AgRISTARS

SR-PO-04024
NAS9-15466

A Joint Program for Agriculture and Resources Inventory Surveys Through Aerospace Remote Sensing

Supporting Research

November 1980

Final Report

Vol. III
Data Processing Research and Techniques Development

by P.E. Anuta, D.A. Landgrebe, H.J. Siegel, P.H. Swain

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

NASA

SR-PO-04024
NAS9-15466
LARS 112880

Final Report

VOL. III. DATA PROCESSING RESEARCH AND TECHNIQUES DEVELOPMENT

P.E. Anuta, D.A. Landgrebe, H.J. Siegel, P.H. Swain

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

November 1980

Star Information Form

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
SR-PO-04024		

4. Title and Subtitle	5. Report Date
VOL. III. Data Processing Research and Techniques Development	November 30, 1980
	6. Performing Organization Code

7. Author(s)	8. Performing Organization Report No.
P.E. Anuta, D.A. Landgrebe, H.J. Siegel, P.H. Swain	112880

9. Performing Organization Name and Address	10. Work Unit No.
Purdue University Laboratory for Applications of Remote Sensing 1220 Potter Drive West Lafayette, Indiana 47906	11. Contract or Grant No. NAS9-15466

12. Sponsoring Agency Name and Address	13. Type of Report and Period Covered
NASA Johnson Space Center Earth Observations Division Houston, Texas 77058	Final, 12/1/79-11/30/80
	14. Sponsoring Agency Code

15. Supplementary Notes

Technical Monitor: J.D. Erickson
Principal Investigator: M.E. Bauer

16. Abstract

This report describes results of several research investigations in the general area of analysis and processing of remotely sensed multispectral data. The first examines two aspects of misregistration: determination of misregistration using properties of Fourier transforms and development of a general model for studying the effects of misregistration on classification accuracy. The second study presents a review of multistage classification and preliminary results for a layered decision algorithm which may increase the accuracy and efficiency over the single conventional single-stage classification approach. The third study describes the results of a more general way to exploit the spatial/spectral context of pixels to achieve accurate classification. The fourth study is an investigation of relaxation labeling processes employing ancillary information to reduce or eliminate the ambiguity of labels assigned to a set of objects.

17. Key Words (Suggested by Author(s))	18. Distribution Statement
Registration, misregistration, error model, multistage classification, Hughes Phenomenon, contextual classification, ambiguity reduction, relaxation labeling, training sample labeling	

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price*
Unclassified		152	

*For sale by the National Technical Information Service, Springfield, Virginia 22161

NASA - JSC

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

A. DETERMINATION OF MISREGISTRATION EFFECTS

Paul E. Anuta
Carlos Pomalaza

This task was formulated to further explore the question of misregistration in multispectral/multitemporal data used in resource application projects. Misregistration can occur between spectral bands in a single acquisition due to sensor and data handling errors or it can occur between acquisitions from different times which have been imperfectly registered. Temporal registration errors are due to a variety of causes relating to both the nature of the data and the registration processor. The ultimate source of error is the change in the scene which occurs between the two times of acquisition. Matching of two images which have undergone change relative to each other will always result in some error. The change in the scene is, of course, related to the information derived by the user, whereas the change is noise to the registration processor. These two viewpoints are in conflict and are an interesting characteristic of this information processing image processing problem.

Several studies have been conducted on the misregistration effects problem but results are few compared for results of studies of, for example, various classifiers and their performance. In 1975 Malila 1 studied the effects of misregistration on crop classification and produced results on proportion of mixed pixels and its effect on class statistics.

Two aspects of misregistration were studied in this project. The first is an approach to determine misregistration using properties of the Fourier transforms of the two images being compared. This is a different approach from the usual correlation process carried out. It warranted investigation since little was known about its characteristics and could potentially offer an improvement in accuracy of misregistration estimation. The second subject is effect of misregistration on classification accuracy. Several empirical studies have been performed and these give only a few samples of error conditions. The work here focused on the problem of developing general models for misregistration effects so that results would be varied over a wide range of cases. Several models were pursued and evaluated. Problems and considerations for adequately general models are discussed.

1. Estimation of Misregistration Using the Fourier Transform

1.1 Development of Difference Spectrum Expressions

The method described here for estimation of misregistration between two images uses properties of the Fourier transforms of the two images. It is assumed that there is no rotation, skew, scale difference or higher order distortion between the images. Thus there are only two variables describing the misregistration: δ_x, translational shift in the x direction and δ_y, translational shift in the y direction. One image is assumed to be the reference and the other is shifted with respect to it. The images we are concerned with are discrete or digital images which are arrays of numerical values. However, the thrust of this research is to find improved methods of determining small fractional sample misregistrations. Thus the continuous image case will be explored first. Then the relationship between the continuous image and fractional misregistration in the discrete case will be explored.

The two images will be represented as $p_{t_1}(x,y)$ and $p_{t_2}(x,y)$. Since we are considering temporal change in the earth scene, the $p_{t_1}(x,y)$ is the reference image sensed at time t_1 and $p_{t_2}(x,y)$ is the same basic image with the temporal change added. Thus:

$$p_{t_2}(x,y) = p_{t_1}(x,y) + \Delta p_{t_1 t_2}(x,y) \qquad \text{A-1}$$

The $\Delta p_{t_1 t_2}(x,y)$ is the change in the image over the time interval $t_1 t_2$ and is due to changes made in the scene by cultural practices, seasonal effects, atmospheric effects, sun angle changes and other events such as floods or fire. The change is the information carrying quantity for the analyst and it is noise for the image registration processor for if there were no change, then registration would be a much easier problem. The misregistration is a shift of $p_{t_2}(x,y)$ with respect to $p_{t_1}(x,y)$. Thus the misregistered image is represented as:

$$p_{t_2}(x,y) = p_{t_1}(x+\delta_x, y+\delta_x) + \Delta p_{t_1 t_2}(x+\delta_x, y+\delta_y) \qquad \text{A-2}$$

The premise for the approach being investigated is that the misregistration can be deduced from the Fourier transforms of the two images. The Fourier transform of $p_{t_2}(x,y)$ is:

$$\begin{aligned} FT\,[p_{t_2}(x,y)] &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} p_{t_2}(x,y)e^{-j2\Pi(ux+vy)}dxdy \qquad \text{A-3} \\ &= FT\,[p_{t_1}(x,y)]e^{j2\Pi(u\delta_x+v\delta_y)} + FT\,[\Delta p_{t_1 t_2}(x,y)]e^{j2\Pi(u\delta_x+\delta_y)} \end{aligned}$$

The misregistration thus manifests itself as a linear phase term in the transform of the misregistered image. Estimation of the phase term due to misregistration is the goal of this approach.

The transform can be expressed as a magnitude and phase for each frequency and the problem is to separate the phase components due to misregistration from those components due to the reference image and the

difference image. The one-dimensional case permits a more convenient exploration of the problem.

$$\begin{aligned} FT[p_{t_2}(x)] &= |P_{t_1}(u)| e^{j\phi_{t_1}(u)} \; e^{j2\Pi u\delta_x} \\ &+ |\Delta P_{t_1t_2}(u)| e^{j\phi_{t_1t_2}(u)} \; e^{j2\Pi u\delta_x} \\ &= |P_{t_1}(u)| \left\{ e^{j\phi_{t_1}(u) + j2\Pi u\delta_x} \right\} \\ &+ |\Delta P_{t_1t_2}(u)| \left\{ e^{j\phi_{t_1t_2}(u) + 2\Pi u\delta_x} \right\} \end{aligned} \qquad \text{A-4}$$

where the capital P denotes the Fourier transform of p(x,y). The phase angle of the misregistered image is obtained by manipulation of the complex variables. Let $\phi'_{t_2}(u)$ be the phase of the misregistered time t_2 image.

Then:

$$\phi'_{t_2}(u) = \tan^{-1}\left[\frac{\text{Im } P_2(u)}{\text{Re } P_2(u)}\right]$$

$$= \tan^{-1}\left[\frac{|P_{t_1}(u)|\sin(\phi_{t_1}(u) + j2\Pi u\delta_x) + |\Delta P_{t_1t_2}(u)|\sin(\phi_{t_1t_2}(u) + 2\Pi u\delta_x)}{|P_{t_1}(u)|\cos(\phi_{t_1}(u) + j2\Pi u\delta_x) + |\Delta P_{t_1t_2}(u)|\sin(\phi_{t_1t_2}(u) + 2\Pi u\delta_x)}\right] \qquad \text{A-5}$$

The problem here is to find δ_x, given the measured phase functions. Since the reference image is known, the magnitude and phase functions $P_{t_1}(u)$ and $\phi_{t_1}(u)$ can be evaluated. The true difference transform is thus what is unknown. Previous work has provided evidence that the temporal change statistics are similar to the statistics of the reference image only with a different spectral amplitude. Assuming the image is a Gaussian process with a Markof spatial correlation characteristic, the

difference image can be assumed to be zero mean. A least squares fit of the linear phase line to the measured phase difference is the approach taken here to estimate the misregistration. It is desired to find the δ_x, which minimizes the expression:

$$\int_{-\infty}^{\infty} (\Delta\phi(u) - 2\Pi\delta_x u)^2 du \qquad \text{A-6}$$

where: $$\Delta\phi(u) = \phi'_{t_2}(u) - \phi_{t_1}(u)$$

This is a least squares fit of the linear phase line to the measured phase difference values. In the discrete case, the integral becomes a summation and the standard method of solution is to differentiate with respect to the unknown parameters and set the result to zero.

$$\varepsilon = \sum_{i=1}^{N} (\Delta\phi_i - 2\Pi\delta_x u_i)^2 \qquad \text{A-7}$$

$$\frac{d\varepsilon}{d\delta_x} = \sum_{i=1}^{N} 2(\Delta\phi_i - 2\Pi\delta_x u_i)(-2\Pi u_i) = 0 \qquad \text{A-8}$$

This results in:

$$\hat{\delta}_x = \frac{1}{2\Pi} \frac{\sum_{i=1}^{N} \Delta\phi_i u_i}{\sum_{i=1}^{N} u_i^2} \qquad \text{A-9}$$

The important characteristic of this estimator is its variance. An expression for the variance of this misregistration estimate will enable comparison to other methods. In particular, in the work by Svedlow[2] an expression was developed for the variance of the registration estimate using a matched filter which was also shown to be the optimum linear process for registration error estimation. This important variance derivation is presented in the Appendix as a reference for the difference image method explored here.

1.2 Variance of the Registration Error Using Matched Filter

The development presented in the Appendix provides results relating the signal-to-noise ratio of the images to the variance of the estimator. The expressions are:

$$\mathrm{Var}[(\hat{\delta}_x - \delta_x)] = \frac{1}{B_x^2\,\mathrm{SNR}}$$

$$\mathrm{Var}[(\hat{\delta}_y - \delta_y)] = \frac{1}{B_y^2\,\mathrm{SNR}} \qquad \text{A-10}$$

where $B_{x,y}$ are the effective bandwidths of the x and y dimension in the noise or temporal difference image. The signal-to-noise ratio is the ratio of energy in the reference image to the noise energy in the random temporal difference image. The $\hat{\delta}_x, \hat{\delta}_y$ are the estimated misregistrations and the δ_x, δ_y are the true misregistrations. Making assumptions that the images are bandlimited and that they have the same spectrum shape with different amplitudes, a simplified expression can be obtained as a function only of the bandlimits ω_x, ω_y. The standard deviation of the estimates then becomes:

$$\mathrm{Std}[(\hat{\delta}_x - \delta_x)] = \frac{1}{2\Pi\omega_x}\sqrt{\frac{3}{\mathrm{SNR}}}$$

$$\mathrm{Std}[(\hat{\delta}_y - \delta_y)] = \frac{1}{2\Pi\omega_y}\sqrt{\frac{3}{\mathrm{SNR}}} \qquad \text{A-11}$$

For the Landsat case, the expressions become:

$$\text{Std lines} = \frac{44.1}{\sqrt{\mathrm{SNR}}} \text{ meters}$$

$$\text{Std cols} = \frac{33.1}{\sqrt{\mathrm{SNR}}} \text{ meters}$$

These functions are plotted in Figure A-1 to show the relationship graphically. It is the purpose of this investigation to obtain the same relationship for the estimator using the Fourier difference spectrum.

1.3 Variance of Registration Estimate Using Difference Phase

The variance of the estimator given in Section 1.1 can be obtained from the variances of the input processes.

$$\mathrm{VAR}[\hat{\delta}_x] = E[(\hat{\delta}_x - \delta_x)^2] = E\left[\left(\frac{1}{2\Pi}\frac{\sum_{i=1}^{N}\Delta\phi_i u_i}{\sum_{i=1}^{N}u_i^2} - \delta_x\right)^2\right] \qquad \text{A-12}$$

$$= E\left[\left(\frac{1}{2\Pi}\frac{\sum_{i=1}^{N}\Delta\phi_i u_i}{\sum_{i=1}^{N}u_i^2}\right)^2\right] - 2E\left[\frac{1}{2\Pi}\frac{\sum_{i=1}^{N}\Delta\phi_i u_i}{\sum_{i=1}^{N}u_i^2}\cdot\delta_x\right]$$

$$+ E\left[\delta_x^2\right]$$

We can assume the δ_x is zero without loss of generality. Then the expression is simply a linear combination of the expectations of squares and cross products of the $\Delta\phi_i$. If we further assume that the $\Delta\phi_1$ are uncorrelated and of equal variance, the variance becomes:

$$\mathrm{VAR}[\hat{\delta}_x] = E[\Delta\phi_1^2]\,\frac{1}{4\Pi^2}\frac{\sum_{i=1}^{N}u_i^2}{\left(\sum_{i=1}^{N}u_i^2\right)^2} = \frac{1}{4\Pi^2}\frac{E[\Delta\phi_1^2]}{\sum_{i=1}^{N}u_i^2} \qquad \text{A-13}$$

The $\Delta\phi_1$ is the result of the phase and magnitude variations as defined in equation A-5. The random process defining the temporal change is assumed to be Gaussian; thus the imaginary and real parts of the Fourier transform are also Gaussian. The ratio of the imaginary to the real parts is a ratio of two Gaussian random variables which results in a Cauchy distributed random variable. The arctangent function applied to a Cauchy random variable results in a uniformly distributed random variable with range $-\Pi/2$ to $\Pi/2$.[3] The variance of the phase is thus $\Pi^2/12$ radians. The variance then of the estimate appears to be independent of the variance of the noise process which is representing the temporal change.

The value of the variance of δ_x is then .8225 radians multiplied by the factor K:

$$K = \frac{1}{4\Pi^2\sum_{i=1}^{N}u_i^2} \qquad \text{A-14}$$

The summation of the frequency values is a summation of fractions up to the value one-half since the sample interval is taken as unity, so the maximum frequency is 1/2: $u = i/N$ $i=1,...N/2$ where N is the total number of points Fourier transformed. Thus we have:

$$K = \frac{1}{4\Pi^2 \frac{1}{N^2} \sum_{i=1}^{N/2} i^2} \qquad \text{A-15}$$

For N=16 K=.008 and for N=32 K=.004; thus the value will be very small for large arrays. The dimension of K is distance units squared and if the one-unit spacing is scaled to the 80 meter Landsat sample spacing and more than 32 points are used, the value of the variance would be less than $.004 \times .8225 \times 6400 \cong 21$ meters or the standard deviation would be 4.6 meters.

This result is not considered to be reasonable since it does not depend on the effect of temporal change in the two images. Clearly higher levels of change in the images will make registration more difficult and in the limiting case of infinite variance for the temporal change, the error should be infinite. This result is correctly predicted by the model described in the Appendix and plotted in Figure A-1. A realistic model for the variance of the error in the phase difference estimate was not developed in the study as desired. The above result is interesting in that if the arctangent function is used, the variance of any phase estimator would appear to be bounded by some small number in the Π radian range. This contradicts the fact that increasing temporal variance makes images increasingly dissimilar. Further work is needed to develop correct analytical expressions for the variance of the proposed misregistration approach.

1.4 Experimental Analysis Method

To explore the behavior of the proposed misregistration method, a numerical evaluation was carried out using registered LACIE segments. Early in the study, ten segments were identified as a test set for the evaluation. As it became apparent that analytical evaluation of the method was not going to be achieved in the course of the study, it did not seem desirable to plan on an extensive numerical evaluation. Some work was carried out to observe the behavior of the phase differences of the images for one segment: 854 in Tippecanoe County, Indiana.

Misregistration estimates were made using the correlation coefficient and a correlation block size of 32 by 32 pixels from ten acquisitions for this segment for the 1978 growing season. A typical correlation function plot is shown in Figure A-2. The peak of the function was interpolated using second degree Lagrange polynomials and the fractional position of the peak was determined. These estimated fractional misre-

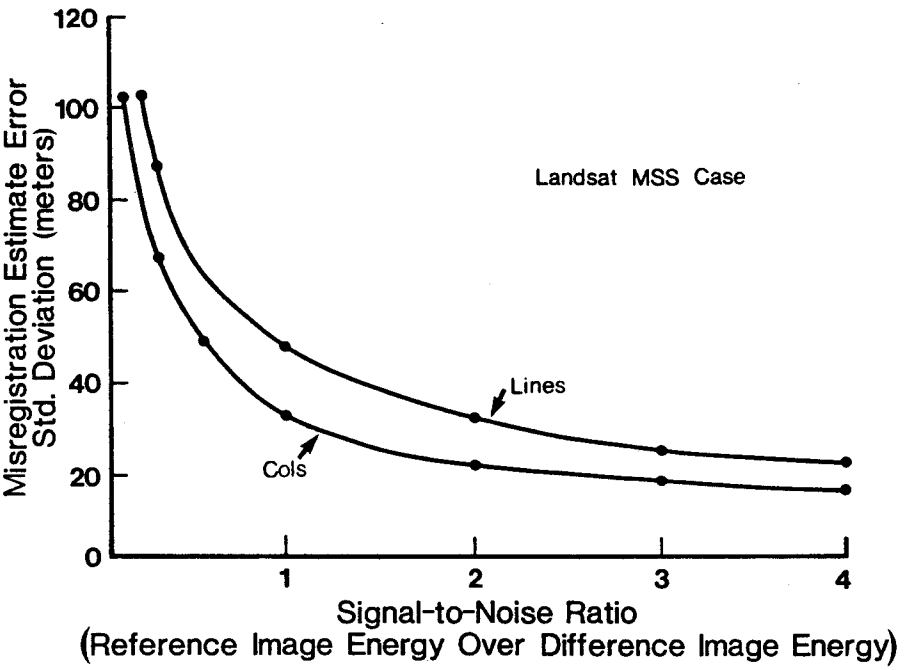


Figure A-1. Standard deviation of estimated misregistration using matched filter registration processor.

CORRELATION SURFACE (CORRELATION COEFFICIENT)

-6	-6	-4	-3	-1	0	-2	-1	-7	-11	-14	-17	-20	-21	-19	-15	-13
-5	-6	-4	-1	-1	0	0	-1	-4	-7	-10	-14	-17	-19	-18	-15	-12
-8	-8	-6	-2	-1	-1	0	2	1	0	-5	-10	-14	-16	-16	-15	-12
-7	-8	-8	-4	-1	0	1	4	4	3	-1	-7	-11	-14	-14	-11	-9
-7	-10	-11	-10	-5	-1	2	9	9	8	4	0	-3	-6	-6	-4	-2
1	-2	-5	-5	0	8	14	24	26	22	16	8	2	0	0	4	5
8	8	6	5	10	17	22	30	34	29	24	15	7	4	4	5	6
9	8	6	5	11	20	31	43	53	49	38	24	13	8	7	4	3
15	15	16	15	17	23	30	39	49	43	33	21	10	6	2	0	0
9	4	3	3	5	13	22	29	37	36	26	14	2	-2	-5	-8	-7
8	4	1	1	1	6	13	19	24	24	14	6	0	-3	-4	-9	-8
2	-2	-7	-6	-5	0	9	15	19	20	13	6	1	-2	-5	-9	-9
-1	-3	-6	-5	-2	2	10	17	19	19	13	6	3	1	-2	-8	-9
-2	-4	-7	-7	-5	-4	4	11	13	13	9	3	1	0	-5	-10	-10
-6	-8	-12	-14	-11	-9	-2	6	9	10	8	5	1	0	-3	-6	-8
-3	-7	-10	-14	-12	-9	-4	3	6	6	5	4	0	-2	-4	-6	-6
-2	-6	-8	-13	-14	-10	-5	1	3	4	3	2	1	0	-1	0	-1

Figure A-2. Correlation function plot for Segment 854 for reference date Aug. 21, 1978 and Aug. 4, 1978 overlay date. Maximum point is at the center column and one row up from the center row. The correlation value at the peak is .54.

gistrations are listed in Table A-1 using a mid-season date of August 21 as the reference. The last row in the table is the August 21 date correlated with itself as a check on the algorithm. The misregistration values are what the correlator estimates as the remaining misregistrations in the LACIE segments after registration by the NASA processor. These estimates are, of course, subject to the error variance associated with this method so cannot be taken as exact values.

The variance of an optimum processor used to register the two images is a function of the signal-to-noise ratio, as defined in the Appendix. This ratio can be estimated from the Fourier transforms of the two images. Since the registration method being investigated requires the Fourier transform of the images, the transforms of the fourteen acquisitions were computed and stored on tape. Ten of these are used in the study. The dates from September 27 on were not used as they were judged to be too late in the season to offer good results with an algorithm we knew little about. The Fourier data file information is presented in Table A-2. The transform dimensions are 64 x 64 points. The next larger size, 128 x 128, could not be used since the data sets had only 117 lines. Gray-scale images of two acquisitions are shown in Figures A-3 and A-4 for dates August 21 and August 4, respectively. Channel 2 (Band 5) was used for the study as it displays good road structure relative to the IR bands and better signal-to-noise ratio than Channel 1 (Band 4). A gray-scale reproduction of the magnitude of the FFT is shown in Figure A-5 for the August 21 date. This image is typical of all transforms of this type of scene and is presented for its pictorial value. The two general loci of brightness, one running top to bottom and the other faintly left to right, represent the road structure.

The misregistration estimator was programmed and applied to the Fourier transform files. The phase difference of the transforms is computed for each frequency value. The evaluation was begun with the August 21 data set misregistered one line and one column with respect to itself to give an image pair of known misregistration and no temporal

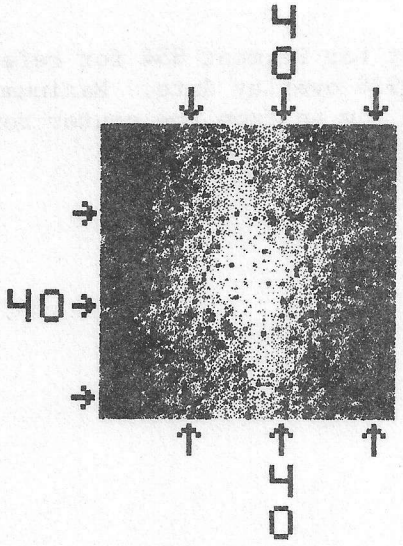


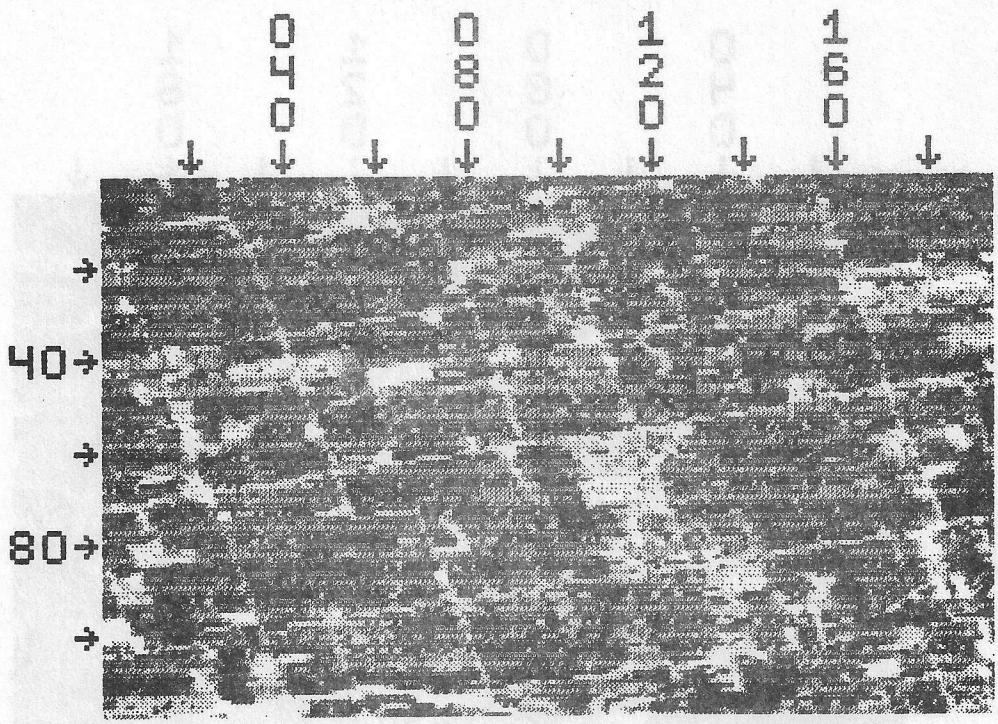
Figure A-5. Fourier transform magnitude for Segment 854, Aug. 21, 1978, obtained from the first 64 lines and columns of Channel 2.

Table A-1. Misregistration errors in sample segment obtained from image correlation and using cubic interpolation to find fractional misregistration. Segment 854, Tippecanoe County, Indiana.

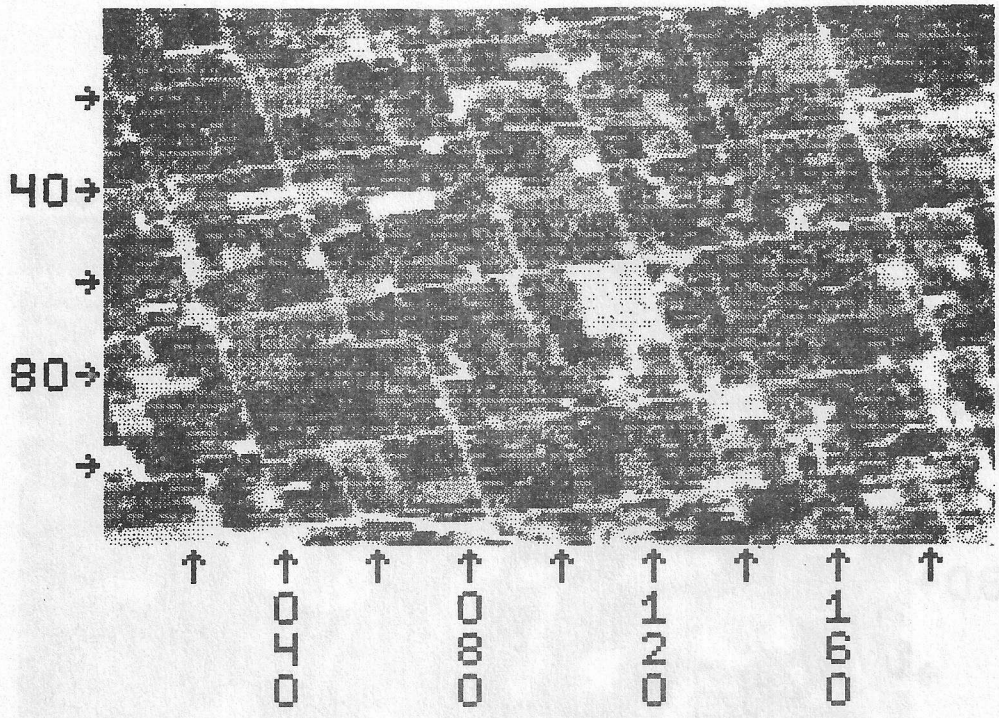
Reference Run is 78542330 Aug. 21, 1978				
Overlay Runs	Date	Discrete Peak Correlation Value	Fractional Misregistration Lines	Columns
78542340	Aug. 21, 1978	.68	-0.54	-0.76
78542430	Aug. 31, 1978	.56	-0.5	0.0
78542510	Sept. 8, 1978	.56	-0.03	-1.40
78542520	Sept. 9, 1978	.40	-0.05	-1.23
78542690	Sept. 26, 1978	.27	0.09	-0.50
78541610	June 10, 1978	.28	0.535	-0.50
78541970	July 16, 1978	.32	0.27	+0.70
78542070	July 26, 1978	.25	0.21	-1.30
78542160	Aug. 4, 1978	.54	-0.67	0.21
78542330	Aug. 22, 1978	1.00	0.0	0.0

Table A-2. Fourier transform files for Segment 854 (Tippecanoe County, Indiana). Channel 2 (.6 - .7μm) used for transform. (Data on LARS Tape 26 at time of publication.)

Data Set ID No.	File No.	Date	Additional Information
78542332	1	Aug. 21, 1978	64x64 points
78542342	2	Aug. 22, 1978	"
78542432	3	Aug. 31, 1978	"
78542512	4	Sept. 8, 1978	"
78542522	5	Sept. 9, 1978	"
78542692	6	Sept. 26, 1978	"
78542702	7	Sept. 27, 1978	"
78543062	8	Nov. 2, 1978	"
78543512	9	Dec. 17, 1978	"
78543592	10	Dec. 25, 1978	"
78541612	11	June 10, 1978	"
78541970	12	July 16, 1978	"
78542072	13	July 26, 1978	"
78542162	14	Aug. 4, 1978	"
78542334	15	Aug. 21, 1978	Origin shifted one line and one column

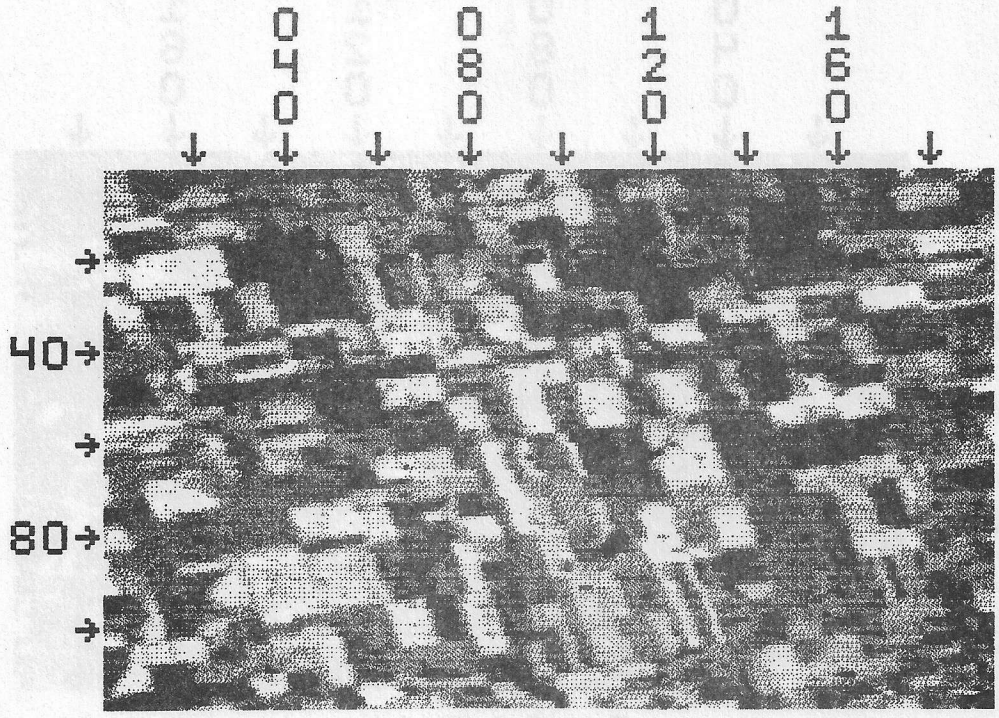


Channel 1 (Band 4)

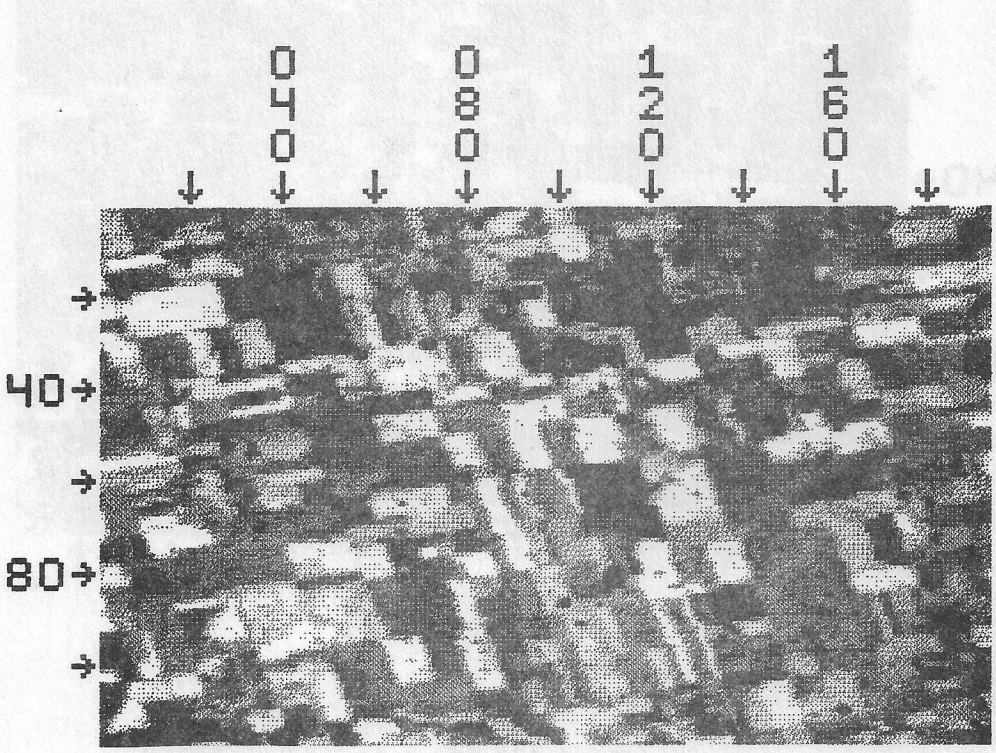


Channel 2 (Band 5)

Figure A-3. Gray-scale images of Segment 854 for Aug. 21, 1978 used as a reference image.

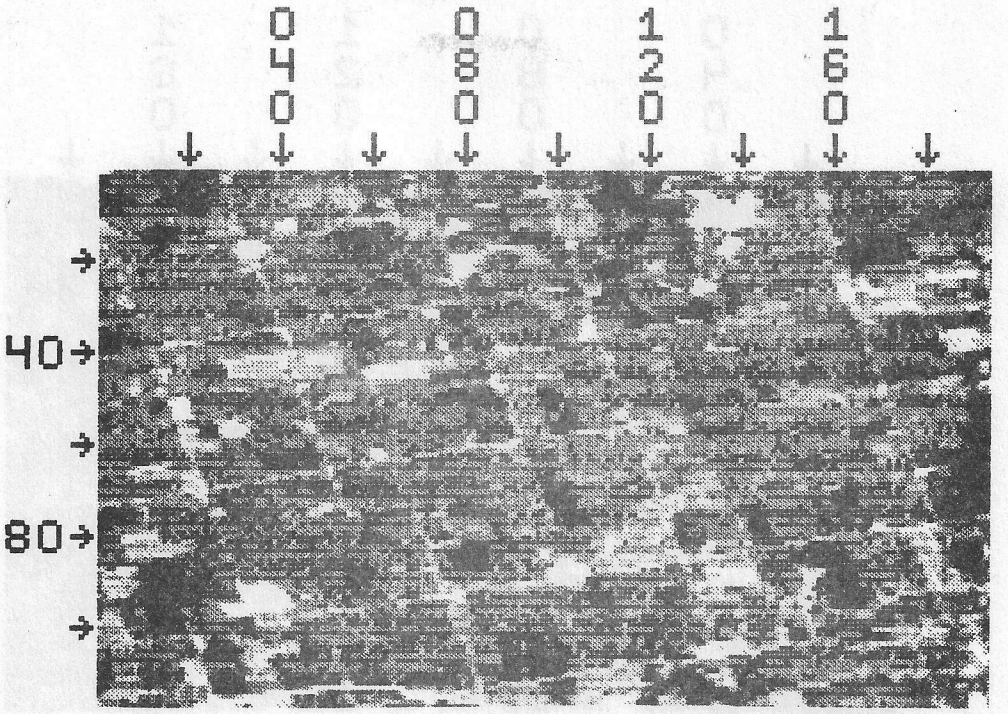


Channel 3 (Band 6)

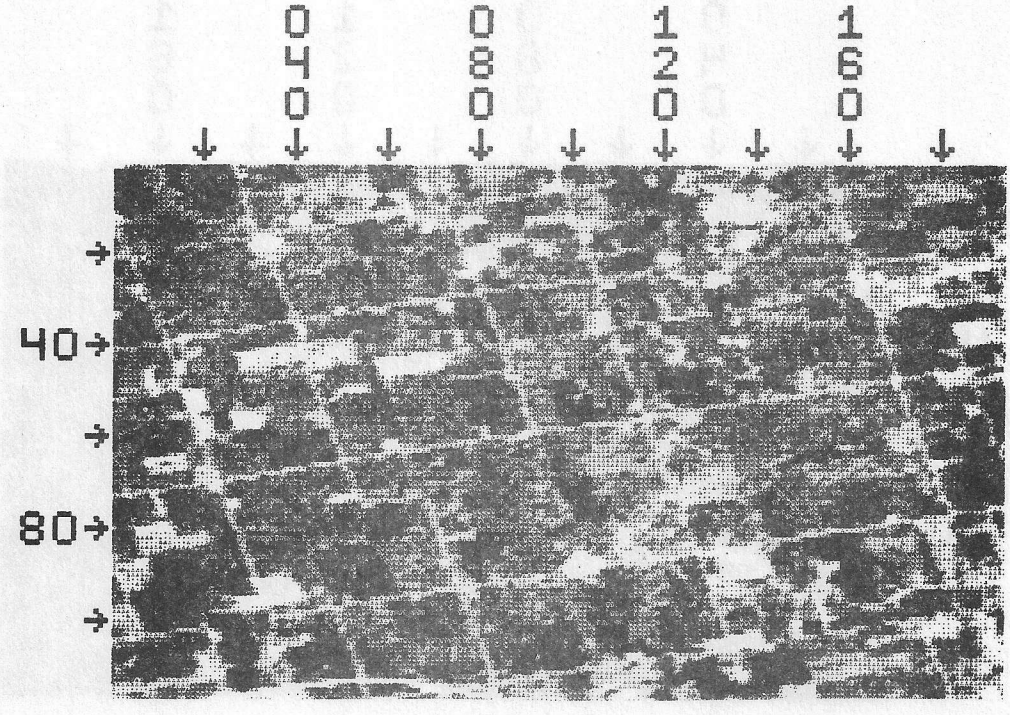


Channel 9 (Band 7)

Figure A-3 (continued).

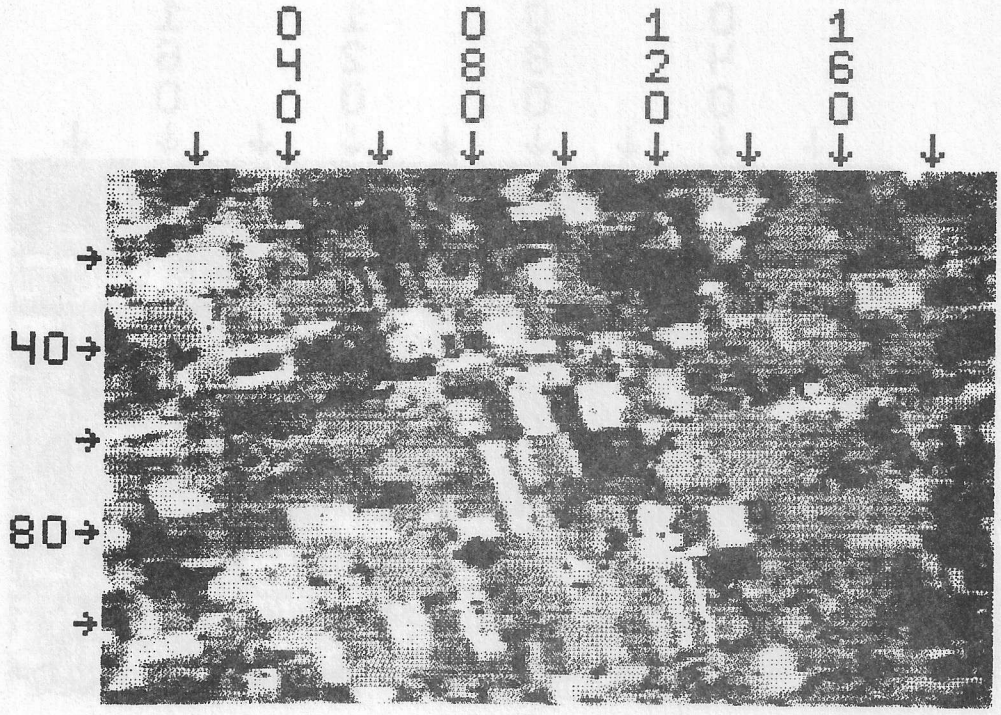


Channel 1 (Band 4)

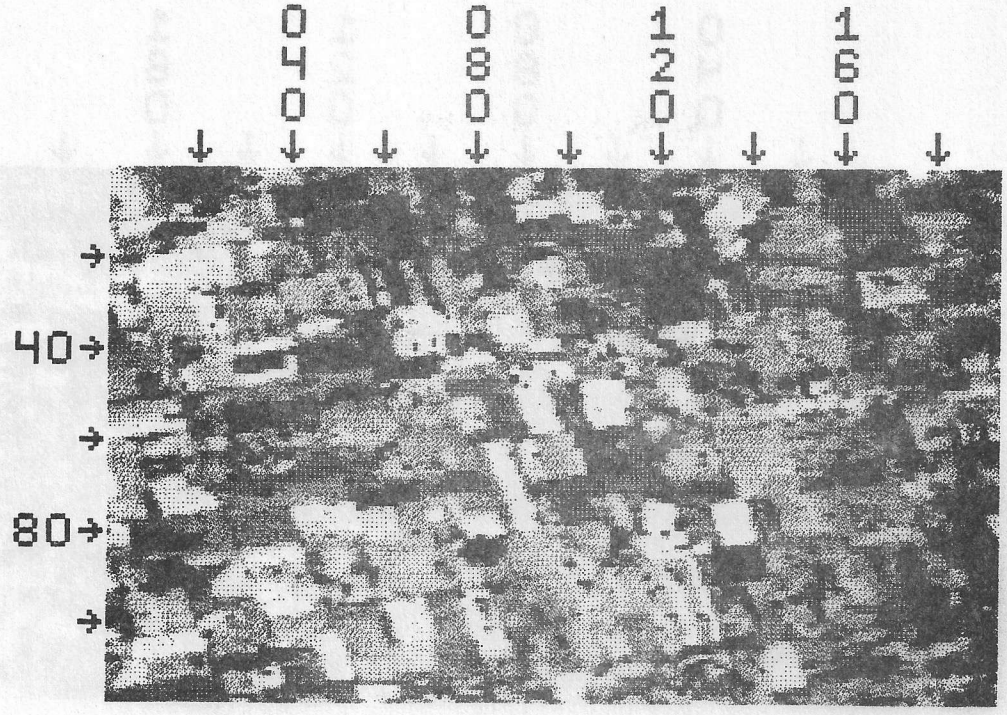


Channel 2 (Band 5)

Figure A-4. Gray-scale images of Segment 854 for the Aug. 4, 1978 acquisition.



Channel 3 (Band 6)



Channel 4 (Band 7)

Figure A-4 (continued).

change. The linear phase term should have a slope of unity for this case and at the maximum frequency of 1/2 the value should be:

$$2\pi u \delta_x = 2\pi x \tfrac{1}{2} x 1 = \pi$$

Thus the phase difference curve should go from 0 to π and the linear phase line should pass through these points.

The phase angles used for the estimation for columns are all values for which the line frequency is zero and vice versa. The phase difference angles for the column direction are plotted in Figure A-6 and those for the line direction are plotted in Figure A-7. The desired linear phase line with a unity slope is shown starting at zero and ending at pi. Phase angles are close to the expected curve at low frequencies but deviate widely at high frequencies. Restricting the least squares curve fit to only the lower half of the frequency curve reduces the effect of this dispersion somewhat. The column misregistration estimate in this case is .97 and the line estimate is .65 pixels.

The Fourier transform program assumes a two-dimensional periodic image; thus the shift of one pixel here introduces new data and essentially adds noise to the transform of the second image. This was suspected of causing the variability in phase so a special data set was constructed which was periodic with a period of 64 pixels. The time-two transform was then taken of a 64 x 64 block shifted one line and column in the periodic image data. The phase differences are plotted in Figures A-8 and A-9; a strict linear phase line is observed in each case. The curve fit results also produced exact 1.000 line and 1.000 column estimated registration which is the correct and expected result. Thus we concluded the algorithm was properly implemented and could be tested on the other segment acquisitions.

The algorithm was tested on the ten acquisitions for which correlations were run. The results are listed in Table A-3 along with the correlation results. It is apparent that no readily recognizable agreement exists between the correlation and phase difference predictor results. The wide variability of the phase angle for the test case where one new line and column were introduced (Figures A-6 and A-7) suggests that there may be serious problems with this approach. This is reinforced by the results from the nine other acquisitions. Based on these results, the decision was made not to pursue this Fourier transform method further in this task and to instead concentrate resources on the question of misregistration effects on classification accuracy. Further study of Fourier transform methods of determining misregistration is recommended to determine the cause of the observed variability and possibly define an algorithm which will perform satisfactorily. Theoretically, this should be a good method of determining small misregistrations because of the linear phase properties of translation into the Fourier domain. The investigation conducted into models for misregistration effects on classification accuracy will be discussed in the remainder of this task report.

