Automatic Interpretation of LANDSAT Images Using
Context Sensitive Region Merging

by

Dale William Starr

B.Math, University of Waterloo, 1974

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
in
THE DEPARTMENT OF COMPUTER SCIENCE

We accept this thesis as conforming
to the required standard

THE UNIVERSITY OF BRITISH COLUMBIA
April, 1976

© Dale William Starr, 1976
In presenting this thesis in partial fulfilment of the requirements for
an advanced degree at the University of British Columbia, I agree that
the Library shall make it freely available for reference and study.
I further agree that permission for extensive copying of this thesis
for scholarly purposes may be granted by the Head of my Department or
by his representatives. It is understood that copying or publication
of this thesis for financial gain shall not be allowed without my
written permission.

Department of Computer Science

The University of British Columbia
2075 Wesbrook Place
Vancouver, Canada
V6T 1W5

Date Sept. 3, 1976
Abstract

Automatic interpretation of images from Earth Resources Technology Satellite-1 (ERTS-1, now called LANDSAT) can be used in a variety of applications with considerable accuracy. Most systems, however, classify strictly on a point by point basis, making no use of any spatial knowledge. Standard photo-interpretation techniques are combined with some techniques from artificial intelligence to produce an increase in accuracy over a point-by-point classification method. Traditional classification methods are used to obtain an initial segmentation of the image. Then, a controlled region merging process allows the regions with unambiguous interpretations to influence the interpretation of neighbouring ambiguous regions, thereby introducing considerable context sensitivity into the interpretation process. Results are given of an experiment to interpret areas of different forest cover.
Acknowledgement

I would like to thank Dr. Alan Mackworth for his many ideas, his patience, and the financial support provided by his NRC Operating Grant A9281. I would also like to thank Dr. Bary Pollack for reading this thesis and Dr. Murtha of the Faculty of Forestry for providing the ground truth data so essential to this study.
# Table of Contents

Introduction ........................................................................................................ 1
Objectives of Remote Sensing ........................................................................ 7
Uses of Remote Sensing .................................................................................. 9
Background ....................................................................................................... 15
Region Merging ............................................................................................... 19
The Method ....................................................................................................... 22
Results .............................................................................................................. 36
Conclusions ..................................................................................................... 41
Beyond the Present ......................................................................................... 43
Bibliography ..................................................................................................... 44
Appendix 1: Description of Programs ............................................................ 46
  Read From Tape ............................................................................................ 46
  Print Picture .................................................................................................. 48
  Get Statistics ................................................................................................. 51
  Ellipse .......................................................................................................... 53
  Ellipsoid ....................................................................................................... 56
  Initial Classifier ............................................................................................ 59
  The Region Finder ......................................................................................... 61
  The Region Merger ....................................................................................... 66
List of Figures

1. The LANDSAT Satellite and its Field of View .................. 2
2. A Sample LANDSAT Image .......................................... 4
3. Ground-truth Data .................................................. 24
4. Band 5-6 Ellipses of Concentration .............................. 27
5. Band 5-7 Ellipses of Concentration .............................. 28
6. Band 6-7 Ellipses of Concentration .............................. 29
7. Initial Classification with no Ambiguous Regions .............. 38
8. Initial Classification with Ambiguous Regions ................. 39
9. Final Partition After Merging ..................................... 40
List of Tables

I. Suggested Level One Classification Features .............. 7
II. Summary of LANDSAT Uses .................................. 13
III. Initial Classification with Ambiguous Regions .......... 32
IV. Initial Classification with no Ambiguous Regions ...... 36
V. Overprinted Characters for each Reflectance Range ...... 50
Introduction

A picture can be stored on the computer as an array on an \( N \times M \) grid \( G \). (Each point on the grid is called a picture element or pixel). The array is a function on \( G \) whose value at each point is some measure of light at that point. If a simple black and white photograph were stored the value would simply be the brightness of the point. A colour photo would require more information so that the brightness of the colour and light intensity of the point are known for the basic colours. In this way, with the right equipment, one is able to accurately reproduce a picture from the digital data. This is what is done for satellite pictures.

In 1972 Earth Resources Technology Satellite 1 (ERTS-1, now called LANDSAT) was launched by NASA in the United States as part of a program to demonstrate the feasibility and practicality of remote sensing from space. The satellite has an apogee of 917 km., a perigee of 898 km., with a rotational period of 103 minutes. This orbit covers the earth's surface from a latitude of 81 degrees S to 81 degrees N, repeating the coverage every 18 days at the same local time.

The basic component of the satellite, as shown in Figure 1 (from Gower and Daniel (13) and Bernstein (4)), is a multispectral scanner which consists of an oscillating mirror and an optical system which reflect and direct scene radiance values onto a detector array that is sensitive to wavelengths in four spectral bands as follows:
Figure 1

The LANDSAT Satellite and its field of view.
Band 4: Green portion of spectrum .5 - .6 micrometers
Band 5: Red portion .6 - .7
Band 6: Near infrared .7 - .8
Band 7: Further infrared .8 -1.1

Six scan lines are simultaneously swept in each spectral band with each oscillation of the mirror, the 11.56 degree field of view covering a swath of 185 km. on the earth's surface. The output of the detectors is in six bit per word digital form for transmission to ground receiving stations. Each point of the scene has associated with it four pieces of data: the radiance values of the point in each of the spectral ranges defined above. With the LANDSAT-1 equipment a resolution of 4600 square metres per pixel is achieved; plans for future satellites reduce this significantly. A sample LANDSAT image is given in Figure 2, showing southern Vancouver Island and the Strait of Georgia.

Unfortunately, there are many sources of error for the multispectral scanner which must be corrected before the data can be used. Some of these are (from Bernstein (4)):

**Altitude:** Differences of the spacecraft from its nominal altitude cause distortions of scale which must be corrected.

**Attitude:** The axis of the platform is maintained with one axis normal to the surface of the earth and another parallel to the velocity vector of the craft. Distortions result when perturbations to the correct
Figure 2

A Sample LANDSAT Image
headings are experienced.

**Scan Skew:** While the mirror is scanning, the spacecraft is moving along the ground track. Therefore the swath obtained is not normal to the ground track vector but skewed slightly.

**Velocity:** If the velocity changes from the nominal value, the ground track covered by a given number of successive mirror sweeps changes, producing distortions along the ground track.

**Earth Rotation:** As the mirror sweeps, not only does the spacecraft move, but the earth also rotates beneath it. Therefore, there is a gradual westward shift of the ground swath being scanned, causing along scan distortions.

**Mirror Sweep:** Imperfections in the controlling mechanism of the mirror cause the scanning rate to vary, creating along scan distortions.

**Perspective:** For most applications the desired images represent the projection of points on the earth on a plane tangent to the earth at the point directly below the satellite. The data, however, represent a perspective projection as a result of the curvature of the surface.

**Atmospheric Effects:** The light received by the satellite is dispersed and attenuated by the atmosphere. Compensation for this effect is difficult, but if it can be done it enhances the
accuracy of the information received.

Tapes received from the Canadian Centre for Remote Sensing have various corrective operations performed on them to compensate for these imperfections. The data also is put into an eight bit word format and the structure of the data on the tape is completely changed for easier use.
Objectives of Remote Sensing

The objective of automatic interpretation is to produce a meaningful partition of the scene presented, placing each pixel into one of a number of classes, depending on the application of the study. For example, in the periodic tracing of the growth of a city it may be sufficient to have only three categories for a gross study — say residential, industrial-commercial, and other, while a more in-depth study might delve into more specific subclasses of these larger groupings.

In 1971 the Conference on Land Use Information and Classification (15) provided some classification objectives. The participants divided their classes into two levels, the first as given in Table I.
I. Urban and built-up.  
II. Transportation, communication and utilities  
III. Farming  
IV. Grassland  
V. Forest land  
VI. Extractive (mining)  
VII. Water  
VIII. Marshland  
IX. Tundra  
X. Barren land  
XI. Permanent snow fields

Table I
Suggested Level One Classification Features

Level two features were subsections of those of level one. For example, urban was split into residential, commercial, industrial, services, recreational, transportation and other. Similar subsections were proposed for the other groupings where appropriate.

