of MPP Cycles
Compare $G_{(i)}$ to $Max_{(i)}$, that is: Calculate $G_{(i)} - Max_{(i)}$)	3s
Assign "ones" to $B_{(i)}$ where $G_{(i)} > Max_{(i)}$	1
Remove "ones" from previous $B_{(j's)}$ where $G_{(i)} > Max_{(i)}$	2(i - 1)
Replace elements of $Max_{(i)}$ with elements of $G_{(i)}$ where $G_{(i)} > max_{(i)}$	2s

In this table, s is the number of bits in the arrays $G_{(i)}$ and $Max_{(i)}$. For K classes the total number of cycles is:

$$N_C = K(3s + 1 + 2s) + \sum_{i=2}^{K} 2(i - 1)$$

At the MPP's clock frequency, this reduces to

$$T_C = \left[K(5s + 1) + \sum_{i=2}^{K} 2(i - 1) \right] \times 10^{-1} \text{ microseconds}$$

Development of the Timing Formula for Performing the Inventorying Task

In performing the inventorying task, a partial sum array $PS_{(i)}$ for each class is updated as each 128 X 128 sub array of the large 4000 X 4000 spacecraft image is classified. After the classification task is completed on each 128 X 128 sub array, the binary array for each class is added to the partial sum array for that class. After all the 128 X 128 sub arrays of the 4000 X 4000 image have been processed, the elements of the partial sum arrays contain a subtotal of the inventory sums for each class. The elements in each partial sum array must then be summed to obtain the total inventory for each class.

To calculate the number of cycles required to add a binary array to the partial sum array one must consider the way in which the number of bits in the sum increases as binary arrays are added. The maximum bit length of the sum of i one bit numbers is 1 + (first integer less than $\text{Log}_2 i$). The addition of a binary array to an m-bit array in the MPP requires 2 m cycles. Thus the number of cycles required for adding a binary array to the partial sum arrays n times is:

$$N_{BSUM} = \sum_{i=1}^{n} 2(1 + \llcorner \log_2 i)$$

where $\llcorner$ is the symbol for "the first integer less than". Since n gets as large as 1000, we will ease our evaluation of it by approximating the above sum using the integral equation below as an upper bound.

$$M_{BSUM} \geq \int_{1}^{n} 2(1 + \log_2 i)\, di$$

The integral is evaluated as:

$$M_{BSUM} \geq 2n + [2 \log_2 e \times ((n \log_e n) + 1 - n)]$$

Using the MPP's clock time, the time required for this operation is:

$$T_{PS} = [2n + [2 \log_2 e \times ((n \log_e n) + 1 - n)]] \times 10^{-1} \ \mu\text{sec.}$$

The summation of the elements in the 128 X 128 partial sum arrays is accomplished in the MPP by summing over sixteen 32 X 32 sub arrays simultaneously in the array processor portion of the MPP and outputing the 16 sums to an external adder for the final summation. This procedure makes use of the vertical and horizontal connections between each of the MPP's processing elements. It also makes use of the MPP's sixteen output connections from special processing elements at the corner of each of the 32 X 32 sub arrays. The algorithm for performing the summation over the 32 X 32 sub arrays performs two sets of operations each consisting of 5 steps. In the first set the 32 columns of the sub arrays are summed. In the second set, the 32 rows of the resulting sub arrays from the first set are summed. When both sets of operations are completed, the processing elements, in the lower right hand corners of all sixteen 32 X 32 sub arrays contain the sum of all elements in the sub arrays.

Each of the 5 steps in the column summing and row summing follows the following procedure:

1. For the ith step, translate the resulting array from the previous step 2^{i-1} elements ("columns to the right" when column summing; "rows down" when row summing).
2. Add the translated version to the original version.

For example: In the first step, the partial sum array is translated one element to be right and the translated version is added to the partial sum array. In this way, adjacent columns are added. In the second step, the resulting sum is translated two columns to the right and added to the original version. After this step, 4 adjacent columns have been summed. Since the sum is potentially doubled in each step, the number of bit planes in the resulting sum arrays increases by one in each of the 10 steps. Thus, at the ith step of column summing, s + i - 1 binary arrays (making up the sum arrays) must be shifted 2^{i-1} columns. In the row summing, s + i + 4 binary arrays must be shifted 2^{i-1} rows. Translation of a binary array requires one MPP cycle. Thus the number of cycles required at the ith column summing step is:

$$N_{CS_i} = (s + i - 1) \times 2^{i-1} + 3(s + i - 1)$$

where 3 cycles are required for each stage of addition. Similarly,

$$N_{RS_i} = (s + i + 4) \times 2^{i-1} + 3(s + i + 4)$$

cycles are required for the row summations. Therefore, the total number of cycles for the summation of all elements in the 32 X 32 sub arrays is:

$$N_S = \sum_{i=1}^{5} N_{CS_i} + \sum_{i=1}^{5} N_{RS_i}$$

$$= \sum_{i=1}^{5} (s + i - 1) \times 2^{i-1} + 3(s + i - 1) + \sum_{i=1}^{5} (s + i + 4) \times 2^{i-1} + 3(s + i + 4)$$

This simplifys to:

$$NS = \sum_{i=1}^{5} (2s + 2i + 3)(3 + 2^{i-1})$$

The time required to perform the summing operation is therefore

$$T_S = \sum_{i=1}^{5} (2s + 2i + 3)(3 + 2^{i-1}) \times 10^{-1} \ \mu\text{sec}$$

The time required to output the 16 sums from the processing elements is $(s + 10) \times 10^{-1}$ μseconds where $s + 10$ is the bit length of the final sum. Thus the total time for performing the summation of the partial sum arrays is

$$T_{PSS} = \left[s + 10 + \sum_{i=1}^{5} (2s + 2i + 3)(3 + 2^{i-1}) \right] \times 10^{-1} \ \mu\text{sec.}$$

Peter D. Argentiero was born in Philadelphia, PA., in June 1939. He received the B.S. degree in physics from Villanova University in 1961 and the M.A. degree in mathematics from Temple University 1963. He received the Ph.D. degree in mathematics from the Virginia Polytechnic Institute in 1968.

From 1963 to 1967 he was on the faculty of Virginia Polytechnic Institute as an Instructor in mathematics. Since 1967, he has been with NASA at the Goddard Space Flight Center, Greenbelt, MD. At present, he is engaged in research in the areas of pattern recognition and image data evaluation as related to Earth applications missions.

Since 1964, David W. Koch has been employed as a Mathematician/Data Analyst at the NASA/ Goddard Space Flight Center, Greenbelt, Maryland. He is presently engaged in sensor data processing and software development. He obtained a B.A. in Humanities from Johns Hopkins in 1960 and a B.S. in Mathematics from Virginia Polytechnic Institute in 1964. He is also a member of Delta Phi Alpha (German) and Pi Mu Epsilon (Mathematics) honorary societies.

Dr. Strong received his B.S. at the University of Maryland in 1958. At that time, he started work at the Naval Research Laboratory in Washington, DC. designing and testing shipboard communications antennas. In 1960, received his M.S. from the University of Maryland. In 1963, he transferred to Goddard Space Flight Center where he pursued work in telemetry coding techniques. Later work involved image processing using coherent optics. This work led to the development of parallel digital image processing techniques culminating in the concepts for the tse computer which in turn developed into the Massively Parallel Processor. Dr. Strong received his Ph.D. in 1971 from the University of Maryland. He is currently Project Scientist for the MPP project at Goddard.