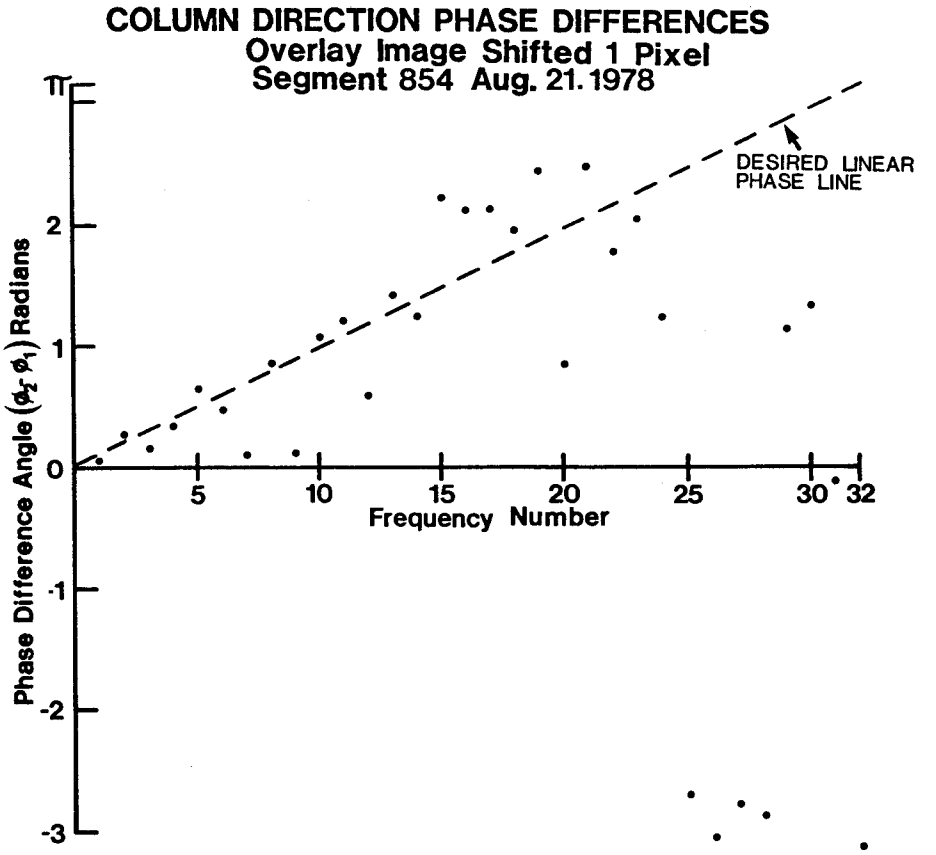


Figure A-6. Phase difference angles for column direction for overlay image shifted one line and one column.

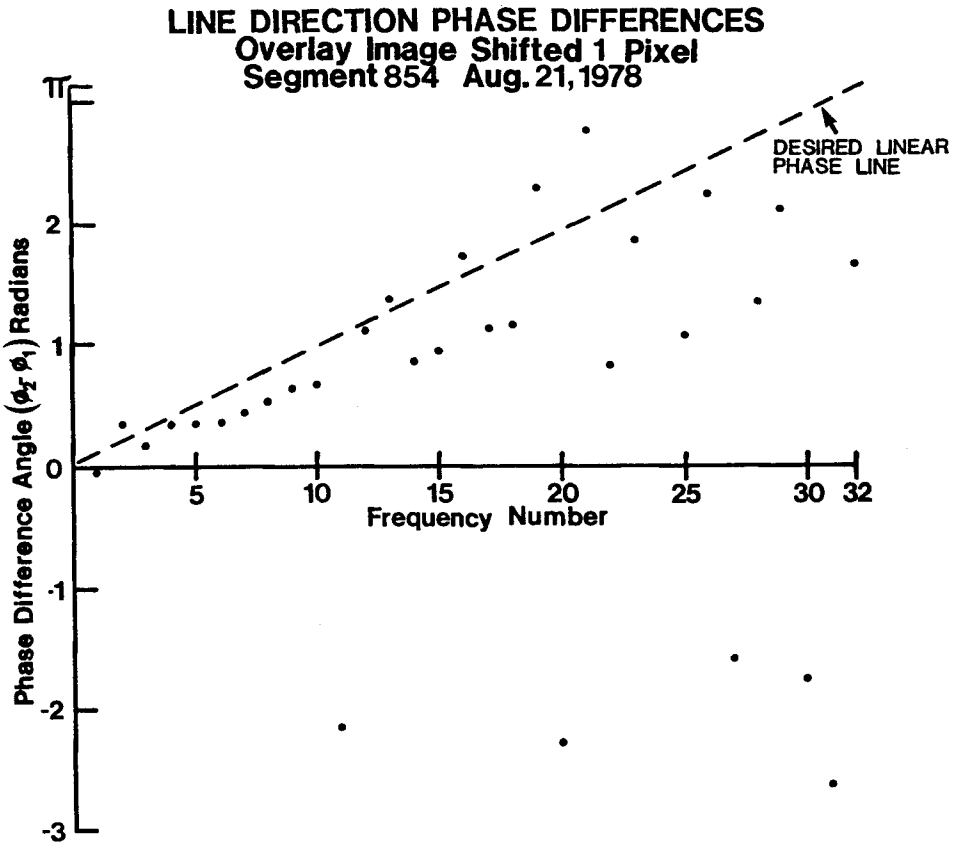


Figure A-7. Phase difference angles for line direction for overlay image shifted one line and one column.

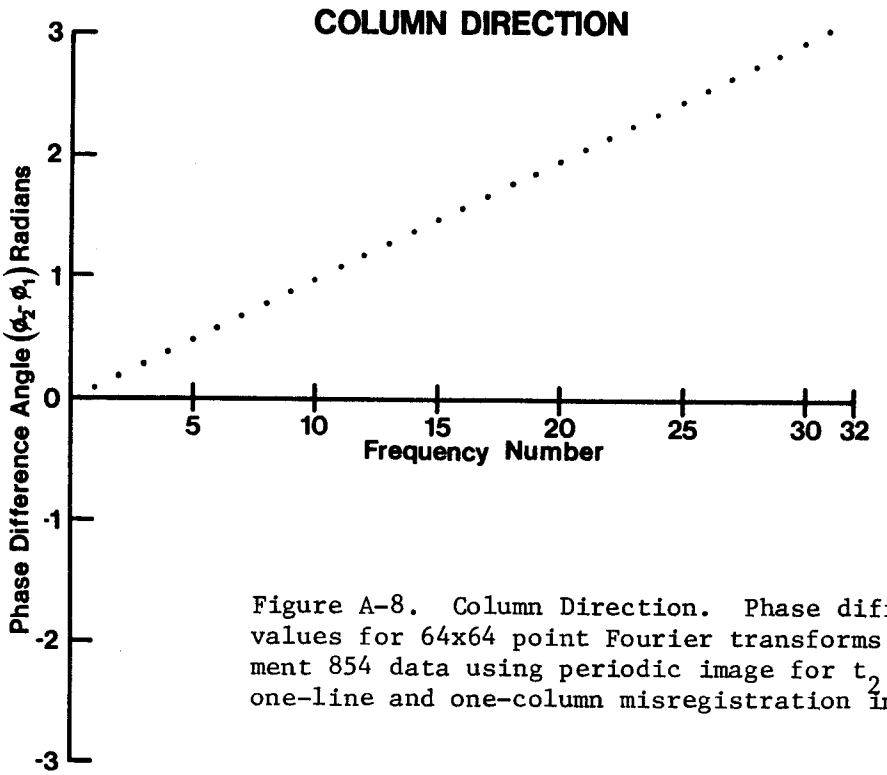


Figure A-8. Column Direction. Phase difference values for 64x64 point Fourier transforms of Segment 854 data using periodic image for t_2 with a one-line and one-column misregistration introduced.

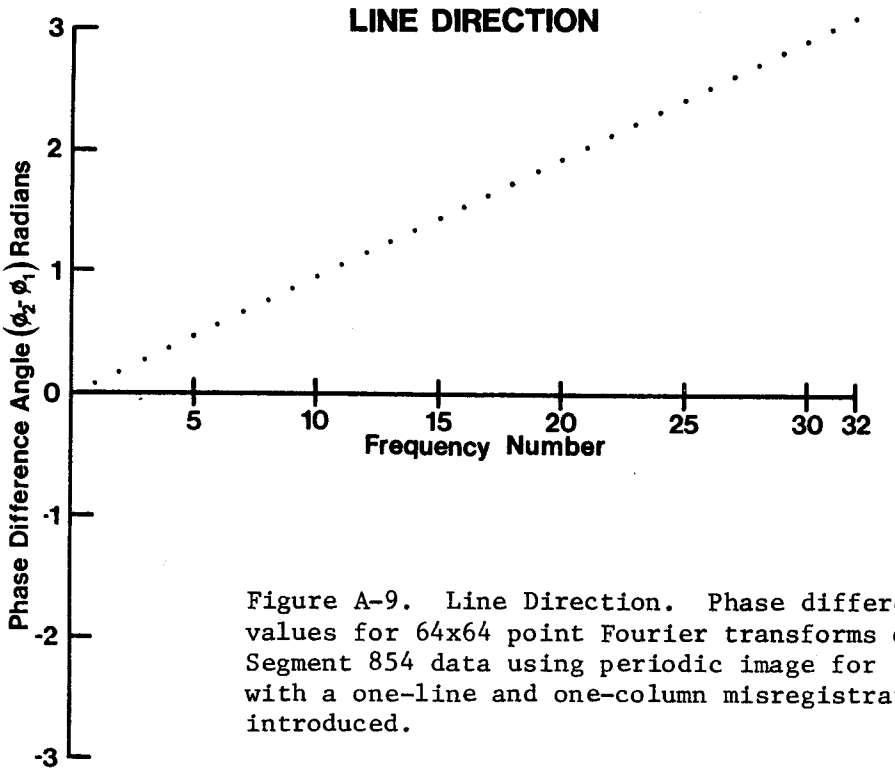


Figure A-9. Line Direction. Phase difference values for 64x64 point Fourier transforms of Segment 854 data using periodic image for t_2 with a one-line and one-column misregistration introduced.

Table A-3. Estimated misregistration for Segment 854 acquisition using phase difference algorithm.

Reference Run is 78542330 Aug. 21, 1980				
Date (1978)	Misregistration From Correlation Lines	Columns	Misregistration From Phase Algorithm Lines	Columns
Aug. 22	-.54	- .76	.08	- .86
Aug. 31	-.5	0.0	-.86	.24
Sept. 8	-.03	-1.40	-.55	.18
Sept. 9	-.05	-1.23	-.08	.05
Sept. 26	.09	- .50	-.54	- .31
June 10	.54	- .50	-.15	1.16
July 16	.27	.70	-.36	1.09
July 26	.21	-1.30	-.18	.72
Aug. 4	-.67	.21	.11	.12

2. Classification Error Models for Misregistration

2.1 Introduction

Spatial misregistration in multitemporal, multispectral scanner data affects the performance of classifier processors working on such data. In order to analyze these effects, it is useful to identify and to study particular cases before developing a general model that can take into account the different parameters that are present in the problem. A discussion of those cases is in the next paragraphs.

2.2 Field-Center Pixels That Remain as That After Misregistration

When the amount of misregistration is such that field-center pixels will remain the same, the major effect on the training statistics computed from those pixels will be in the change of value of the nondiagonal elements of the covariance matrix. Specifically this change occurs in the covariance between misregistered channels. To see this more clearly, let the following conditions be assumed: There are two classes (Class A and Class B), two channels (Channel i and Channel j), and the covariance matrices of the statistics from both classes are the same (K). The class mean vectors will be denoted by $\bar{a}$ and $\bar{b}$, respectively. Assuming equally likely classes, the probability of error in classification (P_E) will be

$$P_E = \tfrac{1}{2}(P_F + P_M)$$

where

P_F = Probability of false alarm (say, Class A when it is Class B)

P_M = Probability of miss (say, Class B when it is Class A)

For the present problem

$$P_F = P_M = Q\left(\tfrac{1}{2}[\bar{\mu}^{-T} K^{-1} \bar{\mu}]^{\frac{1}{2}}\right)$$

where

$$Q(\alpha) = \frac{1}{\sqrt{2\pi}} \int_{\alpha}^{\infty} e^{-x^2/2}\, dx$$

$$\bar{\mu} = \bar{a} - \bar{b} = \begin{bmatrix} a_1 - b_1 \\ a_2 - b_2 \end{bmatrix} = \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix}$$

Due to the misregistration, the covariance matrix K changes to K_{MR}. Assuming, as in [1], a linear decreasing cross correlation between mis-

registered channels, K_{MR} is of the form

$$K_{MR} = \begin{bmatrix} \sigma_1^2 & (1-\varepsilon)\ \rho\ \sigma_1\ \sigma_1 \\ (1-\varepsilon)\rho\ \sigma_1\ \sigma_2 & \sigma_2^2 \end{bmatrix}$$

where ε = amount of misregistration (relative to pixel dimensions), assuming for simplicity $\sigma_1 = \sigma_2 = \sigma$.

$$P_E = Q\left(\frac{1}{2}[\bar{\mu}^T K_{MR} \bar{\mu}]^{\frac{1}{2}}\right)$$

$$= Q\left(\frac{1}{2}\left[\frac{\mu_1^2 + \mu_2^2 - 2\ \mu_1\ \mu_2\ \rho\ (1-\varepsilon)}{\sigma^2\ 1 - \rho^2(1-\varepsilon)^2}\right]^{\frac{1}{2}}\right)$$

The expression above can then be used for computing the probability of error for different combinations of parameters.

Figure A-10 shows the probability of error for different values of the cross-correlation ρ when $u_1 = u_2 = \sigma = 1$. From that figure, it is observed that the effect of the spatial misregistration can be quite different as the cross-correlation value varies. Thus it does not affect it at all if $\rho = 0$. If $\rho = -0.8$ P_E increases as ε increases and finally P_E actually <u>decreases</u> with ε if $\rho = 0.8$. Therefore for this particular case, the presence of misregistration does not necessarily mean degrading the performance of the classifier.

2.3 Boundary Pixels

Boundary pixels, i.e., the ones that are close to the lines that separate different cover types, are affected by spatial misregistration more markedly than field-center pixels by the fact that, in general, they will become mixed-class pixels. The statistics of these pixels (mean vector and covariance matrix) will be different from the ones coming from the training fields which, if the amount of misregistration is not excessive, will have only field-center pixels. To illustrate some of these aspects, let it be assumed that a pixel is lying just next to a boundary between two different classes or cover types. Without misregistration, it would be a pure Class B pixel, but with misregistration

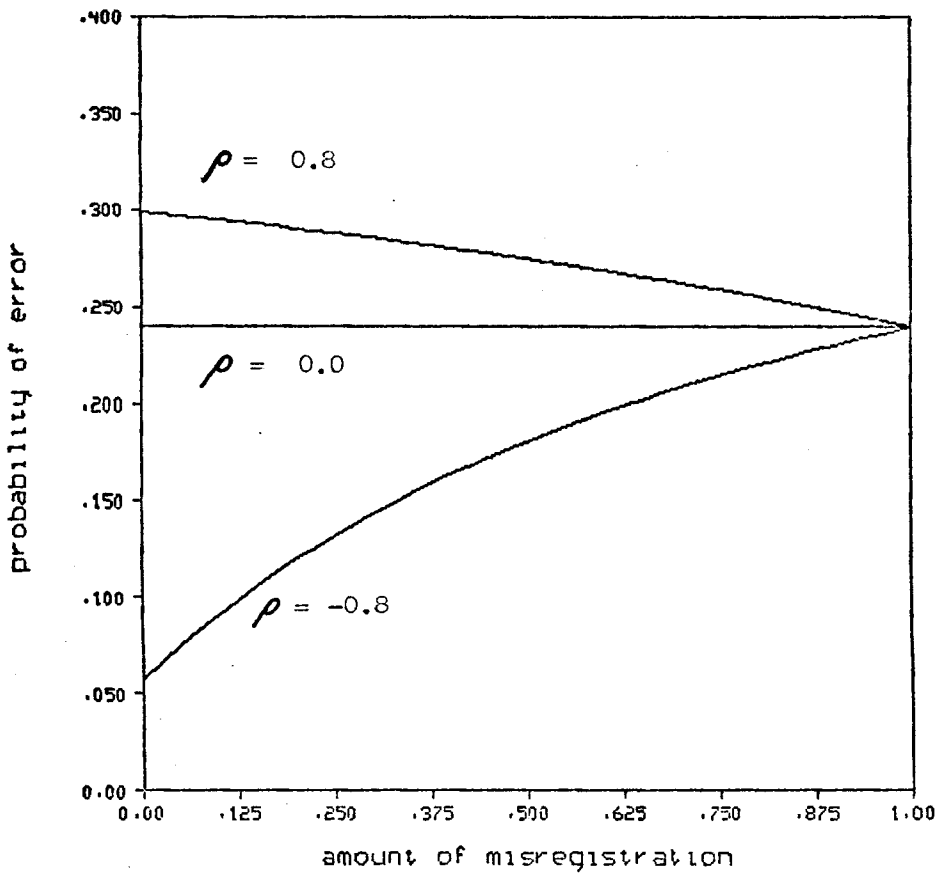


Figure A-10. P_E when classifying field-center pixels.

along one channel it will become a mixed-class pixel. If the covariance matrices of both classes are considered equal, the statistics of such a pixel when the relative amount of misregistration is ε would be

$$\text{mean vector } \bar{m} = \begin{bmatrix} m_1 \\ m_2 \end{bmatrix} = \begin{bmatrix} b_1 \\ (1-\varepsilon)\ b_2 + \varepsilon\ a_2 \end{bmatrix}$$

where it is assumed that the misregistration is along the second channel. The covariance matrix will be the same as the one from the field-center pixels, i.e.,

$$K = \begin{pmatrix} \sigma_1^2 & (1-\varepsilon)\ \rho\sigma_1\ \sigma_2 \\ (1-\varepsilon)\ \rho\sigma_1\ \sigma_2 & \sigma_2^2 \end{pmatrix}$$

The discriminant function based on the statistics coming from the training set is

$$\ell = \mu^T K^{-1} \underset{\underset{B}{\gtrless}}{A} \frac{1}{2} (\bar{a}^T K^{-1} \bar{a} - \bar{b}^T K^{-1} \bar{b})$$

$$E(\ell/B) = \bar{\mu}^T K^{-1} \bar{m}$$

$$\text{Variance } (\ell/B) = \bar{\mu}^T K^{-1} \bar{\mu} = \sigma_x^2$$

$\bar{m}$ can be rewritten as

$$\bar{m} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} + \varepsilon \begin{bmatrix} 0 \\ \mu_2 \end{bmatrix} = \bar{b} + \varepsilon\ \bar{d}_2$$

The probability of false alarm will be

$$P_F = Q\left(\frac{\frac{1}{2}\bar{a}^T K^{-1}\bar{a} - \frac{1}{2}\bar{b}^T K^{-1}\bar{b} - \bar{\mu}^T K^{-1}\bar{m}}{\sigma_x}\right)$$

$$= Q\left(\frac{\frac{1}{2}\bar{\mu}^T K^{-1}\bar{\mu} - \varepsilon\bar{\mu}^T K^{-1}\bar{d}_2}{\sigma_x}\right)$$

assuming $\sigma_1 = \sigma_2 = \sigma$

$$\bar{\mu}^T K^{-1}\bar{\mu} = \frac{\mu_1^2 + \mu_2^2 - 2\mu_1\mu_2(1-\varepsilon)\rho}{\sigma^2(1-(1-\varepsilon)^2\rho^2)}$$

$$\bar{\mu}^T K^{-1}\bar{\mu} = \frac{-\mu_1\mu_2(1-\varepsilon) + \mu_2^2}{\sigma^2(1-(1-\varepsilon)^2\rho^2)}$$

Finally

$$P_F = Q\left(\frac{1}{2}\frac{m_1^2 + \mu_2^2 - 2\mu_1\mu_2(1-\varepsilon)\rho - 2\varepsilon(\mu_2^2 - \mu_1\mu_2(1-\varepsilon)\rho)}{[\mu_1^2 + \mu_2^2 - 2\mu_1\mu_2(1-\varepsilon)]^{\frac{1}{2}}\ [1-(1-\varepsilon)^2 p^2]^{\frac{1}{2}}\sigma}\right)$$

In particular, if $\mu_1 = \mu_2 = 1$

$$P_F = Q\left(\frac{1-\varepsilon}{[2(1+(1-\varepsilon)\rho)]^{\frac{1}{2}}}\right)$$

Figure A-11 illustrates P_F for various values of the cross-correlation ρ and when $\sigma = \mu_1 = \mu_2 = 1$. It is apparent from it that for all cases there is an increase in P_F with ε. The relative amount of this increase is different for different values of ρ.

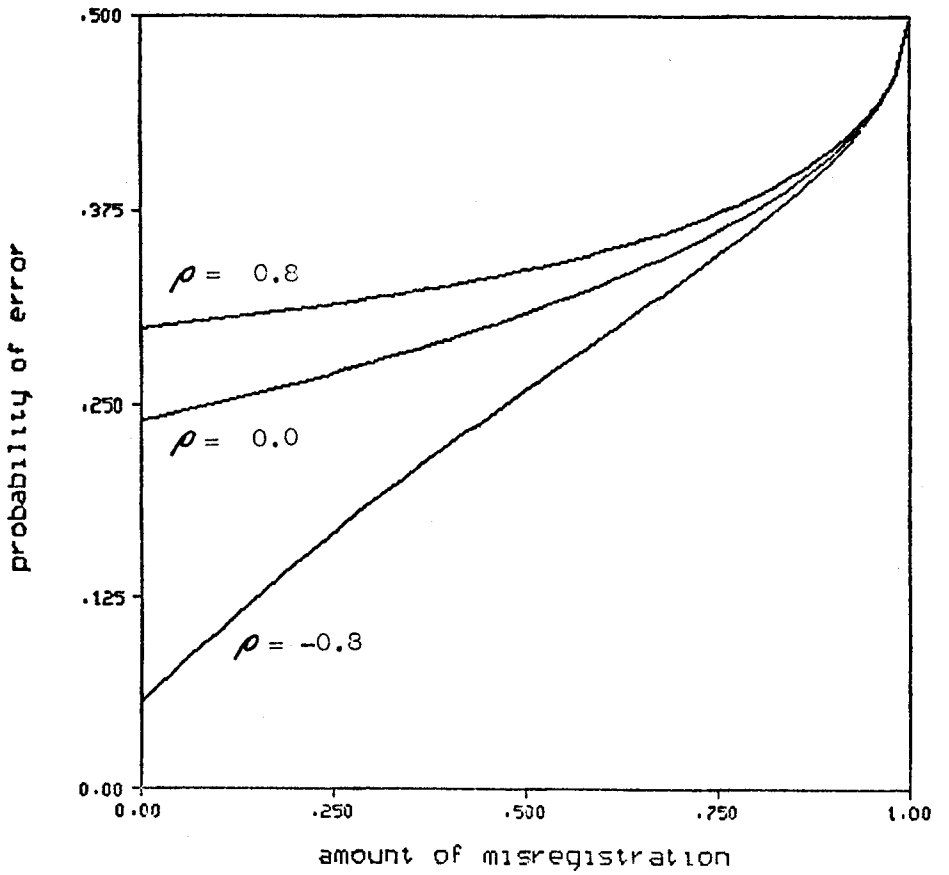


Figure A-11. P_F when classifying boundary pixels.

2.4 Possible Effects When More Than Two Classes Are Present

So far, the effects of misregistration when dealing with two classes and two features have been discussed. Additional effects occur when there are more than two classes. For example, let it be assumed, for simplicity, that there are four classes and also two channels. Let it be assumed that there are equal covariances ($K = \sigma^2 I$) and the disposition of the mean vectors is as in Figure A-12.

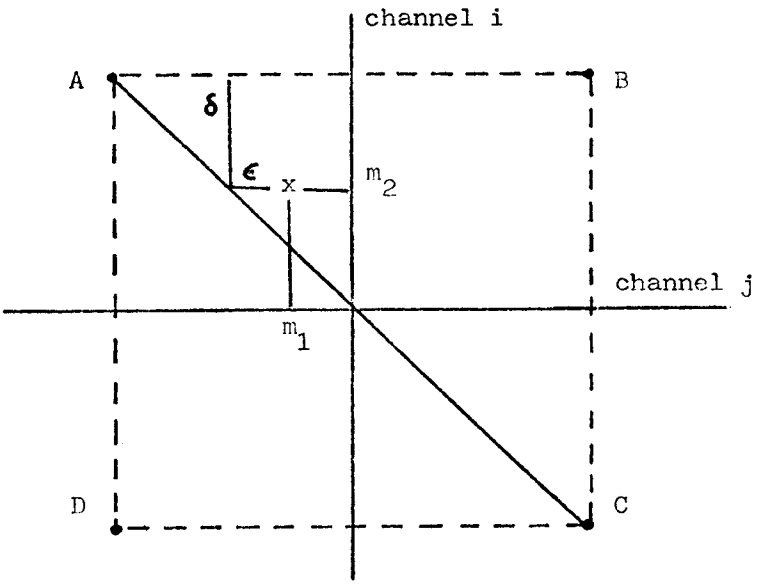


Figure A-12. Locus of mean vectors.

For equally likely classes

$$P_E = 1 - [1-Q(d/2)]^2$$

A mixed-class pixel (let us say between Classes A and C) will have its mean located along the line that goes from the class mean A to the class mean C, as in Figure A-12 where a mixture proportion of $(1-\delta)$ from

Class A plus δ has been assumed. Thus

$$\bar{m}_x = \begin{bmatrix} d/2\ (2\delta-1) \\ \\ -d/2\ (2\delta-1) \end{bmatrix}$$

If in addition there is a relative misregistration ε along the j channel, then

$$\bar{m}_x = \begin{bmatrix} d/2\ [2(\delta+\varepsilon) - 1] \\ \\ -d/2(2\delta-1) \end{bmatrix} = \begin{bmatrix} m_1 \\ \\ m_2 \end{bmatrix}$$

In the present example, the covariance matrix stays the same with or without mixture and misregistration. It turns out now that a mixed-class pixel (such as x in Figure A-12) due to misregistration will have an increasing probability of being assigned to a class completely different from the ones present in the pixel (such class would be Class B in Figure A-12). Class B, therefore, will be assigned, by classification error, more pixels than it would usually get if the configuration of the various cover types were different. For the case shown in Figure A-12, the probability of a pixel with mixture proportion δ and relative misregistration ε of being assigned to Class B is (assuming $\sigma = 1$)

$$P_B = Q(m_1)\ [1 - Q(m_2)]$$

Figure A-13 shows P_B for different values of the mixture proportion δ and when $d = 2$. It is apparent from the figure that the P_B increases steadily as the amount of the misregistration increases. This behavior is intrinsically related to the distribution of the cover types on a particular scene and the nature of their statistics. If, for example, the mixture would have been between Classes A and D, the misregistration would have had no effect whatsoever on P_B. Therefore, given a scene, the classification error for certain classes is going to be differently affected by the misregistration. The conclusion that misregistration effects might result in errors that are noncompensating over a scene was also reported in the simulation study formed in [4].

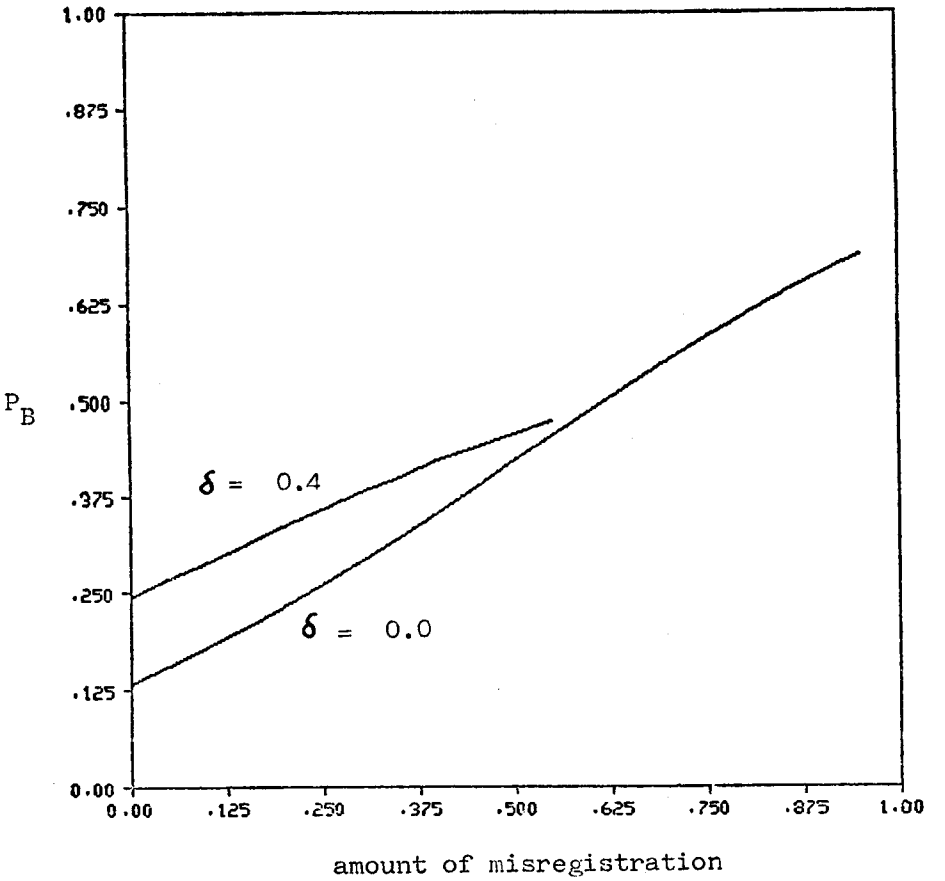


Figure A-13. P_B . Mixed-class pixels, d=2.0.

2.5 Contamination of the Training Statistics Due to Misregistration

One important aspect in the design of the classifier processors is the nature of the training data. It is usually desired that these data be accurately representative of the various classes present in the scene. To have this, one would like to have training-field sizes with the sufficient number of pixels to ensure a good estimation of the classifier parameters, e.g., the mean vectors and covariance matrices. Even if this condition is met in a particular analysis, when a certain amount of spatial misregistration is present, a number of pixels of the training set would have their statistics changed due to class mixture with neighboring fields. The number of these pixels which are the boundary or close to the boundary pixels will depend on the relative sizes and shapes of the pixels and the fields. In many practical applications, it has been found that this number is in general significant, i.e., it can account for 20-40% of the total number of training pixels.

When the training statistics are now being computed from spatial misregistered training fields, the statistics of the boundary pixels, now altered, will contaminate the class statistics. The severity of this contamination depends on the amount of the misregistration and the relative number of boundary pixels involved. To see these effects more clearly, let the following model be assumed. There are two channels and the statistics of Class A are being computed. A certain amount of misregistration occurs in Channel 1 and because of this, some of the pixels from the training data of Class A become mixed-class pixels. Let Class B be the contaminating class. The statistics of the field-center pixels from Class A are $N(\bar{a},K_a)$ and the statistics of the boundary pixels are $N(\bar{m}_x\ K_x)$. $\bar{m}_x$ and K_x will depend on the statistics of Class A and Class B ($N(\bar{b},K_b)$) and the amount of the spatial misregistration. If the proportion of boundary pixels to the total number of training pixels is δ, the statistics of Class A when misregistered will be:

Probability distribution

$$P(x) = (1-\delta)\ N(\bar{a},\ K_a) + N(\bar{m}_x,\ K_x)$$

mean

$$\bar{a}_{MR} = (1-\delta)\ \bar{a} + \delta\ \bar{m}_x$$

covariance

$$K_{a_{MR}} = (1-\delta)\ K_a + \delta K_x + (1-\delta)\ \bar{a}\bar{a}^T + \bar{m}_x\ \bar{m}_x^T - (1-\delta)^2\ \bar{a}\bar{a}^T - 1-\delta))\ \delta\ \bar{a}\ \bar{m}_x^T - (1-\delta)\ \delta\ \bar{m}_x\ \bar{a}^T - \delta^2\ \bar{m}_x\ \bar{m}_x^T$$

$\bar{m}_x$ and K_x can be computed as in previous sections as a function of the relative amount of the misregistration ε. Figure A-14 illustrates an example when the statistics of A and B are

$$\bar{a} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \qquad \bar{b} = \begin{bmatrix} 8 \\ 0 \end{bmatrix} \qquad K_a = K_b = \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix}$$

The values of the parameters δ and ε are 0.4 and 0.3, respectively. The common classifier processors will assume that all distributions are bivariate normal, doing the same here. The ellipses shown for Figure A-14 contain 90% of the total normal mass for each case. It is easy to observe the change in distribution for Class A due to misregistration.

It is also apparent that the performance of the classifier when using the altered statistics will be worse than when no misregistration occurs. This effect has been observed in an analysis performed at LARS where aircraft data have been misregistered in order to study and evaluate the Thematic Mapper specifications[2]. In this analysis, two sets of two bands each were registered with respect to each other. The first set contains a visible and a near infrared band. The second set has a middle and a thermal band. Tables A-4(a,b) show the statistics of a corn class when no misregistration is present and when a relative misregistration of $\varepsilon = 0.3$ (along the scanning track) is set between the two pairs of channels, i.e., Channels 3 and 4 are shifted 0.3 pixel with respect to Channels 1 and 2. The changes in the corresponding elements of the mean vector and covariance matrix are shown in Table A-4(b). Figure A-15 displays this change in the statistics when considering only Channels 1 and 3. As before, the ellipses contain 90% of the total normal mass. The continuous line corresponds to no misregistration and the dashed line corresponds to misregistration as in Table A-4(a).

It is interesting to observe that even for this relative amount of misregistration, the statistics of a particular class can be altered significantly. This usually happens when the amount of training data is not big enough and the training fields available have to be used in most of its extension, making them vulnerable to contamination from neighboring fields due to misregistration.

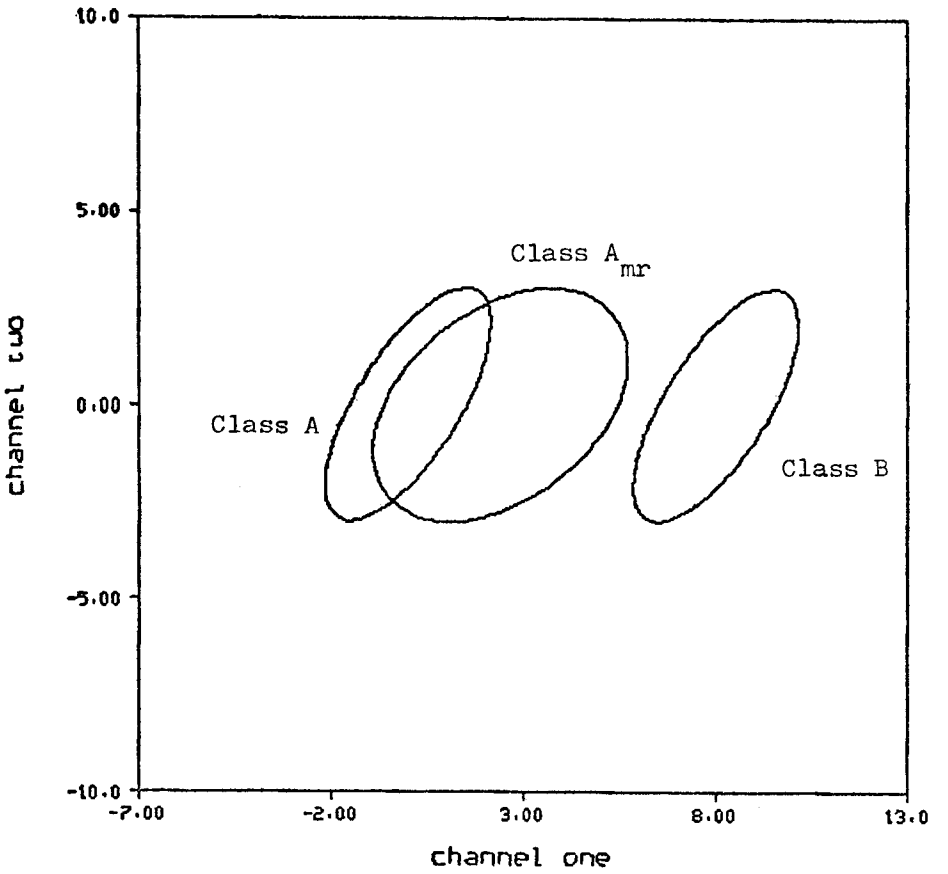


Figure A-14. Statistics change due to misregistration.

Table A-4(a). Statistics of corn class. No misregistration present.

Channel	Mean	Covariance			
		1	2	3	4
1	46.50	2.09	0.90	1.18	-14.77
2	76.83	0.90	8.15	-0.12	12.71
3	52.83	1.18	-0.12	1.96	-13.74
4	59.41	-14.77	12.71	-13.74	246.08

Table A-4(b). Statistics of corn class. Misregistration $\varepsilon = 0.3$ of Channels 3 and 4 with respect to Channels 1 and 2.

Channel	Mean	Covariance			
		1	2	3	4
1	46.50	2.09	0.90	1.90	-15.77
2	76.83	0.90	8.15	-1.00	12.37
3	54.00	1.90	-1.00	4.00	-13.18
4	60.58	-15.77	12.37	-13.18	270.99

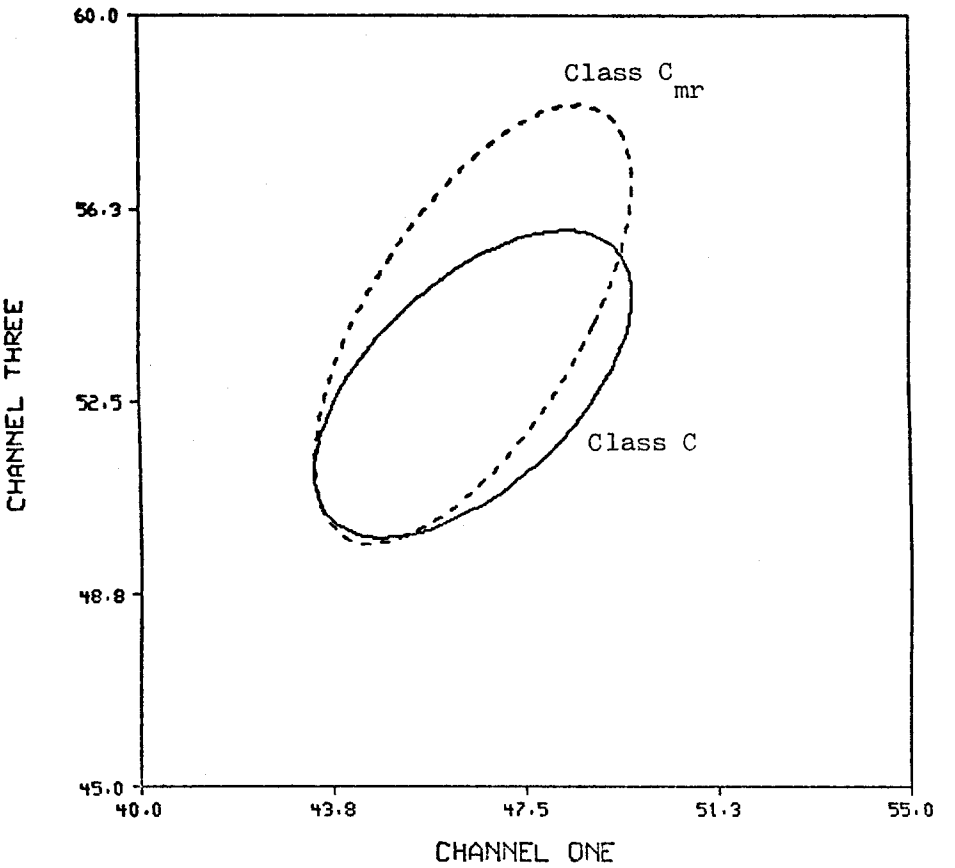


Figure A-15. Statistics change due to misregistration.

2.6 Elements of a General Model for Estimating the Effects of Misregistration

2.6.1 Amount and Direction of the Misregistration

The influence of these factors in the classification error, when the data have a spatial misregistration between channels, is apparent from the equations derived in the previous sections and displayed in Figures A-10 - A-12. That the amount of misregistration is an important factor is intuitively obvious. On the other hand, the effect of the direction of the misregistration is more complex to assess due to the various elements involved. Among them are the relative direction to the scanning track of the sensor and the preprocessing of the collected data before being put in final digital form. In the latter, usually some kind of averaging process is involved and the way this is done is intrinsically related to the spatial direction. Finally, as it has been seen in Section 2.5, the contamination in the training statistics depends on the nature of the contaminating neighboring field. The kind and amount of this contamination will obviously depend on the amount and direction of the spatial misregistration. Experimental studies made at LARS[3] have shown significantly different results when the direction of the misregistration changes even though the amount of it remains the same. Thus, for example, the classifications performance varies when the misregistration occurs along the scanning track, but in one case, let us say, in the positive direction and in the other in the negative direction. Therefore the nature of the fields that contaminate the training statistics is different for both cases.

2.6.2 Relative Sizes of the Fields and the Pixels

The number of boundary pixels is directly dependent on the relative sizes between fields and pixels. Estimation of the expected proportion of boundary pixels and its increase due to misregistration can be computed by using the expressions developed in [1]. The effect of the number of boundary pixels on the classification performance is important since the statistics of these pixels is the most altered when spatial misregistration occurs. As it was explained in Sections 2.3 and 2.4, these border pixels almost invariably are or would become mixed-class pixels whose statistics will not be represented in the training statistics set, the latter coming usually from field-center pixels. This situation will lead to the classification of these pixels with higher probability of error and, as explained in Section 2.4, in a noncompensating fashion. Application of the expressions in [1] to compute the proportion of boundary pixels in a scene used in the experimental study of the Thematic Mapper performed at LARS shows that 43% of the total number of pixels are boundary pixels. When a spatial misregistration occurs along the scanning track direction by the amount $\varepsilon = 0.3$, the

increase in the number of boundary pixels is estimated to be around 20%, i.e., in this case around 50% of the total number of pixels will be boundary pixels. As expected, it was found in that experimental study that the classification performance sharply decreases even when the amount of the spatial misregistration is small.

2.6.3 Statistics of the Various Classes

So far, for the discussions and quantitative analyses performed in the previous sections, it has been assumed that the different classes involved in the analysis have equal covariance matrices. This was done only for the sake of simplicity. A more realistic analysis has to consider the fact that in most real cases, the various classes have different covariance matrices. If in addition to this fact the number of channels is more than just two, the theoretical analysis of the classifier performance even with the assumption of multivariate Gaussian distributions becomes much more complex. Although it might be possible to obtain bounds for the various probabilities of error, it will still require a sizable increase in the number of computations required. A more viable way of tackling this theoretical problem is yet to be obtained.

References

1. Malila, W.A., J.M. Gleason and R.C. Cicone, Investigation of Spatial Misregistration Effects in Multispectral Data. NASA CR-ERIM 109600-68-F. Environmental Research Institute of Michigan, Ann Arbor, MI, May 1976.

2. Svedlow, Martin, C.D. McGillem and P.E. Anuta, Analytical and Experimental Design and Analysis of an Optimal Processor for Image Restoration. LARS Information Note 090776. Laboratory for Applications of Remote Sensing (LARS) Purdue University, West Lafayette, IN 47907, Sept. 1976.

3. Swain, P.H., A Quantitative Applications-Oriented Evaluation of Thematic Mapper Design Specifications. LARS Grant Report 121680, Laboratory for Applications of Remote Sensing (LARS), Purdue University, West Lafayette, IN 47907, Dec. 1980.

4. Cicone, R.C., W.A. Malila, J.M. Gleason and R.F. Nalepka, Effects of Misregistration on Multispectral Recognition. Proc. LARS Symp. on Machine Processing of Remotely Sensed Data, Purdue University, West Lafayette, IN, June 29-July 1, 1976.

APPENDIX

Derivation of the Variance of the Registration Error Estimate

The basic assumption made in this derivation is that in the absence of noise, the output of the processor will be a maximum at the correct translation. No assumptions about the probability distribution of the noise are needed. The signal corresponding to the image to be overlaid is modeled as having two components, the desired signal and additive noise. This signal is passed through a filter and the position where the maximum of the output signal occurs is taken to be the correct registration position. However, since the filter is designed to yield a maximum at the correct delay only in the noise-free case, this observed registration position may differ from the true registration location. The discrepancy between these two positions is the registration error.

First consider the parameters involved.

$f(x,y)$	signal;
$m(x,y)$	additive noise;
$f(x,y) + m(x,y)$	data set to be registered;
$h(x,y)$	filter impulse response;
$g(x,y)$	$f(x,y) * h(x,y)$ = output signal in the absence of noise;
$n(x,y)$	$m(x,y) * h(x,y)$ = output due to the noise input;
$z(x,y)$	$g(x,y) + n(x,y)$ = composite output signal used to estimate the correct registration position;
$(\tilde{x},\tilde{y})$	true registration position;
$(\hat{x},\hat{y})$	estimated registration position.

The derivation proceeds as follows. First expand $g(x,y)$ in a second order Taylor series about $(\tilde{x},\tilde{y})$.

$$\begin{aligned} g(x,y) \simeq\ & g(\tilde{x},\tilde{y}) + g_x(\tilde{x},\tilde{y})[x-\tilde{x}] + g_y(\tilde{x},\tilde{y})[y-\tilde{y}] \\ & + g_{xy}(\tilde{x},\tilde{y})[x-\tilde{x}][y-\tilde{y}] + \tfrac{1}{2}\, g_{xx}(\tilde{x},\tilde{y})[x-\tilde{x}]^2 \\ & + \tfrac{1}{2}\, g_{yy}(\tilde{x},\tilde{y})[y-\tilde{y}]^2 \end{aligned} \qquad \text{I-1}$$

where the subscripts denote the partial derivatives with respect to the corresponding variables,

$$g_x(\tilde{x},\tilde{y}) = \left.\frac{\partial g(x,y)}{\partial x}\right|_{x=\tilde{x},\ y=\tilde{y}}$$

This subscript notation is used for the remainder of this section. Assume that $(x-\tilde{x})$ and $(y-\tilde{y})$ are small enough so that all higher order terms may be neglected.