The basic aim was to achieve level one interpretation through the use of satellite imagery alone and level two through a combination of satellite and air-photo techniques. These aims were slightly conservative though, since success has been achieved in identifying most level two classes solely from satellite imagery at much less cost than if aircraft had been used.
Uses of Remote Sensing from Satellites

One of the main uses of remote sensing from space lies in the area of land-use inventories. Present data that is used to construct an inventory map may be derived from different years and collected by diverse methods. This obviously leads to inaccuracy and high costs, not to mention the time it takes to gather such information. This is especially true in developing countries which may not have the facilities for such data gathering, but to whom the results may be of extreme importance. With an effective system of satellite remote sensing a resource inventory can be done much more quickly and at less cost than by conventional methods with the added benefit of all the data coming from a single source.

One area where this could be important to Canada is in predicting wheat crops throughout the world so that the export market can be estimated. Zwarun (24) reported that in 1975 most predictions were of a large Soviet crop, causing world prices to fall and farmers to make seeding plans in anticipation of the Russian crop. American scientists; however, using satellite data correctly predicted that the Soviet crop would be a failure. Had the Canadians used the same remote sensing data more wheat could have been planted in anticipation of increased export demands.

This ability to determine yields should be of major importance to the world as a whole since most crops can be differentiated using satellite data. In addition to this, the health of most crops also is detectable, enabling blights to be
controlled before they spread over too great an area (see references 2 and 3).

The forest industry also should benefit from applications of remote satellite sensing. Inventories of available wood will be made much less costly and easier to obtain than it presently is, especially in inhospitable mountain areas. Forest fire damage can be quickly mapped by satellites, providing companies with quick estimates of replanting costs.

In much the same way urban land-use can be obtained from LANDSAT data, as demonstrated by Ellefsen et al. (9) and Todd et al. (22). The repetitive coverage is useful for tracing the growth of cities and could prove useful in city planning, although cities usually have much of this data readily available.

Hoffer et al. (16) have demonstrated the effectiveness of providing areas of interest to mining companies even in heavily vegetated areas. While they can't detect minerals they can identify certain geological features which give clues to high potential areas. This could be of particular importance in the Canadian north where vast areas of uncharted land could be viewed by satellite to provide mining concerns with areas where further study should be carried out.

Another use in the geological area is in earthquake studies. Satellite photos provide excellent views of large areas making fault lines readily apparent. This might have use in planning cities so that major population centres or industrial complexes are not placed on or near high risk areas.
Volcanic disturbances also can be detected but in addition to this it may be possible to predict when activity will occur, as outlined by Friedman et al. (11). During periods of repose of active volcanoes, as well as during periods of eruption, a sizeable heat flux reaches the surface. Hopefully this can be monitored and used as a predictive clue to impending activity.

The current energy crisis and rapid increase in world oil and energy prices have had a great impact on the Canadian north in terms of fossil fuel exploitation and in development of hydroelectric projects. There exists a need for information on which to base decisions concerning routing of roads and pipelines. Resource satellites can provide this data in a short time and at a reasonable cost so that detailed mapping can be carried out in the areas of specific interest.

Clough et al. (7) report that a cost benefit study on remote sensing of sea ice forecasted savings to Arctic shipping of $4 million in 1975, rising to $100 million by 1990. This saving is predicting increased use of oil tankers through the Arctic, but if this is not undertaken considerable savings still could be obtained by wheat ships going through Churchill, Manitoba, where the navigation season is very short and any aids in getting through the ice as quickly as possible would be a major help. Gerson (12) has demonstrated that it is possible to map ice fields by computer processing of satellite pictures where the main difficulty lies in differentiating between cloud and the ice. The mapping of ice is not only useful to navigation but also can be used to plan many activities such as
offshore drilling or construction of artificial islands which require good knowledge of freeze-up or break-up conditions.

Ocean currents can be detected because of the differences in temperature between the current and the surrounding water. By studying this it should be possible to gain more knowledge into distributions of fish populations for more effective and ecologically sound fishing. Taking this further into the future it might prove useful, once farming of the sea becomes a reality, to know areas where the temperature is optimal for a certain crop to deliver its maximum yield.

Another area where satellite imagery could prove useful is in monitoring the amount of snow present in springtime so that runoff can be predicted to prevent flooding. At present though, the 18 day cycle time is probably too long for this purpose, but several satellites could be utilized.

Finally, some pollution monitoring could be accomplished through the use of satellite data. On rivers and lakes one can readily detect sources of pollution as a result of sudden changes in colour or temperature of the water. Oil spills from pipelines also could be detected in this way as the oil usually covers a fairly large area. This is presently done by aircraft which patrol the pipelines each week or so but could probably be done using LANDSAT data since only a change from normal has to be detected instead of a whole new classification.

Table 2, from Bernstein (4) summarizes some of the important uses for LANDSAT imagery.
Agriculture, Forestry
Crop census
Crop yield estimates
Identification of vegetation disease
Land use inventory

Hydrology
Water resource inventory
Fresh water source identification
Flood monitoring
Pollution monitoring of lakes and rivers

Oceanography, marine resources
Fish production
Ship routing
Sea state
Ice conditions

Geology
Geologic and physiographic mapping
Mineral exploration
Earthquake area studies
Glacier and volcanic studies
Shoreline erosion

Geography
Land use mapping
Physical geography
Cartography
Urban planning
Demography

Table II
Summary of LANDSAT Uses

As is readily observable the uses of remote sensing from space are many and varied and likely will increase as the resolution of the imagery increases. Their greatest impact may be felt in those countries which at present lack many of the data gathering facilities common in North America. But perhaps the most important use will lie in the area of food production, where satellites can be used to detect unused arable land, predict yields to predetermine distribution, spot blight areas, etc.
Clough (6) reports that the estimated cost of the LANDSAT program in the U.S. will be about $20 to $50 million per annum while the estimated potential benefits should exceed $1.4 billion per annum. As an example, minimizing flood damage may save $300 million and improving forecasts of availability of irrigation water may save $280 million each year in the U.S. Alone. This makes the use of such satellites not only feasible but extremely economically viable.
Background

Systems for automatically interpreting LANDSAT images generally can be divided into two basic groups - supervised and non-supervised classification systems.

Supervised classification requires some input by the experimenter to identify certain training areas in the scene. Each class under consideration has its own areas so that the necessary statistics can be calculated. The means and covariance matrix are calculated for the class within these areas so it is obviously of great importance that the training areas be as homogeneous as possible, consisting solely of that class; otherwise the statistics will not be as accurate as one would like. Once this is carried out for all classes the partitioning of the picture can be done by placing each point in that group which gives it the highest probability of membership. This usually is done using a maximum likelihood function.

Unsupervised classification usually employs cluster analysis (or factor analysis) to automatically group a given set of data into the most spectrally separable clusters using several wavelength bands. Different features of the earth's surface ideally would have their own unique spectral response, thereby creating distinct clusters of points belonging to each feature in the spectral space. Unfortunately, this is seldom the case as the clusters usually overlap to some extent, thereby causing errors in the classifier.

Many studies have used these approaches with considerable
success. Ellefsen et al. (9) and Todd et al. (22) carried out similar studies in urban land-use mapping from satellite data. Todd et al. used a cluster analytic approach with the number of classes set at 14. Results were said to be poor so it was necessary to use ground-truth data to reinforce the clustering approach. Overall over 90% of the points were correctly identified with most errors occurring near the urban-rural fringe where upper income residential areas were classified, not unnaturally, as grassy areas. Ellefsen et al. also used cluster analysis in their study with an overall accuracy of about 87%. Frequent errors occurred in misclassifying old residential areas as wooded, open space; obviously something which easily could be corrected if spatial knowledge was incorporated in the system.

Robertson (18) used a different approach from most systems. The image was recursively partitioned into blocks of image points such that each block contained points from a single class. These blocks were then classified as a whole, using texture and other spatial characteristics to influence the classification. Accuracy was increased slightly over a point by point classifier (with an average accuracy of 82%) at a cost of about ten times the computing time. Tests were carried out on both aircraft and satellite imagery but inexplicably the results from airphotos were only comparable to those from satellite, whereas in most cases airphotos resulted in greater accuracy because of the much smaller area covered by a single pixel.
Gupta et al. (14) also felt that point by point classification was less than optimal. They employed a boundary finder to produce homogeneous regions in the scene. The images under consideration were those of agricultural fields -obviously well suited to the method used since one would expect the boundaries to be relatively straight and therefore easily detectable. Once the closed regions were found classification was done using a maximum likelihood classifier and a minimum distance classifier. Results of the study were extremely good. On a point by point basis an error rate of only 4.1% was observed but this was reduced to 2% with their system. One must bear in mind; however, that the tests were carried out on aircraft imagery so the excellent accuracy is not all that unexpected.