Note that a necessary condition for a maximum is

$$\frac{\partial g(\tilde{x},\tilde{y})}{\partial x} = 0 = \frac{\partial g(\tilde{x},\tilde{y})}{\partial y} .$$

Substitute this result into the equation for $z(x,y)$.

$$\begin{aligned} z(x,y) &= g(\tilde{x},\tilde{y}) + g_{xy}(\tilde{x},\tilde{y})[x-\tilde{x}][y-\tilde{y}] \\ &\quad + \tfrac{1}{2}\, g_{xx}(\tilde{x},\tilde{y})[x-\tilde{x}]^2 \\ &\quad + \tfrac{1}{2}\, g_{yy}(\tilde{x},\tilde{y})[y-\tilde{y}]^2 + n(x,y). \end{aligned} \qquad \text{I-2}$$

Again use the necessary condition for an observed maximum,

$$\partial z(\hat{x},\hat{y})/\partial x = 0 = \partial z(\hat{x},\hat{y})/\partial y,$$

$$\begin{aligned} z_x(\hat{x},\hat{y}) &= 0 = g_{xy}(\tilde{x},\tilde{y})[\hat{y}-\tilde{y}] \\ &\quad + g_{xx}(\tilde{x},\tilde{y})[\hat{x}-\tilde{x}] + n_x(\hat{x},\hat{y}) \end{aligned} \qquad \text{I-3}$$

$$\begin{aligned} z_y(\hat{x},\hat{y}) &= 0 = g_{xy}(\tilde{x},\tilde{y})[\hat{x}-\tilde{x}] \\ &\quad + g_{yy}(\tilde{x},\tilde{y})[\hat{y}-\tilde{y}] + n_y(\hat{x},\hat{y}). \end{aligned} \qquad \text{I-4}$$

Arrange these equations in terms of $(\hat{x}-\tilde{x})$ and $(\hat{y}-\tilde{y})$, the error in the registration.

$$(\hat{x}-\tilde{x}) = \frac{g_{xy}n_y - g_{yy}n_x}{g_{xx}g_{yy} - g_{xy}^2} \qquad \text{I-5}$$

$$(\hat{y}-\tilde{y}) = \frac{g_{xy}n_x - g_{xx}n_y}{g_{xx}g_{yy} - g_{xy}^2} \qquad \text{I-6}$$

where the arguments $(\hat{x},\hat{y})$ and $(\tilde{x},\tilde{y})$ have been left out for notational convenience.

One can now find the variance of the error by taking the expectation of $(\hat{x}-\tilde{x})^2$ and $(\hat{y}-\tilde{y})^2$, where it is assumed that $E[\hat{x}-\tilde{x}] = 0 = E[\hat{y}-\tilde{y}]$.

$$\mathrm{Var}[\hat{x}-\tilde{x}] = E[(\hat{x}-\tilde{x})^2] = \overline{(\hat{x}-\tilde{x})^2} \qquad \text{I-7}$$

$$\mathrm{Var}[\hat{y}-\tilde{y}] = E[(\hat{y}-\tilde{y})^2] = \overline{(\hat{y}-\tilde{y})^2} \qquad \text{I-8}$$

$$\overline{(\hat{x}-\tilde{x})^2} = \frac{g_{xy}^2\,\overline{n_y^2} - 2g_{xy}g_{yy}\,\overline{n_y n_x} + g_{yy}^2\,\overline{n_x^2}}{[g_{xx}g_{yy} - g_{xy}^2]^2} \qquad \text{I-9}$$

$$\overline{(\hat{y}-\tilde{y})^2} = \frac{g_{xy}^2\,\overline{n_x^2} - 2g_{xy}g_{xx}\,\overline{n_y n_x} + g_{xx}^2\,\overline{n_y^2}}{[g_{xx}g_{yy} - g_{xy}^2]^2} \qquad \text{I-10}$$

One may use these equations to calculate the variance of the error, but in doing so, it is found that a filter function must be specified first. This is intrinsic in the parameters in these equations, which is seen more clearly if one writes these terms as a function of the filter (wide sense stationarity is assumed).

$$\overline{n_y^2(\hat{x},\hat{y})} = \int\int\int\int h_y(\hat{x}-\alpha,\hat{y}-\beta)h_y(\hat{x}-\gamma,\hat{y}-\lambda) \cdot R_m(\alpha-\gamma,\beta-\lambda)\,d\alpha\,d\beta\,d\gamma\,d\lambda \qquad \text{I-11}$$

$$\overline{n_y(\hat{x},\hat{y})n_x(\hat{x},\hat{y})} = \int\int\int\int h_y(\hat{x}-\alpha,\hat{y}-\beta) \cdot h_x(\hat{x}-\gamma,\hat{y}-\lambda)R_m(\alpha-\gamma,\beta-\lambda) \cdot d\alpha\,d\beta\,d\gamma\,d\lambda \qquad \text{I-12}$$

$$\overline{n_x^2(\hat{x},\hat{y})} = \int\int\int\int h_x(\hat{x}-\alpha,\hat{y}-\beta)h_x(\hat{x}-\gamma,\hat{y}-\lambda) \cdot R_m(a-\gamma,\beta-\lambda)\,d\alpha\,d\beta\,d\gamma\,d\lambda \qquad \text{I-13}$$

$$g_{xx}(\tilde{x},\tilde{y}) = \int\int h_{xx}(\tilde{x}-\alpha,\tilde{y}-\beta)\,f(\alpha,\beta)\,d\alpha\,d\beta \qquad \text{I-14}$$

$$g_{yy}(\tilde{x},\tilde{y}) = \int\int h_{yy}(\tilde{x}-\alpha,\tilde{y}-\beta)\,f(\alpha,\beta)\,d\alpha\,d\beta \qquad \text{I-15}$$

$$g_{xy}(\tilde{x},\tilde{y}) = \int\int h_{xy}(\tilde{x}-\alpha,\tilde{y}-\beta)\,f(\alpha,\beta)\,d\alpha\,d\beta \qquad \text{I-16}$$

where $R_m(\alpha-\gamma,\beta-\lambda) = \overline{m(\alpha,\beta)m(\gamma,\lambda)}$ I-17

One now has an expression for determining the registration error variance. Equations I-9 and I-10 will allow one to find the variance of the error for any filter function; however, they seem to bear little resemblance to the results in the first section.* To obtain a particular solution, a specific filter function must be chosen. The one that has been picked is intuitively pleasing in two ways: It is an optimum type filter in that it maximizes the signal-to-noise ratio; and it yields an answer in terms of the signal bandwidth and signal-to-noise ratio. It can be shown that this filter minimizes the error variance. This filter is the so-called "matched filter."

Let
$$H(u,v) = \frac{F^*(u,v)\ \exp\ (-j2\pi(\tilde{x}u + \tilde{y}v))}{S_m(u,v)} \qquad \text{I-18}$$

$S_m(u,v)$ Fourier transform of $R_m(x,y)$;

$F(u,v)$ Fourier transform of $f(x,y)$;

$H(u,v)$ Fourier transform of $h(x,y)$.

Substituting this filter function into equations I-9 and I-10, the results simplify to,

$$\overline{(\hat{x}-\tilde{x})^2} = \left[\frac{g_{xy}^2}{g_{yy}} - g_{xx}\right]^{-1} \qquad \text{I-19}$$

$$\overline{(\hat{y}-\tilde{y})^2} = \left[\frac{g_{xy}^2}{g_{xx}} - g_{yy}\right]^{-1} \qquad \text{I-20}$$

This simplification is seen more easily if one first converts equations I-11 through I-16 to the frequency domain and then inserts the matched filter.

One obtains the final result by converting these last two equations to the frequency domain. They then become,

$$\overline{(\hat{x}-\tilde{x})^2} = \left[-\frac{g_{xy}^2}{B_y^2 SNR} + B_x^2\ SNR\right]^{-1} \qquad \text{I-21}$$

$$\overline{(\hat{y}-\tilde{y})^2} = \left[-\frac{g_{xy}^2}{B_x^2 SNR} + B_y^2\ SNR\right]^{-1} \qquad \text{I-22}$$

* Reference [2], pages 18-24.

where

$$B_x = \left[\frac{4\pi^2 \iint u^2 \frac{|F(u,v)|^2}{S_m(u,v)} \, du \, dv}{\iint \frac{|F(u,v)|^2}{S_m(u,v)} \, du \, dv} \right]^{1/2} \qquad \text{I-23}$$

B_x effective bandwidth of input signal in the x-axis direction;

$$B_y = \left[\frac{4\pi^2 \iint v^2 \frac{|F(u,v)|^2}{S_m(u,v)} \, du \, dv}{\iint \frac{|F(u,v)|^2}{S_m(u,v)} \, du \, dv} \right]^{1/2} \qquad \text{I-24}$$

B_y effective bandwidth of input signal in the y-axis direction;

$$SNR = \iint \frac{|F(u,v)|^2}{S_m(u,v)} \, du \, dv \qquad \text{I-25}$$

SNR = output signal-to-noise ratio.

It is seen that the variance of the error is again expressible in terms of the effective signal bandwidth and signal-to-noise ratio. These results are similar to those obtained in the first section,* but the relationships are not quite as simple.

A further simplification can be obtained by making some additional assumptions. The error variance expressions then will be the same as in the first method. These assumptions concern the term $g_{xy}(\hat{x},\hat{y})$ in equations I-21 and I-22. If this term equals zero, then the desired result is obtained. Such a condition involves the quantity $[|F(u,v)|^2]/[S_m(u,v)]$ since $g_{xy}(\hat{x},\hat{y})$ is a function of this quantity. Let $K(u,v) = [|F(u,v)|^2]/[S_m(u,v)]$ for notational convenience. Since $K(u,v)$ is an even function of u and v, in order for $g_{xy}(\hat{x},\hat{y})$ to equal zero it is suffficient that,

$$K(u,v) = K(-u,v) \qquad \text{I-26}$$

or necessary and sufficient that,

$$\int_0^\infty \int_0^\infty uv \, K(u,v) \, du \, dv = \int_0^\infty \int_0^\infty uv \, K(-u,v) \, du \, dv. \qquad \text{I-27}$$

* Reference [2], pages 18-24.

The expressions then become

$$\overline{(\hat{x}-\tilde{x})^2} = \frac{1}{B_x^2 SNR} \qquad \text{I-28}$$

$$\overline{(\hat{y}-\tilde{y})^2} = \frac{1}{B_y^2 SNR} \qquad \text{I-29}$$

An example of when these last assumptions might apply is the following situation. Let F(u,v) and S (u,v) be bandlimited to W_x and W_y in the respective axis directions. And let $[|F(u,v)|^2]/[S_m(u,v)]$ equal a constant. This would occur when the noise spectrum has a shape similar to the signal spectrum. In this case, it might be advantageous to model the two spectra as differing only by a constant factor for simplicity in estimating the variance to be expected. This may be written,

$$\frac{|F(u,v)|^2}{S_m(u,v)} = c, \text{ a constant.} \qquad \text{I-30}$$

From Equation I-25

$$SNR = c \int_{-W_y}^{W_x} \int_{-W_y}^{W_y} du\, dv. \qquad \text{I-31}$$

So,

$$c = \frac{SNR}{4W_x W_y}. \qquad \text{I-32}$$

Then from Equations I-23, I-24 and I-25,

$$B_x^2\, SNR = 4\pi^2\, c \left[\frac{2W_x^3}{3}\right](2W_y) \qquad \text{I-33}$$

$$B_y^2\, SNR = 4\pi^2\, c(2W_x) \left[\frac{2W_y^3}{3}\right]. \qquad \text{I-34}$$

Substituting in the expressions for c, the variances are:

$$\overline{(\hat{x}-\tilde{x})^2} \quad \frac{3}{4\pi^2 W_x^2 SNR} \qquad \text{I-35}$$

$$\overline{(\hat{y}-\tilde{y})^2} = \frac{3}{4\pi^2 W_y^2 SNR} \qquad \text{I-36}$$

The respective standard deviations then are:

$$\text{Standard deviation of } (\hat{x}-\tilde{x}) = \frac{1}{2\pi W_x}\sqrt{\frac{3}{SNR}} \qquad \text{I-37}$$

$$\text{Standard deviation of } (\hat{y}-\tilde{y}) = \frac{1}{2\pi W_y}\sqrt{\frac{3}{SNR}} \qquad \text{I-38}$$

One may obtain a quantitative feel for the values of these expressions by using the sampling intervals for the Landsat-1 data in this example. The sampling interval is about 60 meters along the columns and about 80 meters along the lines. Substituting these values in equations I-37 and I-38, one finds that,

$$\text{Standard deviation of error along the lines} = \frac{44.1}{\sqrt{SNR}} \text{ meters} \qquad \text{I-39}$$

$$\text{Standard deviation of error along the columns} = \frac{33.1}{\sqrt{SNR}} \qquad \text{I-40}$$

A quantitative measure of the registration processor accuracy in terms of the variance of the error of the registration has been derived. With the appropriate assumptions, the variance is shown to be inversely proportional to the square of the effective bandwidth times the signal-to-noise ratio. The final expressions are presented again to emphasize both the form and simplicity of their representation.

$$\text{Var}\left[(\hat{x}-\tilde{x})\right] = \frac{1}{B_x^2 \text{SNR}}$$

$$\text{Var}\left[(\hat{y}-\tilde{y})\right] = \frac{1}{B_y^2 \text{SNR}}$$

This derivation should prove useful in several respects. First of all, it may be a basis for the analysis of different registration systems by providing a way to estimate the expected accuracy of the system. Secondly, it provides a straightforward way of estimating this error.

B. MULTISTAGE CLASSIFICATION

D. A. Landgrebe, M. J. Muasher, P. H. Swain

1. Introduction

A number of different types of classifiers are now in routine use in remote sensing. Most of these classification algorithms, using pattern recognition techniques, can be regarded as "single-stage" classifiers, where an "unknown" pattern is tested against all classes using one feature subset, and then the pattern is assigned to one of the present classes in a single-stage decision procedure.

In recent years, as classification of multispectral data has found a larger number of users and a wider range of applications, the need has been felt for alternate, more powerful techniques than the conventional classifiers, through the use of which more information could be extracted more accurately and/or efficiently from the scene. Some of the reasons that have warranted this need include:

1. The need to extract more detailed information from data. The opportunity to do so results from the emergence of more complex data sets. The growing use of multitype data bases containing Landsat data with a variety of other quantitative geodata together with the anticipated launching of more sophisticated sensors such as the Thematic Mapper result in the opportunity to extract considerably more information from the data.

2. The broadening of the range of applications. As pattern recognition methods have developed, they have found a larger number of users with a wider range of applications. The feedback from these different and versatile uses has indicated problems and needs not initially present.

3. The ever present need for improved classification accuracy. There are some applications for which conventional classifiers have proved to be marginal at best. Some of these are listed in Swain et al. [1] and include multi-image analysis and the use of mixed feature types.

4. The need for improved processing efficiency. The conventional, single-stage, classifiers use only one particular feature subset and are somewhat inefficient, as they must compare an

unknown pattern against all possible classes before assigning that pattern to a particular class.

Because of these and other factors, there has been some research in recent years directed towards developing multistage classifiers, whereby the decision procedures go through several stages before finally assigning a pattern to a class.

The purpose of this research is to develop a layered decision algorithm that can increase the accuracy and efficiency over the conventional single-stage classification approach. Developing such an algorithm requires, among other things, a careful look at some parameters that are crucial to any successful attempt at tackling such a complex problem. In particular, three areas have to be investigated:

1. The development of an adequate training procedure to define an initial set of spectral classes with their respective statistics;

2. The investigation of various error estimators and the development of an adequate performance estimator that can reasonably predict the accuracy or any trends in performance;

3. The development of an algorithm to build a binary tree making use of the above-mentioned methods.

Of these three areas, the most important problem is believed to be the development of an accurate error estimator, especially in the presence of what has come to be known as the Hughes phenomenon (elaborated upon later in the review of literature). Predicting the conditions under which the Hughes phenomenon occurs provides the key to the solution of the problem. Therefore, a considerable portion of the research has been directed towards trying to understand and predict the impact of this phenomenon.

2. Review of Literature

2.1 Training Procedure

Several training methods have been suggested in the literature. We will not attempt to list all of them, but rather will give a background of some of the methods reviewed and used in this work.

The training process is the procedure whereby labeled samples are selected and used to compute class statistics which in turn are used to classify unlabeled (i.e., "unknown") samples.

Several parameter estimation methods (training methods) have appeared in the literature. Sample-partitioning methods, the leaving-

one-out method, clustering are but a few. See, for example, Fukunaga [2] and Duda and Hart [3].

For remote sensing purposes, clustering has been widely used in developing training statistics. Two basic approaches have been: a supervised clustering approach, in which the analyst selects areas of known cover types and then the statistics for these areas are obtained with the aid of a computer; and the non-supervised clustering approach, in which the entire training area is subdivided into clusters by the clustering algorithm and each cluster is then identified by the analyst and given a specific label. The statistics of each cluster corresponding to a cover type or a subclass of a cover type are then calculated. Fleming et al. [4,5] investigated several clustering approaches and their effect on classification accuracy. Among the approaches they used were non-supervised clustering, supervised clustering, modified clustering, mono- (aggregate) cluster blocks, and multi- (class-conditional) cluster blocks.

2.2 Performance Estimators

A key factor in the design of a layered decision algorithm is the ability to predict how the algorithm will perform in terms of accuracy at every node. While optimizing the performance at every node does not necessarily produce a globally optimal tree, it is still a very important and useful step in the design.

Several performance (or error) estimators have appeared in the literature. Again, we will not attempt here to exhaustively list all the contributions made, but rather will give an idea of how the research in this area has progressed.

Performance estimators can be divided into two main categories:

Performance functions which have some sort of direct relationship with the probability of error. Examples are Parzen estimators (see [2]), the k-nearest neighbor error estimator (see [6]). More recently, Mobasseri et al. [7] published an error estimator that computes the minimum probability of error through use of a combined analytical and numerical integration over a sequence of simplifying transformations of the feature space. The results have been shown to be similar to those obtained by conventional techniques. However, the algorithm becomes computationally too inefficient to use as the number of classes and/or features increases. Moore, Whitsitt and Landgrebe [8] (see also Whitsitt and Landgrebe [9]) developed a stratified posterior estimator which, like Mobasseri's, depends only on a given set of statistics. This was later used by Wiersma [10] and both estimators (Mobasseri's and Whitsitt's) were compared in [11] and found to give similar results, with Whitsitt's algorithm being faster in some cases. The former procedure uses a "deterministic" grid to sample the feature space, while the latter uses an internally generated random data base and assigns the

feature vector to the appropriate class via the maximum a posteriori principle. Both procedures assume normal class conditional statistics.

Separability measures, most of which have only a subtle, indirect, and often unknown, relationship to the probability of error. Various separability measures have been in common use in remote sensing applications. Among these are: Divergence [12], Transformed Divergence [13], Jeffreys-Matusita distance [14,15], Bhattacharyya distance [16] and the Mahalanobis distance [17]. (See list in [22].)

Several works have been reported comparing different separability measures and their effects on performance. (See [9,13,18,19,53].)

There are two problems with most of the above distances applied to remote sensing applications: (1) ambiguity and (2) linearity in pairwise error. The term ambiguity implies here that there does not exist a one-to-one relationship between the value of the measure and the probability of error. Linearity means that for some distance measures, when they are averaged over all class pairs, one class pair with large separability value can outweigh the contribution of all the others. Whitsitt [9] developed a distance measure $D_{erf} = \text{erf}(\sqrt{2B})$ where B is the Bhattacharyya distance and erf(·) is the error function. He found that the resulting measure is less ambiguous and more linear than the measure B.

Another key factor in the process of error estimation is the choice of feature subsets. The problems here are twofold:

1. As the number of features becomes large, it becomes desirable to choose a subset of these features that can adequately predict the accuracy. This selection process also can become expensive if one must search through all possible combinations of the feature set. It is desirable, therefore, to have a priori knowledge of the importance of each feature in relation to the probability of error. The Karhunen-Loeve expansion (attributed to Karhunen [20], and Loeve [21]) in pattern recognition literature has historically been used as a feature selection technique. It has the advantage of producing uncorrelated features (in theory, but the features are actually approximately uncorrelated in a practical K-L transformation). In addition, it imposes an ordering on the features in terms of importance in a representation error sense. As a result, first feature is "likely" to be more important than the second in calculating the probability of error, and so on.

2. The probability of error is not necessarily monotonically decreasing as the number of features increases. This is due to a peculiar phenomenon that has come to be known as the Hughes phenomenon. Hughes [23] found that with a fixed and finite training pattern sample, recognition accuracy can first increase as the number of measurements on a pattern increases, but decay with measurement complexity higher than some optimum value. He also reported that for unlimited training data, this does not occur and the recognition accuracy reaches an optimum

only at infinite measurement dimensionality. According to Hughes, if insufficient sample data are available to estimate the pattern probabilities accurately, then a Bayes recognizer is not necessarily optimal. Many papers have since been published on this phenomenon, confirming it or trying to explain why it occurs (see [24-32]). Thus, it appears that a successful design should predict when and if such phenomena occur.

2.3 Multistage Classifiers

In recent years, some work has appeared in the literature aimed at developing multistage classification algorithms. There is much yet to be learned about such algorithms, and no work has been reported claiming optimality (or even close to optimality) of results.

In general, earlier work can be grouped into two main categories:

Sequential classification methods. These can be found in several papers and books (see, for example, [33-35]). Basically, the method consists of observations made on feature measurements, one at a time. After an observation is made, the classifier either reaches a final decision and the process is terminated, or it makes another observation until a final decision is reached.

Hierarchical classification methods. These are subdivided into two categories:

1. Hierarchical clustering methods. Examples of such work are found in Fukunaga [2], Dubes and Jain [36], who present a semi-tutorial review of the state of the art in cluster validity, and Lukasova [37]. In general, hierarchical clustering is designed to generate a classification tree. The "root" node of the tree represents a collection of samples (either a training data set or the entire sample set) and each terminal node represents either an individual sample or a group of samples belonging to some class within the set of classes in the data set. The method attempts to divide the set of samples in each node into disjoint subsets which form new nodes. Defined as such, the method is often nonparametric and depends heavily on the ability of the algorithm to find meaningful divisions of samples that correspond at terminal nodes with meaningful classes.

2. Decision trees and criterion functions. Most of the work done in multistage algorithms belongs to this category. Often, a decision tree is built using an optimization or criterion function that dictates the structure of the tree. It is this kind of approach that will be of greatest concern in this research.

Hierarchical methods differ from sequential methods in certain important respects. While in sequential schemes any class can be accepted at any stage of the measurement process, in hierarchical

schemes certain classes are excluded from consideration at each stage. Also, sequential methods impose a linear ordering on the features. In hierarchical methods, features used along one decision path can be different from those used along another path.

In 1971, Nadler [38] tried to calculate error rates in a hierarchical decision structure under assumptions of statistical independence among the members of the hierarchy. Even under such assumptions, the results assume "small" probabilities of errors at any level.

Several heuristic methods of constructing tree designs have been proposed in the literature. Some studies were done using optimization methods to automate the classifier design procedure, but the assumptions made were often too restrictive. Meisel and Michalopoulos [39] in 1973 presented a two-stage partitioning algorithm for the design of an optimal binary tree. In the first stage, a suboptimal sufficient partition is obtained. The second stage optimizes the result of the first stage through a dynamic programming approach. The method allows only for linear discriminant functions to partition the space, certainly a suboptimal and too restrictive condition.

In 1974, Wu et al. [40] reported on a decision tree approach with direct application to multispectral data analysis. Several design procedures were proposed (one of which is manual), with special emphasis on a heuristic, machine-implemented approach. The optimality criterion used is a weighted sum of computation cost and accuracy. Results were presented which showed superiority in efficiency (but infrequently in accuracy) over the conventional classifier. The criterion function used, as it cannot predict beforehand the structure of the tree below that node, assumes all the nodes below the node under consideration are terminal nodes, and hence is necessarily suboptimal. Later papers have appeared that have pointed to applications using this particular classifier [41,42].

In 1976, You and Fu [43] presented a linear binary tree classifier that uses linear discriminant functions at decision stages with an application to multispectral remotely sensed data. The procedure includes a grouping algorithm, a separability measure, and an error minimization procedure using the Fletcher-Powell algorithm [44]. Again, the procedure is certainly suboptimal because of the assumption of linearity. Results reported, though, show that this classifier is much faster and more accurate than the maximum likelihood classifier with the same number of features. This is due to the fact that the procedure uses different feature subsets (with a restriction on their number) at each node, compared with only one feature subset used in the one-stage maximum likelihood classifier.

Kulkarni and Kanal [45] used dynamic programming and branch-and-bound methodologies in the design of hierarchical classifiers. The criterion of optimality they used is a weighted sum of the probability of error and the average measurement cost incurred in classifying a random sample. The design assumes that the features used at the nodes are statistically independent and that the decision at each node is a function

of only that particular feature observation, the design using only one best feature at each tree node. Further, the design of the optimal tree assumes a very low error rate for the tree, a very restrictive assumption since in many cases a high error rate is specifically the reason why a layered classifier was selected, i.e., to improve the accuracy. Although the authors presented some methods to reduce the complexity of their design algorithms, the examples they used involve only a small number of classes and features.

In 1977, Parkih [46] compared several classification techniques of clouds, including hierarchical design. However, his paper offers no new insights or major results that would help improve the state of the art.

Also in 1977, Sethi and Chatterjee [47] developed an algorithm for the design of an efficient decision tree with application to pattern recognition problems involving discrete variables. A criterion function was defined to estimate the minimum expected cost of a tree in terms of the weights of its terminal nodes and costs of the measurements, which then was used to establish the search procedure for the efficient decision tree. The concept of prime events was used to obtain the number of nodes and the corresponding weights in the design sample. No optimality claim was made, but the procedure was found to lead to the optimal tree in most of the cases. The procedure uses only one feature at every node, and its applicability to remotely sensed multispectral data is very doubtful.

In 1978, Breiman [48] presented a procedure for building a binary classification tree. He used a criterion function that is only a function of the parent node and the two descendent nodes. He used one best feature at every node. He also reported on another regression algorithm developed at Survey Research Center, University of Michigan [49], in which the criterion function tries to reduce the variances of the two descendent nodes as much as possible from the variance of the parent node.

Rounds [50] in 1979 developed a binary decision tree algorithm, but again one feature is selected at every node. The approach is a nonparametric one, based on the Kolmogorov-Smirnov criterion.

3. Hughes Phenomenon: Work Accomplished

As mentioned earlier, a considerable portion of this research is directed towards understanding the Hughes phenomenon. Figure B-1 illustrates the phenomenon conceptually. In the presence of a limited training sample size, the mean recognition accuracy as a function of the measurement complexity (number of features for our purposes) exhibits a peaking effect. Contrary to intuition, the mean accuracy does not always increase with additional measurements. Further, peaking of the curve shifts up and to the right as the number of samples increases, disappearing in the case of an infinite number of training samples (complete knowledge of the underlying distributions).

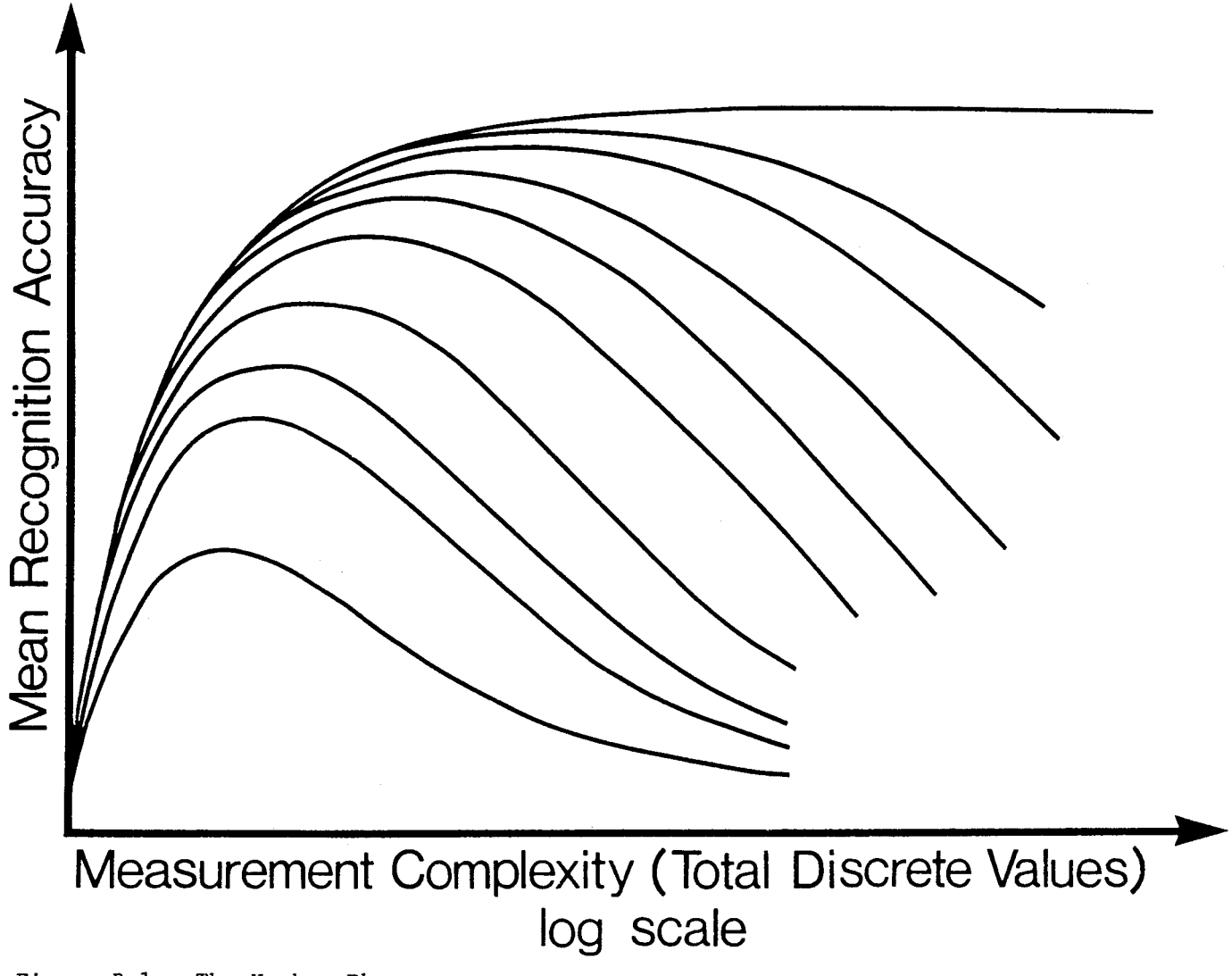


Figure B-1. The Hughes Phenomenon.

Figure B-2 suggests one possible explanation for this phenomenon. Figure B-2a shows class separability plotted vs. dimensionality. As dimensionality increases, so does class separability (a nondecreasing function of dimensionality) until it saturates, and any further increase in dimensionality does not have a significant effect on class separability. But this is not the only effect on the mean accuracy. With the presence of a fixed, limited training sample size, any increase in dimensionality necessarily results in a degradation in the accuracy of statistics estimation of the class distributions. Thus, conceptually, one should expect a curve similar to that of Figure B-2b.. Further, as the number of samples increases, the curve should shift to the right, i.e., for any given dimensionality, the larger sample size should provide a better estimate of the true distributions. Assuming these two effects are the dominant effects on accuracy, adding the two effects results in Figure B-2c, a curve similar to Figure B-1. Based upon this concept of the phenomenon, the solution to the problem lies in being able to predict quantitatively how the number of samples present affects the accuracy of the estimated statistics .

3.1 Simultaneous Diagonalization: Introduction

The key problem to solve, then, is the development of an error estimator that accurately predicts the Hughes phenomenon. Working with multiple features, several properties are desired in these features which will make further analysis easier:

Uncoupled (Independent) Features. Uncoupling of features from one another simplifies analysis a great deal as it permits evaluating the effect of each feature separately from other features.

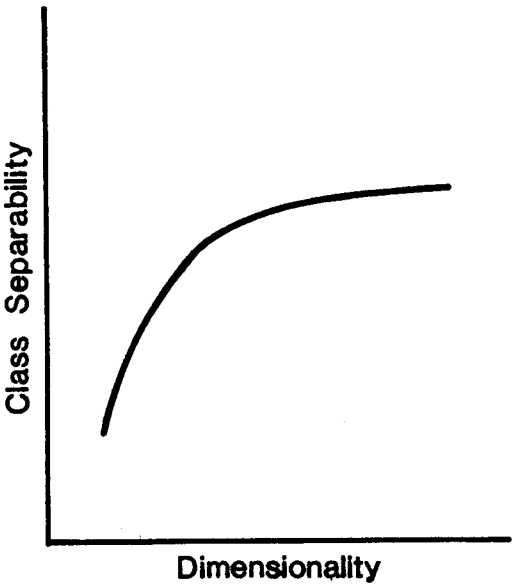
Ordered Features. If the features can be ordered, or at least approximately so, in terms of their effect on the probability of error, then the process of feature selection would be made easier.

Optimal Separability. The features should be optimal with respect to the probability of error for two distributions at hand. Putting it in different words, the feature subset should be tailored to the separability of the two distributions.

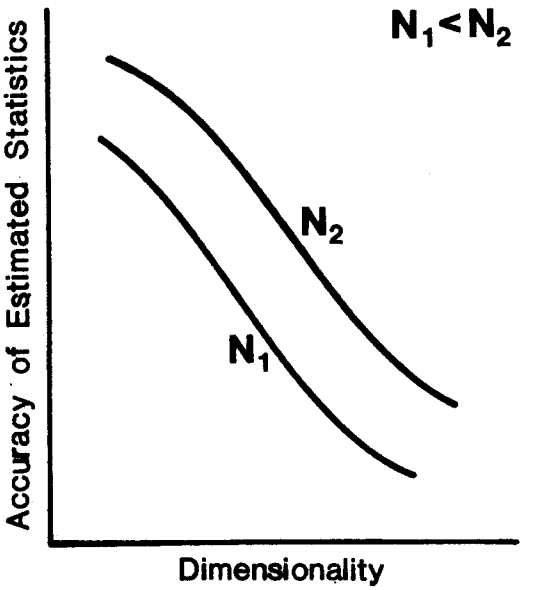
To this end, a technique known as a "simultaneous diagonalization" [51,52] is discussed in the next section.

3.2 Simultaneous Diagonalization: Theory

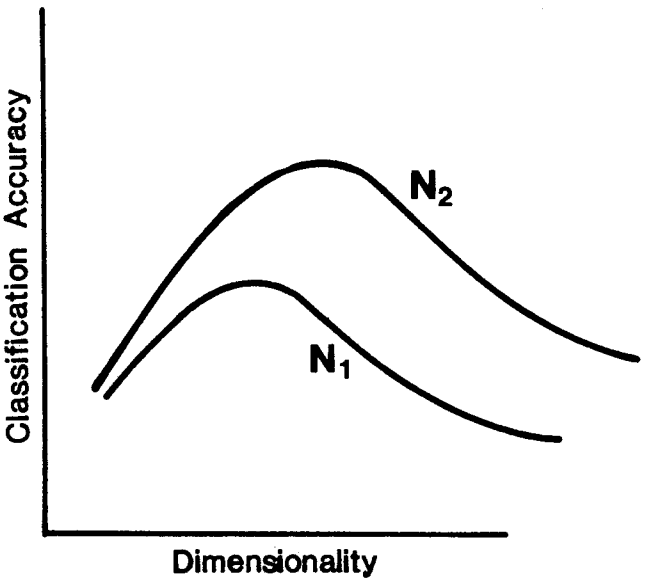
Let $\hat{\Sigma}_1$ and $\hat{\Sigma}_2$ be the estimated covariance matrices for classes 1 and 2, respectively. We seek a transformation matrix A such that



B-2a.



B-2b.



B-2c.

Figure B-2. Explanation of the Hughes phenomenon.

$$A\hat{\Sigma}_1 A^T = I \qquad A\hat{\Sigma}_2 A^T = \Lambda$$

where I is the identity matrix and Λ is a diagonal matrix.

This transformation would uncouple the features, while not affecting the probability of error because the latter is invariant under linear transformations. We proceed to find such a transformation as follows. (For more details, see [2], pp. 31-35.)

Let θ and Φ be the eigenvalue and eigenvector matrices of $\hat{\Sigma}_1$, respectively; then

$$\theta^{-\frac{1}{2}}\Phi^T\hat{\Sigma}_1\Phi\theta^{-\frac{1}{2}} = I \text{ ..... (1)} \qquad (\Phi^T\hat{\Sigma}_1\Phi = \theta)$$

$$\theta^{-\frac{1}{2}}\Phi^T\hat{\Sigma}_2\Phi\theta^{-\frac{1}{2}} = K \text{ ..... (2)} \qquad K \text{ is a general matrix}$$

Next, we desire to diagonalize K. To find eigenvalues of K, it is necessary to solve the equation

$$|K - \lambda I| = 0 \text{ ..... (3)}$$

Replacing K and I in (3) by (1) and (2), we get

$$|\theta^{-\frac{1}{2}}\Phi^T\hat{\Sigma}_2\Phi\theta^{-\frac{1}{2}} - \lambda\theta^{-\frac{1}{2}}\Phi^T\hat{\Sigma}_1\Phi\theta^{-\frac{1}{2}}| = 0$$

Or

$$|\theta^{-\frac{1}{2}}\Phi^T|\ |\hat{\Sigma}_2 - \lambda\hat{\Sigma}_1|\ |\Phi\theta^{-\frac{1}{2}}| = 0$$

Since $\theta^{-\frac{1}{2}}\Phi^T$ is nonsingular, it follows that

$$|\hat{\Sigma}_2 - \lambda\hat{\Sigma}_1| = 0$$

or

$$|\hat{\Sigma}_1^{-1}\hat{\Sigma}_2 - \lambda I| = 0$$

So, only the eigenvalue and eigenvector matrices of $\hat{\Sigma}_1^{-1}\hat{\Sigma}_2$ need be calculated.

The eigenvalue matrix is then Λ, and the transpose of the eigenvector matrix, A^T, serves as the transformation matrix.

The idea behind simultaneous diagonalization is to transform the original features into a new space where the features are independent and then choose a subset of these features in the new space which is optimal with respect to the probability of error. This is illustrated in Figure B-3.

4. Performance Estimator: Approximation to the Probability of Error

4.1 The Likelihood Function

One of the most serious difficulties facing researchers in trying to estimate the probability of error in multidimensional analysis is the need to carry out a multiple integration on the multivariate probability density function. It would be much easier if one could work with a function that is one-dimensional and carries all the information present. Fortunately, such a function does exist and is given by

$$h(x) = -\ln(p(x/\omega_1)/p(x/\omega_2))$$

Since the assumption of class-conditional normal distributions is valid in this case, h(x) becomes:

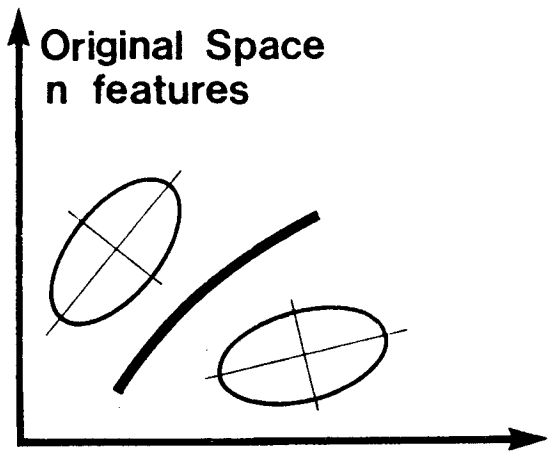
$$h(X) = \tfrac{1}{2}(X-\hat{M}_1)^T\hat{\Sigma}_1^{-1}(X-\hat{M}_1) - \tfrac{1}{2}(X-\hat{M}_2)^T\hat{\Sigma}_2^{-1}(X-\hat{M}_2) + \tfrac{1}{2}\ln\frac{|\hat{\Sigma}_1|}{|\hat{\Sigma}_2|} - \ln\frac{\hat{p}(\omega_1)}{\hat{p}(\omega_2)}$$

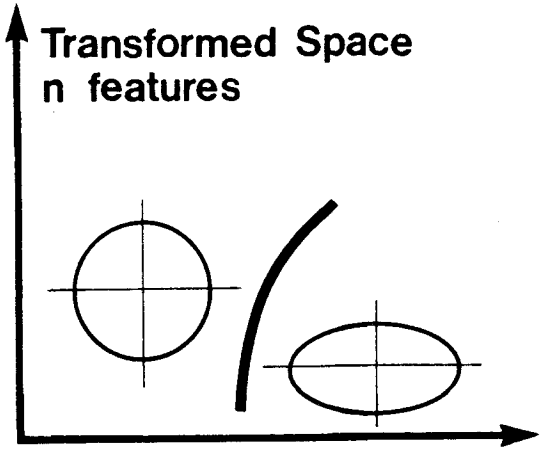
where $\hat{M}_i$ is the class-conditional mean of class i and $\hat{\Sigma}_i$ is the covariance matrix. Note that h(x) is a scalar random variable, regardless of the dimensionality of x.

The problem then is to know, or to estimate, the probability density function of h(x). Once that is known, the probability of error can be obtained by carrying out a scalar integration.

4.2 Feature Selection

Before proceeding to discuss the approximation algorithms to estimate the probability of error, we digress briefly to discuss how the features are ordered.





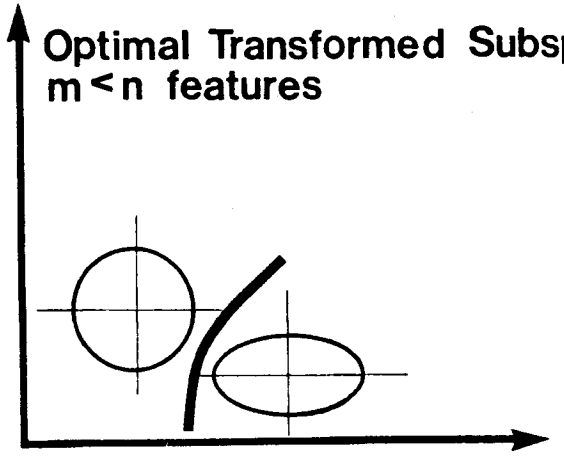


Figure B-3. Delineation of optimal subspace by simultaneous diagonalization.

The literature offers many studies made on comparing different separability measures and their effectiveness in choosing the best feature subset (see [9,13,18,53,54]). It appears that the Bhattacharyya distance is one of the most suitable separability measures for distinguishing between classes. Thus, it will be used as a basis for feature selection. The fact that the features are independent allows us to determine the effect of each feature on the probability of error separately.

The Bhattacharyya distance for two normal distributions can be expressed as follows:

$$B = \frac{1}{8}(\hat{M}_1-\hat{M}_2)^T\left(\frac{\hat{\Sigma}_1+\hat{\Sigma}_2}{2}\right)^{-1}(\hat{M}_1-\hat{M}_2) + \frac{1}{2}\ln\frac{|\frac{1}{2}(\hat{\Sigma}_1+\hat{\Sigma}_2)|}{|\hat{\Sigma}_1|^{\frac{1}{2}}|\hat{\Sigma}_2|^{\frac{1}{2}}}$$

After the simultaneous diagonalization transformation, however, B can be expressed as:

$$B = \sum_{i=1}^{n}\left[\frac{1}{4}\frac{(d_{1i}-d_{2i})^2}{\lambda_i+1} + \frac{1}{2}\ln\left(\frac{1}{2}\left(\frac{1}{\lambda_i^{\frac{1}{2}}}+\lambda_i^{\frac{1}{2}}\right)\right)\right]$$

where d_{ij} is the jth element of the transformed class-conditional mean: $D_i = A^T\hat{M}_i$; and λ_i is the ith diagonal element of Λ.

Thus, it is clear that for every feature i, B can be calculated separately. The feature with the largest B is the best feature, the one with the second largest is the second best, and so on. Also, the two best are the best two, and so on.

4.3 Approximation Algorithms

Figure B-4 shows the probability density functions for h(x) given either class 1 or 2. The probability of error is then the area under the two curves (multiplied by the prior probabilities). The objective is to develop an algorithm which will approximate the class-conditional probability density functions of h(x) and, hence, the probability of error.

Normal Assumption. When $\hat{\Sigma}_1 = \hat{\Sigma}_2$, h(x) becomes a linear function of x and hence is normally distributed. In general, $\hat{\Sigma}_1$ does not equal $\hat{\Sigma}_2$ and h(x) is not normal. An algorithm was developed and tested under the assumption that h(x) is normally distributed, but results showed it to be a very poor approximation to the probability of error and hence the algorithm was not further analyzed.

Discriminant function:

$$h(X) = -\ln \frac{\hat{p}(X|\omega_1)}{\hat{p}(X|\omega_2)}$$

$$= (X-\hat{M}_1)^T \hat{\Sigma}_1^{-1} (X-\hat{M}_1) - (X-\hat{M}_2)^T \hat{\Sigma}_2^{-1} (X-\hat{M}_2) + \ln \frac{|\hat{\Sigma}_1|}{|\hat{\Sigma}_2|} - \ln \frac{\hat{P}(\omega_1)}{\hat{P}(\omega_2)} \begin{matrix} \omega_1 \\ < \\ > \\ \omega_2 \end{matrix} 0$$

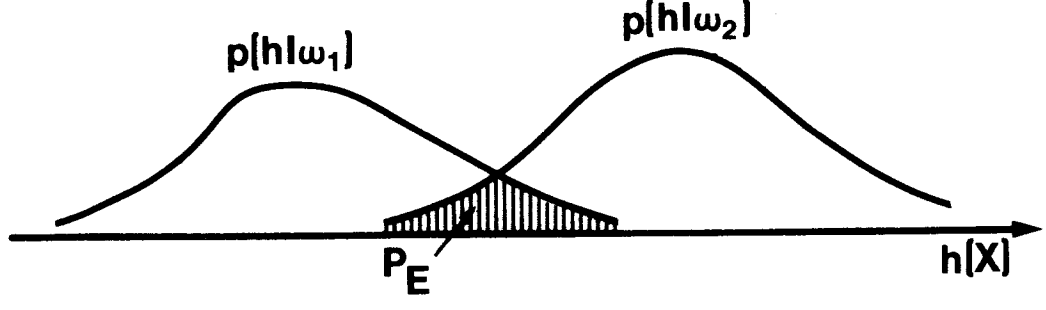


Figure B-4. Probability density functions of $h(X|\omega_i)$ and the probability of error.

Modified-Gamma Distribution Assumption. Fukunaga and Krile [52] developed an algorithm that approximates h(x) by convolving the densities of n (number of dimensions) noncentral χ^2 variables having multiplicative constants, and adding a shift parameter, by considering a general gamma function,

$$g(h) = \begin{cases} \dfrac{h^{\alpha} e^{-h/\beta}}{\beta^{\alpha+1}\,\Gamma(\alpha+1)} & \text{for} \quad h \geq 0 \\ 0 & h < 0 \end{cases}$$

The parameters α and β can be determined so that the means and the variances of the true distributions match those of the approximation.

Initial testing of the algorithm was done using four examples. These were all based on aircraft multispectral scanner data, LARSYS run 71053900. The area is in Tippecanoe County, Indiana, and the data set was part of the 1971 Corn Blight Watch Experiment. The data were taken in August, 1971, and have 12 channels.

Results are plotted in Figure B-5. The first example contains wheat and corn, with 265 samples of wheat and 569 samples of corn. The second example also has wheat and corn, but with 20 samples of each. The third example has wheat and corn with 13 samples of each. And the last example has corn combined with pasture as one class (720 samples) and forest as the other class (483 samples).

5. Discussion of Results to Date

The probability of correct classification is plotted against the best n channels, n=1,2,...,12, as determined by the Bhattacharyya distance. In all four examples shown, the black lines represent the results obtained using the modified-gamma distributions approximation.

Procedural and programming problems remain in transforming the data; therefore, these results are provisional at this time. These problems are being studied but are unresolved as yet. To provide a basis for comparison, however, the best n channels for the original data were computed (using the Transformed Divergence in the LARSYS SEPARABILITY processor) and used for determining experimental results (dotted lines). While the values of the probability of correct classification thus obtained do not exactly correspond to the values obtained using the transformed features, one might expect them to be quite close as both feature subsets are near-optimal. Results involving the transformed features will be reported when the remaining problem is solved.

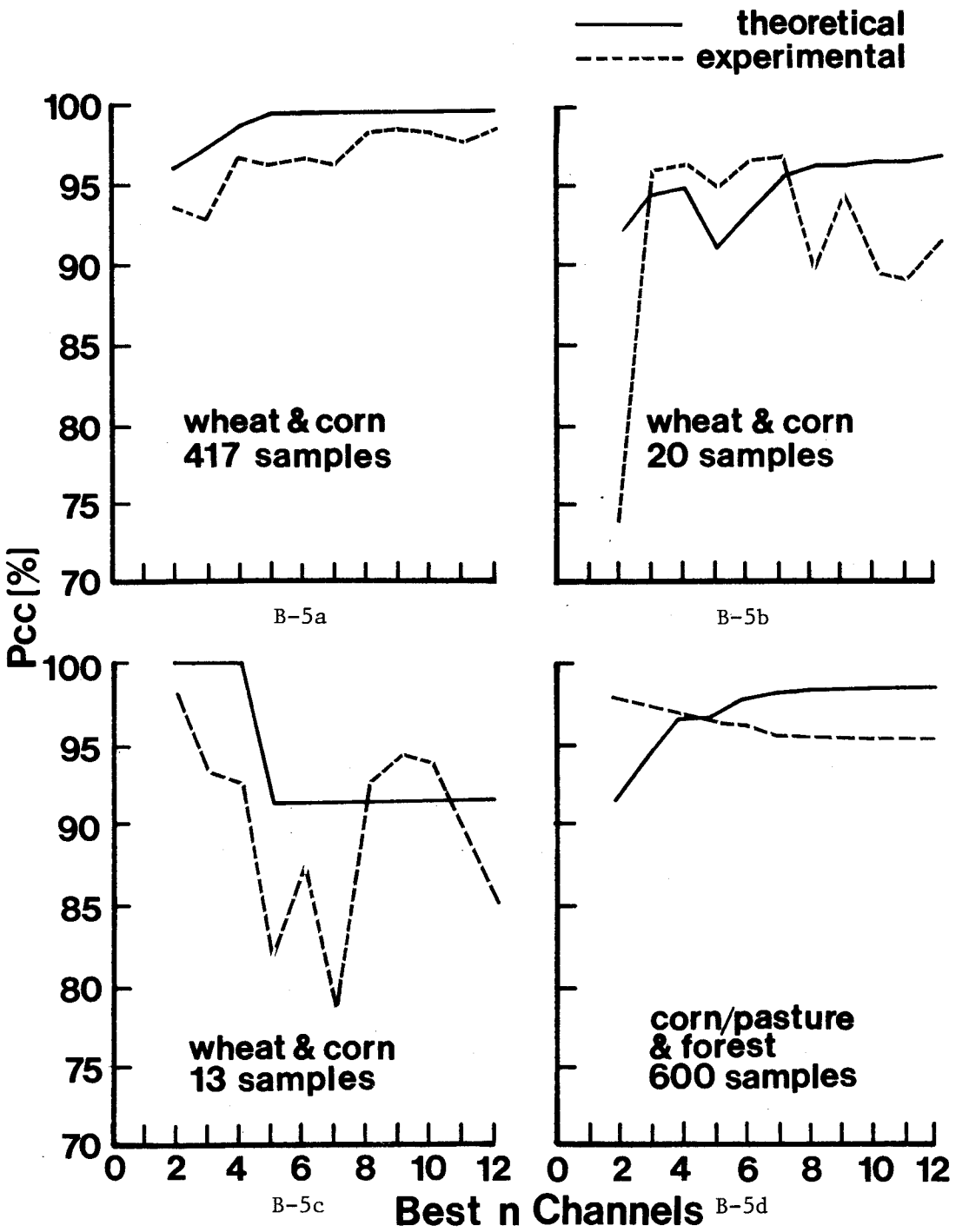


Figure B-5. Theoretical and experimental results of the probability of correct classification vs. the best N channels.

Figure B-5a shows wheat and corn where there are enough training samples to estimate the distributions, so that a Hughes phenomenon does not occur. We see that the approximation is a very good one in this case, especially for large dimensionality, where the algorithm is more accurate.

Looking at Figure B-5b, we see again that up to the best 7 channels, the approximation is very good (except at the low dimensionality of 2). After the best 7 channels, the approximation predicts a saturation of P_{cc}, while experimental results show a decrease in the accuracy. The decrease is expected for large dimensionality as the low number of training samples in this case (20 samples per class) is not sufficient to estimate the distributions at high dimensionality, i.e., there is a Hughes phenomenon occurring here.

The same effect is also noticed in Figure B-5c, where the algorithm saturates at high dimensionality, but experimental results show a decrease in accuracy. Actually, the decrease starts at a lower dimensionality (2 features) here because of the lower number of training samples (13 samples per class).

Figure B-5d shows an example where corn and pasture are combined to form one class while forest is the other class. Here, although the training samples are sufficient to estimate the distributions, combining corn and pasture results in a distinctly non-normal distribution. Hence, the algorithm fails to detect the slight Hughes peaking that does occur.

It appears that the next step is to modify the algorithm so that it does take into account the number of training samples and its effect on the accuracy of the estimated statistics for different dimensionality. More examples will be used to test the algorithm and provide conclusive results. Furthermore, tests must be conducted to examine how the program performs under different conditions (high and low separability, number of training samples, etc.).

6. Simulation Algorithm

In remote sensing data analysis, several assumptions are made that are not always precisely met. These assumptions often include: that the training classes are normally distributed; that the training data are representative of the area to be classified; that the number of classes present is known; and that all pixels are pure (one class only). In testing new algorithms, deviations from the assumptions may obscure the action of the new process. At times it is not clear whether a particular result is due to the aspects of the algorithm or to the extent the data set deviates from the assumptions.

A method was developed to obtain an artificial data set through simulation. While retaining the natural spatial and spectral informa-

tion in the scene by basing the simulation on a classification, the data set provides the analyst with an exact number of classes in the scene, normal distributions of these classes, independent measurements and "pure pixels." The simulation method was reported in LARS Technical Report 070980 [55] , and the program will be used for testing the error estimator after it is further developed.

References

1. Swain, P.H. and H. Hauska. 1977. The Decision Tree Classifier: Design and Potential. IEEE Trans. Geos. Elect. GE-15(3).

2. Fukunaga, K. 1972. Introduction to Statistical Pattern Recognition. Academic Press, New York.

3. Duda, R.O. and P.E. Hart. 1973. Pattern Classification and Scene Analysis. Wiley, New York.

4. Fleming, M.D., J.S. Berkebile, and R.M. Hoffer. 1975. Computer-Aided Analysis of Landsat-1 MSS Data: A Comparison of Three Approaches, Including a 'Modified Clustering' Approach. Laboratory for Applications of Remote Sensing, Purdue University, West Lafayette, Indiana. LARS Information Note 072475.

5. Fleming, M.D. and R.M. Hoffer. 1977. Computer-Aided Analysis Techniques for an Operational System to Map Forest Lands Utilizing Landsat MSS Data. Laboratory for Applications of Remote Sensing, Purdue University, West Lafayette, Indiana. LARS Information Note 112277.

6. Fukunaga, K. and D.L. Kessell. 1973. Nonparametric Bayes Error Estimation Using Unclassified Samples. IEEE Trans. Infor. Theory. IT-19:434-400.

7. Mobasseri, B.G. and C.D. McGillem. 1979. Multiclass Bayes Error Estimation by a Feature Space Sampling Technique. IEEE Trans. Systems, Man and Cybernetics. SMC-9(10):660-665.

8. Moore, D.S., S.J. Whitsitt, and D.A. Landgrebe. 1976. Variance Comparisons of Unbiased Estimators of Probability of Correct Classification. IEEE Trans. Infor. Theory. IT-22:102-105.

9. Whitsitt, S.J. and D.A. Landgrebe. 1977. Error Estimation and Separability Measures in Feature Selection for Multiclass Pattern Recognition. Laboratory for Applications of Remote Sensing, Purdue University, West Lafayette, Indiana. LARS Publication 082377.

10. Wiersma, D.J. and D.A. Landgrebe. 1978. The Analytical Design of Spectral Measurements for Multispectral Remote Sensor Systems. Laboratory for Applications of Remote Sensing, Purdue University, West Lafayette, Indiana. LARS Technical Report 122678.

11. Mobasseri, B.G., D.J. Wiersma, E.R. Wiswell, D.A. Landgrebe, C.D. McGillem, and P.E. Anuta. 1978. A Multispectral Scanner System Parameter Study and Analysis Software System Description. Laboratory for Applications of Remote Sensing, Purdue University, West Lafayette, Indiana. LARS Contract Report 112678.

12. Marill, T. and D.M. Green. 1963. On the Effectiveness of Receptors in Recognition Systems. IEEE Trans. Infor. Theory. IT-9:11-17.

13. Swain, P.H. and R.C. King. 1973. Two Effective Feature Selection Criteria for Multispectral Remote Sensing. Laboratory for Applications of Remote Sensing, Purdue University, West Lafayette, Indiana. LARS Information Note 042673.

14. Jeffreys, H. 1948. Theory of Probability. Oxford University Press.

15. Matusita, K. 1951. On the Theory of Statistical Decision Functions. Ann. Instit. Stat. Math. (Tokyo). 3:17-35.

16. Bhattacharyya, A. 1943. On a Measure of Divergence Between Two Statistical Populations Defined by Their Probability Distributions. Bul. Calcutta Math. Soc. 35:99-109.

17. Mahalanobis, P.C. 1936. On the Generalized Distance in Statistics. Proc. National Inst. Sci. (India). 12:49-55.

18. Swain, P.H., T.V. Robertson, and A.G. Wacker. 1971. Comparison of the Divergence and B-Distance in Feature Selection. Laboratory for Applications of Remote Sensing, Purdue University, West Lafayette, Indiana. LARS Information Note 020871.

19. Kailath, T. 1967. The Divergence and Bhattacharyya Distance Measures in Signal Selection. IEEE Trans. Communication Technology. COM-14(1):52-60.

20. Karhunen, K. 1947. Uber Lineare Methoden in Der Wahrscheinlichkeitsrechnung. Amer. Acad. Sci., Fennicade. Ser. A,I, 37:3-79. (Transl: Rand Corp., Santa Monica, California, Rept. T-131, August 1960.)

21. Loeve, M. 1963. Probability Theory. Van Nostrand, Princeton, New Jersey.

22. Kanal, L. 1976. Patterns in Pattern Recognition: 1968-1974. IEEE Trans. Infor. Theory. IT-20(6):697-722.

23. Hughes, G.F. 1968. On the Mean Accuracy of Statistical Pattern Recognizer. IEEE Trans. Infor. Theory. IT-14(1):55-63.

24. Abend, K. and T.J. Harley, Jr. 1969. Comments "On the Mean Accuracy of Statistical Pattern Recognizers." IEEE Trans. Infor. Theory (correspondence), pp. 420-421.

25. Chandrasekaran, B. and T.J. Harley, Jr. 1969. Comments "On the Mean Accuracy of Statistical Pattern Recognizer." IEEE Trans. Infor. Theory (correspondence), pp. 421-423.

26. Kanal, L. and B. Chandrasekaran. 1971. On Dimensionality and Sample Size in Statistical Pattern Classification. Pattern Recognition. 3:225-236.

27. Van Campenhout, J.M. 1978. On the Peaking of the Hughes Mean Recognition Accuracy: The Resolution of an Apparent Paradox. IEEE Trans. Systems, Man and Cybernetics. SMC-8(5):390-395.

28. Kulkarni, A.V. 1978. On the Mean Accuracy of Hierarchical Classifiers. IEEE Trans. Computers. C-27(8):771-776.

29. Raudys, S.J. 1979. Determination of Optimal Dimensionality in Statistical Pattern Classification. Pattern Recognition. 11:263-270.

30. Trunk, G.V. 1979. A Problem of Dimensionality: A Simple Example. IEEE Trans. Pattern Analysis and Machine Intelligence. PAMI-1:306-307.

31. Raudys, S. and V. Pikelis. 1980. On Dimensionality, Sample Size, Classification Error, and Complexity of Classification Algorithm in Pattern Recognition. IEEE Trans. Pattern Analysis and Machine Intelligence. PAMI-2(3).

32. El-Sheikh, T.S. and A.G. Wacker. 1980. Effect of Dimensionality and Estimation on the Performance of Gaussian Classifiers. Pattern Recognition 12:115-126.

33. Wald, A. 1947. Sequential Analysis. Wiley, New York.

34. Fu, K.S., Y.T. Chien and G.P. Cardillo. 1967. A Dynamic Programming Approach to Sequential Pattern Recognition. IEEE Trans. Computers. C-16(6):790-803.

35. Fu, K.S. 1968. Sequential Methods in Pattern Recognition and Machine Learning. Academic Press.

36. Dubes, R. and A.K. Jain. 1979. Validity Studies in Clustering Methodologies. Pattern Recognition. 11:235-254.

37. Lukasova, A. 1979. Hierarchical Agglomerative Clustering Procedure. Pattern Recognition. 11:365-381.

38. Nadler, M. 1971. Error and Reject Rates in a Hierarchical Pattern Recognizer. IEEE Trans. Computers. C-20:1598-1601.

39. Meisel, W.S. and D.A. Michalopoulos. 1973. A Partitioning Algorithm with Application in Pattern Classification and the Optimization of Decision Trees. IEEE Trans. Computers. C-22:93-103.

40. Wu, C.L., D.A. Landgrebe and P.H. Swain. 1974. The Decision Tree Approach to Classification. Laboratory for Applications of Remote Sensing, Purdue University, West Lafayette, Indiana. LARS Information Note 090174.

41. Swain, P.H., C.L. Wu, D.A. Landgrebe, and H. Hauska. 1975. Layered Classification Techniques for Remote Sensing Applications. Laboratory for Applications of Remote Sensing, Purdue University, West Lafayette, Indiana. LARS Information Note 061275.

42. Bartolucci, L.A., P.H. Swain, and C.L. Wu. 1976. Selective Radiant Temperature Mapping Using a Layered Classifier. IEEE Trans. Geoscience Electronics. GE-14:101-106.

43. You, K.C. and K.S. Fu. 1976. An Approach to the Design of a Linear Binary Tree Classifier. Proc. Conference on Machine Processing of Remotely Sensed Data. June 29-July 1, 1976. IEEE Catalog No. 76CH1103-1 MPRSD.

44. Fletcher, R. and M.J.D. Powell. 1963. A Rapid Descent Method for Minimization. Computer Journal. 6:163-168.

45. Kulkarni, A.V. and L.N. Kanal. 1976. An Optimization Approach to Hierarchical Classifier Design. Proc. Third Int. Joint Conf. on Pattern Recognition (Coronado, California), IEEE Catalog No. 76CH/140-3C.

46. Parikh, J. 1977. A Comparative Study of Cloud Classification Techniques. Remote Sensing of Environment. 6:67-81.

47. Sethi, I.K. and B. Chatterjee. 1977. Efficient Decision Tree Design for Discrete Variable Pattern Recognition Problems. Pattern Recognition. 9:197-206.

48. Breiman, L. 1978. Growing Trees to Analyze High Dimensional Data. Technology Service Corporation Report. TSC-CSD-IN-024.

49. Sonquist, J.A., E.L. Baker, and J.N. Morgan. 1973. Searching for Structure. Survey Research Center, Institute for Social Research, University of Michigan, Ann Arbor, Michigan.

50. Rounds, E.M. 1979. A Combined Nonparametrical Approach to Feature Selection and Binary Decision Tree Design. Proc. IEEE Computer Society Conference on Pattern Recognition and Image Processing, Chicago, Illinois, August 6-8, 1979.

51. Kullback, S. 1959. Information Theory and Statistics. p. 195. Wiley, New York.

52. Fukunaga, K. and T. Krile. 1969. Calculation of Bayes Recognition Error for Two Multivariate Gaussian Distributions. IEEE Trans. on Computers. C-18(3).

53. Wacker, A.G. and D.A. Landgrebe. 1971. The Minimum Distance Approach to Classification. Laboratory for Applications of Remote Sensing, Purdue University, West Lafayette, Indiana. LARS Information Note 100771.

54. Swain, P.H. and S.M. Davis, eds. 1978. Remote Sensing: The Quantitative Approach. pp. 164-174. McGraw-Hill, Inc., New York.

55. Muasher, M. and P. Swain. 1980. A Multispectral Data Simulation Technique. Laboratory for Applications of Remote Sensing, Purdue University, West Lafayette, Indiana. LARS Technical Report 070980.

C. CONTEXTUAL CLASSIFICATION

P.H. Swain, H.J. Siegel, J.C. Tilton, and B.W. Smith*

Multispectral image data collected by remote sensing devices aboard aircraft and spacecraft are relatively complex data entities. Both the spatial attributes and spectral attributes of these data are known to be information bearing [1], but to reduce the magnitude of the computations involved, most analysis efforts have focused on one or the other. Only within the last few years have serious efforts been made to utilize them jointly. For example, one approach uses the spectral homogeneity of "objects," such as agricultural fields, to segment the scene and then uses sample classification to assign each object as a whole, rather than its individual pixels (picture elements), to an appropriate ground cover class [2]. Another approach involves extraction of features based on gray-tone spatial-dependency matrices from which texture-like characteristics are developed [3].

In this project we are developing a more general way to exploit the spatial/spectral context of a pixel to achieve accurate classification. Just as in written English one can expect to find certain letters occurring regularly in particular arrangements with other letters (qu, ee, est, tion), so certain classes of ground cover are likely to occur in the "context" of others. The former phenomenon has been used to improve character recognition accuracy in text-reading machines. We have demonstrated that the latter can be used to improve accuracy in classifying remote sensing data. Intuitively this should not be surprising since one can easily think of ground cover classes more likely to occur in some contexts than in others. One does not expect to find wheat growing in the midst of a housing subdivision, for example. A close-grown, lush vegetative cover in such a location is more likely the turf of a lawn.

Classification algorithms such as the contextual classifier under development in this project (and even much simpler algorithms used for remote sensing data analysis) typically require large amounts of computation time. One way to reduce the execution time of these tasks is through the use of parallelism. Various parallel processing systems that can be used for remote sensing have been built or proposed. The Control Data Corporation Flexible Processor system is a commercially available multiprocessor system which has been recommended for use in remote sensing. PASM is a proposed multimicroprocessor system for image processing and pattern recognition. The way in which parallel machines such as the CDC Flexible Processor system and PASM can exploit parallelism to perform contextual classifications is under examination in this project.

The theoretical foundations of this contextual classifier and preliminary experimental results from applying the contextual classifier to a variety of very different data sets were detailed in last year's final report issued in November 1979, along with a description of the CDC Flexible Processor and how it can be used to implement contextual classification. This report summarizes the work reported in November 1979 and presents

---

* Substantial contributions to this research by Dr. Stephen B. Vardeman are gratefully acknowledged.

the work completed in this contract year. Section 1 summarizes the theoretical basis and the classification model while Section 2 presents the results of studies into methods of overcoming the practical implementation problems indicated by the preliminary results of last year's final report. Section 3 gives examples of contextual classifiers. In Section 4, uniprocessor algorithms for performing contextual classifications are presented and their computational complexity is analyzed. The CDC Flexible Processor system is described in Section 5 and the use of this system in performing the contextual classifiers of Section 3, based on the algorithms from Section 4, is the topic of Section 6. The SIMD mode of parallel processing is defined and PASM is overviewed in Section 7. Section 8 uses the uniprocessor algorithms of Section 4 as a basis for developing parallel (SIMD) algorithms for performing contextual classifications.

1. Theoretical Basis and Classification Model

Consistent with the general characteristics of imaging systems for remote sensing, we assume a two-dimensional array of $N=N_1\times N_2$ random observations X_{ij} having fixed but unknown classification ϑ_{ij}, as shown in Figure C-1. The observation X_{ij} consists of n measurements (usually containing spectral and/or temporal information), while the classification ϑ_{ij} can be any one of m spectral or information classes* from the set $\Omega = \{\omega_1,\omega_2,\ldots,\omega_m\}$.

$$\begin{array}{|cccc|}\hline \vartheta_{11} & \vartheta_{12} & \cdots & \vartheta_{1N_2} \\ \vartheta_{21} & \vartheta_{22} & \cdots & \vartheta_{2N_2} \\ \vdots & & & \\ \vartheta_{N_1 1} & & \cdots & \vartheta_{N_1 N_2} \\ \hline \end{array}$$

Figure C-1. A two-dimensional array of $N=N_1\times N_2$ pixels.

Let $\underline{X}$ denote a vector whose components are the ordered observations:

$$\underline{X} = [X_{ij} \mid i=1,2,\ldots,N_1; j=1,2,\ldots,N_2]^T.$$

Similarly, let $\underline{\vartheta}$ be the vector of states (true classifications) associated with the observations in $\underline{X}$:

$$\underline{\vartheta} = [\vartheta_{ij} \mid i=1,2,\ldots,N_1; j=1,2,\ldots,N_2]^T.$$

The following notation will be useful. Let $\underline{\vartheta}^p \in \Omega^p$ and $\underline{X}^p \in (R^n)^p$ stand respectively for p-vectors of classes and n-dimensional measurements; each component of $\underline{\vartheta}^p$ is a variable which can take on any classification value; each component of $\underline{X}^p$ is a random n-dimensional vector which can take on values in the observation space.

Let the action (classification) taken with respect to pixel (i,j) be denoted by $a_{ij}\in\Omega$. We restrict the action a_{ij} to be a function of a specified subset of observations in $\underline{X}$. This subset includes, along with X_{ij}, p-1 observations spatially near to, but not necessarily adjacent to, X_{ij}. These p-1 observations serve as the spatial context for X_{ij} and are taken from the same spatial positions relative to pixel position (i,j) for all i and j. Call this arrangement of pixels together with X_{ij} the p-context array, several examples of which are shown in Figure C-2. Group the p observations in the p-context array into a vector of observations $\underline{X}_{ij} = (X_1,X_2,\ldots,X_p)^T$ and let $\underline{\vartheta}_{ij}$ be the vector of true but unknown classifications associated with the observations in $\underline{X}_{ij}$. Note that the $\underline{\vartheta}_{ij}$ and $\underline{X}_{ij}$ are the particular instance of $\underline{\vartheta}^p$ and $\underline{X}^p$ associated with pixel position (i,j). Correspondence of

---

* Spectral classes are spectrally differentiable subclasses of information classes (the classes of interest).

the components of $\underline{\vartheta}_{ij}$, $\underline{X}_{ij}$, $\underline{\vartheta}^p$ and $\underline{X}^p$ to the positions in the p-context array is fixed but arbitrary except that the pixel to be classified will always correspond to the p^{th} components.

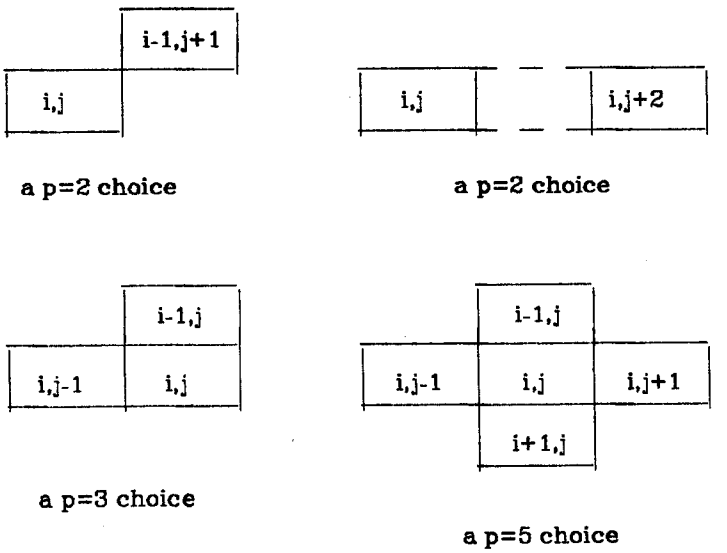


Figure C-2. Examples of p-context arrays.

Let the loss suffered by taking action a_{ij} be denoted by $\lambda(\vartheta_{ij}, a_{ij})$ for some fixed non-negative function $\lambda(\cdot,\cdot)$. The expected average loss (or risk) suffered over the N classifications in the classification array is

$$R_{\underline{\vartheta}} = E\,\frac{1}{N}\sum_{i,j}\lambda(\vartheta_{ij}, a_{ij}(\underline{X}_{ij})) \tag{1}$$

where the expectation is with respect to the distribution of the p-context array.

Now consider finding an optimal decision rule of the form

$$a_{ij}(\underline{X}_{ij}) = d(\underline{X}_{ij}) \tag{2}$$

for a fixed function $d(\cdot)$ mapping p-vectors of observations to actions so that $R_{\underline{\vartheta}}$ is minimized. If we require that the distribution of the p-context array is spatially invariant, i.e. the value of the probability density for $\underline{X}_{ij}$ depends only on the measurement values in $\underline{X}_{ij}$ and the set of classifications in $\underline{\vartheta}_{ij}$ and not the location (i,j), the risk, $R_{\underline{\vartheta}}$, can be written as

$$\begin{aligned} R_{\underline{\vartheta}} &= \sum_{\underline{\vartheta}^p \in \Omega^p} C(\underline{\vartheta}^p)\int \lambda(\vartheta_p, d(\underline{X}^p))\, f(\underline{X}^p \mid \underline{\vartheta}^p)\, d\underline{X}^p \\ &= \int \sum_{\underline{\vartheta}^p \in \Omega^p} C(\underline{\vartheta}^p)\lambda(\vartheta_p, d(\underline{X}^p))\, f(\underline{X}^p \mid \underline{\vartheta}^p)\, d\underline{X}^p \end{aligned} \tag{3}$$

where $G(\underline{\vartheta}^p)$, the *context distribution*, is the relative frequency with which $\underline{\vartheta}^p$ occurs in the array $\underline{\vartheta}$, and ϑ_p is the p^{th} element of $\underline{\vartheta}^p$. For any array $\underline{\vartheta}$, a decision rule $d(\underline{X}^p)$ minimizing $R_{\underline{\vartheta}}$ can be obtained by minimizing the integrand in equation (3) for each $\underline{X}^p$; thus for a specific $\underline{X}_{ij}$ (an instance of $\underline{X}^p$), an optimal action is:

$$d(\underline{X}_{ij}) = \text{the action (classification) } a \text{ which minimizes}$$
$$\sum_{\underline{\vartheta}^p \in \Omega^p} G(\underline{\vartheta}^p)\lambda(\vartheta_p, a) f(\underline{X}_{ij} \mid \underline{\vartheta}^p). \tag{4}$$

In practice, a "0-1 loss function" is usually assumed, i.e.,

$$\lambda(\vartheta, a) = \begin{cases} 0, & \text{if } \vartheta = a \\ 1, & \text{if } \vartheta \neq a \end{cases}.$$

Then equation (4) simplifies and the decision rule becomes:

$$d(\underline{X}_{ij}) = \text{the action } a \text{ which } \textit{maximizes}$$
$$\sum_{\substack{\underline{\vartheta}^p \in \Omega^p, \\ \vartheta_p = a}} G(\underline{\vartheta}^p) f(\underline{X}_{ij} \mid \underline{\vartheta}^p). \tag{5}$$

We now assume class-conditional independence for the observations. This assumption means that the joint class-conditional density over the p-context array can be written as

$$f(\underline{X}_{ij} \mid \underline{\vartheta}^p) = \prod_{k=1}^{p} f(X_k \mid \vartheta_k) \tag{6}$$

where X_k and ϑ_k are the k^{th} elements of $\underline{X}_{ij}$ and $\underline{\vartheta}^p$, respectively. Evidence that this is a reasonable assumption may be found in reference [4]. With this assumption, the decision rule in equation (5) becomes:

$$d(\underline{X}_{ij}) = \text{the action } a \text{ which maximizes}$$
$$\sum_{\substack{\underline{\vartheta}^p \in \Omega^p, \\ \vartheta_p = a}} G(\underline{\vartheta}^p) \prod_{k=1}^{p} f(X_k \mid \vartheta_k). \tag{7}$$

The optimal choice of $d(\cdot)$ cannot be implemented in practice since it depends on $G(\underline{\vartheta}^p)$ and the $f(X_k \mid \vartheta_k)$ which are unknown. Methods for estimating the $f(X_k \mid \vartheta_k)$ are well established from considerable experience in using the conventional no-context maximum likelihood decision rule [1]. Methods for estimating $G(\underline{\vartheta}^p)$ are not so well established. We can, however, expect that, at least for large $N = N_1 \times N_2$, a decision rule in which $G(\underline{\vartheta}^p)$ is replaced by an estimate $\hat{G}(\underline{\vartheta}^p)$ based on the $\underline{X}_{ij}$ will have risk $\hat{R}_{\underline{\vartheta}}$ approximating that of the optimal rule. This "bootstrap effect" is one aspect of this problem explored in the following sections.

2. Experimental Results

In last year's final report we explored the effectiveness of contextual classification as applied to the analysis of multispectral remote sensing data. In doing so we used three simulated data sets and two real Landsat data sets. The simulated data sets demonstrated the effectiveness of the classifier given that the underlying assumptions mentioned in Section 1 were satisfied.

The preliminary results reported last year indicated three main problem areas for further research. These were:

1. A generally applicable method was needed for making adequate estimates of the context distribution;
2. The originally implemented algorithm was too computationally intensive for general use in practical classification problems; and
3. It was not clear what effect the assumptions made in developing the classification model had upon the performance of the context classifier.

Several different approaches for dealing with the above problems have been thought through, and some of these approaches have been investigated this past year. How these approaches relate to the main research problems and to our major goals of (a) advancing the theoretical understanding of this problem, and (b) developing a contextual classification algorithm for use in practical problems is illustrated in Figure C-3.

2.1 Context Distribution Estimation

Simulated data sets were utilized in the earliest experiments exploring the effectiveness of classifying multispectral remote sensing data using context classification as defined by the set of discriminant functions in equation (7). This was done so as to demonstrate the effectiveness of the classifier given that the underlying assumptions in the classification model are satisfied. At first, the context distribution was found by simple tabulation from the true classification used as a template for the data simulation. As reported last year, the classifier was very effective when the context distribution was determined in this way.

When dealing with real data, there is no direct way of determining the context distribution. We cannot tabulate the context distribution from the true classification since the true classification is not known. (If it were known, we would not have to perform a classification!) As mentioned in the last section, we should be able to base an adequate estimate of the context distribution on the data, $\underline{X}_{ij}$, or, more practically, on representative sections from the data designated as a training set. The most straightforward way to develop an estimate of the context distribution from the training set would be to perform a conventional no-context classification of the training set and use the context distribution as tabulated from this classification as an estimate of the true context distribution. One could then further refine this estimate of the context distribution by making another estimate from the contextual classification, and even iterate in this way until no further improvements in classification accuracy were obtained.

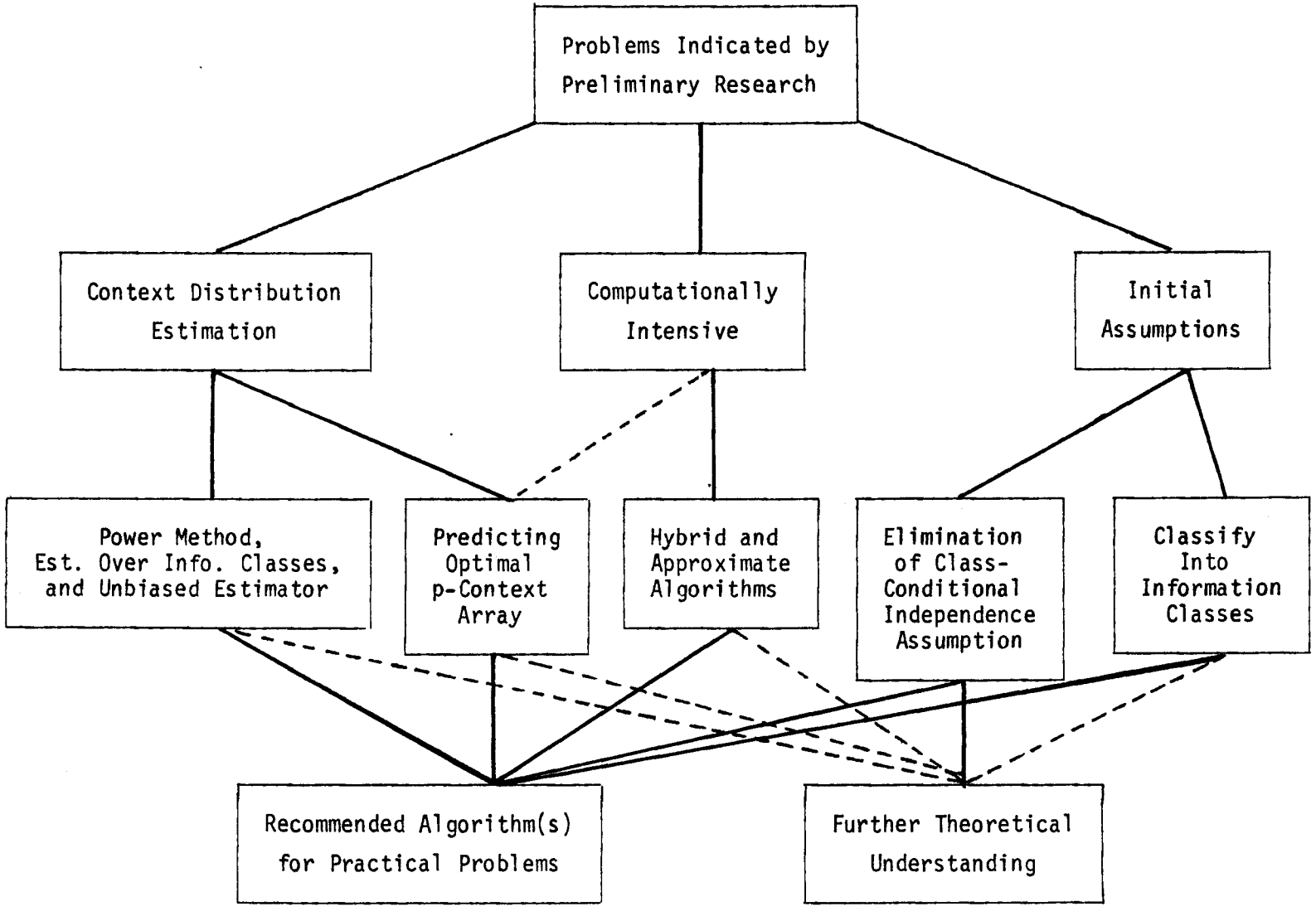


Figure C-3. Interrelationships among topics of research.

This iterative bootstrap method was tested on one simulated data set and two real data sets. As reported last year, this method gave excellent results on the simulated data set, but rather poor results on the real data sets.

The poor results using the iterative bootstrap method on the real data sets led to a proposed alternative method for estimating the context distribution. This method is based on the idea that ground truth information, if available, should improve the context distribution estimate when incorporated into the estimate. The proposed ground-truth-guided method for estimating the context distribution incorporates the ground truth into the context distribution estimate in the following manner:

1. Perform a conventional no-context classification of the training set using uniform prior probabilities as before, but with the following twist: The classifier is allowed to choose only among spectral classes associated with the information class designated by the ground truth.
2. Estimate the context distribution by tabulation from the resulting 100-percent accurate classification of the training set.
3. Classify the entire scene with the contextual classifier and evaluate the results over a test set disjoint from the training set.

Since last year's report, the ground-truth-guided method was tested on a 50-pixel-square LACIE data set. (This Landsat data set is a large-field agricultural scene from Hodgman County, Kansas. See last year's final report for a detailed description of this data set.) The ground-truth-guided no-context classification was performed as described over the first 25 lines of the data set and the context distribution was estimated over those 25 lines. Contextual classifications of the scene were performed and classification accuracy* was evaluated over the last 25 lines. The results (Figure C-4) show that this is a very effective way to estimate the context distribution. This easily implemented automatic method for estimating the context distribution produced on this data set an estimate of the context distribution which in turn produced contextual classifications with significant improvement in classification accuracy over the conventional uniform-priors no-context classification.

While this method can produce good estimates of the context distribution, it suffers the limitation that an adequate number of sufficiently sized *blocks* of ground truth are needed to make an accurate estimate of the context distribution. The method cannot be used when blocks of ground truth test data are not available such as with the other data set available (the Bloomington, Indiana data set described in last year's report). A generally applicable method such as the iterative bootstrap method is still needed for such a case.

The poor performance of the iterative bootstrap method on real Landsat data as compared to simulated data may be due to the fact that the class-conditional probabilities, $f(X_k \mid \vartheta_k)$, were not modeled adequately for real Landsat data. Since the $f(X_k \mid \vartheta_k)$ were modeled exactly in the simulated data case, the initial no-context uniform-priors classification was accurate enough for the iterative bootstrap method to work. The

---

* Classification accuracy can be tabulated in two ways. *Overall accuracy* is simply the overall number of correct classifications divided by the total number attempted. *Average- by- class accuracy* is obtained by first computing the accuracy for each class and taking the arithmetic average of the class accuracies. The latter is significant when the classification results exhibit a tendency to discriminate in favor of or against a subset of the classes

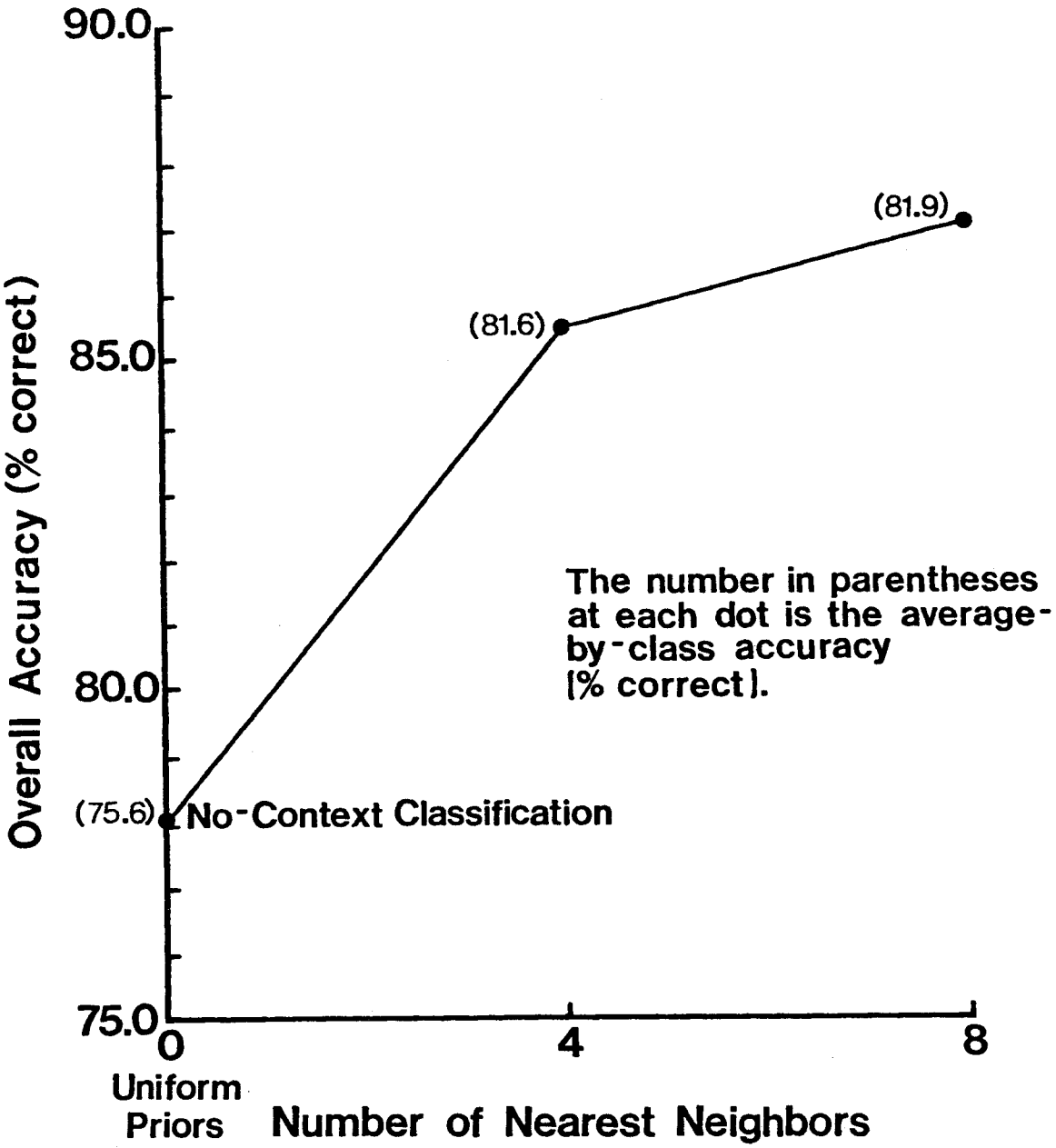


Figure C-4. Performance using the ground-truth-guided method for estimating the context distribution (LACIE data).

bootstrap effect did not perform satisfactorily in the real Landsat data cases because the initial classifications were not accurate enough due to inaccuracies in modeling the $f(X_k \mid \vartheta_k)$. The poor initial classifications in the real data cases produced estimates of the context distribution, $G(\underline{v}^p)$, which contained more erroneous class configuration counts than in the simulated case, which in turn gave poorer context classifications results.

There are several ways in which the context distribution estimates from no-context classifications of real data could be "cleaned up". One could employ a procedure which deletes all class configurations with counts below a certain threshold. Or one could divide the count for each class configuration by a fixed number and take the integer part of the result as the new count, deleting all class configurations with counts that become zero.

Both of the aforementioned clean-up procedures could result in totally eliminating rarely occurring but valid classes from context classifications using the cleaned up estimate of the context distribution. A context distribution estimate clean-up method that does not suffer from this problem is the "Power Method."

The Power Method raises the relative frequency count for each class configuration to a power and uses the result as the context distribution estimate. For powers greater than one, the class configurations with larger counts are favored more heavily than those with relatively small (and possibly erroneous) counts. Conversely, for powers less than one, the class configurations with large counts are not so heavily favored. Going to the extreme of a power of zero results in all class configurations being equally favored as in a uniform-priors no-context classification. In no case is a class configuration deleted from the context distribution estimate.

The Power Method was tested on the Bloomington, Indiana data set. (This Landsat data set is from an area to the southeast of Bloomington, Indiana, containing approximately equal amounts of urban and agricultural area. The agricultural portion of this scene has generally smaller fields than in the LACIE data set.) The results of this test are reported in detail in [5]. Using the Power Method with two-nearest-neighbors context (neighbors to the north and east) based on an estimate of $G(\underline{v}^p)$ from the no-context uniform-priors classification, the best classification was produced using a power of 5, for which an overall accuracy of 87.0 percent and average-by-class accuracy of 86.1 percent was achieved. This compares to the 85.3 percent overall accuracy and 84.8 average-by-class accuracy acheived in two iterations of the bootstrap method. (Further iterations produced no significant improvement.) The original no-context uniform-priors classification had an overall accuracy of 83.1 percent and an average-by-class accuracy of 82.7 percent.