Bajcsy and Tavakoli (1) carried out an interesting study to identify bridges, islands, rivers, and lakes from satellite imagery. They used a world model to describe the objects of interest. For example, a bridge has the spectral response of the land; it has a thin, elongated shape; it is connected to the land on its short sides and surrounded by water on its longer sides. Similarly, every object of interest to the study was described in the program. Initial segmentation of the picture was fairly simple, needing only to differentiate between land and water although it had to be quite sensitive in order to find evidence of bridges which were narrower than the pixel size. A skeleton operator was then used to find the shape of the regions in the picture which were then used in
connection with the model. An hypothesis was made about a figure and then compared with the model in an ordered way. For example, bridges were identified first and then removed from the picture so that rivers and islands can be identified. After all this was done their program was able to identify all the bridges they had hoped for, plus some others which they did not expect to find because of their small width.

The preceding study made the most use of context to interpret the scene. However, it had the disadvantage that in an unstructured scene one would be hard pressed to come up with an equivalent world model. However, scene analysis techniques from artificial intelligence seem to provide a good basis for use in satellite image interpretation.
Region Merging

Brice and Fenema (5) described a method of scene analysis involving the use of regions and region merging to reduce the number of small regions in a picture into larger, more meaningful pieces. (A region simply is a connected set of pixels). Initially the picture was partitioned into regions of equal grey value or radiance.

This; however, did not generally result in a partition allowing simple interpretation of the scene. Many false regions were created because of uneven illumination, noise, etc. A first approach to solving this would be to group points if their features are not too different. However, this may tend to produce regions that extend across the natural boundaries of the scene if two actual regions are not too different across the separating boundary.

Their approach; therefore, was to start with atomic regions and use global criteria to merge them. The first pass merged regions in such a way that the resulting boundary was shortened. Even if the boundary between two regions was weak, the two were joined only if their resulting boundary did not grow too quickly. More precisely, their so-called phagocyte heuristic was to merge adjacent regions R_i and R_j only if W/Pm > THETA1, where Pm = min(Perim_i, Perim_j), THETA1 is the threshold, and W is the length of the weak boundary between the two regions. (A weak boundary is defined as the boundary vector between two pixels having a difference in gray value less than some strength threshold).
This heuristic removed many of the superfluous regions from the picture, but many false separations still existed so another heuristic; the weakness heuristic, was applied. This joined two regions only if \( \frac{W}{I} > \Theta_2 \), where \( W \) was again the weak portion of the intersection \( I \) of the two regions. After these two heuristics have been applied most of the natural regions were present, with very few regions that weren't actually in the scene.

Feldman and Yakimovsky (10) have done much work on a semantics based region analyzer. While their aims were different (they were interested in understanding everyday scenes) the ideas seem to be readily applicable to LANDSAT imagery. A major problem in region analysis is that most systems use absolute criteria for deciding what constitutes a region. Feldman and Yakimovsky attempt to vary these criteria with context. For example, certain shades of green, brown and yellow might be merged into a single area of grass in one picture, while remaining completely separate entities in another, or even in different areas of the same picture.

As in Brice and Fenema, they began with atomic regions which were sequentially merged into larger regions. Initially they did a pass which merged regions based on a weakest boundary first criteria. In other words, the pair of regions with the weakest boundary was merged into one region. This process was stopped fairly early with conservative thresholds for merging so that it did not produce false merges.

The semantic controlled program was then called to produce
the final picture using various heuristics to choose the regions to merge. This process depended on semantic information known by the program which included such items as shape, position, boundary smoothness, etc. with some knowledge of the relationships between possible objects in the scene with respect to these features. Their final results were quite impressive in the simplicity and correctness, with the system able to correctly identify most objects in the scene with few false regions. Hopefully satellite imagery interpretation can be taken to such elegant heights.

Tennenbaum and Weyl (21) incorporated ideas from both the above to analyze everyday scenes. Their system used input from the experimenter to supply an interpretation when the system reached points of uncertainty. With this approach a user could quickly outline major regions of a picture which would then be used as an initial partition, especially in scenes which are too complex to process by computer alone.

While the above systems are used to analyze everyday scenes, the structures they produce are exactly what is needed for satellite imagery. Feldman and Yakimovsky take a region of highest confidence and assign to it the most likely interpretation. This allows the probability of adjacent regions to be changed by considering the inter-relationships between various possible interpretations. This is the idea I wished to apply to satellite photo-interpretation.
The Method

The aim was to be able to show that integration of some scene analysis ideas from artificial intelligence could be applied to the problem of classifying LANDSAT photos. Hopefully an increase in accuracy and a simpler picture would result over the point by point method. Dr. Murtha of the Faculty of Forestry provided a detailed ground-truth map of a forested area on Vancouver Island so this became the test area, with the aim of classifying regions of old growth, second growth, recent logging, and water. This ground-truth data provided training areas where the statistics of these four classes could be determined. This is necessary since a supervised maximum likelihood classifier is used.

In order to use this ground-truth data, it first must be transformed into machine readable form. This can be accomplished with the aid of a digitizer in the Mechanical Engineering Department. First though, some points must be established as ground control points. These are points which are readily identifiable on both the ground-truth data and the digital data. In this case there are several small lakes which stand out clearly from other features on a computer print-out of the scene (see Appendix 1, Print Picture). Therefore it is a simple matter to determine the exact position of a number of pixels in the picture matrix. Once the ground-truth data is put into matrix form it then becomes a matter of transforming the points in the ground truth to the corresponding point of the digital data.
It is possible on the digital data to find two ground control points which lie in the same row of the matrix. This defines a line on the visual ground truth data which can be used as a base line for digitizing.

Assume the area of interest covers $x$ cm. on the visual data and $y$ rows of pixels in the actual picture matrix. Then each row of the matrix corresponds to $x/y$ cm. of visual data. Therefore, in order to digitize the visual data a scan is started at the origin and sweeps across the base line, recording the position of each boundary encountered plus a number indicating the type of region to the right of the boundary. Once this is done for the base line the next sweep is made along a parallel line $x/y$ cm. above the last one, which corresponds to the next row in the matrix. This is done for as many rows as necessary, thereby providing a matrix of ground truth data. However, these points are determined by an $(x,y)$ position which is measured in inches from the origin. This must be transformed to the co-ordinate system used by the actual data. This transformation was done using a least squares fit providing results which were at most 1% off the correct values for the ground control points. The results are shown in Figure 3, showing only the area of the study.

Once this is accomplished it is possible to provide certain areas where the surface features are known, in order to establish means and the covariance matrix for the reflectance vectors for the features. Then, assuming a multi-normal distribution about the class means, a point is assigned to a
Figure 3
Ground-truth Data

- Second Growth
- Old Growth
- Recent Logging
- Water
particular class on the basis of a multi-normal probability function. In particular, there will be a function for each class, each taking the form as follows: (see Steiner (19))

\[ f(X,S) = \frac{1}{\sqrt{2\pi} |S|} \exp \left\{ -\frac{\sum \lambda_{ij} (x - \mu_i)(x - \mu_j)}{2|S|} \right\} \]

where \(|S|\) is the determinant of \(S\), which is the covariance matrix for the class.

\[
S= \begin{bmatrix}
\sigma_{11} & \sigma_{12} & \cdots & \sigma_{1m} \\
\sigma_{12} & \sigma_{22} & \cdots & \sigma_{2m} \\
\vdots & \vdots & \ddots & \vdots \\
\sigma_{1m} & \sigma_{2m} & \cdots & \sigma_{mm}
\end{bmatrix}
\]

\(\lambda_{ij}\) is the cofactor of \(S\) in the \(i\)th row and \(j\)th column. \(\mu_i\) and \(\mu_j\) are the means of the \(i\)th and \(j\)th elements of the feature vectors respectively.

The feature vector is simply the data known for each point of the picture. From the actual data, four pieces of information are known—the reflectance values in each of the four bands. Only bands five, six, and seven were used though, since the values for band four were extremely close for all pixels and provided little additional information.