A second iteration of estimating $G(\underline{v}^p)$, this time over four-nearest-neighbors context, was then made based on first-iteration classifications using $G(\underline{v}^p)$ raised to various powers. The second estimate of $G(\underline{v}^p)$ based on the classification using the first estimate raised to a power of 10 produced the best classification results with an overall accuracy of 88.5 percent and an average-by-class accuracy of 87.5 percent (using $G(\underline{v}^p)$ raised to a power of 5).

A test of the Power Method was also performed on the LACIE data set. The results were similar to the Bloomington, Indiana, data set results. Again using two-nearest-neighbor context (neighbors to the east and west), the best classification was produced using a power of 7. Here the overall and average-by-class accuracies were 83.7 percent

and 73.8 percent, respectively, as compared to overall and average-by-class accuracies of 78.7 and 72.0 percent, respectively, for the uniform-priors no-context case (evaluated over the entire scene). The best second iteration-result, using four-nearest-neighbor context, was produced with an estimate of $G(\underline{\vartheta}^p)$ made from the power of 15 first iteration classification and raised to a power of 10. This classification had an overall accuracy of 86.7 percent and average-by-class accuracy of 75.6 percent for an improvement of 8.0 percent and 3.6 percent, respectively, in overall and average-by-class accuracies. This compares to improvements of 1.8 percent and 1.0 percent, respectively, in overall and average-by-class accuracies produced by the iterative bootstrap method when evaluated over the entire scene.

Prior to making the second-iteration estimates of $G(\underline{\vartheta}^p)$ in the above tests, it was assumed that a more accurate classification would necessarily produce a better estimate of $G(\underline{\vartheta}^p)$. The results quoted here indicate this is not always the case. This makes the Power Method more difficult to use, since classifications must be made using estimates of $G(\underline{\vartheta}^p)$ based on several classifications from the previous iteration in order to find the best estimate.

Estimation Over Information Classes. Instead of just "cleaning up" the context distribution through using such a method as the Power Method discussed in the previous section, one could instead try to arrive at a better original estimate of the context distribution that may not need to be cleaned up (or may need less cleaning up).

Up to this point we have assumed spectral-class context to be significant. It may be, rather, that the information-class context is the important factor. For the common case where the number of spectral classes may be half or a third the number of spectral classes, estimating over information classes rather than spectral classes leads to a large reduction of dimensionality of the context distribution. The large dimensionality of the context distribution in the spectral class formulation may in and of itself be a significant source of estimation error. If this is indeed the case, the lower dimensionality of the context distribution estimated over information classes should lead to a more accurate estimate.

Let the set $\Omega=\{\omega_1,\omega_2,\ldots,\omega_m\}$ represent spectral classes and let the set $\Gamma=\{\gamma_1,\gamma_2,\ldots,\gamma_n\}$, $n \leq m$, represent information classes. Let $\underline{\vartheta}^p \in \Omega^p$ and $\underline{\zeta}^p \in \Gamma^p$ stand for p-vectors of classes over spectral and information classes, respectively. If we assume that the spectral classes carry no contextual information outside of that carried by their information class membership, we can calculate the context distribution over spectral classes, $G(\underline{\vartheta}^p)$, from the context distribution over information classes, $H(\underline{\zeta}^p)$, as follows:

$$G(\underline{\vartheta}^p) = H(\underline{\zeta}^p) \prod_{k=1}^{p} p(\vartheta_k \mid \zeta_k). \tag{8}$$

The $p(\vartheta_k \mid \zeta_k)$ represent the various probabilities that given a particular information class, ζ_k, the spectral class, ϑ_k, will be observed. Our context classification decision rule from equation (7) can now be rewritten to explicity show that the context distribution estimation is being done over information classes, viz:

$$d(\underline{X}_{ij}) = \text{the action } a \in \Omega \text{ which maximizes}$$

$$\sum_{\substack{\underline{\vartheta}^p \in \Omega^p, \\ \vartheta_p = v}} H(\underline{\zeta}^p) \prod_{k=1}^{p} p(\vartheta_k \mid \zeta_k) \prod_{k=1}^{p} f(X_k \mid \vartheta_k) \tag{9}$$

Spectral classes do carry some contextual information. One would expect then, that if the context distribution estimates are very accurate, the spectral-class estimate using equation (7) would produce better results than the information-class estimate using equation (9). This is precisely what happens when the context distributions are determined from the true classification for the simulated data set. Using two neighbor context (north and west neighbors), the spectral-class estimate produced overall and average-by-class accuracies of 93.0 and 78.4 percent. The corresponding information-class estimate result was 91.2 and 74.0 percent. As expected, the information-class estimate produced a significantly less accurate classification.

It would now be interesting to note what happens when the uniform-priors no-context classification is used to form the context distribution estimate for the simulated data set. Using two-neighbor context (north and west neighbors), the spectral-class estimate of the context distribution produced overall and average-by-class accuracies of 78.4 and 81.1 percent. The corresponding information-class estimate result was 79.8 and 81.7 percent. Here the information-class estimate produced a slightly more accurate classification.

These simulated data results would seem to indicate that although the information-class estimate context distribution produces less accurate classifications when the context distribution is known exactly, the information-class estimate is less sensitive to errors in the estimate of the context distribution when it must be estimated imprecisely as from a uniform-priors no-context classification. In the later case, the information-class context distribution estimate may produce more accurate classifications than the spectral-class estimate.

The first real-data test was performed using the Bloomingtion, Indiana data set. For two-neighbor context (north and west neighbors), the spectral-class estimate produced overall and average-by-class accuracies of 84.5 and 84.2 percent. The corresponding information-class estimate result was 85.9 and 85.8 percent. These results are quite similar to the two-neighbor simulated data-results.

A test was also performed using four-nearest-neighbor context. The spectral-class context distribution calculated from the information-class estimate by equation (8) had to be thresholded in this case, i.e. context vectors, $\underline{v}^p$, with relative frequency of occurance of less than a threshold value (here 6×10^{-6}) were eliminated from the sum in equation (9). If a nonthresholded context distribution would be used here, there would be so many separate context vectors to sum over in equation (9) that the computer program would take an impractical amount of time, even over a small 50-pixel square test area.

The four-nearest-neighbor spectral-class estimate produced overall and average-by-class accuracies of 84.5 and 84.1 percent. The thresholded information-class estimate produced accuracies of 88.2 and 88.7 percent. This rather substantial increase in classification accuracy was found to be even further improved by combining the Power Method with estimation of the context distribution over information classes. (In implementing the Power Method elements of $C(\underline{v}^p)$ calculated from equation (8) were raised to a power rather than elements of $H(\underline{\zeta}^p)$.) Using a power of 7 in this case produced overall and average-by-class accuracies of 89.6 and 89.5 percent. These accuracies matched those produced in two iterations of the Power Method when spectral class estimates of the context distribution were used. Additional iterations in either case produced no further improvement in classification accuracies. The Bloomington, Indiana data set

results using four-nearest-neighbor context are summarized in Figure C-5.

The same tests were repeated using the LACIE data set. For two-neighbor context (north and west neighbors), the spectral-class estimate produced overall and average-by-class accuracies of 80.0 and 72.1 percent. The corresponding information-class estimate produced accuracies of 80.4 and 72.4 percent. This accuracy improvement is much smaller than that obtained with the Bloomington, Indiana data set, and may not even be statistically significant. In the four-nearest-neighbor-context case, the information-class estimate (thresholded at 6×10^{-5} and 4×10^{-5}) produced poorer accuracies than did the spectral-class estimate.

Whether or not the information-class estimate of the context distribution produces better results than the spectral-class estimate is clearly data-set dependent. It may be that a highly variable data set, such as the Bloomington, Indiana data set, will generally produce better results using an information-class estimate. Conversely, a less variable data set, such as the LACIE data set, may generally produce better results using a spectral-class estimate. Data sets in the Tippecanoe County area of Indiana are presently being developed for tests which may confirm this result.

Unbiased Estimator. Another method for arriving at a better original estimate of the context distribution may be to use an unbiased estimator for estimating the spectral- or information-class context distribution. In all cases studied to date, estimates of the context distribution have simply been tabulated from some template classification. Simple tabulation produces a biased estimate of the context distribution. It would seem worthwhile to look into unbiased estimators of the context distribution (such as in Van Ryzin[6]) as such estimators may produce significantly better estimates of the context distribution. It is anticipated that research into this area will be completed in the coming year.

Predicting the Optimal p-Context Array. In all of the tests of the contextual classifier described so far, the size and shape of the p-context array was chosen somewhat arbitrarily. For the data sets studied, experience has generally shown that nearest-neighbors provide the most useful contextual information. When context arrays of fewer than four nearest neighbors are used, however, it is not clear which neighbors should be used. This could be a problem because the preliminary research indicated that, for the first iteration of the bootstrap method, a smaller p-context array (p = 2 or 3) was generally most effective, and which nearest neighbors were used as context seemed to make some difference in classification results.

One could discover the optimal p-context arrays at each iteration by simply performing a large number of contextual classifications over a training set. This could be quite time consuming, however. A more desirable solution would be to predict the optimal p-context array at each iteration from some "context measure" of the data before actual classifications are performed.

Suppose that the context distribution, $C(\underline{v}^p)$ is such that it can be written in product form, i.e.,

$$C(\underline{v}^p) = C_1(\underline{v}')\cdot C_2(\underline{v}'') \tag{10}$$

where $\underline{v}'$ and $\underline{v}''$ are, respectively, q and p-q vectors of classes. The elements of $\underline{v}'$ are identical to the first q elements of $\underline{v}^p$, and the elements of $\underline{v}''$ are identical to the last p-q

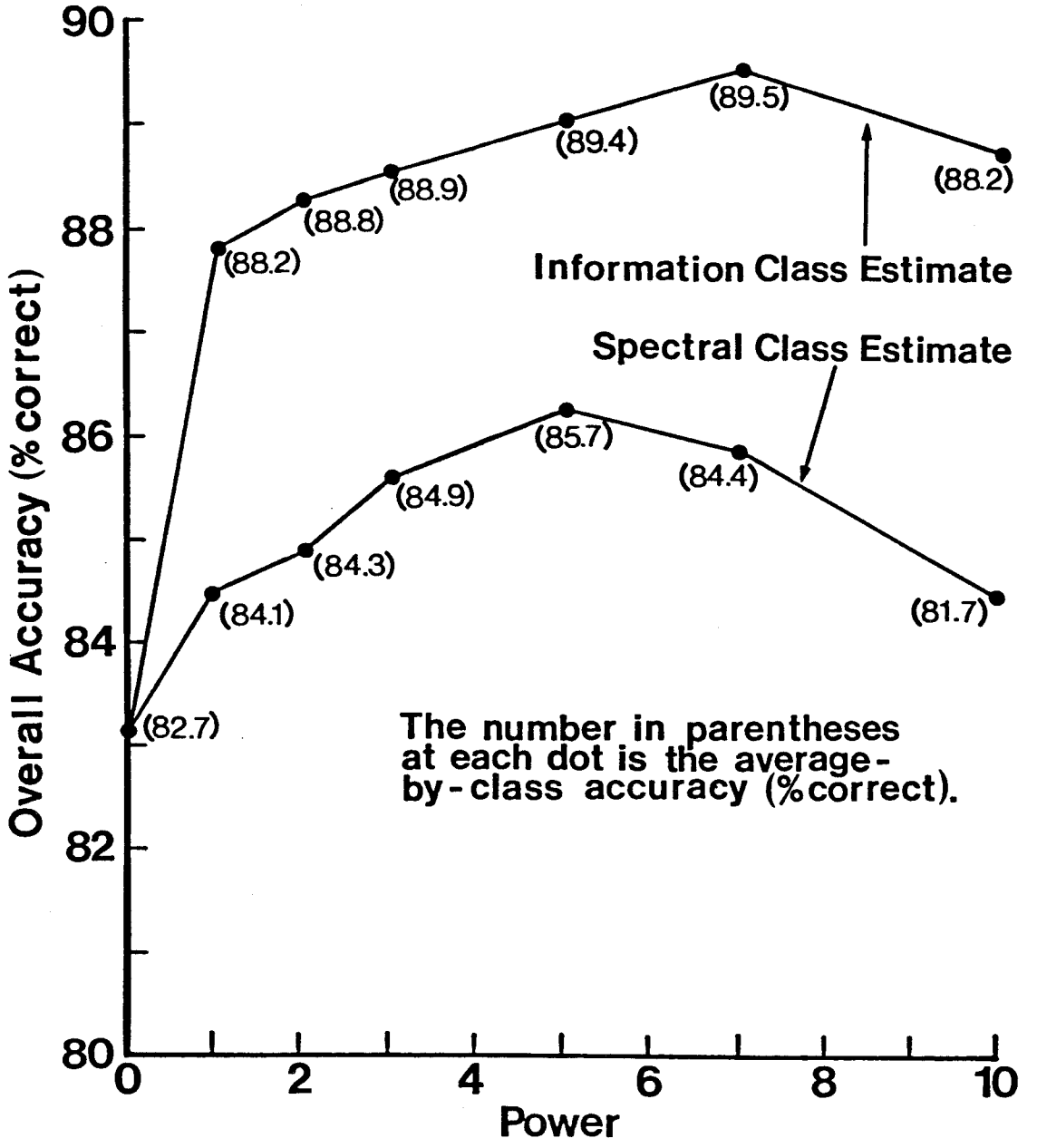


Figure C-5. Summary of four nearest neighbor context classification results from the Bloomington, Indiana data set. Here the Power Method is combined with both spectral class and information class estimates of the context distribution as tabulated from the uniform-priors no-context classification. Note that the power of zero result is equivalent to the uniform-priors no-context classification.

elements of $\underline{\vartheta}^p$. If this factorization can indeed be realized, equation (7) can be rewritten as

$$d(\underline{X}_{ij}) = \text{the action } a \text{ which maximizes}$$

$$\left[\sum_{\underline{\vartheta}' \in \Omega^q} G_1(\underline{\vartheta}') \prod_{k=1}^{q} f(X_k \mid \vartheta_k)\right] \cdot \left[\sum_{\substack{\underline{\vartheta}'' \in \Omega^{p-q}, \\ \vartheta_p = a}} G_2(\underline{\vartheta}'') \prod_{k=q+1}^{p} f(X_k \mid \vartheta_k)\right] \tag{11}$$

where the ϑ_i, i=1,2,...,p, are the elements of $\underline{\vartheta}^p$. Since the term in the first set of brackets is independent of the decision a, it is just a constant term relative to the decision process and can be ignored when classifying the point at (i,j).

If $G(\underline{\vartheta}^p)$ can be factored as in equation (10), it is clear that the distribution $G(\underline{\vartheta}^p)$ is one of independence of $\underline{\vartheta}'$ and $\underline{\vartheta}''$. This suggests that a measure of nonredundant contextual information from the pixel positions in $\underline{\vartheta}'$ as compared to that from the pixel positions in $\underline{\vartheta}''$ would be a measure of departure from independence of $\underline{\vartheta}'$ and $\underline{\vartheta}''$ in the distribution $G(\underline{\vartheta}^p)$. One measure of this departure is

$$\Delta G_q^p = \sum_{\underline{\vartheta}^p \in \Omega^p} \left(G_1(\underline{\vartheta}') \cdot G_2(\underline{\vartheta}'') - G(\underline{\vartheta}^p) \right)^2 \tag{12}$$

where $G_1(\underline{\vartheta}')$ and $G_2(\underline{\vartheta}'')$ are the marginals of $G(\underline{\vartheta}^p)$. Here the departure of the factorization of $G(\underline{\vartheta}^p)$ into its marginals from a true factorization is defined as the context measure ΔG_q^p.

To investigate the use of the context measure ΔG_q^p in predicting the optimal p-context array we use the following approach. Establish $\underline{\vartheta}''$ as a fixed (p-q)-dimensional classification vector we shall call the "core array". Note that this core array must include the pixel which is to be classified. Calculate the values of ΔG_q^p for various q-dimensional classification vectors $\underline{\vartheta}'$ with elements distinct from the core array. (Only those q-dimensional arrays that are expected to add significant contextual information need be investigated.) The best p-context array would be the (p-q) pixel locations of $\underline{\vartheta}''$ combined with the q pixel locations of the $\underline{\vartheta}'$ that produced the largest value for ΔG_q^p.

The predictor ΔG_q^p cannot be used to predict the optimal size of the p-context array for there is nothing in the theoretical development of ΔG_q^p that would indicate anything like a threshold value for ΔG_q^p. Actually, the values of ΔG_q^p are totally incomparable for different sized p-context arrays. For a fixed size p-context array, and a fixed core array, however, the relative values of ΔG_q^p can be used to indicate which data point(s) would be best to use as additional context.

ΔG_q^p was tested as a context measure to predict the best p-context array in terms of relative pixel locations as shown in Figure C-6. Usually pixel location 5 was the pixel to be classified. In some cases pixel location 1 was used as the pixel to be classified. For a more detailed presentation of test results see [5].

The first test of ΔG_q^p was performed on the simulated data with context distributions estimated by tabulation from a perfect template. A comparison of various data points as one neighbor context was performed in which ΔG_q^p did quite well in predicting classification accuracy. This test was repeated using context distributions estimated from the uniform-priors no-context classification. Here the values of ΔG_q^p did not

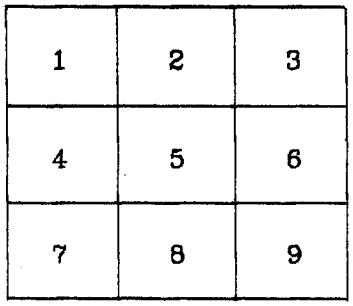


Figure C-6. Pixel locations used in testing ΔG_q^p.

correlate as closely with classification accuracy, but ΔG_q^p still did fairly well in predicting the best data point to use as one neighbor context.

Tests using the Bloomington, Indiana and LACIE data sets did not turn out as well as the simulated data tests. Here the values of the context measure ΔG_q^p did not seem to correlate with the accuracy measurements. It seems that in these real data cases, the context as estimated from the no-context classification contains enough error to confuse the context measure ΔG_q^p. For ΔG_q^p to function properly as a predictor of the optimal p-context array, the contextual information contributed by the $\underline{v}'$ pixel locations must not be so erroneous as to actually decrease classification accuracy.

Further tests with the Power Method on the two real data sets were performed to see how significant this failure of ΔG_q^p to predict some best p-context array is in these cases. Table C-1 summarizes the results of two iterations of the Power Method in which various two-neighbor contexts were used in the first iteration. Four-nearest-neighbor context was used for the second iteration.

For nearest-neighbor context, the choice of 1st-iteration context for the LACIE data set makes virtually no difference in terms of 2nd iteration accuracies. There are some differences in the Bloomington data set results. As indicated by by ΔG_q^p, the non-nearest-neighbor case (1st iteration pixel locations 3 and 7) produced a lower 2nd iteration accuracy. Comparing the values of ΔG_q^p among nearest-neighbor locations, however, gives no indication that pixel locations 4 and 8 would produce better classification results than any other combination of nearest-neighbor pixel locations. The fact that ΔG_q^p produced larger values for pixel locations 4 and 6 versus pixel locations 2 and 8 for both real data sets is not reflected at all in the accuracy results presented in Table C-1.

It should be noted that the Bloomington data set results were evaluated from just over half the pixels in the 50-pixel square scene (1317 pixels) while the LACIE data set was evaluated from ground-truth over the entire 50-pixel square scene. Also, the Bloomington data set ground truth was derived from aircraft infrared photography while the LACIE ground truth was from a ground survey. The combination of these facts may serve to make the Bloomington data set results sufficiently noisy to make the variations in the accuracies displayed in Table C-1 statistically insignificant.

Table C-1. Power Method results varying pixel locations of the two-neighbors used for first-iteration context. Classified pixel location is location 5. Second iteration uses four-nearest-neighbor context.

Data Set	1st Iteration Context Pixel Locations	Best Power 1st Iteration	Best Power 2nd Iteration	2nd Iteration Accuracy, %	
				Overall	Average-By-Class
LACIE	2 & 4	15	10	86.7	75.6
LACIE	2 & 8	15	10	86.7	75.6
LACIE	4 & 6	15	10	86.7	75.6
Bloomington	2 & 6	10	5	88.5	87.5
Bloomington	2 & 8	10	5	88.6	87.8
Bloomington	4 & 6	7	3	88.2	88.2
Bloomington	4 & 8	10	5	89.7	89.2
Bloomington	3 & 7	7	3	87.2	87.1

If indeed no one particular nearest-neighbor is better as context in these two real data cases, it remains to be explained why ΔG_q^p consistently produced a larger value for pixel locations 4 and 6 versus pixel locations 2 and 8 on both the Bloomington data set and the LACIE data set.

It is interesting to note that the Landsat sampling rate is significantly finer along the east-west direction than for the north-south direction. With such a difference in sampling rate we might expect from ΔG_q^p exactly the kind of behaviour we have observed with these two data sets.

The research reported above raises doubt as to the usefulness of ΔG_q^p as a predictor of the optimal p-context array. A closer look at the problem seems to indicate, fortunately, that such a predictor may not be needed. It may indeed be the case that for general data sets, when less than four-neighbor context is used, any particular choice of nearest neighbors as context will be as good as any other choice of nearest neighbors.

2.2 Reducing Computation Time

The contextual classifier is very computationally intensive, typically requiring a large amount of computer time. To reduce execution time, one could exploit the latest improvements in the raw speed of computer components and/or one could take advantage of special computer architectures involving multiple processing elements. This type of approach is discussed in Sections 3 through 8. Here we will limit ourselves to discussing modifications to the context classification algorithm itself aimed at reducing the computation necessary to perform a contextual classification.

Hybrid Algorithm. At least two different approaches to such a modification of the context classification algorithm are possible. One way to produce classifications of accuracy comparable to the original contextual classification algorithm but with less computation may be to use a "hybrid" algorithm. Such an algorithm would use a conventional no-context classifier whenever that classifier can classify a given point "confidently", resorting to the contextual classifier only on "difficult" pixels.

A simple test of the "confidence" of classification by a conventional no-context classifier (as in [1]) would be to compare the difference between maximum discriminant function and the next largest discriminant function for the classifier at a given point with a threshold value. If the value equals or exceeds the threshold value, the conventional classification would be used. Otherwise, the contextual classifier would be employed. Such a method should save considerable computation time depending on the percentage of pixels that are classfied conventionally. Classification accuracy should not suffer significantly because the pixels classified "confidently" by the conventional no-context classifier presumably would have been classified identically by the contextual classifier.

Approximate Algorithm. Another approach to reducing the computation required to do contextual classification is to look for a less computationally intensive algorithm which approximates the original contextual classification algorithm. Such an approximate algorithm has been studied during this contract year. For a detailed report on this approximate algorithm see [7]. If such an algorithm produces classifications that do not differ significantly in accuracy from those produced by the original algorithm, the approximate algorithm would be preferred for practical applications.

In studying the original algorithm, it was discovered that a single term of the sum in equation (7) usually dominated all other terms for each possible classification action a. This observation suggested the following approximation to the decision rule $d(\cdot)$ given in equation (7):

$$d(\underline{X}_{ij}) = \text{the action } a \text{ which maximizes, for all } \underline{\vartheta}^p \in \Omega^p \text{ with } \vartheta_p = a,$$

$$G(\underline{\vartheta}^p) \prod_{k=1}^{p} f(X_k \mid \vartheta_k). \tag{13}$$

Comparing equations (7) and (13), we see that the summation in equation (7) is eliminated. This not only saves the computer time needed to perform the summation, but also allows the use of a discriminant function proportional to (13) which is much easier to calculate. Again, see [7] for the details. Based on these considerations, we would expect that an approximate algorithm based on (13) will take less computation time than the original algorithm for any data set. However, the effect of the approximation on classification accuracy may be data dependent.

It now remains to be tested empirically whether the lower computational requirements of the approximate algorithm result in a significant savings in computer costs when compared to the original algorithm and whether the classifications produced by the approximate algorithm differ significantly from the classifications produced by the original algorithm.

The approximate algorithm was compared for accuracy with the original algorithm in tests using the simulated data set and the real data sets mentioned earlier. Included in the comparisons were algorithms that take only the three or five maximum terms in the summation in equation (7). These additional algorithms serve to give an indication of how many terms in the summation are needed to produce classifications equivalent to those produced by the original algorithm. See [7] for a detailed presentation of experimental results.

The approximate algorithm performed very well when compared to the original algorithm, with overall accuracy differing by less than 0.2 percent for all data sets. The results also show that in the two real data sets, the five largest terms of the sum in equation (7) are all that are needed to produce identical classifications to those produced by the full sum (the original algorithm).

The approximate algorithm was compared in terms of computer timings with the original algorithm on the simulated data set and the two real Landsat data sets. Highly optimized versions of each algorithm (written in the "C" programming language) were run on PDP-11/45 and PDP-11/70 computers.

The length of time the classifier takes to process a 50-pixel square data set varies depending primarily on the number of nonzero elements of the context distribution. (The number of terms that need to be evaluated in the sum in equation (7) and the number of terms to be compared in the maximization of equation (13) are equal to the number of nonzero elements in the context distribution.) In the test the ratio of timings between the two programs remained fairly consistent, however, across all data sets and for both computers. The results show that the approximate algorithm averaged less than half the real and user time* taken by the original algorithm. This amounts to a significant improvement in computation time.

Experimental results from one simulated and two real data sets show that on these data sets the approximate algorithm takes significantly less computer time while producing classifications that do not differ significantly in accuracy from classifications produced by the original algorithm. By the nature of the approximate algorithm, it is expected that similar time savings will occur when the approximate algorithm is used on other data sets. Whether or not the accuracy results presented here can be expected with other data sets depends on the extent to which the data sets tested here are representative of remotely sensed data in general. We believe that they are fairly representative. Further tests are planned to confirm that the approximation does not significantly affect classification accuracy.

---

* Real time is the time the program is running in the computer including the time the program is swapped out for other tasks. User time is essentially time spent doing computations.

2.3 Study into Initial Assumptions

Certain assumptions were made in the initial development of the context classifier (see Section 1). Probably the most important assumption was class-conditional independence for the observations as expressed by equation (6).

Elimination of Class-Conditional Independence Assumption. For contextual classifications using an arbitrary p-context array, the class-conditional density $f(\underline{X}_{ij}|\underline{v}^p)$ of equation (5) could be estimated by clustering in a similar manner to the way the densities $f(X_k|\vartheta_k)$ of equation (7) are estimated (see [1]). In this case, however, the clustering must be done over the $n \times p$ dimensional $\underline{X}_{ij}$ rather than the n-dimensional X_k. Significant clusters of the observation vectors, $\underline{X}_{ij}$, could then be identified with a particular classification vector, $\underline{v}^p$, and a normal density model for $f(\underline{X}_{ij}|\underline{v}^p)$ could be determined. Clustering done in such a way would provide class-conditional densities $f(\underline{X}_{ij}|\underline{v}^p)$ without the independence assumption for use in comparison to classifier tests using class-conditional densities assumed to be independent among all image locations.

The use of the class-conditional density $f(\underline{X}_{ij}|\underline{v}^p)$ presents the practical problem of effectively working with a multispectral data set with a large number of channels. Some of the dimensionality reduction techniques used in working with other large-dimensioned data sets may also be useful in this case.

Research into the class-conditional independence assumption is planned for later in the coming year.

Classification Over Information Classes. Not explicitly pointed out in Section 1 is the assumption that results in the original context classifier classifying into spectral classes rather than information classes. Since classification results are normally evaluated with respect to information classes rather than spectral classes, it may prove beneficial to classify directly over information classes. Scholz et al [8] reported a small improvement in classification accuracy in certain cases where a maximum likelihood no-context classification was done directly into information classes rather than into spectral classes. It would be of interest to test if classifying over information classes would significantly improve the performance of the contextual classifier.

Let the set $\Omega=\{\omega_1,\omega_2,\ldots,\omega_m\}$ represent spectral classes and the set $\Gamma=\{\gamma_1,\gamma_2,\ldots,\gamma_n\}$, $n \leq m$, represent information classes. Note that each element of Γ is a set of spectral classes where the γ_i are such that if $\omega_i \in \gamma_j$ then $\omega_i \notin \gamma_k$ for $k \neq j$ and $\bigcup_{j=1}^{n} \gamma_j = \Gamma$.

Where the possible actions are defined over information classes and the contextual information is defined in terms of spectral classes, the decision rule is obtained by maximizing a function as in equation (5) summed over the spectral classes contained in the action (information class) considered. Invoking the class-conditional independence assumption as in equation (7), the decision rule becomes:

$$d(\underline{X}_{ij}) = \text{the action } a \in \Gamma \text{ which maximizes}$$

$$\sum_{v \in a} \sum_{\substack{\underline{v}^p \in \Omega^p, \\ \underline{v}_p = v}} G(\underline{v}^p) \prod_{k=1}^{p} f(X_k|\vartheta_k) \quad . \tag{14}$$

where the σ are the spectral classes comprising information class α. This decision rule can also be used in the case where information classes are assumed to carry all of the contextual information by using equation (8) to calculate $G(\underline{v}^{(p)})$.

The information-class decision rule was compared to the spectral-class decision rule using a simulated data set and the two real Landsat data sets. Table C-2 lists the results obtained using the simulated data set and spectral-class context. Results were similar for the two real data sets and when information-class context was used.

Table C-2. Comparison of classification results from using the information-class decision rule of equation (14) versus results from using the spectral-class decision rule of equation (7). Simulated data and spectral-class context were used.

Classification	Accuracy, %			
	Information Class Rule		Spectral Class Rule	
	Overall	Ave-by-Class	Overall	Ave-by-Class
uniform-priors, no-context	72.1	78.2	70.4	77.5
estimated-priors, no-context	87.8	65.6	87.5	65.4
2-nearest-neighbors (west and north)	93.2	78.5	93.0	78.4
4-nearest-neighbors	97.1	87.5	97.1	87.5
8-nearest-neighbors	98.2	92.0	98.2	92.0

The results indicate that while there may be some advantage in using the information class decision rule when classification accuracies are low, there is apparently no advantage to using the information class decision rule when classification accuracies are high. We would actually be disadvantaged by using the information class rule if we would want to use an iterative process using spectral class context, for we would not have a spectral class classification for use in continuing the iterative process.

3. Examples of Contextual Classifiers

This section reviews the theory presented in Section 1 and gives examples of the calculations needed to perform contextual classifications. This will serve as background for the implementations sections that follow.

The image data to be classified are assumed to be a two-dimensional I-by-J array of multivariate pixels. Associated with the pixel at "row i" and "column j" is the multivariate measurement n-vector $X_{ij} \in R^n$ and the true class of the pixel $\theta_{ij} \in \Omega = \{\omega_1,\ldots,\omega_C\}$. The measurements have class-conditional densities $f(X|\omega_k)$, $k = 1,2,\ldots,C$, and are assumed to be class-conditionally independent. The objective is to classify the pixels in the array.

In order to incorporate contextual information into the classification process, when each pixel is to be classified, p-1 of its neighbors are also examined. This neighborhood, including the pixel to be classified, will be referred to as the p-array. Intuitively, to classify each pixel, the contextual classifier computes the probability of the given observed pixel being in class k by also considering the measurement vectors (values) observed for the neighbor pixels in the p-array. Specifically, for each pixel, for each class in Ω, a discriminant function g is calculated. The pixel is assigned to the class for which g is greatest. Each value of g is computed by summing the weighted probabilities of the p-1 neighbor pixels occurring in all possible classification states. This is described below mathematically for pixel (i,j) being in class ω_k.

The description is followed by an example to clarify the notation used. Further details may be found in [5,9-11].

$$g_k(\underline{X}_{ij}) = \sum_{\substack{\underline{\theta}_{ij} \in \Omega^p, \\ \theta_{ij} = \omega_k}} \left[\prod_{\ell=1}^{p} f(X_\ell|\theta_\ell) \right] G^p(\underline{\theta}_{ij})$$

where

$X_\ell \in \underline{X}_{ij}$ is the measurement vector from the ℓth pixel in the p-array (for pixel (i,j))

$\theta_\ell \in \underline{\theta}_{ij}$ is the class of the ℓth pixel in the p-array (for pixel (i,j))

$f(X_\ell|\theta_\ell)$ is the class-conditional density of X_ℓ given that the ℓth pixel is from class θ_ℓ

$G^p(\underline{\theta}_{ij}) = G^p(\theta_1,\theta_2,\ldots,\theta_p)$ is the a priori probability of observing the p-array $\theta_1,\theta_2,\ldots,\theta_p$.

Within the p-array, the pixel locations may be numbered in any convenient but fixed order. The joint probability distribution G^p is referred to as the context distribution. The class-conditional density of pixel measurement vector x given that the pixel is from class k is:

$$f(x|k) = e^{-[\log|\Sigma_k| + (x-m_k)^T \Sigma_k^{-1} (x-m_k)]/2}$$

where the measurement vector for each pixel is of size four, Σ_k^{-1} is the inverse covariance matrix for class k (four-by-four matrix), m_k is the mean vector for class k (size four vector), "T" indicates the transpose, and "log" is the natural logarithm. This is the same function as used for maximum likelihood classification [1].

To clarify the computation of the discriminant function, consider the following example. Let the context array (neighborhood) be the p=3 choice shown in Figure C-7. This type of neighborhood is called horizontally linear[10]. As shown in the figure, the pixels are numbered such that the pixel (i,j) to be classified is associated with X_2 and θ_2, pixel (i,j-1) is associated with X_1 and θ_1, and pixel (i,j+1) is associated with X_3 and θ_3. Assume there are two possible classes: $\Omega = \{a,b\}$. Then the discriminant function for class b is explicitly:

$$g_b(\underline{X}_{ij}) = \sum_{\substack{\underline{\theta}_{ij} \in \Omega^3, \\ \theta_2 = b}} \prod_{\ell=1}^{3} f(X_\ell|\theta_\ell) \quad G^3(\underline{\theta}_{ij})$$

$$= f(X_1|a)f(X_2|b)f(X_3|a)G(a,b,a)$$

$$+ f(X_1|a)f(X_2|b)f(X_3|b)G(a,b,b)$$

$$+ f(X_1|b)f(X_2|b)f(X_3|a)G(b,b,a)$$

$$+ f(X_1|b)f(X_2|b)f(X_3|b)G(b,b,b)$$

(Recall $G^3(\underline{\theta}_{ij}) = G(\theta_1, \theta_2, \theta_3)$ is the a priori probability of observing the specific neighborhood configuration $(\theta_1, \theta_2, \theta_3)$.) After computing the discriminant functions of g_a and g_b for pixel (i,j), pixel (i,j) is assigned to the class which has the larger discriminant value.

Consider the case where there is a non-linear three-by-three context array (neighborhood), i.e., p=9. This is shown in Figure C-8. The pixels are numbered such that pixel (i,j) is the pixel to be classified and it is numbered as pixel 5 in the neighborhood. The indices and numbers of the other pixels are shown in Figure C-8. This numbering will be used to demonstrate the computations required by the general formula above.

1 (i,j-1)	2 (i,j)	3 (i,j+1)

Figure C-7. A p=3 context array (neighborhood). The number above the row and column indices is the pixel number for that position in the neighborhood.

1 (i-1,j-1)	2 (i-1,j)	3 (i-1,j+1)
4 (i,j-1)	5 (i,j)	6 (i,j+1)
7 (i+1,j-1)	8 (i+1,j)	9 (i+1,j+1)

Figure C-8. Three-by-three neighborhood for classifying pixel (i,j). The number above the row and column indices is the pixel number for that position in the neighborhood.

For the sake of a simple example, assume there are only two classes: $\Omega = \{a,b\}$. Let the pixel number ℓ be associated with X_ℓ and θ_ℓ, $1 \leq \ell \leq 9$. Then the discriminant function for pixel (i,j) being in class b is explicitly:

$$g_b(\underline{X}_{ij}) = \sum_{\substack{\theta_{ij} \in \Omega^9 \\ \theta_5 = b}} \left[\prod_{\ell=1}^{9} f(X_\ell | \theta_\ell) \right] G^9(\underline{\theta}_{ij}).$$

This is shown in its expanded form in Figure C-9.

As is demonstrated by the figure, the number of product terms is 2^8. In general, the number of product terms will be C^{p-1} where C is the number of classes being considered and p is the size of the neighborhood. This must be repeated for each class for each pixel to be classified. Thus theoretically, $C*C^{p-1} = C^p$ of these p+1 element products must be computed for each pixel. Typically $10 \leq C \leq 60$ for the analysis of Landsat data. This demonstrates the potential computation complexity of contextual classification.

In the following sections, uniprocessor, CDC Flexible Processor system, and SIMD algorithms for the size three linear and size nine non-linear neighborhood classifiers will be presented. The improvement in the execution speed of the algorithms provided by the Flexible Processor system and SIMD approaches will be discussed.

4. Uniprocessor Contextual Classifiers

This section presents high-level language algorithms for executing the size three and size nine neighborhood contextual classifiers that were described in the previous section. These uniprocessor algorithms will be used as a basis for the multiprocessor algorithms that follow.

Algorithm 1, shown in Figure C-10, is one way to implement the contextual classifier using a horizontally linear neighborhood of size three which was described in Section 3. This implementation is a straightforward approach presented to clarify the computations necessary. After presenting this algorithm, a second, more efficient algorithm will be given.

First consider the main loop of Algorithm 1, shown in Figure C-10. Let the original image to be classified be an I-by-J array called A. Columns 0 and J-1, the two side edges of the image, are not classified since these pixels will not have both right and left neighbors. The variable "value" will contain the maximum "g" (discriminant function)

$$f(X_1|a)f(X_2|a)f(X_3|a)f(X_4|a)f(X_5|b)f(X_6|a)f(X_7|a)f(X_8|a)f(X_9|a)G(a,a,a,a,b,a,a,a,a)$$

$$+\ f(X_1|a)f(X_2|a)f(X_3|a)f(X_4|a)f(X_5|b)f(X_6|a)f(X_7|a)f(X_8|a)f(X_9|b)G(a,a,a,a,b,a,a,a,b)$$

$$+\ f(X_1|a)f(X_2|a)f(X_3|a)f(X_4|a)f(X_5|b)f(X_6|a)f(X_7|a)f(X_8|b)f(X_9|a)G(a,a,a,a,b,a,a,b,a)$$

$$+\ f(X_1|a)f(X_2|a)f(X_3|a)f(X_4|a)f(X_5|b)f(X_6|a)f(X_7|a)f(X_8|b)f(X_9|b)G(a,a,a,a,b,a,a,b,b)$$

$$\vdots$$

$$+\ f(X_1|b)f(X_2|b)f(X_3|b)f(X_4|b)f(X_5|b)f(X_6|b)f(X_7|b)f(X_8|b)f(X_9|a)G(b,b,b,b,b,b,b,b,a)$$

$$+\ f(X_1|b)f(X_2|b)f(X_3|b)f(X_4|b)f(X_5|b)f(X_6|b)f(X_7|b)f(X_8|b)f(X_9|b)G(b,b,b,b,b,b,b,b,b)$$

Figure C-9. The calculations required for the discriminant $g_b(\underline{X}_{ij})$ for a size nine neighborhood and two classes. Note that the "..." represents $2^8 - 6 = 250$ product terms.

Main Loop for Algorithm 1

```
/* straightforward uniprocessor implementation of a contextual
   classifier using a size three horizontally linear neighborhood */
for i = 0 to I-1 do      /* row index */
    begin
    for j = 1 to J-2 do /* column index */
        begin            /* for each pixel */
        value = -1       /* initialize variable to hold max "g" */
        class = -1       /* initialize variable for class with max "g" */
        for k = 1 to C do  /* for each class */
            begin
            current = g(i,j,k)  /* compute "g" for pixel i,j and
                                   class k */
            if current > value  /* compare new "g" with max */
                then
                    begin
                    value = current  /* update max */
                    class = k        /* update max class */
                    end
            end
        print pixel (i,j) is classified as "class"
        end
    end
```

Figure C-10(a). Main loop of Algorithm 1, a straightforward uniprocessor implementation of a contextual classifier for a horizontally linear neighborhood of size three.

Discriminant Function Calculation

```
function g(i,j,k)  /* discriminant for pixel (i,j) and class k */
sum = 0  /* initialize sum, used to accumulate g(i,j,k)  */
for r = 1 to C do  /* all possible classes for pixel (i,j-1)  */
    begin
    for q = 1 to C do  /* all possible classes for pixel (i,j+1)  */
        begin
        sum = compf(i,j-1,r) * compf(i,j,k)
              * compf(i,j+1,q) * G(r,k,q) + sum
        end
    end
return (sum)  /* sum contains value of g(i,j,k)  */
```

Class-Conditional Density Calculation

```
function compf(a,b,k)  /* for pixel (a,b), class k  */
x = A(a,b)  /* x is the pixel (a,b) measurement vector  */
expo = -[log|Σ_k| + (x-m_k)^T Σ_k^-1 (x-m_k)]/2
return (e^expo)  /* return value of f(A(a,b)|k)  */
```

Figure C-10(b). Algorithm 1 subroutines "g(i,j,k)" and "compf(a,b,k)."

value calculated for pixel (i,j). This variable may be updated as the "g" for each class calculated. The variable "class" is the class associated with "value." In the main loop, "g(i,j,k)" is a call to a function to calculate the discriminant function for pixel (i,j) and class k. This function is called I * (J-2) * C times, once for each class for each pixel being classified.

Consider the calculation of g(i,j,k), shown in Figure C-10. The class of pixel (i,j) is held constant at k, while all other possible class assignments are considered for pixels (i,j-1) and (i,j+1). For each assignment of classes for the pixels neighboring pixel (i,j), of which there are C*C, the product of the class-conditional densities ("compf") is weighted by "G(r,k,q)", the a priori probability of observing the 3-array $(\omega_r,\omega_k,\omega_q)$. The "G" array is predetermined and prestored. For each call to "g(i,j,k)" the value of "sum" for that i, j, and k is calculated. "Sum" is then returned as the value of "g(i,j,k)." In this straightforward version of the g(i,j,k) routine, the function to compute a class-conditional density ("compf") is called C*C times each time "g" is called.

Now consider the "compf" routine. This calculates the class-conditional density for pixel (a,b) and class k using the following equation:

$$f(x|k) = e^{-[\log|\Sigma_k|+(x-m_k)^T\Sigma_k^{-1}(x-m_k)]/2}$$

where x is the measurement vector for pixel (a,b) and is of size four, Σ_k^{-1} is the inverse covariance matrix for class k (four-by-four matrix), m_k is the mean vector for class k (size four vector), "T" indicates the transpose, and "log" is the natural logarithm. For each class, the algorithm uses $\log|\Sigma_k|$, Σ_k^{-1}, and m_k as precomputed constants. For each call to "compf (a,b,k)," the value of "e^{expo}" for that a, b, and k is calculated. "e^{expo}" is then returned as the value of "compf(a,b,k)."

Algorithm 1 executes the "compf" subroutine I * (J-2) * C^3 times. Since for each pixel there are C "f"s (class-conditional densities), this approach is inefficient by a factor of C^2. Algorithm 2 rectifies this problem by saving certain "f" values rather than recalculating them.

Algorithm 2, shown in Figure C-11, implements the contextual classifier without the redundant executions of "compf" that occur in Algorithm 1. Let "hold(m,k)" be a two-dimensional array of size three-by-C. i.e., $0 \leq m \leq 2$ and $1 \leq k \leq C$. For m=cr, hold(cr,k) is a vector of length C containing the class-conditional density values ("compf"s) for the pixel (i,j) ("cr" is an abbreviation for center). For example, hold(cr,1) is the class-conditional density for pixel (i,j) and class 1. hold(lt,k) and hold(rt,k) are the analogous vectors for the pixel (i,j-1) (the left ("lt") neighbor) and pixel (i,j+1) (the right ("rt") neighbor), respectively. By using this array to save the class-conditional densities, each density (for a given pixel and class) is calculated only once, instead of C^2 times.

Main Loop for Algorithm 2

```
/* Algorithm 1 without redundant "compf" calculations */
for i = 0 to I-1 do /* row index */
    for k = 1 to C do /* for each class */
        for m = 0 to 2 do hold(m,k) = compf(i,m,k) /* cols.0-2 */
    lt = 0 /* hold(lt,k) is left neighbor */
    cr = 1 /* hold(cr,k) is pixel being classified */
    rt = 2 /* hold(rt,k) is right neighbor */
    for j = 1 to J-2 do /* column index */
        value = -1; class = -1 /* max "g" and class */
        for k = 1 to C do /* for each class */
            /* "g" for pixel i,j class k */
            current = g'(lt,cr,rt,k)
            if current > value /* compare with max */
                then value = current; class = k
        print pixel (i,j) is classified as "class"
        if j ≠ J-2 then /* not last column */
            /* update hold pointers */
            tp = lt; lt = cr; cr = rt; rt = tp
            for k = 1 to C do /* compf's for next col */
                hold(rt,k) = compf(i,j+2,k)
```

Figure C-11(a). Main Loop of Algorithm 2.

Discriminant Function Calculation for Algorithm 2

```
function g'(lt,cr,rt,k)
        /* discriminant for pixel "cr" (whose neighbors
           are "lt" and "rt") and class k */
sum = 0  /* initialize sum, used to accumulate g'(lt,cr,rt,k) */
for r = 1 to C do  /* all possible classes for pixel (i,j-1) */
    begin
    for q = 1 to C do  /* all possible classes for pixel
                          (i,j+1) */
        begin
        if G(r,k,q) ≠ 0  /* do not do multiplications if G = 0 */
            then
            sum = hold(lt,r) * hold(cr,k) * hold(rt,q)
                  * G(r,k,q) + sum
        end
    end
```

Figure C-11(b). Algorithm 2 subroutine "g'(lt,cr,rt,k)."

The main loop of Algorithm 2 is modified to calculate the class-conditional densities for the first three columns each time a new row is considered (i.e., each time "i" is incremented). Each time a new pixel in a given row is to be classified (i.e., just before "j" is incremented), these values are updated. In particular, the "cr" vector becomes the "lt" vector, the "rt" vector becomes the "cr" vector, and a new "rt" vector is computed.