In order to assign a point to a class, a traditional point by point method would determine the probability that the point belongs to each class and then assign it to the class which gave it the highest probability of membership. This basic technique was augmented as follows. If \(p_1\), \(p_2\), \(p_3\), and \(p_4\) are
the probabilities received by a point from the four probability functions, the greatest of them, \( p_{\text{max}} \), must be greater than some threshold before the point will be assigned to the class which produced \( p_{\text{max}} \).

More precisely, if \( p_{\text{max}} \geq k \frac{(p_1+p_2+p_3+p_4)}{100} \) (where \( k \) is the threshold value) the point is placed in the class producing \( p_{\text{max}} \). If not, it is assigned to a new class which depends on \( p_{\text{max}} \) and the second highest probability. These new classes (called ambiguous classes) are used when the classifier is not sure about the class in which to place a point. For example, if a point receives its highest probability of membership from class 1, but also receives a fairly high one from class 2, it is assigned to a class which contains all points which are likely class 1 or 2, but about which some doubt exists.

This method increases the number of classes from the basic four to a maximum of ten; specifically type 1, type 2, type 3, type 4, type 1 or 2, type 1 or 3, or type 2 or 3. There are other possibilities but one need include only those combinations for which some statistical overlap occurs. For example, it is not necessary to include classes which could be water (type 4) and any other type since this confusion seldom, if ever, occurred. Figures 4, 5, and 6 show this more clearly.

These three figures illustrate the extent of the statistical overlap. A probability ellipsoid is one way of showing the cluster of points belonging to a particular class when the feature vector has three elements. The motivation for this in two dimensions (which easily can be extended to more)
New growth

Recent logging

Old growth

Water

Figure 4

Band 5-6 Ellipses of Concentration
Figure 5

Band 5-7 Ellipses of Concentration
Figure 6

Band 6-7 Ellipses of Concentration

New growth
Recent logging
Old growth
Water
is to find a geometric representation of the concentration of a
given distribution about the centre of gravity \((m_1, m_2)\). In
other words, one wants a curve enclosing that point such that
if a mass is uniformly distributed over the area bounded by the
curve, it will have the same first and second moments as the
given distribution. In general that curve is undetermined so
one is restricted to finding an ellipse with the required
property. The ellipse shows a contour of constant \(f(x, S) = p\)
where \(p\) increases as one gets closer to the centre.

Given a certain mean vector and covariance matrix these
ellipsoids therefore show the general distribution of points
belonging to the class. The closer a point falls to the centre
of the ellipsoid, the more likely it is an element of that
class. These ellipsoids are defined by the equation
\[
\sum_{i,j} \lambda_{ij} x_i x_j = n+2
\]
where \(n\) is the dimension. In two dimensions this is
\[
\frac{1}{1-\rho^2} \left\{ \frac{(x - \mu_x)^2}{\sigma_x^2} - 2\rho \frac{(x - \mu_x)(y - \mu_y)}{\sigma_x \sigma_y} + \frac{(y - \mu_y)^2}{\sigma_y^2} \right\} = 4
\]
where \(\rho = \sigma_x / \sigma_y, \sigma_z\) (see Cramer, 4).

However, three dimensions are somewhat difficult to plot
so the projections of the ellipsoids on the three axes were
taken and that is what appears in the figures. They show
respectively the projection on the band 5-6 plane, the band 5-7
plane, and the band 6-7 plane.

As is readily observable there is much overlap among the
three types of forest cover, but especially between types 1 and 2 (second growth and old growth), and types 1 and 3 (second growth and recent logging). However, the area represented by water is completely separable from the others, indicating little confusion in classifying that particular feature.

The threshold value $k$, for classifying into the four main classes, was determined experimentally by classifying according to the above criteria and examining the results. With too high a value, very few points are placed in classes 1 through 4, although one could be almost certain they belonged to the assigned class. As the value is lowered, the number of points assigned to the first four groups increases and the percentage correctness drops. A value of 0 for $k$ would classify all points into the four main categories alone, which is just what a straight point by point classifier does. Clearly it is desirous to have enough points in the strong regions (regions whose classification is fairly certain) to provide a good base for merging ambiguous ones, but at the same time keep the correctness of the points so classified at a high level. The value of $k$ was finally set at 70%, with a resultant initial classification as follows.
In the initial classification, of all the points classified as being type 1, 63.1% were actually this type (according to the ground-truth), 27.1% were type 2, and 9.6% were type 3. Overall correctness for the points classified in the four main groups was 77%.

Once the initial classification is completed the regions must be found, with each region containing connected points which were assigned to the same class. The region finder is a fairly straightforward program which provides input to the merging program. It begins in the upper left corner of the matrix of points which are labeled by class number. The program then follows the outer boundary of all points with the
same class number until it reaches the point where it began. The list of boundary elements is written into a file for later input into the region merger. As the program goes around the boundary of a region it also enters a region number into a matrix which it uses to keep track of where it has been. All points within the region are filled in with the same region number and the program will proceed to search for the next region by moving along the row it is processing until a point is found which has no region number and therefore has not yet been touched. This is continued until the whole matrix is scanned, producing a list of all the regions in the scene.

The merging program is then called to get rid of the ambiguous regions in the best possible way. Several passes are used, each pass being slightly less stringent than its predecessor in its criteria for merging. In the first pass ambiguous regions will be merged into unambiguous ones only if

a) the ambiguous region received the highest probability of membership from the same class as the strong region;

b) the weak common boundary between the two regions is greater than some threshold (a weak boundary element will have the relative differences of each component of the feature vector for both points on the boundary less that some strength threshold);

c) if the two regions are merged, the new region formed by merger must remain a strong region (which means that its highest probability of
membership into one of the four groups must be greater than k% of the total probabilities, where k is the same threshold as established above); and

d) the average reflectance values of the three spectral bands used are close for both regions. (Close here means that the relative difference between each component of the average vector is less than some strength threshold).

The actual action of the program is as follows. Two linked lists of region descriptors are kept—the ambiguous regions and the strong regions. The program starts at the beginning of the ambiguous region list and works its way to the end for as many times as there are regions merged in the process. The first thing which is done is to obtain a list of all the strong regions which have some weak boundary elements in common with the ambiguous region. This is accomplished by tracing around the boundary of the region, checking each pair of opposite boundary points to see if the boundary between them is weak. If so, the number of weak elements in common for the ambiguous region and the strong region to which the point belongs is incremented by one. Once this is done for the complete boundary of the ambiguous region, the program will merge the ambiguous region with the strong region which is the most likely candidate.

This factor is determined by the closeness of the average feature vectors for the two regions and also by the percentage of the boundary which the two have in common. For example, if
two strong regions have average feature vectors which are equally close to that of the ambiguous region, the one chosen for merging will be the one which has a greater percentage of weak boundary elements in common. In this way the strong region which is most like the ambiguous one will be chosen to merge.

When the actual merging is done the program goes through the matrix in which each element reflects the region to which the particular point belongs and changes all points belonging to the ambiguous region to now belong to the newly expanded strong region. The boundary of the new region also is determined by tracing it out. As mentioned earlier, this process continues until no more regions are merged in a scan through the ambiguous region list.

A second pass is then carried out in which the same criteria for merging hold as in pass one, with the exception that the first criterion is relaxed somewhat. Whereas in pass one a weak region had to have its greatest probability of membership come from the same class as the strong region, pass two requires only the highest or second highest probability to come from the same class as the strong region. This then will merge any regions which were not "close" enough to a strong region in the first pass, but which are still fairly close in their average vectors to an adjacent strong region. This pass also is done so long as regions are merged.
Results

When a simple point by point classification method was used, placing each point in one of four classes (no ambiguous classes), a success rate of 71% was achieved as given below.

<table>
<thead>
<tr>
<th>Classified as</th>
<th>Correct classification (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>56.0</td>
</tr>
<tr>
<td>2</td>
<td>16.7</td>
</tr>
<tr>
<td>3</td>
<td>13.5</td>
</tr>
<tr>
<td>4</td>
<td>1.5</td>
</tr>
</tbody>
</table>

Table IV
Initial Classification with no Ambiguous Regions

The success figure may seem a little low, but this probably is a result of the considerable statistical overlap among the three types of forest cover. Nevertheless, the merging algorithm increased the accuracy to 79%, which represents a 28% decrease in the error rate.

A second positive result was the simplification of the final picture over the point by point classification. Typically about 150 regions were found using that technique in a picture of 2500 pixels, as shown in Figure 6, placing each point in one of the four main categories of old growth, second growth, recent logging, and water. Figure 7 shows the results when the ambiguous regions were included in the initial
classification, resulting in over 350 regions. The final partition, shown in Figure 8, resulted in about 70 regions.

The system consists of a number of programs mostly written in ALGCLW-F with a few FORTRAN service and plot programs. These programs are described in Appendix 1. The point by point classification for 2500 pixels requires about 10 seconds of CPU time on the IBM 370/168. The merging program takes about 55 sec. In total and 150k of storage, including the initial classification and region finding. These values increase significantly as the size of the picture being processed increases since with more regions the program will have to spend more time searching through lists. Therefore, if a large picture is to be processed it would be advisable to break it up into smaller pieces and process each one in turn.
Figure 7

Initial Classification with no Ambiguous Regions
Figure 8
Initial Classification with Ambiguous Regions
Figure 9

Final Partition After Merging
Conclusions

Even with the minimal semantic knowledge employed, a significant increase in accuracy over the point by point method was achieved by the region merger. One could only hope for a greater improvement in areas where there is more semantic knowledge available. For example, parks within a city would be classified, context-free, as agricultural areas in all likelihood, while the mere fact of their position precludes this possibility. For another example, a highway may become blended into surrounding vegetation as it becomes narrower than the area covered by one pixel, but if the program knew that a road should be continuing in the area, such points could be identified correctly since the pixels would be darker than the surrounding ones.

Of course, the amount of semantic knowledge one is able to include is heavily dependent on the picture domain being studied as well as the intended application, but one can only hope to improve accuracy with such techniques.

Future LANDSAT satellites should provide even better performance since the resolution will be finer, with each pixel covering less area. As this happens there is less chance of a pixel hitting two distinct features such as occurs when a road becomes narrower than the pixel size. Of course, with smaller pixels there will be much more data per unit area of the earth's surface and a corresponding increase in computing time. However, it would not always be necessary to look at individual pixels, but rather group them together into larger units, going
to pixel resolution only near boundaries and other areas where accuracy demands.
Beyond the Present

If one were to work on a picture where there is considerable semantic knowledge the program should be improved greatly by infusing some of this knowledge. For example, in mapping wheat fields on the prairies one can be fairly sure that the field will be rectangular with straight boundaries. This fact alone could be useful in influencing the interpretation of ambiguous regions within an area known fairly surely to be wheat.

Unfortunately, this knowledge is not always as easy to determine as it is when interpreting everyday scenes. In them for example, one might know that sky is above the trees, trees are usually found growing on grass or ground, and cars are found on roads. In many cases though, the little semantic information is known about the various elements of the scene covered by a satellite photo.

Possibly a cluster analytic approach may yield slightly better results than the supervised maximum likelihood classifier used. With cluster analysis the data forms its own statistical clusters while a supervised approach first determines the classes and then gathers the statistics, which may form some unnatural groupings. However, the semantic-based region segmentation approach would be equally as valid for such an initial classification.
Bibliography


12. Gerson, D.J. Computer estimation of the presence of sea ice


Appendix 1: Description of Programs

Read From Tape

This FORTRAN program is used to read the data from a portion of a LANDSAT tape to a file AREA2. First, the tape must be mounted by running the program DLIB:DATA and supplying the desired tape identifier. The tape then will be mounted under the pseudo device name *DLIB*.

The user is then asked to supply a starting row and column for the portion he wishes to save. It also asks for the size of the picture in terms of the number of rows and columns. A skip value is then requested. This value is used to indicate the spacing to be used when reading in the data. For example, a value of one will read in every point of the area, while a value of two reads in every second point of every second row, and so forth. No matter the value of this variable though, it does not affect the number of rows which are read in. In other words, if a size of 100 rows was specified, and a skip value of two was used, you would still have 100 rows of data input, which would represent 200 rows of the actual picture.

On the tape three files are first skipped. These files contain information records which generally are of little concern to the user. Details can be found in the ERTS Data Users Handbook, available for use in the Data Library. Then, the appropriate number of records are skipped so that the tape is positioned at the correct record of the picture.

As a check, for the first record of data read, the row
number from the tape is printed. This value is always found in bytes 71 and 72 of the first record for the row. Bytes 107 and 108 hold a number which indicates the start of data in each of the records for that row of the picture. Values previous to this position in the record are all zero.

For some reason, the values for band 6 occur two bytes earlier than band 5, while those of band 7 are 4 bytes earlier, so care must be taken to ensure that the correct data is read in for each pixel.

The records for each of bands 5, 6, and 7 finally are read into the LOGICAL*1 vectors PIXEL1, PIXEL2, PIXEL3 (using the MTS read routine) and the appropriate columns placed in array BUF, which is then output. Only 25 points are output to any line of the file AREA2 because of a limitation of ALGCLW-F. While I could have put the data into a sequential file of appropriate line length, ALGOLW-F doesn't allow you to read or write a line longer than 255 characters.
Print Picture

This FORTRAN program produces an overprinted character representation of a LANDSAT scene using data from band 6 since it seems to give the best visual results. Because of the dimensions of a square on a page of computer output each pixel is not printed or else the picture would appear longer than it actually is. More specifically, only six rows are printed per inch while ten columns are printed in the same space. Therefore when printing the picture, for every five rows of actual data, only three will be printed. A limit is also placed on the width of the picture produced although this could easily be changed. The picture only can be 780 pixels in width, which results in a maximum width of six pages of output since 130 are put on each page.

First, the variables ROWBEG, ROWEND, COLBEG, and COLEND are input to define the dimensions of the picture to be printed in terms of the original picture's rows and columns. The program then goes into a loop, covering five rows of data at a time, but only processing every other one in a call to GETLIN, a routine which takes care of most of the work.

GETLIN: This routine processes one line of data. The first record is read from tape in order to determine the starting point of the data (held in bytes 107 and 108 of the first record for each row of the picture). For band 6, the starting point of the data is four less than in band 4, but there are always two counter bytes at the beginning of each
record so that explains the calculation of the actual starting position of the area of interest, indicated in MSTART. The data for band 5 is skipped and then the band 6 record read into the array PIXEL. Each point of interest in this array is then assigned a certain combination of overprinted characters by the routine ASSIGN, which places the characters in the array PICT.

The data is then printed out into separate files, one file holding a strip of the picture 130 characters wide. First, one line is printed with a "9" control character, which will suppress page skips, followed by seven lines with a "+" control character to overprint.

ASSIGN: This routine decides the overprinting to be done for a specific reflectance value given in the variable SUM. The character printed for each value is given in the following table from MacLeod (17).
<table>
<thead>
<tr>
<th>Value Range</th>
<th>Characters</th>
<th>Percent of box covered</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt; 125</td>
<td>AX'HBBVC</td>
<td>100</td>
</tr>
<tr>
<td>120-125</td>
<td>.XO'HBV</td>
<td>97</td>
</tr>
<tr>
<td>115-120</td>
<td>.XO'HB</td>
<td>93</td>
</tr>
<tr>
<td>109-114</td>
<td>.XO'HC</td>
<td>89</td>
</tr>
<tr>
<td>103-108</td>
<td>.-XO'</td>
<td>85</td>
</tr>
<tr>
<td>97-102</td>
<td>.=+O'</td>
<td>79</td>
</tr>
<tr>
<td>91-96</td>
<td>.+O'</td>
<td>67</td>
</tr>
<tr>
<td>85-90</td>
<td>+O'</td>
<td>64</td>
</tr>
<tr>
<td>77-84</td>
<td>O+</td>
<td>60</td>
</tr>
<tr>
<td>73-78</td>
<td>O=</td>
<td>56</td>
</tr>
<tr>
<td>67-72</td>
<td>O-</td>
<td>53</td>
</tr>
<tr>
<td>61-66</td>
<td>M</td>
<td>45</td>
</tr>
<tr>
<td>55-60</td>
<td>A</td>
<td>42</td>
</tr>
<tr>
<td>49-54</td>
<td>X</td>
<td>40</td>
</tr>
<tr>
<td>43-48</td>
<td>Z</td>
<td>37</td>
</tr>
<tr>
<td>37-42</td>
<td>1</td>
<td>33</td>
</tr>
<tr>
<td>31-36</td>
<td>)</td>
<td>29</td>
</tr>
<tr>
<td>25-30</td>
<td>+</td>
<td>25</td>
</tr>
<tr>
<td>19-24</td>
<td>=</td>
<td>22</td>
</tr>
<tr>
<td>13-18</td>
<td>-</td>
<td>15</td>
</tr>
<tr>
<td>1-12</td>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>

Table V

Overprinted Characters for each Reflectance Range
Get Statistics

This program, written in ALGOLW-F, calculates statistics for the areas of the picture which are to be the learning areas of ground-truth data in order to determine the statistics for the probability functions. First, as usual, the beginning row and column values for the data held in file AREA2 must be input. The reflectance data is then read into the array VALUES, three values per pixel.