The new discriminant function calculation, g', does not call the subroutine "compf." It gets the values it needs from the "hold" array. For each call to "g'(lt,cr,rt,k)," the value of "sum" for that k is calculated. "Sum" is then returned as the value of "g'(lt,cr,rt,k)."

In the function "g(i,j,k)" for Algorithm 1, the computation of "sum" involved $3*C^2$ floating point multiplications and C^2 floating point additions. To reduce the number of floating point operations in "g'(lt,cr,rt,k)" for Algorithm 2, the sum is updated only if "G(r,k,q)" is non-zero.

The same "compf" routine is used for both Algorithms 1 and 2. Algorithm 1 calls this routine $I*(J-2)*C^3$ times, while Algorithm 2 calls it only I*(J-2)*C times.

The serial complexity of Algorithm 2 can be calculated in terms of assignment statements, multiplications, additions, and "compf" calculations. To set "hold" for new rows, 3*I*(C+1) assignments and I*C*3 calls to "compf" occur. For each pixel, at most C+1 assignments to "value" and "class" occur, C assignments to "current" occur, and C calls to "g'(lt,cr,rt,k)" occur. In addition, for each row, "hold" is updated J-3 times, each update using C+4 assignments and C calls to "compf." Each execution of "g'(lt,cr,rt,k)" uses $3*C^2$ multiplications, C^2 additions, and C^2+1 assignments. Thus, the total complexity for Algorithm 2 is:

$I(J(C^3+5C+6)-2C^3-8C-13)$	assignments;
$3C^3I(J-2)$	multiplications;
$C^3I(J-2)$	additions; and
$I*J*C$	"compf" calculations.

The growth is proportional to $I*J*C^3$ assignments, multiplications, and additions, and I*J*C "compf" calculations.

Algorithm 2 can be extended for a non-linear contextual classifier with a neighborhood of size nine (as shown in Figure C-8). The complexity of the algorithm would have growth proportional to $I*J*C^9$ assignments, multiplications, and additions. The number of "compf" calculations would still be I*J*C.

For the size nine square neighborhood case, "hold" would be a (2*J+3)-by-C array (assuming the neighborhood window moves along rows). The "C" term is for holding the C "compf" values that are calculated for

a pixel. The 2*J+3 pixels whose "compf" values are stored in "hold" are chosen to make it unnecessary to perform redundant "compf" calculations. In general, when classifying pixel (i,j), "hold" has the "compf" values for pixels j-1 to J-1 of row i-1, pixels 0 to J-1 (all of) row i, and pixels 0 to j+1 of row i+1. After the classification of pixel (i,j), the values for pixel (i-1,j-1) are removed from "hold" and values for (i+1,j+2) are added. When the pixels on a new row are to be classified, call it i', then the values for pixels (i'-2,J-3), (i'-2,J-2), and (i'-2,J-1) are removed and the values for (i'+1, 0), (i'+1, 1) and (i'+1, 2) are added. (This assumes row i is classified after i -1.) If J > I, then moving the neighborhood window along columns would save space, since "hold" would then be of size (2*I+3)C. Given this, the rest of transforming Algorithm 2 for the size nine square neighborhood case is straightforward.

In summary, Algorithm 2 for implementing a contextual classifier based on horizontally linear neighborhood of size three using a uniprocessor system was presented and its computational complexity analyzed. The extension of Algorithm 2 to implementing a contextual classifier based on a non-linear square neighborhood of size nine, using a uniprocessor was discussed. In the following sections, the ways in which the CDC Flexible Processor system and an SIMD machine can be used to reduce the execution time of these classifiers is examined.

5. The CDC Flexible Processor System

The Control Data Corporation Flexible Processor system is a commercially available multiprocessor system which has been recommended for use in remote sensing[12-14]. In this section, the system is briefly overviewed.

The basic components of a Flexible Processor (FP) are shown in Figure C-12. An example of the way in which N FPs may be configured into a system is shown in Figure C-13. There can be up to 16 FPs linked together, providing much parallelism at the processor level. The FPs can communicate among themselves through the high-speed ring or shared bulk memory. The clock cycle time of each FP is 125nsec (nanoseconds). Since 16 FPs can be connected in a parallel and/or pipelined fashion, the effective throughput can be drastically increased, resulting in a potential effective cycle time of less than 10nsec.

An FP is microprogrammed in micro-assembly language, allowing parallelism at the instruction level. For example, it is possible to conditionally increment an index register, do a program jump, multiply two 8-bit integers, and add two 32-bit integers -- all simultaneously. This type of operational overlap, in conjunction with the multiprocessing capability of the FPs, greatly increases the speed of the FP array.

The following list summarizes the important architectural features of an FP:

DATA PATHS IN A FLEXIBLE PROCESSOR

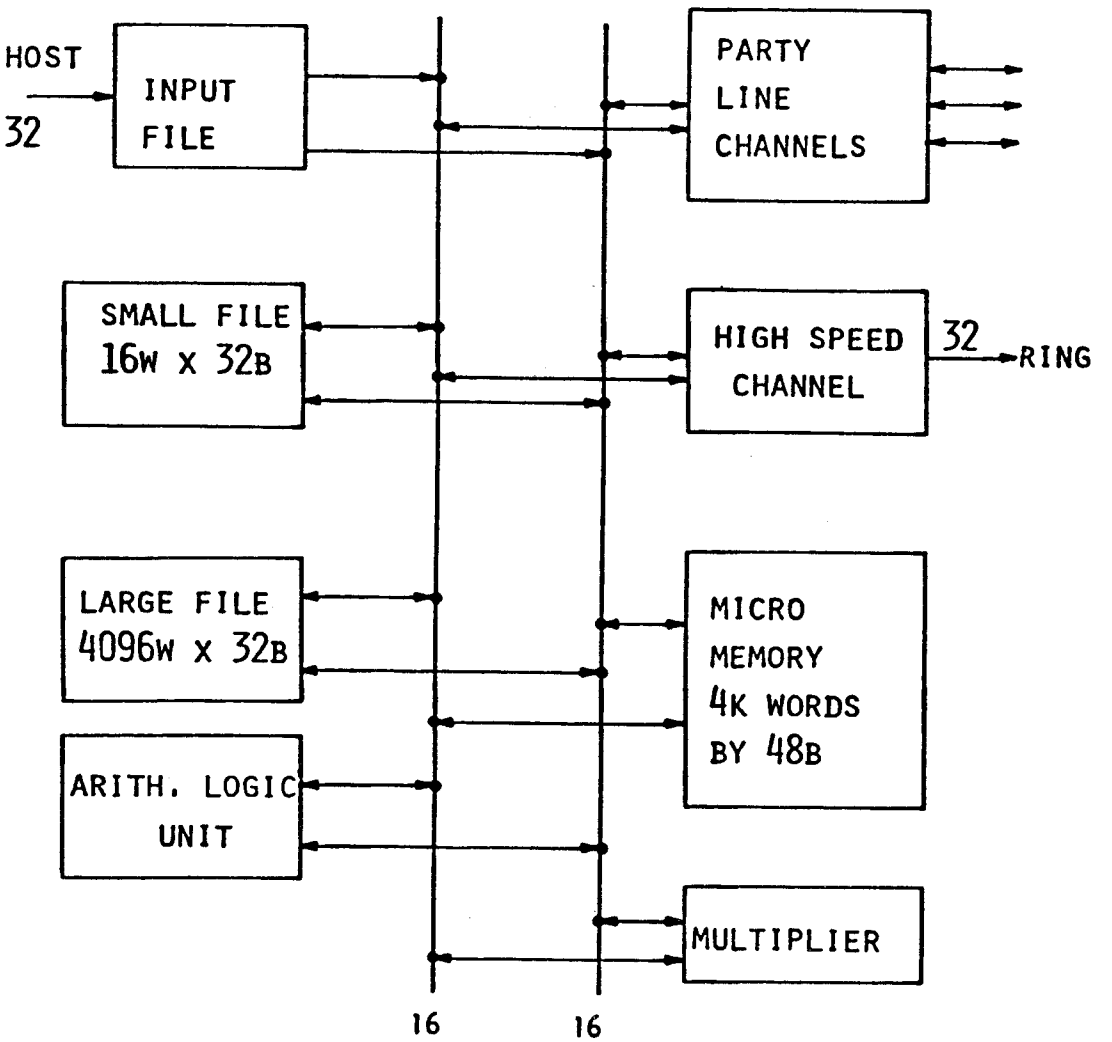


Figure C-12. Data path organization in the CDC Flexible Processor.

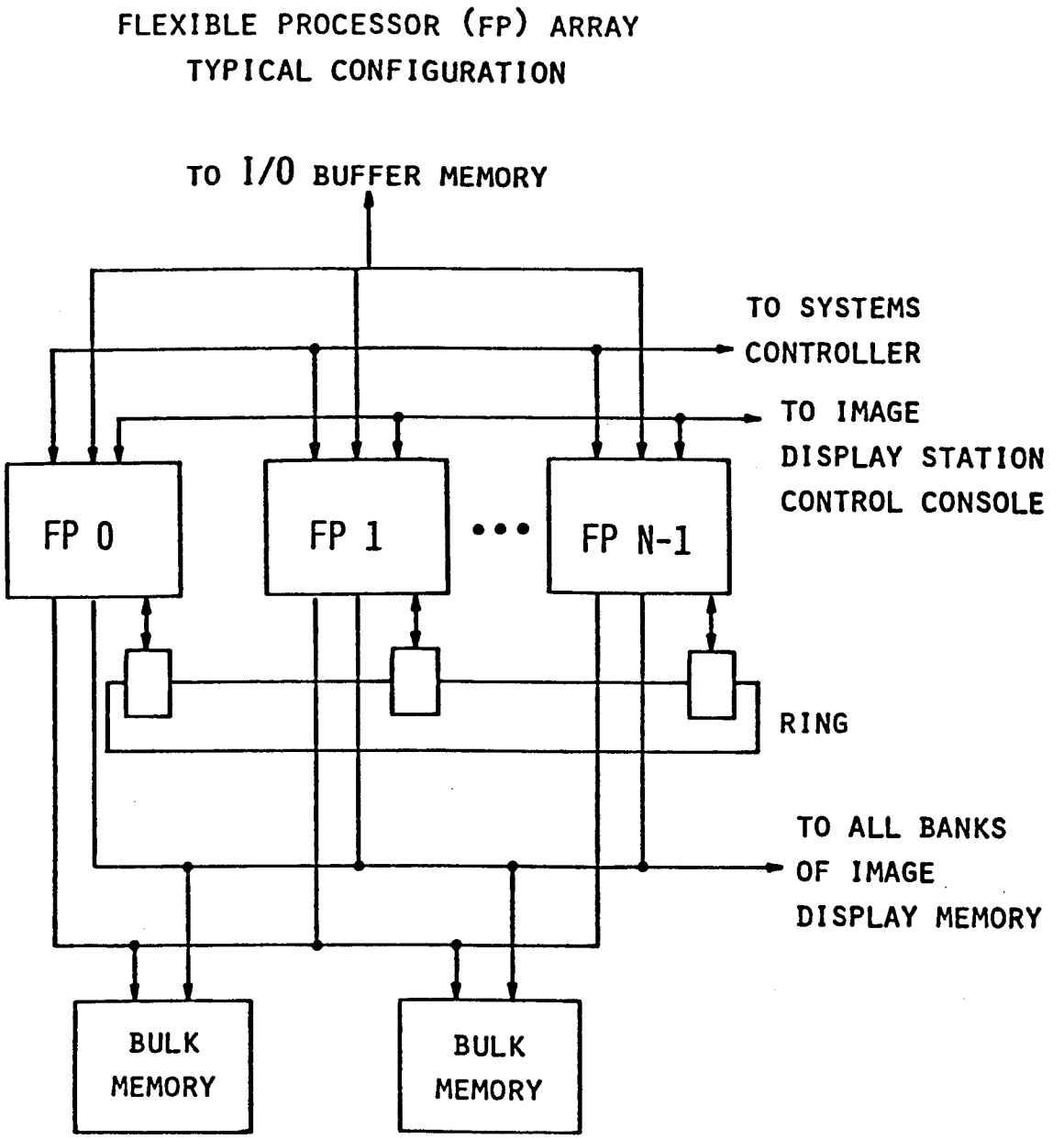


Figure C-13. Block diagram of typical Flexible Processor array.

User microprogrammable.
Dual 16-bit internal bus system.
Able to operate with either 16- or 32-bit words.
125nsec clock cycle.
125nsec time to add two 32-bit integers.
250nsec time to multiply two 8-bit integers.
Register file (with 60nsec access time) of over 8,000 16-bit words.

In order to debug, verify, and time FP algorithms, we have developed a simulator for an array of up to 16 FPs. We have also developed an assembler for the micro-assembly language. The assembler and simulator run under the UNIX operating system at Purdue. They are described in [15,16]. Their use in programming and executing a maximum likelihood classifier is discussed in [10,16].

The experience gained through the use of the simulator has made evident the following advantages and disadvantages of the system.

Advantages:

Multiple processors (up to 16).
User microprogrammable -- parallelism at the instruction level.
Connection ring for inter-FP communications.
Shared bulk memory units.
Separate arithmetic logic unit and hardware multiply.

Disadvantages:

No floating-point hardware.
Micro-assembly language -- difficult to program.
Program memory limited to 4k micro-instructions.

This brief overview was provided as background for the following section. Further details can be found in the references mentioned.

6. Flexible Processor System Implementation of Contextual Classifiers

Consider using a Flexible Processor system to implement the contextual classifier based on a horizontally linear neighborhood of size three (as shown in Figure C-7). Divide the A-by-B image into subimages of B/N rows A pixels long, as shown in Figure C-14. Assign each subimage to a different FP. The entire neighborhood of each pixel is included in its subimage. Each FP can therefore execute the uniprocessor algorithm presented in Section 4 on its own subimage. No interaction between FPs is needed, i.e., each FP can process its subimage independently.

An FP micro-assembly language version of Algorithm 2 (Figure C-11) has been coded and debugged. The LARS Flexible Processor micro-assembler

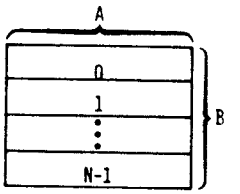


Figure C-14. An A-by-B image divided among N Flexible Processors.

Figure C-15. Horizontally linear neighborhoods. Each box is one pixel.

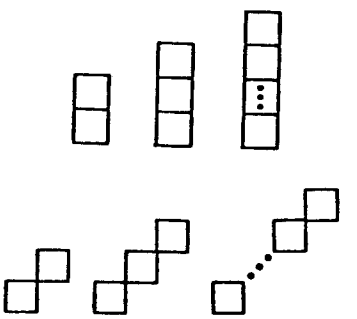


Figure C-16. Vertically linear and diagonally linear neighborhoods. Each box is one pixel.

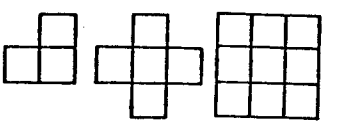


Figure C-17. Non-linear neighborhoods. Each box is one pixel.

and simulator are being used to gather statistics on the execution time for the size three horizontally linear neighborhood contextual classifier. Because each Flexible Processor is microprogrammable, determining program correctness and analyzing execution times is done through the use of the micro-assembler and simulator. The current implementation of the contextual classifier uses 780 microprogram instructions, stored in the micro memory (see Figure C-12). Execution times per pixel vary because all floating point operations are done in the software. The classification time associated with the first pixel on a line is different from the classification of the rest of the pixels on the same line. This difference is accounted for by the three-pixel window. Data must be calculated for each of the pixels in the window for the first pixel on the line while for the rest, data must be calculated for only one pixel.

The format of the data words of the pixel measurement vectors, covariance matrices, etc., consists of a 14-bit two's complement exponent and a 17-bit sign-magnitude mantissa. The covariance matrices, logarithms of the determinants of the covariance matrices, a priori probabilities (G^p), and the hold array are all stored in the Large File (see Figure C-12). In this way, each FP has all the information it needs for performing the classification on its subimage.

The contextual classifier requires a matrix multiplication of the form:

$$(X-M)^T \quad (C) \quad (X-M),$$

where X-M is the normalized data vector, and C is a symmetric matrix. Consider the standard method of matrix multiplication.

$$\begin{bmatrix} X_1 & X_2 & X_3 & X_4 \end{bmatrix} \times \begin{bmatrix} C_1 & C_2 & C_4 & C_7 \\ C_2 & C_3 & C_5 & C_8 \\ C_4 & C_5 & C_6 & C_9 \\ C_7 & C_8 & C_9 & C_{10} \end{bmatrix} \times \begin{bmatrix} X_1 \\ X_2 \\ X_3 \\ X_4 \end{bmatrix}$$

An intermediate result is:

$$\begin{bmatrix} X_1 & X_2 & X_3 & X_4 \end{bmatrix} \times \begin{bmatrix} C_1X_1+C_2X_2+C_4X_3+C_7X_4 \\ C_2X_1+C_3X_2+C_5X_3+C_8X_4 \\ C_4X_1+C_5X_2+C_6X_3+C_9X_4 \\ C_7X_1+C_8X_2+C_9X_3+C_{10}X_4 \end{bmatrix}$$

From the intermediate result, it can be seen that this matrix multiplication requires 20 floating point multiplies and 15 floating point additions.

Carrying the matrix multiplication to its conclusion and grouping-like terms yields:

$$C_1X_1^2+2C_2X_1X_2+2C_4X_1X_3+2C_7X_1X_4+C_3X_2^2$$

$$+2C_5X_2X_3+2C_8X_2X_4+C_6X_3^2+2C_9X_3X_4+C_{10}X_4^2$$

Factoring out like terms yields:

$$X_1[C_1X_1+2C_2X_2+2C_4X_3+2C_7X_4]$$

$$+X_2[C_3X_2+2C_5X_3+2C_8X_4]$$

$$+X_3[C_6X_3+2C_9X_4]+C_{10}X_4^2$$

This implementation requires 14 floating point multiplies and 9 floating point adds in addition to 6 fixed multiplies requiring one step. The total savings in process calls is about 30%. Since the floating point operations are done in the software and not in the hardware, it is important that these are as efficient as possible.

In the Landsat data used in the testing described in [11], the percentage of a priori probabilities (G^9s) that were non-zero was about 1%. Thus, most of the G^9s that are stored are zeroes. The memory requirements of the classifier can be reduced greatly if the zero values are ignored. One method for the "data compaction" would be useful if less than 50% of the data were non-zero. Consider the following format for a neighborhood of size three.

Large File

exponent of G(0,0,0)	mantissa of G(0,0,0)
exponent of G(0,0,1)	mantissa of G(0,0,1)
:	:
exponent of G(0,0,C)	mantissa of G(0,0,C)
exponent of G(0,1,0)	mantissa of G(0,1,0)
:	:
exponent of G(0,1,C)	mantissa of G(0,1,C)
:	:

```
exponent of G(0,C,C)      mantissa of G(0,C,C)
        :                         :
exponent of G(C,C,C)      mantissa of G(C,C,C)
```

This is the current format for the data to be stored. However, if the data are stored as follows:

```
                    Large File
          First non-zero r,k,q
exponent of G(r,k,q)      mantissa of G(r,k,q)
         Second non-zero r,k,q
exponent of G(r,k,q)      mantissa of G(r,k,q)
         Third non-zero r,k,q
exponent of G(r,k,q)      mantissa of G(r,k,q)
```

then the data space can be greatly compressed. Each entry is a pair consisting of class specification of the neighborhood and the a priori probability corresponding to that configuration of classes. The three values r, k, and q must be stored because zero values for G(r,k,q) are not stored. Thus it is impossible to determine the value of r, k, and q through indexing. Since the total pathwidth of the Large File is 32 bits, r, k, and q can occupy a total of up to 32 bits. Because r, k, and q must be the same size (the $\lceil \log_2 C \rceil$), each can be up to 10 bits. This allows for up to 1024 classes per pixel. For a window of size nine with eight-nearest neighbors, this would allow for eight classes per pixel. The test area in [11] was classified with an eight-nearest neighbor window with 14 classes. The total number of non-zero a priori probabilities was 2134. When compared with 14^9, which is the number of distinct types of neighborhoods, this is much less than 1%. Thus, this implementation will save space. The cost of such an operation is minimal, as (in the size three case) the r,k,q (neighborhood class specification) can be used as the index variables, which will actually save time in the form of memory fetches. This will be demonstrated later.

Consider subroutine g for the calculation of the discriminant function, as is given in Algorithm 1. This function will go through each of the a priori probabilities, whether zero or not, and will calculate X(r)*Y(k)*Z(q)*G(r,k,q)+sum. If the a priori probability is zero for a given r,k,q, there will be four floating point operations that will be executed needlessly. Further, if the number of non-zero a priori probabilities is small (less than 50%), then this will use a large amount of space for storage of zeroes.

Consider the following function (used with the proposed storage format) which is a variation of subroutine g' of Algorithm 2:

```
function g'(k)
sum=0;
load r,k,q;     /*this is stored in the Large File*/
old=k;          /*class of discriminant*/
while (k=old)
    begin
    sum=X(r)*Y(k)*Z(q)*G(r,k,q)+sum;
    load r,k,q;
```

```
        end;
    return (sum);
    end function
```

By the nature of the function call, k will be held constant for all values of r and q. Thus, if k is placed in the most significant position of the r,k,q combination, the numbers can be stored in increasing order. For example, if r=1, k=3, and q=5, the number stored in the Large File would be 315. Since the r,k,q and the corresponding a priori probability are stored contiguously in the Large File, the pointers to r,k,q (the neighborhood class specification) and G(r,k,q) can be combined into one pointer. The ability of the FP array to read from the Large File and increment the Large File index register simultaneously allows the programmer to read the data and update the pointer in one program step [16].

The subimage data itself would be stored in a bulk memory (see Figure C-13). A multiple FP configuration which associates one bulk memory with each FP would be best for this application.

For testing this FP contextual classifier program, the classification of two rows of eight pixel measurement vectors (stored in the Large File) using four classes was evaluated. The data were actual Landsat data, as used in [11]. Evaluation of the serial Algorithm 2 from Section 4 showed that a PDP-11/70 required .073 second per pixel, while a single FP required .050 second per pixel. Furthermore, lack of exponent range in the 11/70 floating point hardware yielded the incorrect results due to rounding error. To overcome this error, by normalizing data it would require approximately an extra .030 second per pixel; thus the FP is about 100% faster. The floating point is implemented in software in the FP and uses a 14-bit exponent to overcome this problem. These tests are by no means exhaustive. The simulator must run for many hours just to obtain a result for one pixel. Further testing is in progress.

Using .1 second per pixel as a rough approximation of the PDP processing time, and .05 second per pixel as a rough approximation of a single FP processing time, a 16 FP configuration, where each processor had its own bulk memory, would perform contextual classifications at a rate of 320 pixels per second as opposed to 10 pixels per second for a single PDP-11/70.

Consider horizontally linear neighborhoods in general, such as those shown in Figure C-15. When using N FPs together to process an image, each FP handles 1/N-th of the image. Therefore, nearly a factor of N improvement is attained over the time required for one FP to implement the contextual classifier. (A perfect factor of N improvement occurs if B is a multiple of N. The minor degradation in performance when B is not a multiple of N is discussed in [10,16].) Vertically linear and diagonally linear neighborhoods (Figure C-16) can be processed in a manner similar to that for horizontally linear neighborhoods [10,16].

Consider non-linear neighborhoods, that is, neighborhoods which do not fit into one of the linear classes. For example, all of the neighborhoods in Figure C-17 are non-linear. It can be shown that there is no way to partition an image into N (not necessarily equal) sections such that a contextual classifier using a non-linear neighborhood can be performed without sharing data among FPs. In order to minimize computation time, the FPs to which pixels are assigned will depend upon the particular image size, number of FPs used, the time required for inter-FP communications, and the shape and size of the neighborhoods. The rest of this section is an examination of one way to implement a contextual classifier based on the square non-linear neighborhood of size nine shown in Figure C-8.

The speed at which the contextual classifier runs depends on the hardware organization of the FP system. Since the floating point algorithms are implemented in the software, it is necessary to exploit the MIMD[17] nature of the array. Floating point routines done in software require a variable amount of time based on the number of shifts required to normalize the data. This can cause a bottleneck in the processing if one FP is required to wait for another. Synchronization can require large amounts of time if the full 16 processor array is used.

For the purposes of this report, consider a non-linear neighborhood as shown in Figure C-8. Each box represents one pixel, while the numbers in each box refer to the numbering used to distinguish the various pixels. The use of a non-linear context neighborhood implies that certain data must be shared among the FPs. For an example, assume that the data for pixels 1, 2, 4, 5, 7, and 8 are stored in FP K, and that the data for pixels 3, 6, and 9 are stored in FP K+1. FP K will need to communicate with FP K+1 to obtain the data necessary to classify pixel 5.

The amount of inter-FP communication needed to perform a contextual classification is dependent on the particular implementation chosen. Figures C-18 and C-19 show two methods for dividing an I-by-J image among the FPs, $I \geq J$. Figure C-18 assumes that there are P*Q=16 processors, and that I/P and J/Q are both integers. Call this the checkerboard allocation. This gives each processor a rectangular subimage of IJ/PQ pixels. Figure C-19 divides the image into PQ stripes of I/PQ lines per subimage. Call this the striping allocation. Each method of division has its advantages. The rectangular subimage requires that, in general, data for

$$2 * ((I/P) + (J/Q)) + 4$$

pixels be transferred into each FP. Striping the same image requires

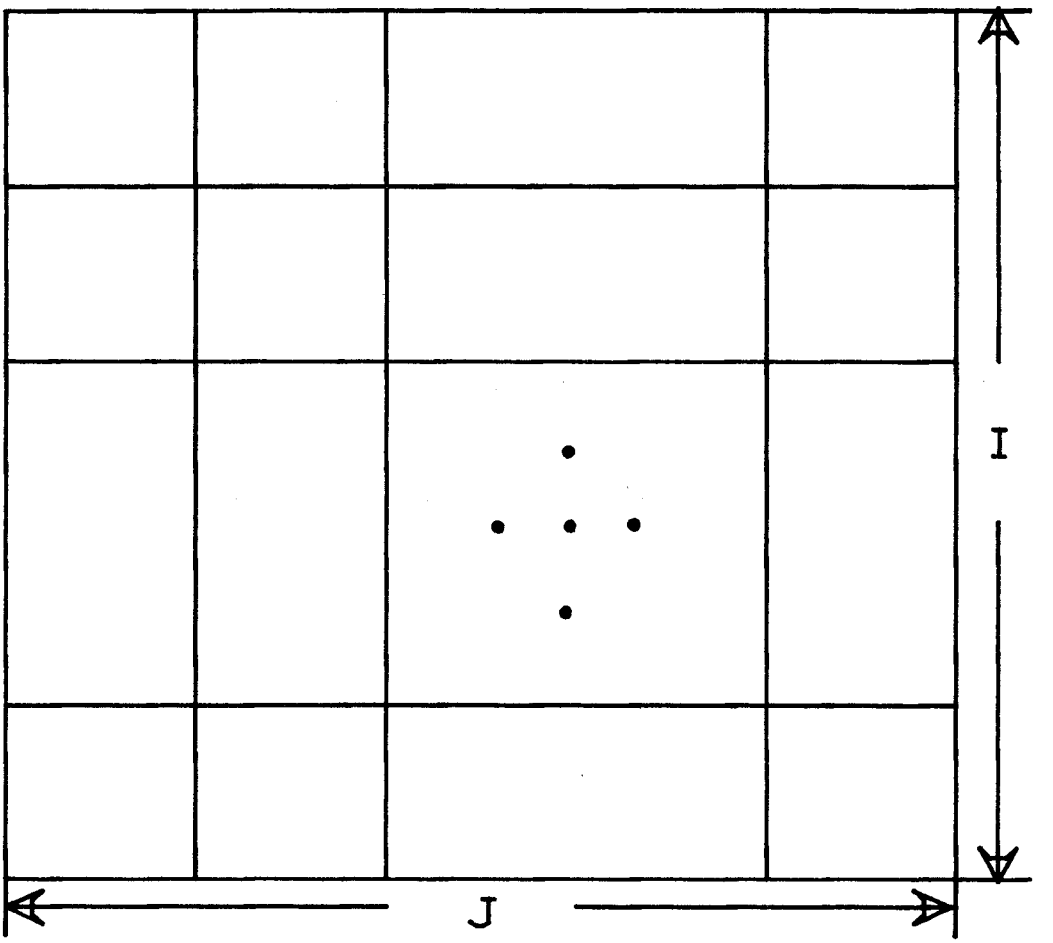


Figure C-18. Checkerboard scheme for classifying an image.

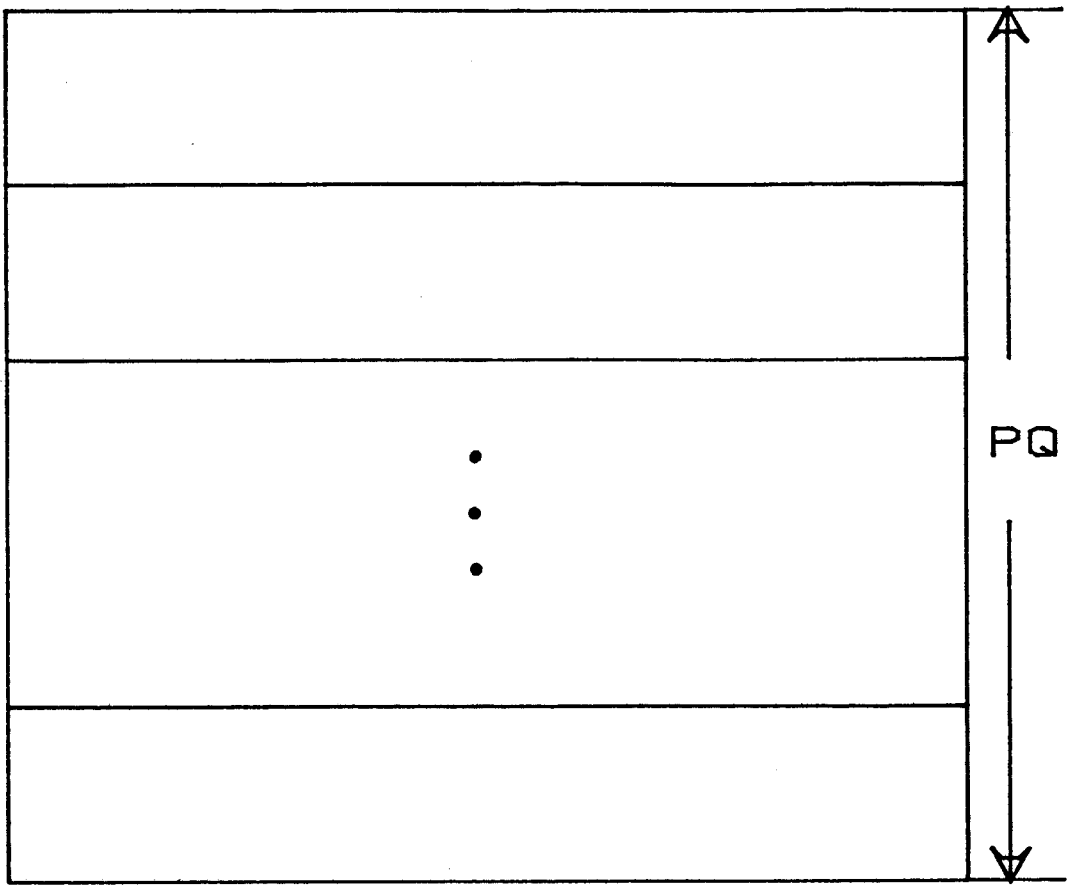


Figure C-19. Striping scheme for classifying an image.

that 2J pixels be transferred from one FP to another. For

$$J > \left\lceil \frac{Q[(I/P)+2]}{Q-1} \right\rceil$$

which will, in practice, be true, the rectangular subimage requires fewer overall data transfers. However, using the checkerboard scheme to minimize total data transfers requires that a given FP communicate with eight other processors. For example, consider the scheme in Figure C-20. FP K holds the data for pixels 4, 5, 7, and 8. The data for pixels 1 and 2 are held in processor K-I, while the data for pixel 3 are held in processor K-I+1. Further, the data for pixels 6 and 9 are held in processor K+1.

The FP array can be constructed with a high-speed inter-FP communication link. Figure C-13 shows that the configuration of the communication link is in a ring. According to [12,13], the data transfer rate of the ring is 64M bytes/sec. This part of the hardware is capable of doing the data transfers, but it requires the synchronization of the FPs, forcing one FP to wait for another, wasting time.

An FP is capable of addressing up to three channels of 16-by-128K bytes of bulk memory each[12,13] (Figure C-13). The sharing of bulk memory is another potential scheme that can be used for shared data. The data for each corner pixel in a given checkerboard rectangle will be accessed by four FPs, as is shown in Figure C-20. The possibility for contention is great. Further, since there are four corners in a given rectangle, a FP would be required to share data with up to eight other FPs.

The checkerboard scheme, because of reduced edge area, will require fewer data transfers. The necessary data transfers are complicated in that they require synchronization of the FPs, negating many of the gains of MIMD (asynchronous) parallel processing.

If the image is divided in the striped scheme, as shown in Figure C-19, each FP will have to communicate with, at the most, two other processors. This requires that FP K communicate with FP K-1 and FP K+1, reducing the communications requirement by a factor of four over the checkerboard scheme.

One possible implementation is shown in Figure C-21. If border areas are stored in the joint memory banks, a processor will begin processing in banks of bus 1. Processing will continue through half the banks in bus 1 to bank 0 on bus 2. After all the data in the banks on data bus 2 have been processed, processing will continue to the banks on bus 3.

Allowing 25% of FP i's data to be stored in the shared banks on bus 1, 50% of the data to be stored in the local banks on bus 2, and 25% of

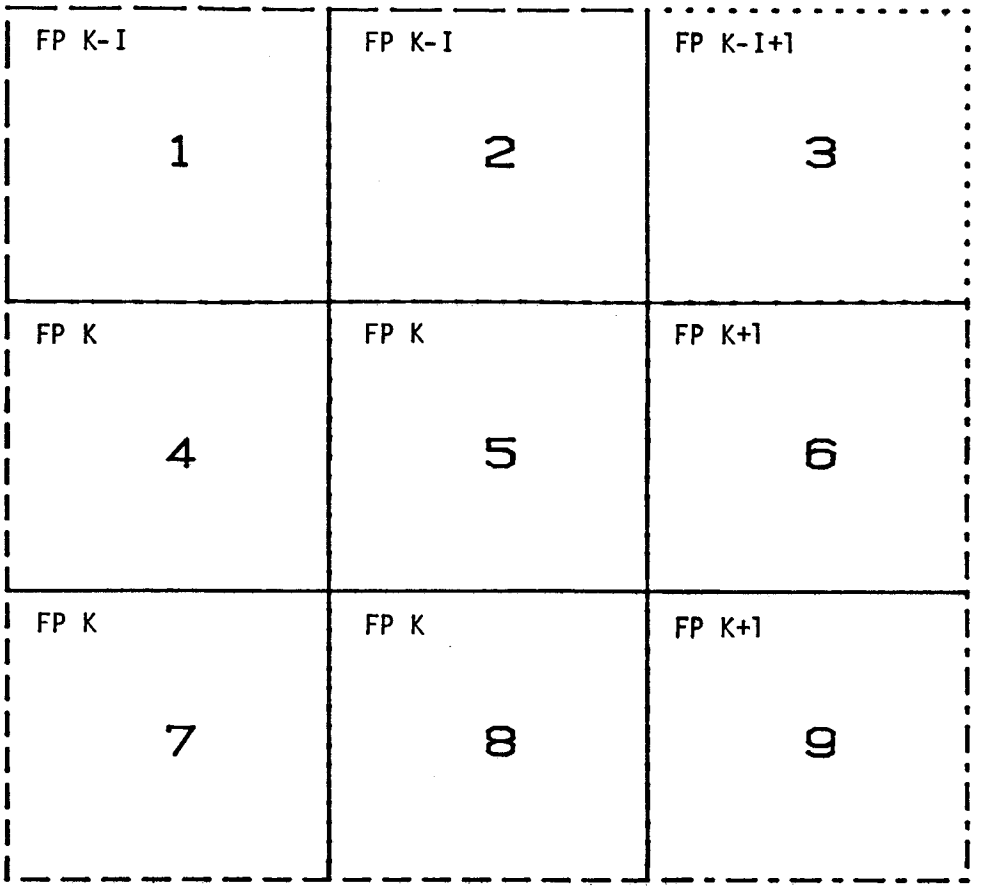


Figure C-20. Demonstration of necessity for inter-FP communications with checkerboard allocation. Note that data for pixel 5 are needed to classify pixels 1 through 9.

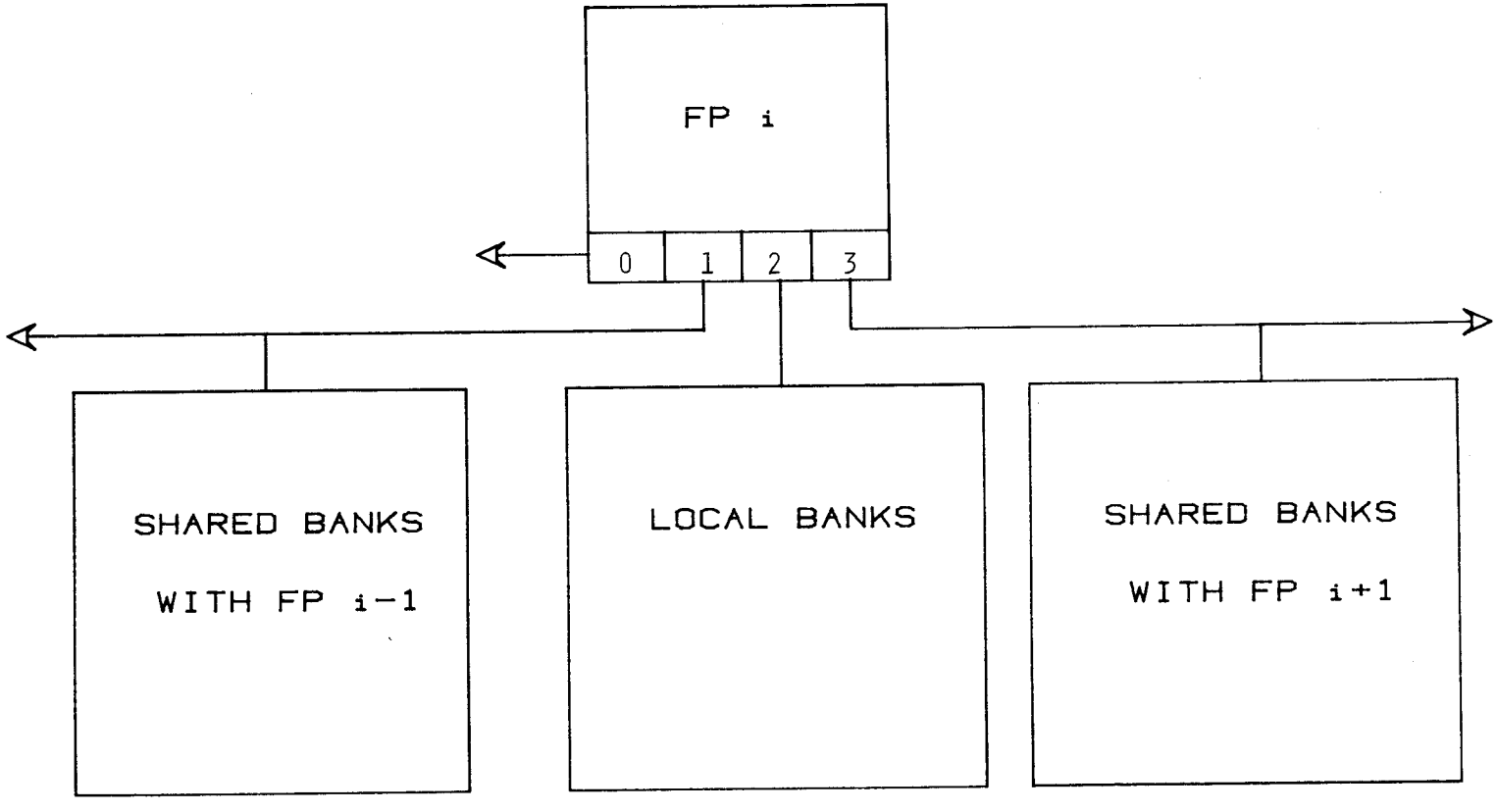


FIGURE C-21. OVERVIEW OF FLEXIBLE PROCESSOR CONFIGURATION.

the data to be stored in the shared banks on bus 3, no contention will occur. Consider that for processor i to "catch up" with processor i+1, processor i will have to process more than 75% of its data in the time it takes processor i+1 to process less than 25% of its data. Thus, contention, which is a common problem in MIMD systems, is not a problem in this implementation.

The "compf" for a pixel is the same, whether it is in one window or another. Thus, all the pixels along the borders will have redundant calculations performed by both FPs. If the first FP to access data for a pixel stores the "compf" values in the shared memory, the redundant calculations can be eliminated, creating a higher utilization of the FP system.

An FP will be allowed to address only half of its memory banks at one time. This is done to facilitate double buffering. The other half will be accessible by the host. This allows, for example, the FP to be classifying the current image while the host unloads and stores the results of the previous classification and then loads the next image to be processed. All memory is dual port, so the host can load the memory without the intervention of the FPs. Single-port memory can be used; however, this may result in decreased efficiency of the overall system. Effective use of cycle stealing should reduce the time losses.

In Figure C-22, L can be taken to be any even number less than or equal to 16. For the purposes of this report, let L=16. This will result in the maximum storage capabilities for the entire system. The memory banks are arranged as shown in Figure C-22 to show which banks will be used by a FP while the host is loading the other banks in the memory.

The eight-nearest-neighbor contextual classifier is similar to the previously discussed linear case. Differences arise in the calculation of the discriminant function, the method of updating the data for a given window, and the method of data storage.

The calculation of the discriminant function for a given class requires that the class-conditional densities must be used from the eight surrounding pixels, instead of the class-conditional densities for the pixels on the left and the right. From probability and measure theory, it can be seen that the increase in calculations is exponential instead of linear. Further, the number of stored a priori probabilities ($G^P s$) increases by the same factor. Thus, the contextual classifier with a size nine neighborhood requires C^9 stored a priori probabilities, while the same classifier with a size three neighborhood requires only C^3, where C is the number of classes. In order to remain within the space limitation of the Large File, C must be kept very small. If the previous method of storing only the non-zero a priori probabilities is used, C can be allowed to grow.

Two constraints must be considered with respect to the storage of G: (1) the space limitation of the Large File, and (2) the limit of 32 bits to specify the neighborhood of classes associated with a given

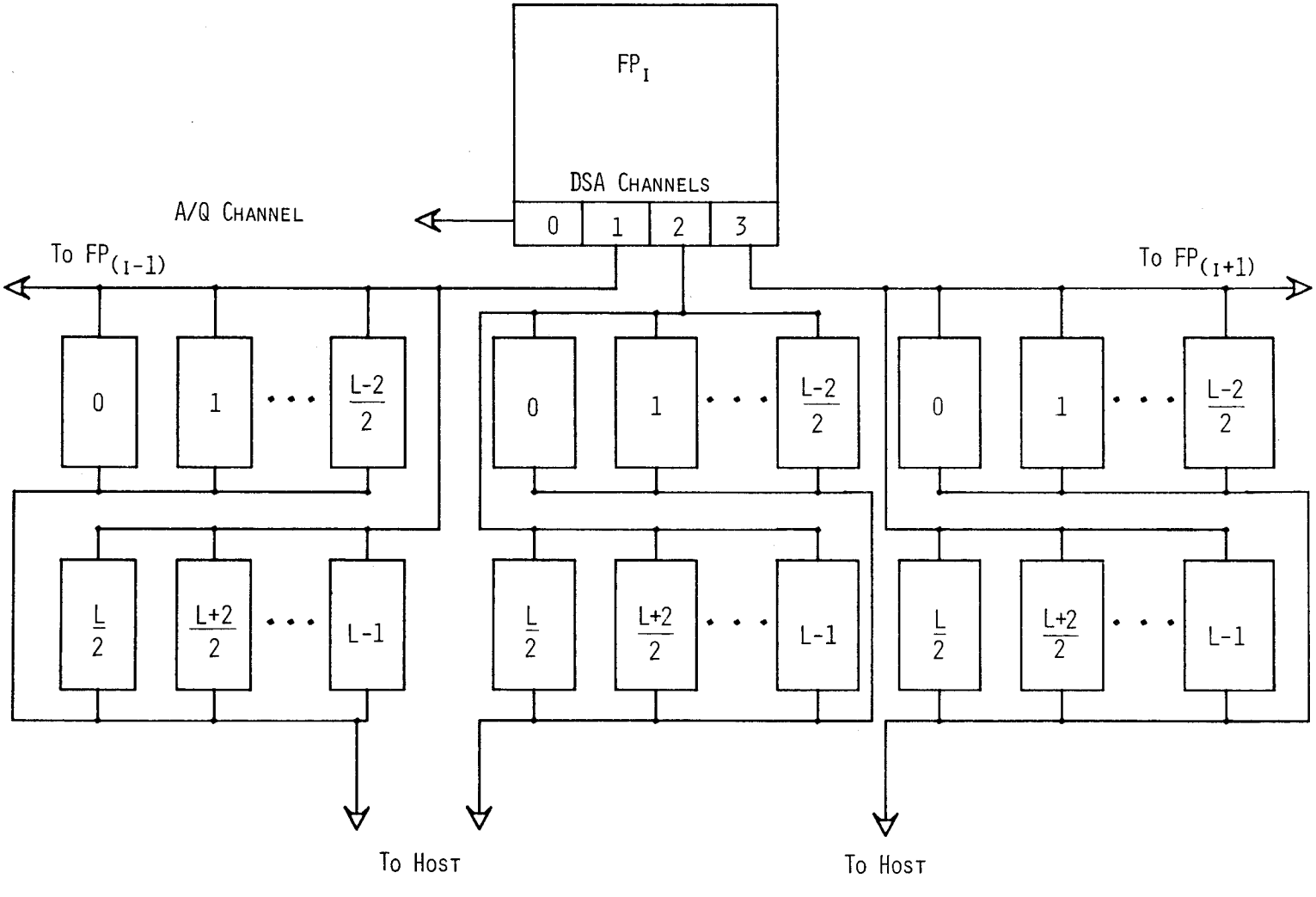


Figure C-22. Potential memory organization for striping scheme.

non-zero G value. The Large File size may be too small to store all the non-zero G's and the associated neighborhood class specification. When this occurs, the G array can be stored in bulk memory, adding 250nsec for each pixel for each non-zero G value. The other constraint is that using only 32 bits to specify the neighborhood classes for each non-zero G value limits the number of classes for a size nine neighborhood to $2^3=8$. However, a second 32-bit specification word could be used for each non-zero G value, allowing up to 7 bits per pixel in the neighborhood, i.e., up to $2^7 = 128$ classes.

The final difference between the linear neighborhood and the non-linear neighborhood is that when the window is moved, the data in the linear case are shifted from pixel 3 to pixel 2 and from pixel 2 to pixel 1 (see Figure C-7), while in the non-linear case, the data must be moved from pixel 3 to pixel 2, pixel 2 to pixel 1, pixel 6 to pixel 5, pixel 5 to pixel 4, pixel 9 to pixel 8, and pixel 8 to pixel 7 (see Figure C-8). Further, in the current FP implementation, with each move of the window, the size nine neighborhood must calculate "compf" values for three pixels while the linear neighborhood must calculate "compf" values for only one. Due to the way in which the image is subdivided among FPs, the window will be moved along columns instead of rows.

Timings run with data from Landsat data used in [11] show that, on the average, the FP implementation of the four class, size nine square neighborhood contextual classifier requires 194264 + (1028 X number of non-zero a priori probabilities) machine cycles for classifying one pixel. Only the non-zero a priori probabilities are stored and, as indicated in Algorithm 2, only the non-zero a priori probabilities are going to affect the computation time. The tests were run with approximately .7% of G's being non-zero. Using test data, the FP implementation required 2568926 machine cycles to perform a contextual classifier for a four class, size nine square neighborhood on a given pixel. This translates to 0.321 second per pixel. The above timing is based on only 18 test data samples, as the classifier is run on a simulator in a time-shared environment and requires a long execution time. Further tests are being run to achieve a more accurate estimate. The above timings assume that an FP has to load all data from the bulk memory, and that all data are preformatted according to the needs of the program.

When the class of a given pixel has been determined, the window must be repositioned so as to classify the next pixel. Changing the context of a pixel will not change the "compf" values for that pixel, as the "compf" values are dependent only on the measurement vector and class dependent data. Thus, moving a window does not require recalculation of the "compf" values for each pixel within that window. A size nine square window will use "compf" values for a given pixel nine times: three times in the top row, three times in the middle row, and three times in the bottom row. Maximum throughput can be achieved by storing the data for each line. Thus, it is necessary to calculate only one set of "compf" values for each move of the window. This was discussed in Section 3.

The current algorithm does not store the "compf" values for two lines as the bulk memories are not yet implemented. This requires two redundant calculations of "compf" values per pixel; thus the speed of the algorithm is not maximal. Future plans include upgrading the simulator to handle the large bulk memories and implementing the contextual classifier with no redundant calculations.

In summary, the organization of an FP system given above will allow contention-free sharing of data. This means that N FPs will be able to operate N times faster than one FP. Furthermore, the double-buffered memory scheme will allow the loading of images to be processed and storage of results by the host to be overlapped with the classification operation of the FPs.

7. SIMD Machines and PASM

The acronym SIMD stands for "single instruction stream -- multiple data stream"[17]. Typically, an SIMD machine is a computer system consisting of a control unit, N processors, N memory modules, and an interconnection network. The control unit broadcasts instructions to all of the processors, and all active processors execute the same instruction at the same time. Thus, there is a single instruction stream. Each active processor executes the instruction on data in its own associated memory module. Thus, there is a multiple data stream. The interconnection network, sometimes referred to as an alignment or permutation network, provides a communications facility for the processors and memory modules. Examples of existing SIMD machines include the Illiac IV and STARAN[18,19].

One way to model the physical structure of an SIMD machine is shown in Figure C-23. As indicated, there are N processing elements (PEs), numbered from 0 to N-1, where each PE consists of a processor with its own memory. The PEs receive their instructions from the control unit and communicate through the interconnection network.

To demonstrate how SIMD machines operate, consider the following simple task. Assume that A, B, and C are each one-dimensional arrays (vectors) and that the task to be performed is the elementwise addition of A and B, storing the result in C. In a uniprocessor system, this can be expressed as:

```
for i = 0 to N-1 do
    C(i) = A(i) + B(i)
```

This computation will take N steps on a serial machine.

Assume that A, B, and C are stored in a SIMD machine, with N PEs, such that A(i), B(i), and C(i) are all stored in the memory of PE i, $0 \leq i < N$. To perform an elementwise addition of the vectors A and B and store the result in C, all PEs would execute (simultaneously):

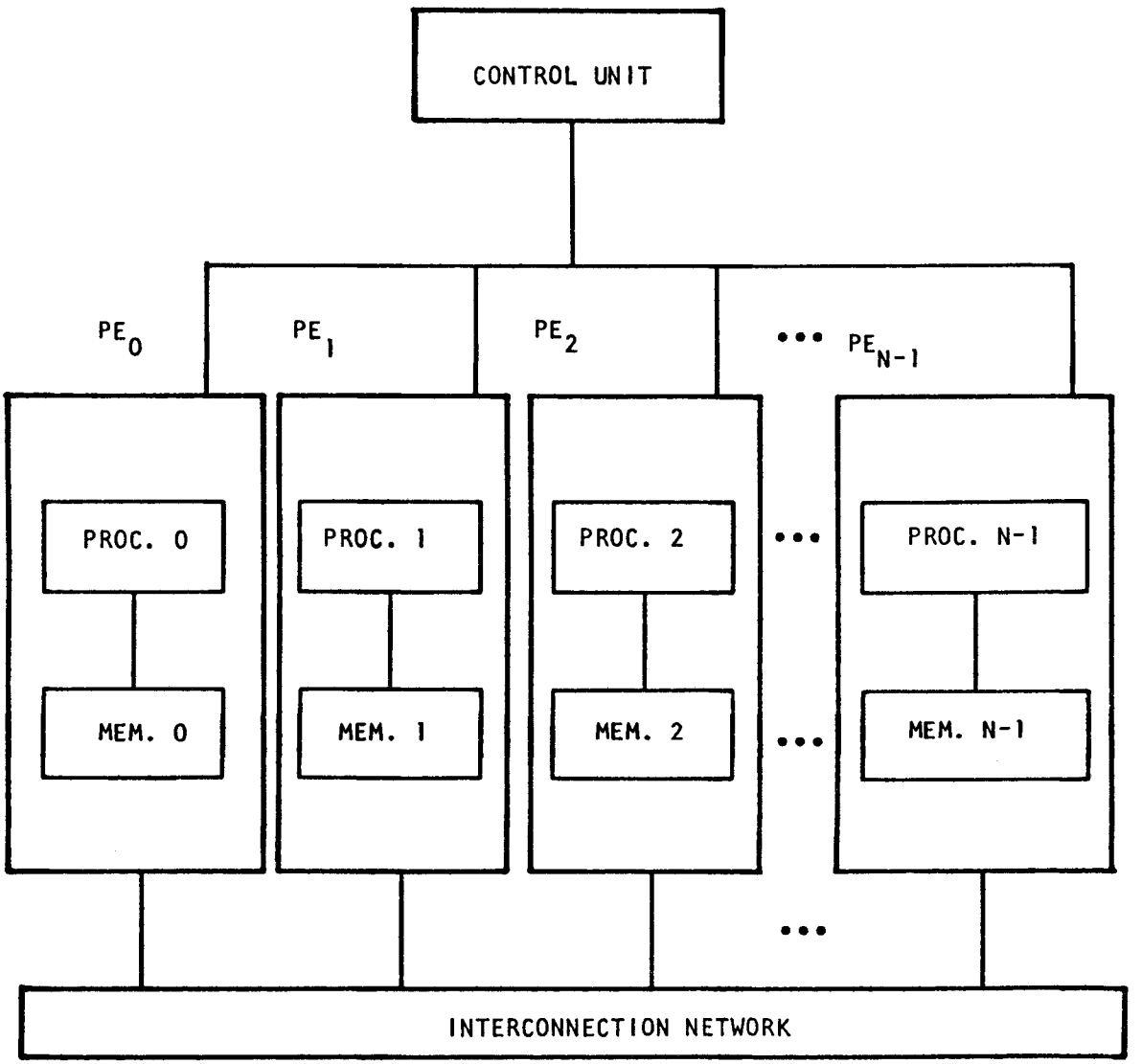


Figure C-23. A general model of an SIMD machine.

$$C = A + B$$

with PE i doing the addition of A(i) and B(i), storing the result in C(i). Thus, in this case, the SIMD machine does in one step a task requiring N steps on a serial processor.

Consider a variation on this example. Assume the N-step serial task is:

$$\underline{\text{for}}\ i = 1\ \underline{\text{to}}\ N-1\ \underline{\text{do}}$$

$$C(i) = A(i) + B(i-1)$$

$$C(0) = A(0)$$

Given the data allocation above (i.e., A(i), B(i), and C(i) in PE i) an SIMD machine does this task in three different steps:

1. The value of B(i-1) is moved, through the interconnection network, from PE i-1 to PE i, $1 \leq i < N$. Most proposed and existing SIMD interconnection networks can do this in one parallel data transfer[20].

2. In PE i, add A(i) to B(i-1) and store the result in C(i), $1 \leq i < N$ (PE 0 is disabled).

3. Enable only PE 0 and execute C=A.

Thus, this example demonstrates the need for the interconnection network and methods for disabling PEs.

This simple example was provided to familiarize the reader with the concept of the SIMD mode of parallel processing. More complex examples involving image processing and feature extraction can be found in [21-24].

PASM is a dynamically reconfigurable multimicroprocessor system being developed at Purdue University for image processing and pattern recognition tasks[25-31]. The PASM design will support as many as 1024 processors. Other computer architects have proposed parallel processing systems with 2^{14} to 2^{16} microprocessors[32,33]. When considering SIMD machine implementations of contextual classifiers in the following sections, these implementations will be based on PASM.

PASM can operate in SIMD mode or the asynchronous multiple instruction stream -- multiple data stream (MIMD)[17] mode. PASM is a PArtitionable SIMD/MIMD system, that is, it can be partitioned into SIMD and/or MIMD systems of varying sizes.

The rest of this section will be a brief overview of PASM. The overview is limited to those aspects of PASM that are needed to understand the SIMD algorithms in the following sections.

Figure C-24 is a block diagram of PASM. The basic system components are the Parallel Computation Unit, the Micro Controllers, the Control Storage, the Memory Management System, the Memory Storage System, and the System Control Unit.

The heart of the system is the Parallel Computation Unit (PCU), which contains N processors, N memory modules, and the interconnection network. The PCU processors are microprocessors that perform the actual computations. The PCU memory modules are used by the PCU processors for data storage in SIMD mode. The interconnection network provides a means of communication among the PCU processors and memory modules. PASM uses data conditional and PE address masks to activate and deactivate PCU processors in SIMD mode[28,34].

The Micro Controllers (MCs) are a set of microprocessors which act as the control unit for the PCU processors in SIMD mode. Control Storage contains the programs for the Micro Controllers. The Memory Management System controls the loading and unloading of the PCU memory modules. It employs a set of cooperating dedicated microprocessors. The Memory Storage System stores these files. Multiple devices are used to allow parallel data transfers. The System Control Unit is a conventional machine, such as a PDP-11, and is responsible for the overall coordination of the activities of the other components of PASM.

The processors, memory modules, and interconnection network of the PCU are organized as shown in Figure C-25. A pair of memory units is used for each PCU memory module so that data can be moved between one memory unit and the Memory Storage System while the PCU processor operates on data in the other memory unit. This is controlled by the Memory Management System. The processors communicate through the interconnection network. One network being considered is a multistage implementation of the "PM2I" network[35,36] called the Augmented Data Manipulator (ADM) network[20,36-42]. The other network being considered is a multistage implementation of the "cube" network[35,36] called the Generalized Cube network[20,39-41].

This very brief overview of PASM is provided as background for the following sections. More details about PASM can be found in the references mentioned above.

8. SIMD Implementation of Contextual Classifiers

This section examines the implementation of the contextual classifiers discussed in Sections 2 and 3 on a microprocessor-based SIMD machine. First consider the SIMD implementation of a contextual classifier based upon a horizontally linear neighborhood with p=3. The approach to decomposing the task will be similar to that used in Section 5 for the FP system. In both cases, the image is divided into N subimages, and each subimage is assigned to a different processor for classification computations. However, there are three main differences:

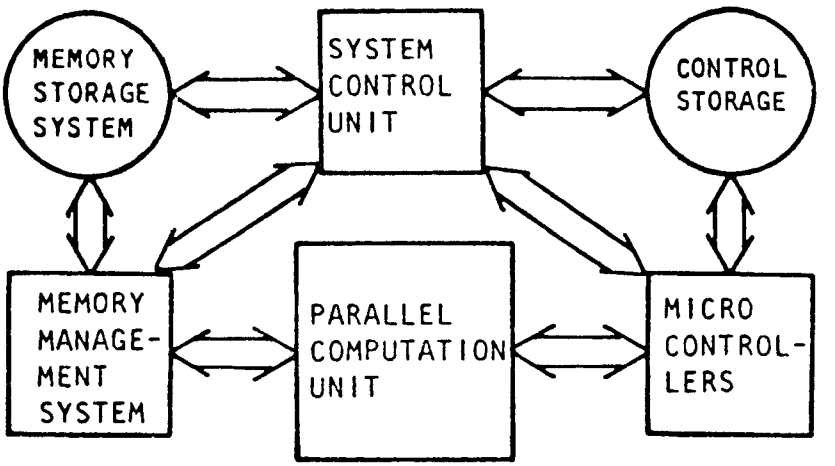


Figure C-24. Block diagram overview of PASM

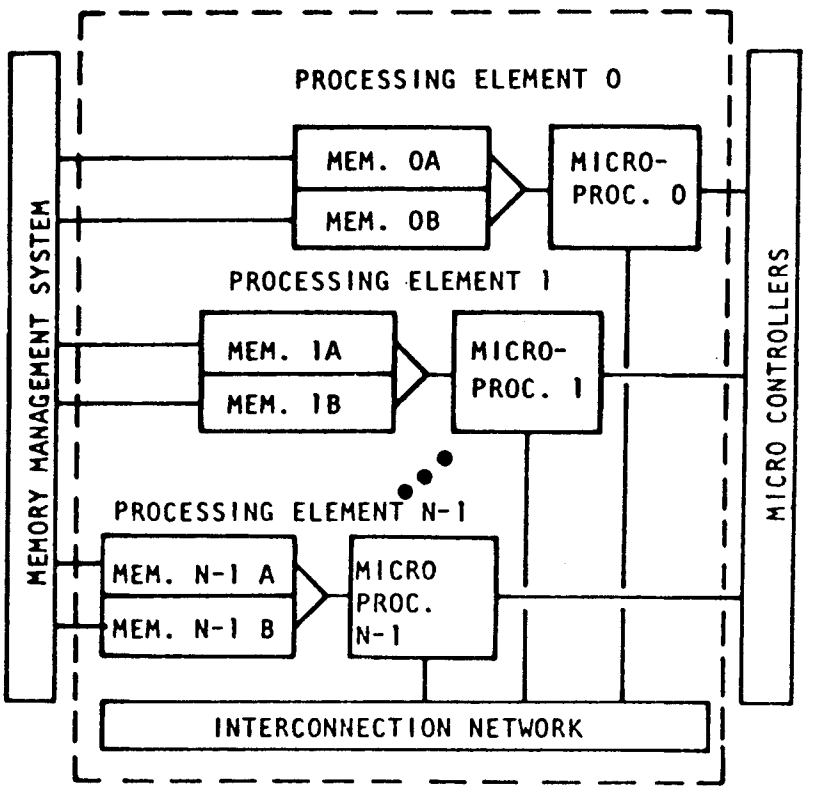


Figure C-25. The Parallel Computation unit.

1. It is technologically and economically feasible to construct a multimicroprocessor SIMD machine with many more than 16 processors. Therefore, while the "N" for the FP system is limited by 16, the "N" for the multimicroprocessor system could be as large as 256, 512, or 1024.

2. The differences in computational capabilities between an FP and an off-the-shelf microprocessor must be considered. For example, depending on the microprocessor chosen, 16 FPs may be faster than 32 microprocessors.

3. In the SIMD mode of parallelism, the program (Algorithm 2) is stored in the control unit, not in each microprocessor. The control unit broadcasts the instructions to the microprocessors. The control unit would also store the G^P array, broadcasting the appropriate array element to all the microprocessors when it is needed. The control unit (MCs) can decode the neighborhood class specification for the next non-zero G value while the PEs in the PCU are performing the calculations for the current non-zero G. In the FP system, each FP would store a copy of the program and must store or have access to the G^P array.

Thus, an SIMD machine can be used to perform the contextual classification based on a horizontally linear neighborhood of size three without any inter-PE communication. As in the case of using the FP system to implement the classifier, the implementation using an SIMD machine with N microprocessors can achieve as much as a factor of N improvement over the use of a single microprocessor. The exact improvement will be a function of the image size and N.

To attain a perfect factor of N improvement, B (in Figure C-14) would have to be a multiple of N. Since N in the SIMD case would be a multiple of the N in the FP case, this is less likely to occur. When B is not a multiple of N, then (a) some PEs may have to process more rows than others (leaving some PEs underutilized), or (b) each PE would process a subimage including a partial row (requiring inter-PE data transfers). The alternative that is best would depend on the image size, the way in which subimages are allocated to PEs, N, the processor speed, and the interconnection network speed. The situation for vertically linear and diagonally linear neighborhoods is similar.

Now consider the SIMD implementation of a size nine non-linear square neighborhood contextual classifier on PASM. The approach taken is somewhat different from that for the CDC FP system because:

1. The processors are synchronized.

2. There is no directly-wired shared memory between processors.

The I-by-J image is divided into N subimages, each an $(I/\sqrt{N})$-by-$(J/\sqrt{N})$ array (i.e., the image is divided like a checkerboard). This is shown in Figure C-26. Each PE stores one such subimage. All of the PEs execute Algorithm 2, modified for a size nine square neighborhood. This is done in similar manner to the linear case (i.e., the control unit will store the program and the G^P array, etc.), except now inter-PE data transfers will be necessary.

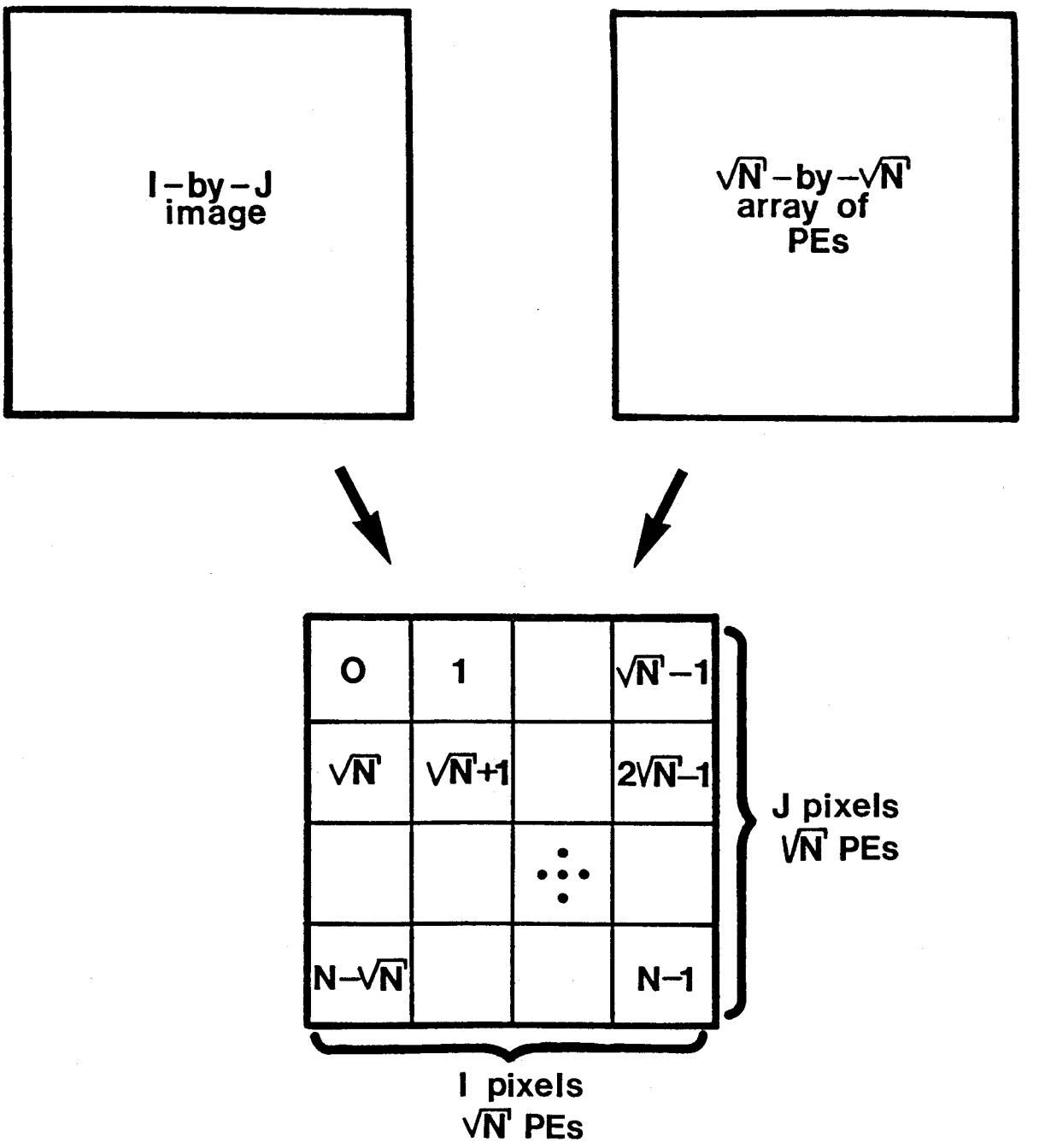


Figure C-26. Dividing an image into subimages using a "checkerboard" pattern. Each square represents one PE with a (I/√N)-by-(J/√N) image. The number in the square is the PE number.

Each PE can classify all the pixels in its subimage which are not on the subimage edges. All PEs can do this simultaneously. Thus, these $N * (((I/\sqrt{N})-1) * ((J/\sqrt{N})-1))$ pixels can be classified in the time it would take one PE to classify $((I/\sqrt{N})-1) * ((J/\sqrt{N})-1)$ pixels. A factor of N speedup is attained. (This will be degraded somewhat if I and J are not divisible by N.)

To classify subimage edge pixels, the PEs must share data by passing information through the interconnection network. For example, in order for PE 0 to classify pixel $(0, (J/\sqrt{N})-1)$ it needs to get the "compf" values for pixel $(0, J/\sqrt{N})$ from PE 1.

The simplest way to do this is to have each PE first compute and store the "compf" values for their edge pixels in a $2(I/\sqrt{N}) + 2(J/\sqrt{N})$ vector called EDGE. Later, when a PE needs the "compf" values for these pixels in order to classify pixels in its own subimage columns 1 and $(J/\sqrt{N})-1$ and rows 1 and $(I/\sqrt{N})-1$, they are fetched from EDGE, not recomputed.

Immediately after a PE calculates the "compf" values for edge pixels of its subimage and saves them in EDGE, it sends copies of these values to the appropriate "adjacent" PE. For example, assume that for rows 0 to $(I/\sqrt{N})-1$ PE 0 contains column $(J/\sqrt{N})-1$ of the original image and PE 1 contains column $(J/\sqrt{N})$ of the original image. Then PE 0 will send PE 1 the "compf" values for rows 0 to $(I/\sqrt{N})-1$ of column $(J/\sqrt{N})-1$ and PE 1 will send PE 0 the compf values for the same rows of column $(J/\sqrt{N})$. Each PE would save the value it receives in a vector OUTEREDGE. Each PE would then access its own OUTEREDGE vector when it is ready to classify its edge pixels. In general, PE i would send subimage column $(J/\sqrt{N})-1$ edge data to PE i+1, for all i, $0 \leq i < N$, simultaneously. (The PEs which contain pixels from column J-1 of the original image would not have to send data.) A similar transfer is done for the other three edges and the four outer-edge "corners." This method requires only $(2(I+J)/\sqrt{N})+4$ parallel data transfers. No redundant "compf" calculations are made.

A checkerboard division of the image was used since, in general, it requires fewer inter-PE transfers than dividing the image by rows or columns. Giving each PE J/N columns would imply that 2J inter-PE transfers would be needed. Similarly, giving each PE I/N rows would imply that 2I inter-PE transfers would be needed.

The timing complexity of this size nine neighborhood SIMD algorithm is as follows:

1. Growth proportional to $(I/\sqrt{N}) * (J/\sqrt{N}) * C^9 = (I*J*C^9)/N$ assignments, multiplications, and additions.

2. Exactly $(I/\sqrt{N}) * (J/\sqrt{N}) * C = (I*J*C)/N$ "compf" calculations.

3. Exactly $(2(I+J)/\sqrt{N})+4$ inter-PE data transfers.

For arithmetic operations and "compf" calculations, a perfect factor of N speedup is attained. This is done at the "cost" of $(2(I+J)/\sqrt{N})+4$ inter-PE transfers. These data transfers are negligible when compared with the (I*J*C)/N "compf" computations.

In summary, both linear and non-linear neighborhood SIMD contextual classification algorithms have been presented. It has been shown that the SIMD mode of parallelism and contextual classification are well suited for each other.

9. Summary

The preliminary results from the first year of study into the contextual classifier indicated three main problem areas for research, many of which have been addressed in this year's research. In studying methods for estimating the context distribution, the ground-truth-guided method has been shown to be an effective way of estimating the context distribution when adequate ground truth is available. When this is not the case, a combination of the Power Method with estimation over information classes can give good results. Because the latter method does not give as consistently good results as the ground-truth-guided method, research is continuing in this area. Development of an unbiased estimator of the context distribution is being considered as a possible approach.

An approximation to the full contextual classification algorithm has been shown to reduce computation time by about one-half and does not significantly affect classification accuracy. A hybrid combination of this approximate algorithm with a conventional no-context classifier offers prospects for further significant reduction in computation time.

A third area of research concerns the major assumptions made in the derivation and implementation of the contextual classification algorithm. Of most theoretical interest is the assumption of class-conditional independence of the observations as expressed by eq. (6). The effects of this assumption will be investigated later in the coming year. Another assumption made was that classifications should be made into spectral classes rather than information classes. It has been shown that there is some advantage to classifying into information classes when classification accuracies are low, but this advantage disappears at higher accuracies.

Algorithms for performing contextual classifications using a size three horizontally linear neighborhood were presented. Algorithm 1 was a straightforward approach; Algorithm 2 was a more efficient approach that avoided unnecessary calculations. The serial complexity for performing Algorithm 2 on an I-by-J image with C classes was shown to have a growth proportional to $I*J*C^3$ assignments, multiplications, and additions, and I*J*C "compf" calculations. The way in which Algorithm 2 could be extended to a size nine non-linear square neighborhood was discussed.

The use of N CDC Flexible Processors to implement contextual classifiers, based on linear neighborhoods, or size nine square non-linear neighborhoods, which are N times faster than a single FP, was explained. The type of FP system configuration which would be well suited for each was described. Timing results obtained through the use of our FP simulator were presented.

The use of N microprocessors in the SIMD mode of parallel processing to do linear classifications N times faster than a single microprocessor was discussed. For the case of a non-linear square neighborhood, the use of SIMD parallelism greatly reduces the execution time required. A perfect reduction in execution time of a factor of N is not attained due to the "parallel overhead" of inter-PE transfers needed.

Through the use of parallel computer systems, such as the CDC Flexible Processor system and PASM, the types of computations required for contextual classifiers and other computationally demanding remote sensing processes can be implemented efficiently. This not only reduces the computation time required to do contextual classification but also allows the investigation of techniques which must otherwise be considered infeasible.

References

1. Swain, P.H. and S.M. Davis, eds. 1978. Remote Sensing: The Quantitative Approach. McGraw-Hill, Inc., New York.

2. Kettig, R.L. and D.A. Landgrebe. 1976. Classification of Multispectral Image Data by Extraction and Classification of Homogeneous Objects. IEEE Trans. Geos. Elect. Vol. 14, pp. 19-26.

3. Haralick, R.M., K. Shanmugam and I. Dinstein. 1973. Textural Features for Image Classification. IEEE Trans. Systems, Man and Cybernetics, Vol. 3, pp. 610-621.

4. Yamamoto, H. 1979. A Method of Deriving Compatibility Coefficients for Relaxation Operators. Comp. Graph. Image Processing. Vol. 10, pp. 256-271.

5. Tilton, J.C., P.H. Swain, and S.B. Vardeman. 1980. Context Distribution Estimation for Contextual Classification of Multispectral Image Data. Proc. of 1980 Machine Processing of Remotely Sensed Data Symp., June 3-6, 1980. IEEE Catalog No. CH1533-99 MPRSD.

6. VanRyzin, J. 1966. The Compound Decision Problem With m x n Finite Loss Matrix. Annals of Mathematical Statistics. Vol. 37, pp. 412-424.

7. Tilton, J.C. 1980. Contextual Classification of Multispectral Image Data: Approximate Algorithm. Laboratory for Applications of Remote Sensing (LARS), Purdue University, West Lafayette, In 47907. AgRISTARS Report SR-PO-00491; also LARS Technical Report 081580.

8. Scholz, D., N. Fuhs, M. Hixson, and T. Akiyama. 1979. Evaluation of Several Schemes for Classification of Remotely Sensed Data: Their Parameters and Performance. Laboratory for Applications of Remote Sensing (LARS), Purdue University, West Lafayette, IN 47907. LARS Technical Report 041279.

9. Swain, P.H., H.J. Siegel, and B.W. Smith. A Method for Classifying Multispectral Remote Sensing Data Using Context. Proc. of 1979 Machine Processing of Remotely Sensed Data Symp., June 1979, pp. 343-353. IEEE Catalog No. 79CH1430-8 MPRSD.

10. Swain, P.H., H.J. Siegel, and B.W. Smith. Contextual Classification of Multispectral Remote Sensing Data Using a Multiprocessor System. IEEE Trans. on Geoscience and Remote Sensing, Vol. GE-18, No. 2, Apr. 1980, pp. 197-203.

11. Swain, P.H., S.B. Vardeman, and J.C. Tilton. Contextual Classification of Multispectral Image Data. Laboratory for Applications of Remote Sensing (LARS), Purdue University, West Lafayette, IN 47907, Jan. 1980, 34 pp. LARS Contract Report 011080.

12. Control Data Corp., Cyber-Ikon Image Processing System Design Concepts. Digital Systems Division, Control Data Corp., Minneapolis, MN, Jan. 1977.

13. Control Data Corp., Cyber-Ikon Flexible Processor Programming Textbook. Digital Systems Division, Control Data Corp., Minneapolis, MN, Nov. 1977.

14. Kast, J.L., P.H. Swain, and T.L. Phillips. The Feasibility of Using a Cyber-Ikon System as the Nucleus of an Experimental Agricultural Data Center. Laboratory for Applications of Remote Sensing (LARS), Purdue University, West Lafayette, IN 47907, Feb. 1978. LARS Contract Report 021678.

15. Swain, P.H., P.E. Anuta, D.A. Landgrebe, and H.J. Siegel. Vol. III: Processing Techniques Development, Part 2: Data Preprocessing and Information Extraction Techniques. Laboratory for Applications of Remote Sensing (LARS), Purdue University, West Lafayette, IN 47907, Nov. 1979. LARS Contract Report 113079.

16. Smith, B.W., H.J. Siegel, and P.H. Swain. A Multiprocessor Implementation of a Contextual Image Processing Algorithm. AgRISTARS Report SR-PO-00474 (also LARS Technical Report 070180), Laboratory for Applications of Remote Sensing (LARS), Purdue University, West Lafayette, IN 47907, July 1980.

17. Flynn, M.J. Very High-Speed Computing Systems. Proc. of the IEEE, Vol. 54, Dec. 1966, pp. 1091-1909.

18. Bouknight, W.J., et al. The Illiac IV System. Proc. of the IEEE, Vol. 60, Apr. 1972, pp. 369-388.

19. Batcher, K.E. STARAN Parallel Processor System Hardware. AFIPS Conference Proc., Vol. 43: 1974 Natl. Computer Conf., May 1974, pp. 405-410.

20. Siegel, H.J. Interconnection Networks for SIMD Machines. Computer, Vol. 12, No. 6, June 1979, pp. 57-65.

21. Feather, A.F., L.J. Siegel, and H.J. Siegel. Image Correlation Using Parallel Processing. Proc. of Fifth Internatl. Conf. on Pattern Recognition, Dec. 1980, pp. 503-507.

22. Mueller. P.T., Jr., L.J. Siegel, and H.J. Siegel. Parallel Algorithms for the Two-Dimensional FFT. Proc. of Fifth Internatl. Conf. on Pattern Recognition, Dec. 1980, pp. 497-502.

23. Siegel, L.J., P.T. Mueller, Jr., and H.J. Siegel. FFT Algorithms for SIMD Machines. Proc. of 17th Annual Allerton Conf. on Communications, Control, and Computing, University of Illinois, Urbana, IL, Oct. 1979, pp. 1006-1015.

24. Swain, P.H., H.J. Siegel, and J. El-Achkar. Multiprocessor Implementation of Image Pattern Recognition: a General Approach. Proc. of Fifth Internatl. Conf. on Pattern Recognition, Dec. 1980, pp. 309-317.

25. Mueller, P.T., Jr., L.J. Siegel, and H.J. Siegel. A Parallel Language for Image and Speech Processing. Proc. of IEEE Computer Society's Fourth Internatl. Conf. on Computer Software and Applications (COMPSAC '80), Oct. 1980, pp. 476-483.

26. Siegel, H.J. Preliminary Design of a Versatile Parallel Image Processing System. Proc. of Third Biennial Conf. on Computing in Indiana, Indiana University, Bloomington, IN, Apr. 1978, pp. 11-25.

27. Siegel, H.J., F. Kemmerer, and M. Washburn. Parallel Memory System for a Partitionable SIMD/MIMD Machine. Proc. of 1979 Internatl. Conf. on Parallel Processing, Aug. 1979, pp. 212-221. IEEE Catalog No. 79CH1433-2C.

28. Siegel, H.J., P.T. Mueller, Jr., and H.E. Smalley, Jr. Control of a Partitionable Multimicroprocessor System. Proc. of 1978 Internatl. Conf. on Parallel Processing, Aug. 1978, pp. 9-17. IEEE Catalog No. 78CH1321-9C.

29. Siegel, H.J. and P.T. Mueller, Jr. The Organization and Language Design of Microprocessors for an SIMD/MIMD System. Proc. of Second Rocky Mt. Symp. on Microcomputers: Systems, Software, Architecture, Aug. 1978, pp. 311-340. IEEE Catalog No. 78CH1387-0.

30. Siegel, H.J., L.J. Siegel, F. Kemmerer, P.T. Mueller, Jr., H.E. Smalley, Jr., and S.D. Smith. PASM: A Partitionable Multimicroprocessor SIMD/MIMD System for Image Processing and Pattern Recognition, School of Electrical Engineering, Purdue University, West Lafayette, IN 47907, TR-EE 79-40, Aug. 1979, 69 pp.

31. Siegel, H.J., L.J. Siegel, R.J. McMillen, P.T.Mueller, Jr., and S.D. Smith. An SIMD/MIMD Multimicroprocessor System for Image Processing and Pattern Recognition. Proc. of 1979 IEEE Computer Society Conf. on Pattern Recognition and Image Processing, Aug. 1979, pp. 214-224. IEEE Catalog No. 79CH1428-2C.

32. Pease, M.C. The Indirect Binary N-Cube Microprocessor Array. IEEE Trans. on Computers, Vol. C-26, No. 5, May 1977, pp. 458-473.

33. Sullivan, H., T.R. Bashkow, and D. Klappholz. A Large-Scale Homogeneous, Fully Distributed Parallel Machine. Proc. of Fourth Annual Symp. on Computer Architecture, Mar. 1977, pp. 103-124. IEEE Catalog No. 77CH1182-5C.

34. Siegel, H.J. Controlling the Active/Inactive Status of SIMD Machine Processors. Proc. of 1977 Internatl. Conf. on Parallel Processing, Aug. 1977, p. 183. IEEE Catalog No. 77CH1253-4.

35. Siegel, H.J. Analysis Techniques for SIMD Machine Interconnection Networks and the Effects of Processor Address Masks, IEEE Trans. on Computers, Vol. C-26, No. 2, Feb. 1977, pp. 153-161.

36. Siegel, H.J. A Model of SIMD Machines and a Comparison of Various Interconnection Networks, IEEE Trans. on Computers, Vol. C-28, No. 12, Dec. 1979, pp. 907-917.

37. McMillen, R.J., G.B. Adams III, and H.J. Siegel. Permuting With the Augmented Data Manipulator Network. Proc. of 18th Annual Allerton Conf. on Communication, Control, and Computing, University of Illinois, Urbana, IL, Oct. 1980, to appear.

38. McMillen, R.J. and H.J. Siegel. MIMD Machine Communications Using the Augmented Data Manipulator Network, Proc. of Seventh Annual Internatl. Symp. on Computer Architecture, May 1980, pp. 51-58. IEEE Catalog No. 80CH1494-4.

39. Siegel, H.J. The Theory Underlying the Partitioning of Permutation Networks. IEEE Trans. on Computers, Vol. C-29, No. 9, Sept. 1980, pp. 791-801.

40. Siegel, H.J. and S.D. Smith. Study of Multistage SIMD Interconnection Networks. Proc. of Fifth Annual Symp. on Computer Architecture, Apr. 1978, pp. 223-229. IEEE Catalog No. 78CH1284-9.

41. Smith, S.D. and H.J. Siegel. Recirculating, Pipelined, and Multistage SIMD Interconnection Networks. Proc. of 1978 Internatl. Conf. on Parallel Processing, Aug. 1978, pp. 206-214. IEEE Catalog No. 78CH1321-9.

42. Smith, S.D., H.J. Siegel, R.J. McMillen, and G.B. Adams III. Use of the Augmented Data Manipulator Multistage Network for SIMD Machines. Proc. of 1980 Internatl. Conf. on Parallel Processing, Aug. 1980, pp. 75-78. IEEE Catalog No. 80CH1569-3.

D. AMBIGUITY REDUCTION FOR TRAINING SAMPLE LABELING

D. A. Landgrebe and H. M. Kalayeh

Relaxation labeling processes are iterative techniques which can employ sources of ancillary information to reduce or eliminate the ambiguity of labels of a set of objects. Our main objective is to improve the performance of a spectrally determined classifier by probabilistic relaxation operations. In that regard, we analyze some relaxation algorithms, explain the parameters which are important in controlling these algorithms, present some means by which they can be utilized in earth observational data analysis, and briefly introduce some new algorithms.

1. Introduction

Our goal has been to improve the accuracy of a primary classification by an updating of a decision function based on the relevant information about the scene. Relaxation labeling procedures use two sources of information, an initial (ambiguous) labeling, and information embedded in ancillary data. Our work began by using spatial context as the source of the ancillary information. It was later extended to the use of elevation as well.

In the past five years the probabilistic relaxation processes have been extensively used in picture processing [1,2], especially for line and curve enhancement [3,4,5]. The convergence properties of relaxation have been investigated by Rosenfeld, Zucker, and many others [2,8]. It is suggested that relaxation algorithms, with some modification, can be used for post classification processing of multispectral class maps [12]. One of the important results which has been found [13] is that during early iterations the accuracy of classification will be improved, but after a few iterations the accuracy may begin to deteriorate. The dotted curve of Figure D-1 shows an example of this. This result shows the effect of a straightforward application of probabilistic relaxation to classification results of a LACIE segment in Kansas. As we see, the labeling error exhibits a minimum at a specific iteration and the final error, though an improvement over its initial value, is worse than necessary. This suggests that the relaxation process should be modified in some way or be stopped after a minimum error is observed. However, up to now there has not been any optimal criterion known for doing this. By a concentrated effort to understand how deteriorations occur it was

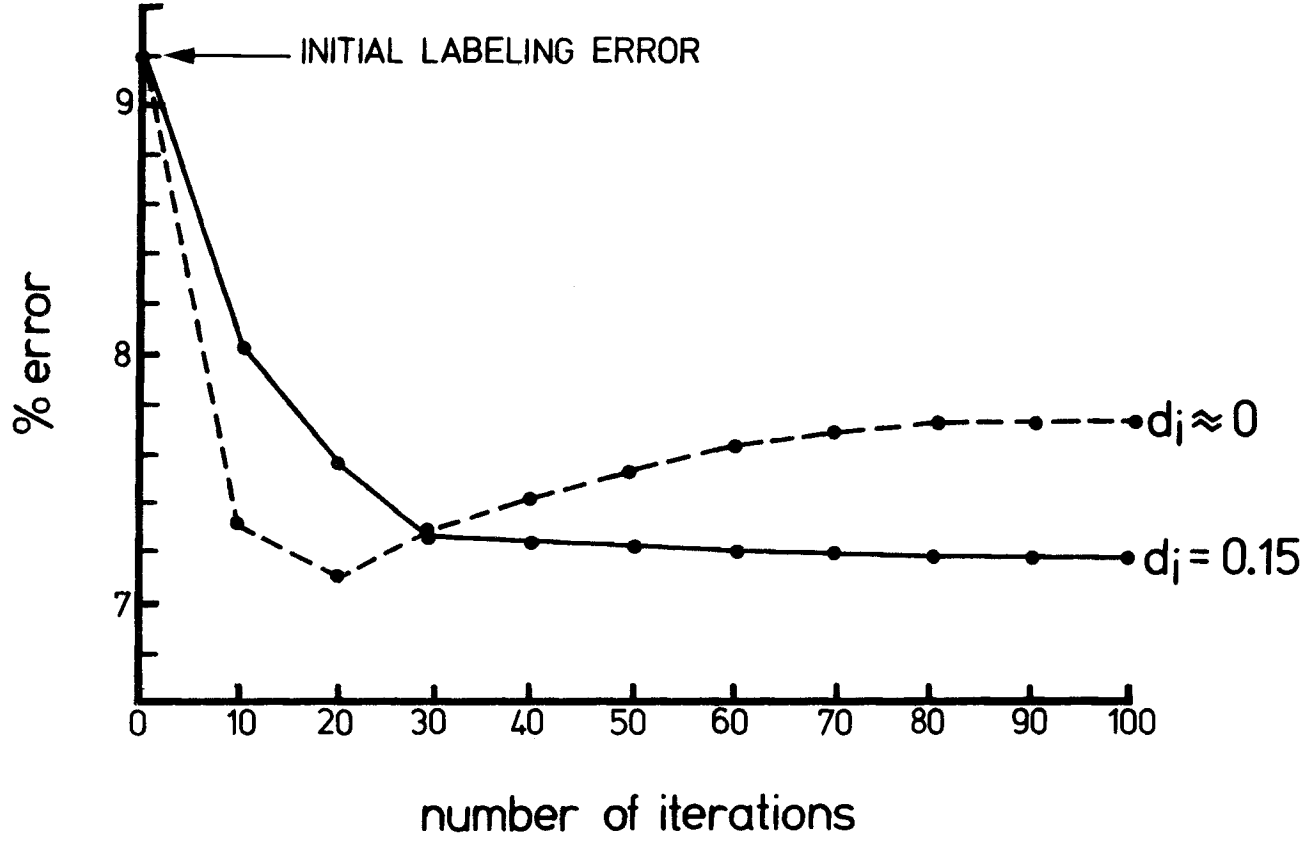


Figure D-1. Label error for the original relaxation algorithm (dotted curve) and an improved procedure (solid curve) for a wheat/nonwheat classification exercise. The image consists of 22932 pixels which were labeled initially as wheat or nonwheat by using minimum distance to means classifier on multitemporal Landsat acquisitions over Kansas.

possible to determine modifications for the existing relaxation algorithms which provide improved performance. The solid curve of Figure D-1 shows the results for one such case [13]. This is but one example of the need for the thorough study of the relaxation process in remote sensing context. This has been the intent of the work reported herein.

In the following is provided further details of the algorithms and the study results, including some new algorithm approaches which show further promise.

2. Some of the Relaxation Algorithms

Let us first consider the probabilistic relaxation algorithms of Rosenfeld, Hummel and Zucker [1,12]. Let $P_i^k(\lambda)$ denote the probability that on the kth iteration the ith pixel of a scene is from class λ. Then define

$$P_i^{k+1}(\lambda) = \frac{Q_i^k(\lambda)P_i^k(\lambda)}{\sum_\lambda Q_i^k(\lambda)P_i^k(\lambda)} \quad \ldots\ldots (1)$$

where $Q_i^k(\lambda)$ is called the neighborhood function and is defined by

$$Q_i^k(\lambda) = \sum_j d_j \sum_{\lambda'} P_{ij}(\lambda|\lambda')P_j^k(\lambda') \quad \ldots\ldots (2)$$

In this equation $P_{ij}(\lambda|\lambda')$ is the probability that pixel i is from class λ given that pixel j is from λ'. The d_{ij} are a set of weighting constants which satisfy $\Sigma d_{ij} = 1$.