Now the form of input data must be described. (The input is in a file attached to SCARDS). The first row will be the value of the type of the area which is delineated in the following rows. This value must be negative though, so the program can tell when it gets to the next type. Each row following the type value (until the next negative number) consists of a row value, (in terms of the ground truth data this time, so the values will range from 1 to the number of rows) followed by a starting and finishing column value. The program simply goes to the indicated row of the ground truth data and extracts the points from the given column range and keeps track of the totals for each position of the feature vector. This is done until a value of -5 is encountered, indicating the end of the data. At this point the mean values are calculated and the program goes back to the top of the input data to begin calculating the covariances and standard deviations. This is accomplished by putting a $CONTINUE WITH FILENAME statement at the end of the data file.

Once the covariances and standard deviations have been
calculated for all the classes, they are output to a temporary file (STATS) in the following order for each row.

1. Standard deviation of first element of feature vector
2. Standard deviation of second element of feature vector
3. Standard deviation of third element of feature vector
4. Covariance between the first and second elements
5. Covariance between the first and third elements
6. Covariance between the second and third elements
7. Mean value of first element of feature vector
8. Mean value of the second element
9. Mean value of the third element
**Ellipse**

This program, written in FORTRAN, is used to draw the two-dimensional projections of the ellipsoids of concentration onto one of the three possible planes. The program first reads in the statistics defining the ellipses and begins at the left side and calculates the two Y values for that X value (or one if the derivative is zero at that point). These values are kept in an array POINTS with two Y values for each X value. It steps moves to the right and repeats the operation until the X value reaches its maximum.

The equation for an ellipse in two space is

\[
\frac{1}{1-\rho^2} \left\{ \frac{(x-u_x)^2}{\sigma_1^2} - \frac{2\rho(x-u_x)(y-u_y)}{\sigma_1 \sigma_2} + \frac{(y-u_y)^2}{\sigma_2^2} \right\} = 4
\]

which becomes

\[
\frac{x^2}{\sigma_1^2} - \frac{2\rho xy}{\sigma_1 \sigma_2} + \frac{y^2}{\sigma_2^2} - 4(1-\rho^2) = 0
\]

when centered at the origin. To find the maximum X values the derivative with respect to Y is taken and set to zero as follows.

\[
\frac{2x}{\sigma_1^2} \frac{dx}{dy} - \frac{2\rho x}{\sigma_1} - \frac{2\rho y}{\sigma_2} \frac{dx}{dy} + \frac{2y}{\sigma_2} = 0
\]

\[
\frac{dx}{dy} \left( \frac{2x - 2\rho y}{\sigma_1^2} \right) = \frac{2\rho x - 2y}{\sigma_1 \sigma_2}
\]

\[
\frac{dx}{dy} = \left( \frac{2\rho x - 2y}{\sigma_1 \sigma_2} \right) / \left( \frac{2x - 2\rho y}{\sigma_1^2} \right) = 0
\]
This is then used to solve for $Y$.

$$\frac{2\rho x}{\sigma_1^2} = \frac{2y}{\sigma_2^2} \quad y = \frac{\sigma_2^2 \rho x}{\sigma_1 \sigma_2}$$

Substituting into the original equation to solve for the minimum and maximum values of $X$.

$$\frac{x^2}{\sigma_1^2} - \frac{2\rho x (\frac{\sigma_2^2 \rho x}{\sigma_1 \sigma_2})}{\sigma_1^2} + \frac{\rho^2 x^2}{\sigma_1^2} - 4(1-\rho^2) = 0$$

$$\frac{x^2}{\sigma_1^2} - \frac{\rho^2 x^2}{\sigma_1^2} - 4(1-\rho^2) = 0$$

$$\frac{x^2}{\sigma_1^2} (1-\rho^2) = 4(1-\rho^2)$$

$$x = \pm 2\sigma_1$$

The statistics must be in a file attached to unit 1 in the program, with a number of rows corresponding to the number of classes. Each row has the following data:

- Standard deviation of first variable
- Standard deviation of second variable
- Covariance between the two
- The mean of the first variable
- The mean of the second variable

To plot the ellipses the program starts at the beginning of the array `POINTS` and moves to position `(POINTS(1,1), POINTS(1,2))`. It then draws to each subsequent point `(POINTS(i,1), POINTS(i,2))` until it reaches the end. This will draw half of the ellipse so it starts at the end and draws back to the beginning of the figure, joining points `(POINTS(i,1), POINTS(i,3))` since the third column of this array
holds the other y value for each x.
ELLIPSCID

This FORTRAN program displays three dimensional probability ellipsoids of concentration on the Adage graphics terminal. These images may then be rotated by the user to give some idea of the statistical overlap among the classes. The routines used are from UBC G (Three Dimensional Graphics Subroutines). Since only line drawings can be displayed on the Adage it is impossible to show surfaces. Instead, equally spaced ellipses are drawn to give the effect, as if one were to take a clear balloon and draw parallel lines on its surface.

The user is asked to input the number of ellipses to draw per ellipsoid, and the number of points per ellipse. Of course, the more points used the longer it takes but the smoother the picture. The statistics are read in from the file STATISTICS, created by the statistics gathering program. The matrix SIGMA holds the covariance values while MU is used to hold the average reflectance values.

The variables LAM11, LAM22, etc. are the cofactors of the indicated element of the covariance matrix. For example, LAM23 is the cofactor of the (2,3) position of the matrix. DET holds the determinant for the matrix of each class.

In order to do the actual plotting the program works as follows. First, an ellipsoid is used with a centre of gravity at the origin, since this makes the equations easier to work with. This is then translated to the point (MU_X, MU_Y, MU_Z), which is the actual centre. The minimum and maximum Z values are then easily calculated as 5*DET/LAM33 and -5*DET/LAM33.
respectively. The program then moves along the Z-axis, drawing ellipses in successive planes Z=Zval.

**The Procedures**

**Ellipse:** This subroutine calculates an ellipse in the Z=Zval plane where Zval is the current value of Z which is passed to the routine. First, the maximum and minimum X values are calculated in order to determine the starting and ending points for the ellipse in the X direction.

The original equation is

\[ \lambda_{11}x^2 + \lambda_{12}y^2 + \lambda_{13}z^2 + 2\lambda_{12}xy + 2\lambda_{13}xz + 2\lambda_{23}yz = 5D \]

where \( \lambda_{ij} \) is the cofactor of the \((i,j)\) position of the matrix and \( D \) is the determinant. Differentiating with respect to \( Y \) gives

\[
2\lambda_{11}x \frac{dx}{dy} + 2\lambda_{12}y + 2\lambda_{13}z + 2\lambda_{23}y \frac{dx}{dy} = 0
\]

Solving for the derivative and setting it to zero to get the extrema gives

\[
\frac{dx}{dy} = \frac{-\lambda_{12}y - \lambda_{13}z}{\lambda_{11}x + \lambda_{12}y} = 0
\]

\[
y = \frac{\lambda_{12}Z + \lambda_{13}X}{\lambda_{11}}
\]

Substituting into the original equation to find the extreme values for \( X \) one gets

\[
x^2(\frac{\lambda_{11}}{\lambda_{12}} - \frac{\lambda_{12}^2}{\lambda_{11}}) + x(2\lambda_{13}z - \lambda_{12}^2 \frac{Z}{\lambda_{11}}) + \lambda_{13}z^2 - \frac{\lambda_{23}^2}{\lambda_{11}} - 5D = 0
\]
which easily can be solved as a quadratic equation.

The program then starts at the calculated minimum value for \( X \) and calculates the \( Y \) value at this point. An increment is added to the \( X \) value and the process repeated until \( X \) gets to the maximum value. At that point the ellipse is plotted, or rather placed into the Adage buffer.
**Initial Classifier**

This program, written in FORTRAN, is used to produce a file where each point will be a type number. A threshold value is first read in (from SCARDS) which will determine how large the greatest probability must be in order to classify a point into one of the first four classes.

If a point receives its highest probability of membership from class k, and this probability is greater than the threshold percentage of the total probabilities, the point is assigned to the class. Otherwise it is assigned to a class depending on the two greatest probabilities. If a point by point classification is desired, a threshold value of 0 would be used. The reflectance values are first read into the matrix VALS, ground-truth to matrix TRUTH, and the statistics read in from the file STATISTICS. Then the program runs through each point of the picture matrix, calculating the probabilities in matrix TEMP. This is sorted and if the highest probability is large enough the point will be assigned to one of the four main classes. Otherwise, the class number assigned will depend on the highest and second highest probability.

At the end, a matrix of correctness percentages is printed out. Position (i,j) of this matrix indicates how many of the points classified as type i are actually type j when compared to ground truth data. Perhaps it should be pointed out that the ground-truth data used had seven classifications which were merged into four. This explains the oddities of calculating these values.
GETS: This subroutine calculates the determinants and cofactors for the four covariance matrices for use in the probability function.

F: This is the probability function which calculates the probability that a point with reflectance vector \( X \) belongs to class I.
The Region Finder

The purpose of this ALGOLW-F program is to trace out the regions contained in the scene. The program begins in the top left corner of the matrix and runs around the boundary of the region, which will include all points of the same type which are connected to one another. When the program returns to the starting point this region is finished so it scans along the row until the next point which it hasn't processed is found.

Global Variables

BEGIN_ROW, BEGIN_COL: These two variables indicate, in terms of rows and columns of the actual picture, the beginning row and column of the area of interest.

NUMBER_OF_ROWS, NUMBER_OF_COLUMNS: These integers are the size of the area of interest.

VALUES: This is a STRING(1) array which contains the reflectance values of each point in the area. VALUES(i,j,1) is the reflectance value of the point (i,j) in band 5, VALUES(i,j,2) is the value for band 6, and VALUES(i,j,3) for band 7.

PICT: This integer matrix holds the picture in terms of the class type of each point. In other words, PICT(i,j) is the number of the class to which the point (i,j) has been assigned.

REGIONS: REGIONS(i,j) is the number of the region (internal to the program) to which the point (i,j) has been assigned.
The Procedures

FIND_REGIONS: This is the controlling process. It starts in the upper left corner and calls FOLLOW_BOUNDARY to get the boundary of the region to which that point belongs. When it returns it steps along the same row, asking each point if it is in a region or not (accomplished by a call to NOT_IN_A_REGION). When it finds one which it hasn't processed yet, the process begins anew. At the end, a null line is output to the file REGIONS to indicate the end of the list of region descriptors and the matrix REGIONS is output to the same file.

FOLLOW_BOUNDARY: This procedure follows the boundary of a region. First it moves to left side of the region since the routine always starts by having the first boundary vector going downwards. The routine then enters a loop so long as the Boolean variable FINISHED is false. This loop adds the present region number to the point of the region presently being processed and calls GET_NEW_POINT to find the next point along the boundary and to determine the necessary boundary vectors. POINT_I and POINT_J are the co-ordinates of the present point on the boundary, while BOUNDARY_POINT_I and BOUNDARY_POINT_J are the co-ordinates of the end of the last boundary vector.

The difference in these two values should perhaps be illustrated. Imagine two adjacent points of the same row, (i, j-1) and (i, j). The top of the vertical boundary vector between these two would be (i, j) and the bottom
would be \((i+1,j)\). One therefore can consider two grids involved, one which is the actual pixels, and another which lies between them, representing the boundary grid.

But getting back to the procedure, once the boundary has been completely traced \textsc{fill\_in\_region} is called to do just that. It fills in all points of the region with the same region number (in the matrix \textsc{regions}). Finally, the descriptor of the region is output in a call to \textsc{add\_to\_region\_list}.

\textsc{add\_to\_boundary}: This procedure outputs the end of a boundary vector to the file \textsc{boundarys} and adjusts the extreme points of the region if necessary.

\textsc{add\_to\_region\_list}: This outputs a descriptor of the region to the file \textsc{regions}.

\textsc{move\_to\_edge\_of\_region}: This procedure, as one might imagine, moves from the present position to the left edge of the region. This is done in order to begin the boundary with a vertical vector oriented downwards.

\textsc{get\_new\_point}: Assuming the boundary tracer is presently positioned at \textsc{point\_i} and \textsc{point\_j}, this procedure will move to the next point along the boundary and create any boundary vectors necessary. This is done by calling one of the procedures \textsc{new\_point\_right}, \textsc{new\_point\_left}, \textsc{new\_point\_up}, or \textsc{new\_point\_down}, depending on the direction the boundary tracer is presently following.

\textsc{new\_point\_up}: This procedure determines the next point along the boundary, given that the tracer is presently heading
upwards. The other three routines are very similar and will not be described.

There are four cases to consider. First, the boundary may take a turn to the right; second, it may continue heading upwards; third, it may be at a right hand corner, in which case it turns to the left; and lastly, there may be a singular point jutting above its neighbours. At several points it is possible that the boundary might reach its starting point, at which time the tracer must cease. This is detected whenever the end of the boundary vector becomes equal to (ROW_Org, COL_Org).

FILL_IN_REGION: This procedure fills in all interior points of a region with the same region number (in the matrix REGIONS). While the boundary finder was doing its job, all the boundary points encountered were assigned the value of the region number, so this provides a start for filling in the interior points.

The method is to scan a box formed by the extreme values of the region. If a point is found which is of the correct type, and either the point to the top or to the left belong to the correct region, then the point is also assigned to the region. However, some points still may be missed in this scan so it starts again at the bottom of the box and moves upwards in the same manner. After both scans the area and average feature vectors are known for the region.

GET_PICTURE: This procedure reads in the data necessary to
produce the regions. This data is found in the file AREA, in which the value of a point \((i,j)\) will be the class to which the point belongs. Also, the actual reflectance values are read in from file AREA2 in order that the average reflectance values for the regions can be determined.

First, an appropriate number of rows are skipped in both files since BEGIN_ROW, the beginning of the area of interest may not be the same as ROW_START, which is the beginning row of the data held in the file. The program then reads in 100 rows and columns of data. Any more than this results in a horrendously large number of regions to process.

**OUTPUT_REGION_MATRIX:** This is used to print out the matrix of regions numbers in the file REGIONS. Again, if there are more than 80 columns in the matrix it will be necessary to continue on the next line because of ALGOLW-F line length restrictions.

**ADD_TO_LIST:** This prints the arguments ROW and COL in the file BOUNDARIES. Twelve points per line are allowed before moving to the next line.
The Region Merger

This program, written in ALGOLW-F, does the region merging to produce the final partition of the scene involved. I will first give a general description of the program and then a detailed description of the actual procedures used. The basic structures used in the program begin with two linked lists of regions. One, called STRONG_REGION_LIST, will include all regions whose interpretation is fairly certain. The other, the WEAK_REGION_LIST, will hold all the ambiguous regions.

There are two types of region merging available. The first steps through the STRONG_REGION_LIST, one region at a time, and then goes through WEAK_REGION_LIST searching for the first region which adheres to the merging criteria of that particular pass. When it finds one, the two regions are merged and the WEAK_REGION_LIST has a link removed.

In the second type of merging, the routine steps through the WEAK_REGION_LIST rather than the STRONG_REGION_LIST. The STRONG_REGION_LIST is then searched for all regions which are adjacent to this weak region and the best possible choice from these is taken as the region to merge.

One of these two processes is completed for each pass for as many times as there are regions being merged. As becomes clear though, if the number of regions becomes too large the program spends most of its time list searching. Therefore, if a large picture is to be processed it would be better to divide it into sections and process each section in turn.
Global Variables

NUMBER_OF_ROWS, NUMBER_OF_COLUMNS: These are number of rows and columns in the area being processed.