Let the denominator of the right side of equation (1) be denoted by D_i^k. In matrix-vector notation the equation has the following form:

$$\begin{bmatrix} P_i^{k+1}(\lambda_1) \\ P_i^{k+1}(\lambda_2) \\ \vdots \\ P_i^{k+1}(\lambda_m) \end{bmatrix} = \begin{bmatrix} \frac{Q_i^k(\lambda_1)}{D_i^k} & & 0 \\ & \frac{Q_i^k(\lambda_2)}{D_i^k} & \\ & & \frac{Q_i^k(\lambda_m)}{D_i^k} \\ 0 & & \end{bmatrix} \begin{bmatrix} P_i^k(\lambda_1) \\ P_i^k(\lambda_2) \\ \vdots \\ P_i^k(\lambda_m) \end{bmatrix} \quad \ldots\ldots (3)$$

Let $q^k(\lambda) \triangleq \sum_j d_j P_j^k(\lambda)$ $\ldots\ldots (4)$

$$\begin{bmatrix} Q_i^k(\lambda_1) \\ Q_i^k(\lambda_2) \\ \vdots \\ Q_i^k(\lambda_m) \end{bmatrix} = \begin{bmatrix} P(\lambda_1|\lambda_1) & P(\lambda_1|\lambda_2) \ldots P(\lambda_1|\lambda_m) \\ P(\lambda_2|\lambda_1) & P(\lambda_2|\lambda_2) \ldots P(\lambda_2|\lambda_m) \\ P(\lambda_m|\lambda_1) & \ldots\ldots\ldots P(\lambda_m|\lambda_m) \end{bmatrix} \begin{bmatrix} q^k(\lambda_1) \\ \vdots \\ q^k(\lambda_m) \end{bmatrix} \quad \ldots\ldots (5)$$

where $P_i^k(\lambda)$ is the kth estimate of the probability that λ is the proper label for the ith pixel; $Q_i^k(\lambda)$ is the kth estimate of the neighborhood function; D_i^k is a normalization factor; and m is the number of classes which are present in the data set. Since*

$$q^k(\lambda) = \sum_j d_j P_j^k(\lambda) \rightarrow$$

$$q_i^k(\lambda) = d_1 P_1^k(\lambda) + d_2 P_2^k(\lambda) + d_3 P_3^k(\lambda) + d_4 P_4^k(\lambda) + d_i P_i^k(\lambda) \quad \ldots\ldots (6)$$

To evaluate neighborhood function, $Q_i^k(\lambda)$, d_i in expression (6) can be equal to zero; if this is the case, it has been called exclusive neighborhood [10,11]. Otherwise, it is termed inclusive neighborhood [1].

3. Analysis of the Algorithm

Consider equations (3) and (4):

$$\underline{P}_i^{k+1}(\lambda) = \Lambda^k \cdot \underline{P}_i^k(\lambda) \quad \ldots\ldots (7)$$

$$\underline{Q}_i^k(\lambda) = T \cdot \underline{q}_i^k(\lambda) \quad \ldots\ldots (8)$$

where $\underline{P}_i^k(\lambda)$ is mx1, Λ^k is an mxm diagonal matrix, $\underline{Q}_i^k(\lambda)$ is mx1, and T is mxm.

---

*A neighborhood consisting of the pixels above, below and to both sides of the "current" or ith pixel is assumed here for convenience of discussion although any other neighborhood could be used.

Our main concern here is to understand the effectiveness of the parameters such as conditional probability $P(\lambda|\lambda')$, central pixel weight d_i, and initial probability estimate $P_i^0(\lambda)$.

In probabilistic relaxation processes, first, the estimated initial label probabilities must be found. For this purpose LARSYS software has been revised to output the estimated posterior class probabilities. Consider an m-class labeling task. Let $p(X|\omega_i)$, $P(\omega_i)$ be the density function of the class i and prior probability for class i, respectively. Then $P_i^0(\lambda)$ can be estimated as follows:

$$P_i^0(\lambda_j) \triangleq P(\omega_j|X) = \frac{p(X|\omega_j)\ P(\omega_j)}{\sum_{j=1}^{m} p(X|\omega_j)\ P(\omega_j)} \quad \ldots\ldots (9)$$

Then based on max $P_i^0(\lambda_j)$, $j = 1,2,\ldots m$, the initial labeling may be performed. Note that the Bayes decision rule is used for initial labeling, but imperfect labeling is presumed.

Since the Bayes classification scheme assigns a class label based upon the maximum likelihood only, a previously utilized source of information which the relaxation exploits is the relative distribution of label likelihoods as defined by equation (9).

4. Local Averaging in the Vicinity of Fixed Points and Its Effect on Geometric Features

To study the role of the central pixel weight d_i in the reduction of ambiguity, assume that the relaxation process has reached a point close to fixed point (the stage where the label estimates are at 0,1 is called a fixed point in the process). Near fixed point we can write:

$$\underline{P}_i^{k+1}(\lambda) \cong \underline{P}_i^{k}(\lambda) = \underline{P}_i(\lambda) \rightarrow [I - \Lambda]\ \underline{P}_i(\lambda) = 0 \quad \ldots\ldots (10)$$

$$\underline{Q}_i^{k+1}(\lambda) \cong \underline{Q}_i^{k}(\lambda) = \underline{Q}_i(\lambda) \rightarrow \underline{Q}_i(\lambda) = T\underline{q}_i(\lambda) \quad \ldots\ldots (11)$$

This implies that near a fixed point the label estimate vector $P_i(\lambda)$ and neighborhood functions are constant or changes very little. Consider equation (10) where $\underline{P}_i(\lambda)$ is the eigenvector corresponding to zero eigenvalue of $[I - \Lambda]$ matrix. By considering the relaxation process at its fixed point it is possible to predetermine a suitable value for d_i based upon preserving corner pixels, line ends, or isolated pixels in

the final results (12). Return to equation (10). Since

$$\underline{P}_i(\lambda) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \leftarrow \text{at } \ell^{th} \text{ row}$$

therefore,

$$D_i = \sum_{j=1}^{m} P_i(\lambda_j)\, Q_i(\lambda_j) = Q_i(\lambda_\ell) \neq 0 \quad \ldots\ldots (12)$$

$$\rightarrow I - \Lambda = \begin{bmatrix} 1 - \dfrac{Q_i(\lambda_1)}{Q_i(\lambda_\ell)} & & 0 & \\ & 1 - \dfrac{Q_i(\lambda_2)}{Q_i(\lambda_\ell)} & 0 & \\ 0 & & & 1 - \dfrac{Q_i(\lambda_m)}{Q_i(\lambda_\ell)} \end{bmatrix} \quad \ldots (13)$$

since $\dfrac{Q_i^k(\lambda_j)}{D_i^k} \leq 1$

Therefore, at fixed point

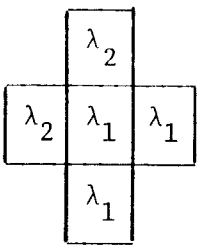
$$\frac{Q_i(\lambda_j)}{D_i} = \frac{Q_i(\lambda_j)}{Q_i(\lambda_\ell)} \leq 1$$

$$\rightarrow \quad 1 - \frac{Q_i(\lambda_j)}{Q_i(\lambda_\ell)} > 0 \qquad j = 1,2,\ldots m; \quad j \neq \ell$$

$$\rightarrow \quad Q_i(\lambda_\ell) - Q_i(\lambda_j) > 0 \qquad \forall j = 1,2,\ldots m; \quad j \neq \ell$$

$$\rightarrow \quad \sum_{k=1}^{m} P(\lambda_\ell|\lambda_k) - P(\lambda_j|\lambda_k)\; q_i(\lambda_k) > 0 \quad \ldots\ldots (14)$$

Now, let us consider 2-class labeling and also let a λ_1 pixel be on the boundary between λ_1 and λ_2 regions. Then assume we have chosen d_i such that λ_1 will be preserved. Therefore, by considering the following geometry at the fixed point,



equation (14) will be:

$$\sum_{k=1}^{2} [p(\lambda_1|\lambda_k) - p(\lambda_2|\lambda_k)]q_i(\lambda_k) > 0$$

$$q_i(\lambda_1) = \frac{P_1(\lambda_1)+P_2(\lambda_1)+P_3(\lambda_1)+P_4(\lambda_1)}{4} + d_i \left[P_i(\lambda_1) - \frac{P_i(\lambda_1) +\ldots+ P_4(\lambda_1)}{4} \right]$$

$P_1(\lambda_1) = 1 \qquad P_1(\lambda_2) = 0$

$P_2(\lambda_1) = 0 \qquad P_2(\lambda_2) = 1$

$P_3(\lambda_1) = 0 \qquad P_3(\lambda_2) = 1$

$P_4(\lambda_1) = 1 \qquad P_4(\lambda_2) = 0$

$P_i(\lambda_1) = 1 \qquad P_i(\lambda_2) = 0$

$$q_i(\lambda_1) = \frac{1}{2} + \frac{d_i}{2}, \qquad q_i(\lambda_2) = \frac{1}{2} - \frac{d_i}{2}$$

$$\Rightarrow [p(\lambda_1|\lambda_1) - p(\lambda_2|\lambda_1)] (\tfrac{1}{2} + \tfrac{d_i}{2}) + [p(\lambda_1|\lambda_2) - p(\lambda_2|\lambda_2)] (\tfrac{1}{2} - \tfrac{d_i}{2}) > 0$$

$$[2p(\lambda_1|\lambda_1) - 1] (\tfrac{1}{2} + \tfrac{d_i}{2}) + [1 - 2p(\lambda_2|\lambda_2)] (\tfrac{1}{2} - \tfrac{d_i}{2}) > 0$$

$$p(\lambda_1|\lambda_1) - p(\lambda_2|\lambda_2) - [1 + p(\lambda_1|\lambda_1) - p(\lambda_2|\lambda_2)] d_i > 0$$

let $p(\lambda_2|\lambda_2) - p(\lambda_1|\lambda_1) = \eta \quad \Rightarrow$

$$d_i > \eta(1 + \eta)^{-1} \qquad \ldots\ldots (15)$$

Equation (15) was first derived in [13]. It allows one to predict a suitable value for d_i in order that λ_1 corner labels not be lost.

Similarly, for the preservation of labels at ends of lines of λ_1 pixels within λ_2 regions can be found from equation (14), which also is given in [13].

$$d_i > \frac{3P(\lambda_2|\lambda_1) - P(\lambda_1|\lambda_1) - 1}{3P(\lambda_2|\lambda_2) - P(\lambda_1|\lambda_1) + 1} \qquad \ldots\ldots (16)$$

Likewise, to preserve individual λ_1 labeled pixel in λ_2 region, it is necessary that

$$d_i > \frac{2P(\lambda_2|\lambda_2) - 1}{2P(\lambda_2|\lambda_2)} \qquad \ldots\ldots (17)$$

Equations (15), (16), and (17) have been used to estimate d_i. Figure D-2 shows the result of a test of this scheme [13]. Figure D-3 shows the error rate vs. d_i for another test with limiting values as determined from equations (15), (16), and (17). Figure D-4 shows labeling error versus number of iterations for selected values of d_i, using the data set of Figure D-1. Note that for d_i less than optimum, labeling error initially decreases, passes through a turning point, and increases again before setting down to a pessimistic final value. For values of d_i near 0.15, the error curve does not exhibit the deterioration phase and has a final value which is almost as low as the minimum in the previous curve. For large d_i, while the curve is monotonically decreasing the final error is larger than necessary. Ultimately for large d_i the curve will remain constant at the initial labeling error.

5. Supervised Label Relaxation

Previously spatial information was utilized to improve classification accuracy. Suppose now there is available a source of ancillary data that has been used to compile another appropriate likelihood measure regarding the labels on each pixel [12]. In supervised relaxation at each iteration the neighborhood function first is modified by the neighborhood data. Then neighborhood function for the label most favored by the ancillary data is increased and others decreased in proportion to their support from the ancillary data source. The relaxation algorithm does not know, of course, which are the correct and which are the incorrect labels. It only "knows" which labels are consistent and which are inconsistent with their neighbors and with the ancillary data. Consequently, an image with initial labeling errors will be iterated

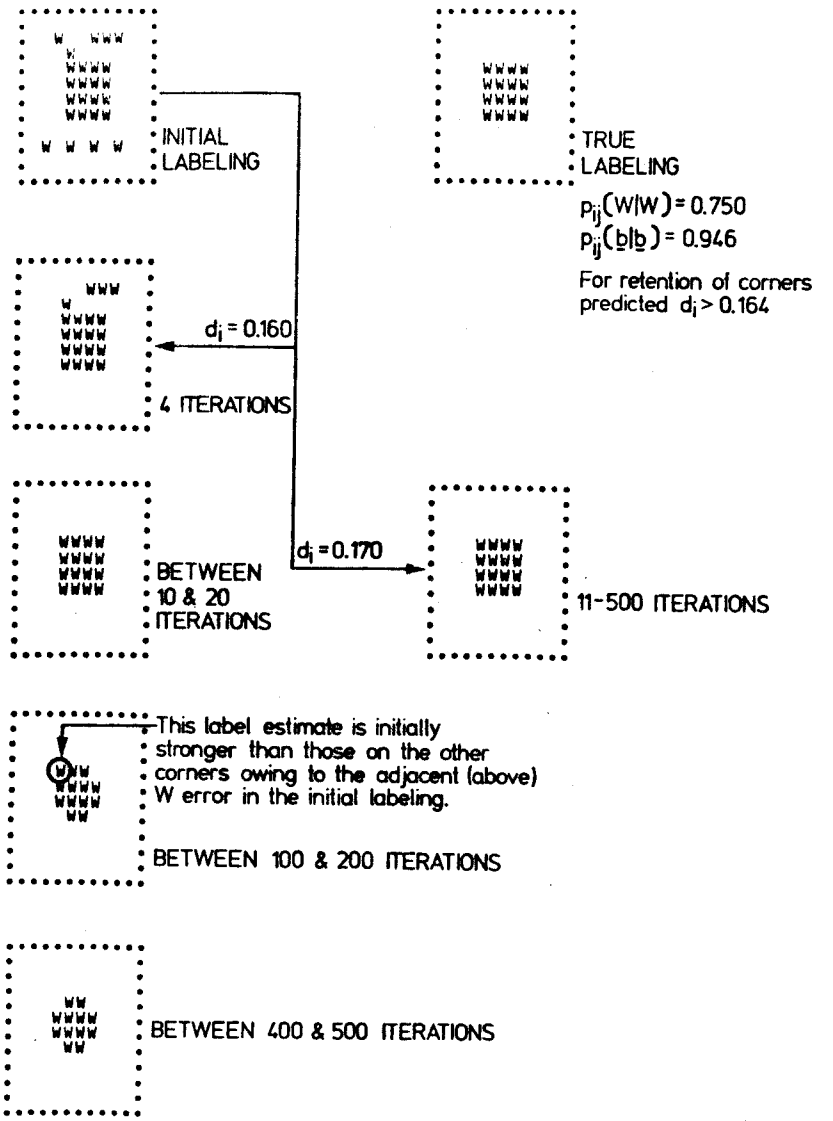


Figure D-2. Using the prediction of equation (15) to avoid loss of corner W labels.

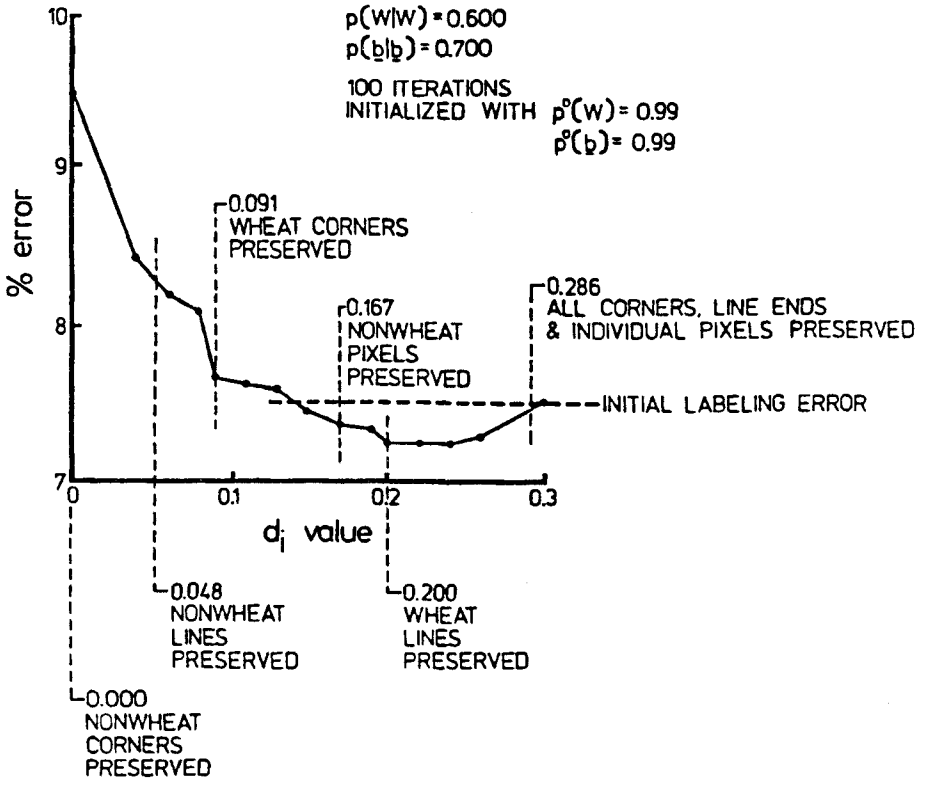


Figure D-3. Remaining labeling error as a function of the central pixel neighbor weight d_i, after 100 iterations of relaxation on the 40x100 pixel small image. (100 iterations are sufficient to achieve the final labeling.) The compatibilities have been chosen as discussed in the text.

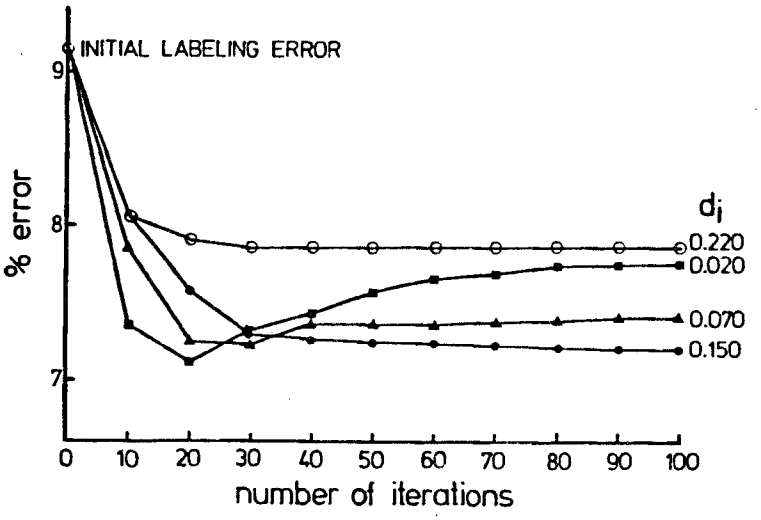


Figure D-4. Labeling error vs. number of iterations when relaxation is applied to the 117x196 pixel image, using several values of d_i.

until consistency between spatial, spectral and ancillary information is achieved. Consider the relaxation algorithm:

$$\underline{P}_i^{k+1}(\lambda) = \Lambda^k \underline{P}_i^k(\lambda)$$

$$\Lambda^k = \begin{bmatrix} \frac{Q_i^k(\lambda_1)}{D_i^k} & & 0 \\ & \frac{Q_i^k(\lambda_2)}{D_i^k} & \\ 0 & & \frac{Q_i^k(\lambda_m)}{D_i^k} \end{bmatrix}$$

$$D_i^k = \sum_\lambda P_i^k(\lambda) Q_i^k(\lambda) \ , \qquad Q_i^k(\lambda) = T\underline{q}^k(\lambda)$$

where $\underline{P}_i^k(\lambda)$, $\underline{Q}_i^k(\lambda)$, D_i^k, T and $\underline{q}^k(\lambda)$ are the same as we defined earlier. Now, let $\Psi_i(\lambda) = 1 + \beta[m\Phi_i(\lambda) - 1]$ be an appropriate likelihood which determines ancillary data influence for the ith pixel with respect to label λ. In this equation $\Phi_i(\lambda)$ is the probability that λ is the correct label for the ith pixel in view of the available ancillary data, m is the number of possible labels, and β is a parameter that adjusts the degree of supervision. As we mentioned, $Q_i^k(\lambda)$ first is modified by $\Psi_i(\lambda)$.

Let $\underline{R}_i^k(\lambda) = \underline{Q}_i^k(\lambda) \cdot \Psi_i(\lambda)$ ..... (18)

where $1 - \beta \leq \Psi_i(\lambda) \leq 1 + \beta(m-1)$ ..... (19)

Therefore

$$P_i^{k+1}(\lambda) = \frac{P_i^k(\lambda) R_i^k(\lambda)}{\sum_\lambda P_i^k(\lambda) R_i^k(\lambda)} \qquad \text{..... (20)}$$

$$D_i^k = \sum_\lambda P_i^k(\lambda) R_i^k(\lambda)$$

To test this scheme multispectral Skylab data from northeast of the Vallecito Reservoir region in the Colorado Rockies was classified into a number of tree species using maximum likelihood classification. Then the classification map so produced was re-arranged for simplicity into the two categories of spruce fir and others. For the region elevation

data as well as a probability model for the occurrence of spruce fir vs. elevation are available. Consequently, elevation was chosen as an ancillary data variable. The result of the test is shown in Figure D-5.

6. Convergence and Fixed Points of Supervised Relaxation

When the supervised relaxation process has converged to a fixed point,

$$\underline{P}_i^{k+1}(\lambda) = \underline{P}_i^{k}(\lambda) = \underline{P}_i(\lambda)$$

where $\underline{P}_i(\lambda) = \begin{bmatrix} 0 \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ 1 \\ 0 \\ \cdot \\ \cdot \\ \cdot \\ \cdot \end{bmatrix} \leftarrow \ell^{th}$ row

As before, we can write:

$$\begin{bmatrix} 1 - \frac{R_i(\lambda_1)}{D_i} & & 0 \\ & 1 - \frac{R_i(\lambda_2)}{D_i} & \\ 0 & & 1 - \frac{R_i(\lambda_m)}{D_i} \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ \cdot \\ \cdot \\ 1 \\ 0 \\ 0 \\ 0 \\ \cdot \\ \cdot \\ 0 \end{bmatrix} = 0$$

$$\rightarrow D_i = R_i(\lambda_\ell) \neq 0, \quad R_i(\lambda_\ell) - R_i(\lambda_j) > 0, \quad \forall_j \quad j \neq \ell \quad \ldots\ldots (21)$$

Equations similar to (15), (16) and (17), found for unsupervised relaxation, can be found for supervised relaxation and are given by:

$$d_i > \frac{\eta + \Psi_i(\lambda_2) - 1}{\eta + \Psi_i(\lambda_2)} \qquad \ldots\ldots (22)$$

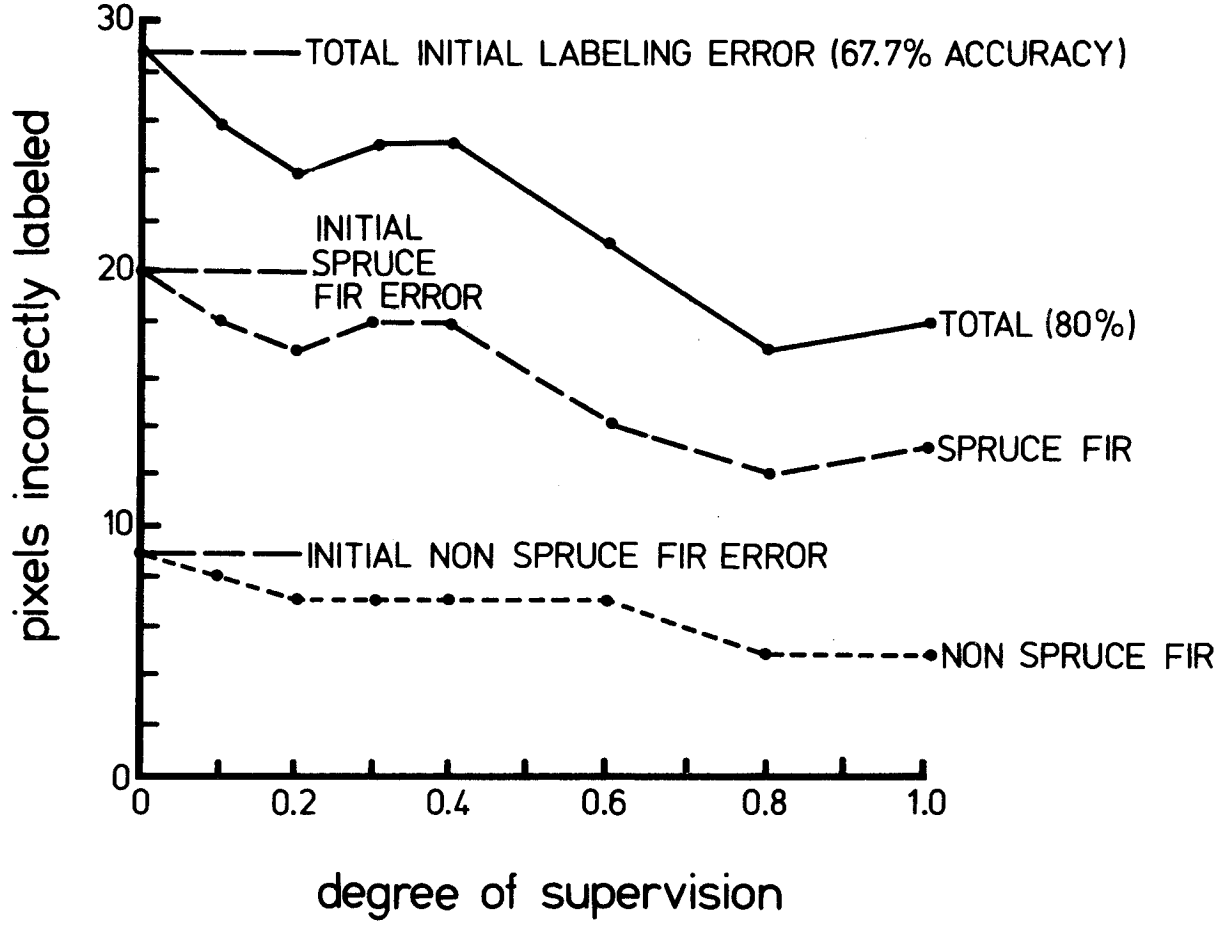


Figure D-5. Labeling error vs. degree of supervision, where the supervisory data used was probability of tree species occurrence vs. elevation. The elevation of each pixel had previously been associated with the multispectral response of that pixel.

This is the condition for a corner pixel to be preserved. Similarly, for a λ_1 line end in a λ_2 region:

$$d_i > \frac{3P(\lambda_2|\lambda_2) - P(\lambda_1|\lambda_1) - 3 + 2\Psi_i(\lambda_2)}{3P(\lambda_2|\lambda_2) - P(\lambda_1|\lambda_1) - 1 + 2\Psi_i(\lambda_2)} \quad \ldots\ldots (23)$$

and for an isolated pixel to be preserved:

$$d_i > \frac{2P(\lambda_2|\lambda_2) - 2 + \Psi_i(\lambda_2)}{2P(\lambda_2|\lambda_2) - 1 + \Psi_i(\lambda_2)} \quad \ldots\ldots (24)$$

7. Modified Probabilistic Relaxation Algorithm

As we said, our objective has been to use all available ancillary information to reduce the initial labeling ambiguity. Several new algorithms have been considered, and we shall describe some of them now. Recall the relaxation algorithms which were used in [12] and [13] and discussed above:

$$P_i^{k+1}(\lambda) = \frac{Q_i^k(\lambda)P_i^k(\lambda)}{\sum_j Q_i^k(\lambda)P_i^k(\lambda)} \quad \ldots\ldots (25)$$

$$Q_i^k(\lambda) = \sum_j d_j \sum_{\lambda'} P_{ij}(\lambda|\lambda')P_j^k(\lambda') \quad \ldots\ldots (26)$$

where the various parameters are as defined previously. Now, it may be argued that an equally logical means for computing $P_i^{k+1}(\lambda)$ (the probability that λ is the proper label for pixel i at k+1th iteration) is

$$\frac{Q_i^k(\lambda)}{\sum_\lambda Q_i^k(\lambda)} = Q_i^k(\lambda)$$

Since $Q_i^k(\lambda) = \sum d_j \sum_{\lambda'} P_{ij}(\lambda|\lambda')P_j^k(\lambda')$

one can write:

$$P_{ij}(\lambda|\lambda')P_j(\lambda') = P_{ij}(\lambda,\lambda') \quad \ldots\ldots (27)$$

$$\sum_{\lambda'} P_{ij}(\lambda,\lambda') = P_{ij}(\lambda) \quad \ldots\ldots (28)$$

Note that $P_{ij}(\lambda)$ still depends on pixel j, one of the neighbors of pixel i, since it is obtained by summing over the labels of pixel j. Our problem is to estimate initial label probabilities iteratively, and we hypothesize that by proper use of spatial and any other source of ancillary information that these estimates converge to the true values. To compute a new estimate $P_i^{k+1}(\lambda)$, we suggest the simplest way is to make an arithmetic average over all neighbors. So, by using arithmetic average

$$P_i(\lambda) = \sum_j d_j P_{ij}(\lambda) \qquad \ldots\ldots (29)$$

Now

$$\sum_j d_j P_{ij}(\lambda) = \sum_j d_j \sum_{\lambda'} P_{ij}(\lambda,\lambda') = \sum_j d_j \sum_{\lambda'} P_{ij}(\lambda|\lambda') P_j(\lambda') \quad \ldots\ldots (30)$$

By letting $P_i^{k+1}(\lambda)$ be our estimate at k+1th iteration, we can write

$$P_i^{k+1}(\lambda) = \frac{Q_i^k(\lambda)}{\sum_\lambda Q_i^k(\lambda)} = Q_i^k(\lambda) \qquad \ldots\ldots (31)$$

Note that $\sum_\lambda Q_i^k(\lambda) = 1$ provided $\sum_j d_j = 1$

If $P_{ij}(\lambda|\lambda')$ is not dependent on i and j, then equation (31) in matrix-vector notation can be written as:

$$\begin{bmatrix} P_i^{k+1}(\lambda_1) \\ P_i^{k+1}(\lambda_2) \\ \vdots \\ P_i^{k+1}(\lambda_m) \end{bmatrix} = \begin{bmatrix} P(\lambda_1|\lambda_1) & P(\lambda_1|\lambda_2) & \ldots & P(\lambda_1|\lambda_m) \\ P(\lambda_2|\lambda_1) & P(\lambda_2|\lambda_2) & \ldots & P(\lambda_2|\lambda_m) \\ \vdots & & & \\ P(\lambda_m|\lambda_1) & P(\lambda_m|\lambda_2) & \ldots & P(\lambda_m|\lambda_m) \end{bmatrix} \begin{bmatrix} \sum_j d_j P_j^k(\lambda_1) \\ \sum_j d_j P_j^k(\lambda_2) \\ \vdots \\ \sum_j d_j P_j^k(\lambda_2) \end{bmatrix} + \begin{bmatrix} d_i P_i^k(\lambda_1) \\ \\ \\ d_i P_i^k(\lambda_m) \end{bmatrix}$$

The supervised form of (31) is

... (32)

$$P_i^{k+1}(\lambda) = \frac{Q_i^k(\lambda)\Psi_i(\lambda)}{\sum_\lambda Q_i^k(\lambda)\Psi_i(\lambda)} \qquad \ldots\ldots (33)$$

where $\Psi_i(\lambda) = 1 + \beta[N\Phi_i(\lambda) - 1]$ as we defined before and the supervised form of original algorithm is

$$P_i^{k+1}(\lambda) = \frac{Q_i^k(\lambda)\Psi_i(\lambda)P_i^k(\lambda)}{\sum_\lambda Q_i^k(\lambda)\Psi_i(\lambda)P_i^k(\lambda)} \quad \ldots\ldots (34)$$

The performance of the two supervised relaxation algorithms will be demonstrated using a forestry classification example. Multispectral Skylab data northeast of the Vallecito Reservoir region in the Colorado Rockies was classified into two categories of spruce fir (Engleman spruce and subalpine fir) and other, after which the ambiguity reduction algorithms were applied. The performance of the two algorithms is given in Figure D-6. The new algorithm has an improved performance over that of the original algorithm.

A simple non-iterative algorithm which was considered in the following

$$P_i(\lambda) = \frac{Q_i(\lambda)\Psi_i(\lambda)}{\sum_\lambda Q_i(\lambda)\Psi_i(\lambda)} \quad \text{or} \quad \max_\lambda Q_i(\lambda)\Psi_i(\lambda) \quad \ldots\ldots (35)$$

It is obvious that the label probability $P_i(\lambda)$ is not a function of iteration, k. In equation (35) effective use of spatial and available ancillary information reduces the initial ambiguity. Again the performance of this new non-iterative scheme was compared to that of a supervised relaxation algorithm [13] where the final labeling was achieved after 80 iterations.

The result is shown in Figure D-7. Note that the performance of the modified algorithm equals or exceeds that of the original one and may suggest the use of a supervised non-iterative approach for reduction of label ambiguity.

8. Conclusion

In the probabilistic relaxation process three new sources of information may be utilized: the initial labeling probability distribution, spatial context and other ancillary data. Since there is always ambiguity and error as a result of the initial labeling, the new sources of information may be useful in reducing both the ambiguity and errors. Several algorithms have been studied for this purpose. So, our new

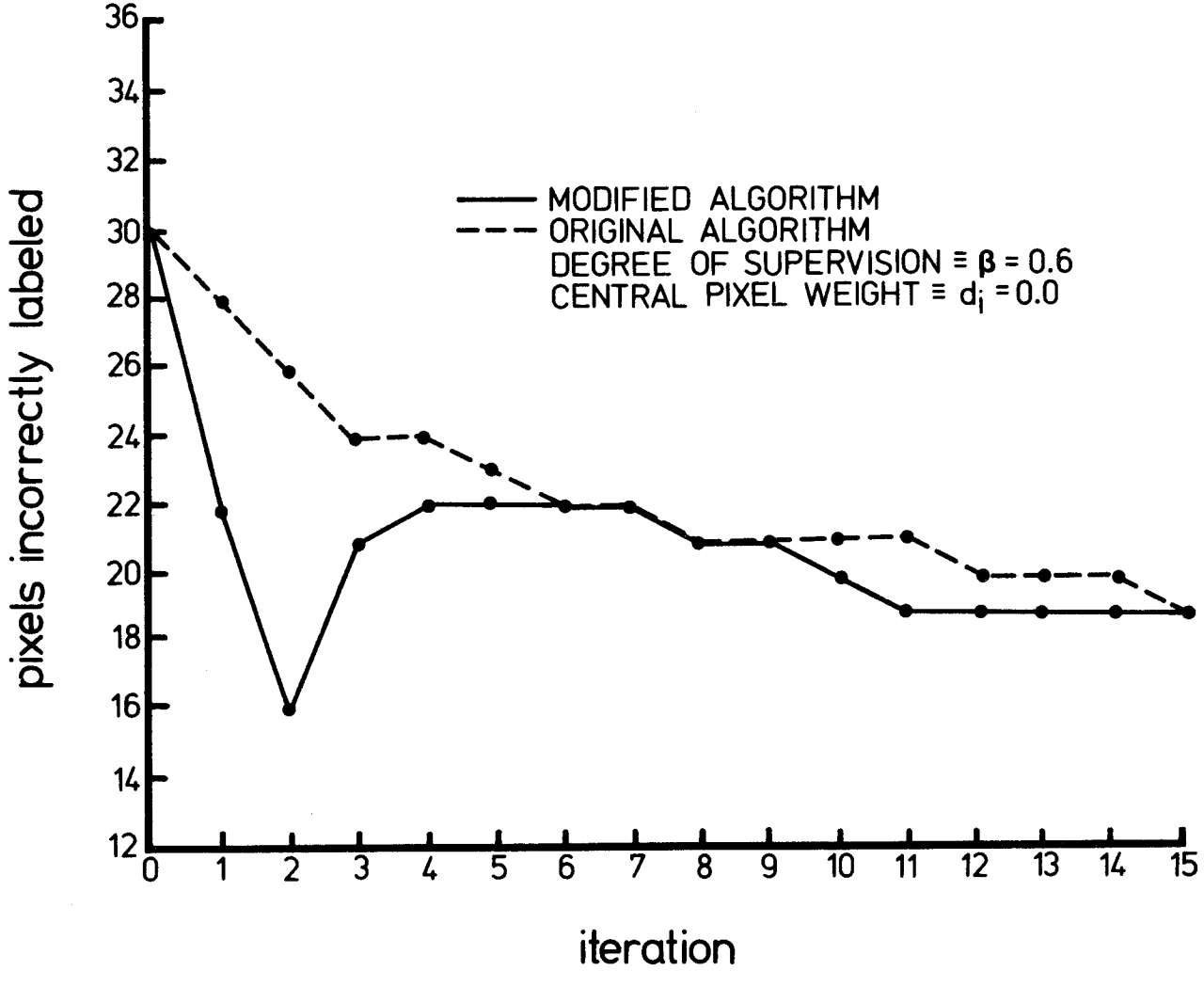


Figure D-6. Comparison of supervised original and modified relaxation algorithm vs. iteration number.

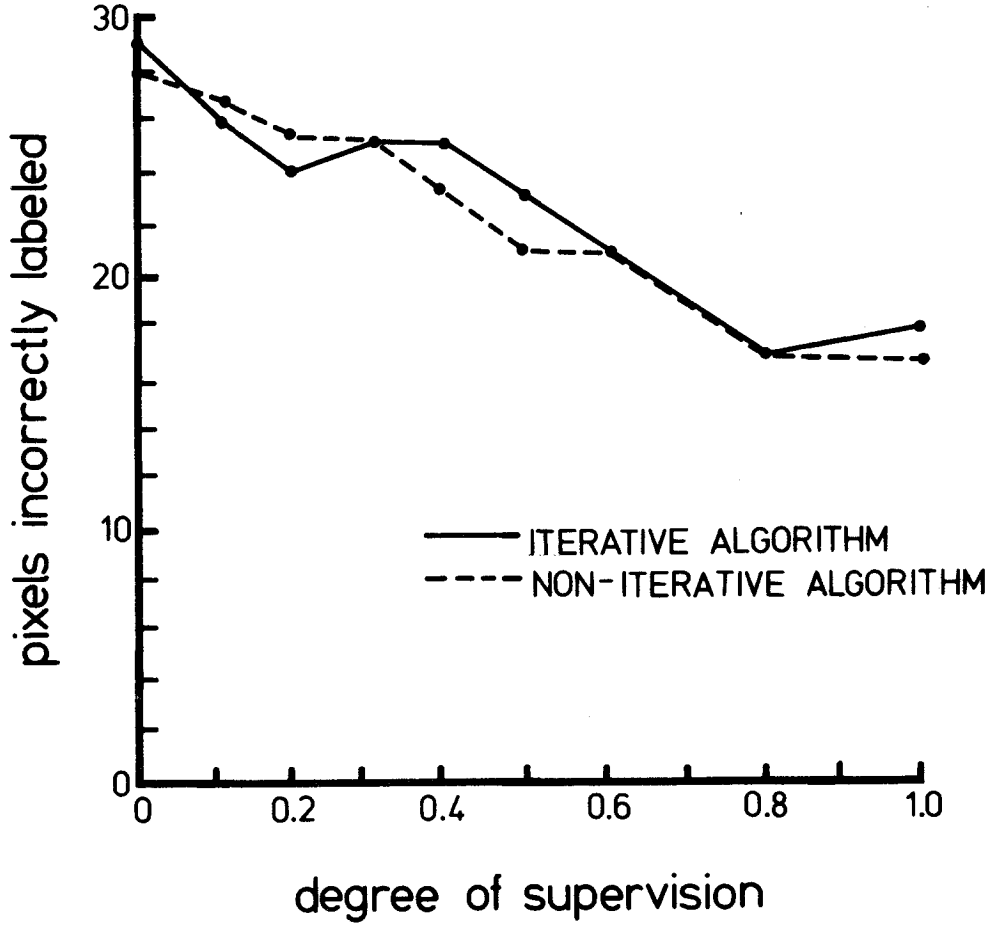


Figure D-7. Labeling error vs. supervision for the original iterative algorithm (80 iterations) and a non-iterative modification.

estimate at each iteration is a function of initial label probability, spatial, and possibly ancillary information. This function has had different forms only because there has not been any criterion of optimality in regard to estimation of label probabilities.

As mentioned before, spatial information is a very important factor in the reduction of ambiguity and the improvement of final labeling accuracy. Spatial information has been incorporated by utilizing the compatibility coefficients $r_{ij}(\lambda|\lambda')$ or $P_{ij}(\lambda|\lambda')$, label probabilities of neighboring pixels, and weighting coefficients d_i. Rather than attempt to seek an optimal way to estimate the d_i, estimation means based upon expected scene geometries has been provided. The third source of information which may be available to us is the ancillary data, and the supervised technique allows relaxation to be controlled from any pixel-oriented external source. There is assumed to be an optimum value of degree of supervision (β) that should be found; the value probably depends on the quality of the ancillary data and its relationship to the classes of interest.

References

1. Rosenfeld, A., R. Hummel and S. Zucker. 1976. Scene Labeling by Relaxation Algorithms. IEEE Trans. Sys. Man, Cyber. SMC-6(6):420-433.

2. Zucker, S., E. Krishnamurthy and R. Haar. 1978. Relaxation Processes for Scene Labeling: Convergence, Speed and Stability. IEEE Trans Sys. Man, Cyber. SMC-8(1):41-48.

3. Zucker, S., R. Hummel and A. Rosenfeld. 1977. An Application of Relaxation Labeling to Line and Curve Enhancement. IEEE Trans. Comp. C-26:394-403, plus correction, pp. 900-929.

4. Schachter, B., A. Lev, S. Zucker and A. Rosenfeld. 1977. An Application of Relaxation to Edge Reinforcement. IEEE Trans. Sys. Man, Cyber. SMC-7(11):813-816.

5. Peleg, S. and A. Rosenfeld. 1978. Determining Compatibility Coefficients for Curve Enhancement Relaxation Processes. IEEE Trans. Sys. Man, Cyber. SMC-8(7):548-555.

6. Lev, A., S. Zucker and A. Rosenfeld. 1977. Iterative Enhancement of Noisy Images. IEEE Trans. Sys. Man, Cyber. SMC-7(6):435-442.

7. Eklundh, J., H. Yamamoto and A. Rosenfeld. 1978. Relaxation Methods in Multispectral Pixel Classification. Technical Report 662. Computer Science Center, University of Maryland, College Park, MD.

8. Eklundh, J. and A. Rosenfeld. 1978. Convergence Properties of Relaxation. Technical Report 701. Computer Science Center, University of Maryland, College Park, MD.

9. Peleg, S. 1979. Monitoring Relaxation Algorithms Using Labeling Evaluation. Technical Report 842. Computer Science Center, University of Maryland, College Park, MD.

10. Zucker, S. and J. Mohammed. 1978. Analysis of Probabilistic Relaxation Labeling Processes. Proc. 1978 IEEE Conf. Pattern Recognition and Image Processing, Chicago, IL, pp. 307-312.

11. Peleg, S. 1979. A New Probabilistic Relaxation Scheme. Proc. 1979 IEEE Conf. Pattern Recognition and Image Processing, Chicago, IL, pp. 337-343.

12. Richards, J.A., D.A. Landgrebe and P.H. Swain. 1981. Pixel Labeling by Supervised Probabilistic Relaxation. IEEE Trans. Pattern Analysis and Machine Intelligence. PAMI-3. Laboratory for Applications of Remote Sensing, Purdue University, West Lafayette, Indiana. LARS Technical Report 022580.

13. Richards, J.A., D.A. Landgrebe and P.H. Swain. 1981. On the Accuracy of Pixel Relaxation Labeling. IEEE Trans. Sys. Man and Cyber. SMC-11(4). Laboratory for Applications of Remote Sensing, Purdue University, West Lafayette, Indiana. Technical Report 030180.

14. Peleg, Samuel. 1980. A New Probabilistic Relaxation Scheme. IEEE Trans. Pattern Analysis and Machine Intelligence. PAMI-2(4).

DISTRIBUTION LIST: VOLUME III

NASA Johnson Space Center

SA: W. Rice
SK: R. Erb
SG: R. MacDonald
SH: J. Erickson
SG3: F. Hall
C. Hallum
R. Heydorn
A. Houston
R. Juday
D. Pitts
M. Steib
G. Badhwar
A. Wacker
SH2: M. Trichel
SH3: J. Dragg
R. Bizzell
R. Eason
JM6: Technical Library (4)
AP: Public Affairs Office

NASA Headquarters

ER-2: P. Thome

NASA Scientific and Technical Information Facility

Q. Holmes:	ERIM
R. Horvath:	ERIM
W. Malila:	ERIM
R. Cicone:	ERIM
T. Minter:	LEMSCO
C. Hay:	Univ. Calif.-Berkeley
F. Ulaby:	Univ. Kansas
J. Smith:	Colorado State Univ.
V. Salomonson:	NASA-GSFC
J. Barker:	NASA-GSFC
W. Alford:	NASA-GSFC
F. Billingsley:	JPL
H. Maurer:	NASA Wallops
L. Guseman:	Texas A&M Univ.
W. Coberly:	Univ. Tulsa
P. Odell:	Univ. Texas
H. Decell:	Univ. Houston
E. Poole:	IBM

Additional volumes available upon request from Purdue/LARS.

Vol. I. Field Research on the Spectral Properties of Crops and Soils
Vol. II. Research in the Application of Spectral Data to Crop Identification and Assessment
Vol. III. Data Processing Research and Techniques Development
Vol. IV. Computer Processing Support