BEGIN_ROW, BEGIN_COL: These two values are input from the file produced by the region finding program. They indicate the row and column of the actual picture matrix where the area of interest begins.

CERTAIN_THRESHOLD: This real variable is the threshold for determining a strong region.

POINT: This record contains the row and column values of a point, along with a link field to create a list. The values ROW_F and COL_F (the F is used to indicate a field of a record) are STRING(1) variables to save core, so the values must be less than 255.

REGION: This record describes the regions. The fields are as follows.

MIN_ROW_F, MAX_ROW_F, MIN_COL_F, MAX_COL_F: These values are the extreme points of the region.

PERIM_F: This is the length of the perimeter.

TYPE_F: This field holds the type of the region. This will be an integer value known to the user. In my case 1 represents second growth, 2 is old growth, 3 is recent logging, 4 is water, 5 is type 1 or 2, 6 is type 1 or 3, and 8 is type 2 or 3

ROW_ORG_F, COL_ORG_F: These indicate the row and column where the boundary of the region begins.

REGION_NUMBER_F: This field is the number of the region
AREA_F: This is the number of pixels in the region.

BAND_5_AVG_F, BAND_6_AVG_F, BAND_7_AVG_F: These values give the averages of the three spectral bands for the points in the region.

POINTER_F: This points to a list of vectors which form the boundary of the region.

NEXT: This is the link to the next region in the list.

REGIONS: This is a matrix which holds the region number of each point. In other words, by looking at the value of REGIONS(i, j) you know to what region the point belongs.

VALUES: This is a three dimensional array which holds the actual spectral values for each point. VALUES(i, j, 1) is the value of point (i, j) for band 5, VALUES(i, j, 2) for band 6, and VALUES(i, j, 3) for band 7.

SIGMA: This matrix holds the covariances for each class. The matrix has six columns since there are four classes. The elements are as follows:

SIGMA(i, 1): The standard deviation of the first element of the feature vector for class i.

SIGMA(i, 2): The standard deviation for the second element.

SIGMA(i, 3): The standard deviation for the third element.

SIGMA(i, 4): The covariance between the first and second elements.

SIGMA(i, 5): The covariance between the first and third
elements.

SIGMA\((i, 6)\): The covariance between the second and third elements.

MU: This matrix holds the mean values of each element of the feature vector for each of the main classes.

DET: This array holds the determinant of the covariance matrix for each class.

S11, S12, S13, S22, S23, S33: These arrays are the cofactors of the indicated position of the covariance matrix. For example, S23\((i)\) is the cofactor of the element \((2, 3)\) for the matrix of class \(i\).

GROUND_TRUTH: This matrix holds the ground-truth values for the area of interest. At present, this matrix consists of one of seven numbers since the ground-truth data had seven classes. Types one and two are the same as type one in my program, types 3 and 4 were merged to type 2, types 5 and 6 to type 3, and type 7 became type 4.

TYPE: This is an array which holds the type of each region. For example, TYPE\((i)\) will be the type of the region numbered \(i\). This is done so that the type can be quickly known, without having to go through the region list until the desired region is found. This array is a STRING(1) variable since the number of types should never reach 255. At present room is made for 500 regions so if the initial partition has more than this the program will have to be changed.
The Procedures

REGION_ELEMENT: This logical procedure determines whether or not the point at the arguments (ROW, COL) is a member of REGION_NUMBER1 or REGION_NUMBER2 (which are also arguments).

FILL_IN_REGION: This procedure fills in the interior points of a region with the same region number. See the description in the Region Finder for more details.

FOLLOW_NEW_BOUNDARY: This procedure traces and records the boundary of a newly merged region. Given two region pointers TEMP1 and TEMP2, the resultant boundary will be the boundary of the merged region. The routine begins at the extreme left side of the region, determined by ISTART and JSTART. The first thing to do is calculate the average reflectance values for the merged region and determine the type from these values in a call to GET_TYPE, the variable SURE used to indicate if the new region is a strong one. If the region is strong (i.e., SURE is true), the second region (TEMP2) is removed from LIST2. (TEMP2_PREV is the record in the list which comes immediately before TEMP2, used for easy removal).

Then the two boundary lists are joined since the new boundary will likely be longer than either of the old ones, but has to be shorter than the sum of the two. FOLLOW_BOUNDARY is then called to do the actual tracing of the boundary. Finally, the various descriptors of the new region are established.
**MERGE_REGION:** This procedure does one complete pass of the simpler merging process. One outer loop continues until the pointer TEMP1 is null by stepping through LIST1 one region at a time. TEMP2 then starts at the top of the second list and a second loop is entered so as long as TEMP2 is not null, or the first region (TEMP1) is not removed in a merge. If the two regions are mergeable (in theory), ADJACENT is called to see if they have enough of their boundaries in common. If they have, FOLLOW_NEW_BOUNDARY is called to get the new boundary of the merged region.

**REMOVE:** This routine removes the region TEMP2 from the list headed by the reference variable LIST. If the region is already the head of the list, the head is changed to the next element. Otherwise the link field of the previous node is changed to the link field of the node being removed.

**ADJACENT:** This logical procedure determines if the two regions denoted by LIST1 and LIST2 are adjacent. This means that a certain number of boundary elements are not only adjacent in the physical sense, but are also weak boundary vectors. In other words the relative difference in reflectance of the two pixels across the boundary is less than STRENGTH_THRESHOLD. The procedure simply determines which region has the shortest perimeter and calls ADJACENT2. This is done since it is quicker to move around the outside of the smaller region.
ADJACENT2: This procedure does most of the work in determining adjacency. It begins by setting the reference variable POINTER to the first point of the boundary list of the region TEMP1. Two basic points are used for the boundary vector, (I1, J1) being one end and (I2, J2) the other. If the vector lies outside of the extreme points of the second region it obviously can't be adjacent, so nothing is done. Otherwise ADJACENT3 is called since adjacency is now possible. It will return with variables INTERSECT and WEAK_COUNT possibly changed. INTERSECT will be the number of boundary vectors the two region have in common. WEAK_COUNT is the number of these vectors which are "weak", according to the previously stated definition of the term. If the WEAK_COUNT becomes greater than a certain percentage of the perimeter (the percentage is held in the variable PHAGOCYTE_THRESHOLD) the two regions are said to be adjacent.

ADJACENT3: This procedure determines if the boundary vector given it (passed in parameters I1, J1, I2, and J2) is adjacent to the region given in TEMP2. The first thing to do is to determine the two pixels on opposite sides of this boundary vector.

To do this an example might be useful. Suppose one has two points, (i,j) and the point to its right (i, j+1). The top of the vertical boundary between them is (i, j+1) and the bottom is (i+1, j+1). Therefore, given any boundary vector, the pixels on either side can be
calculated in a simple fashion.

If the point opposite the vector is an element of the desired region it is then checked for weakness. If not weak, this vector is not adjacent to the desired region.

**MERGE2:** This procedure provides the second type of region merging. The two lists used are passed to the procedure in the variables LIST1 and LIST2. (LIST2 generally will be the WEAK_REGION_LIST). The procedure enters a loop which will look at each element of LIST2 in turn, searching for all the regions in LIST1 which are adjacent to it. This is done by going through the boundary lists of the regions and recording each region which is adjacent and keeping track of the number of adjacent points in the array INTERSECT. INTERSECT(i) will then hold the number of adjacent points which the region TEMP2 has in common with the region numbered i.

Once this is done for the complete boundary list of TEMP2 all adjacent regions are known. This list is then checked to get the region which is most like the weak region and has the greatest boundary in common. This region will be the one to merge with.

**NO_CHANGE_IN_INTERPRETATION:** This logical procedure takes as parameters two regions and will return TRUE if the region TEMP1 will not have its interpretation changed if it were merged with region TEMP2, and FALSE otherwise.

**READ_IN_REGION:** This procedure does some of the input for the program. First, the necessary statistics are read in from
a file STATISTICS, which is created by the program GET_STATS. The form of the data in this file is in four rows, one for each class, where each row has the following information in the order indicated.

Standard deviation of the first element of the feature vector.
Standard deviation of the second element of the feature vector.
Standard deviation of the third element of the feature vector.
Covariance between the first and second elements.
Covariance between the first and third elements.
Covariance between the second and third elements.
Mean value of the first element.
Mean value of the second element.
Mean value of the third element.

Once these are read in the four determinants are calculated, as well as the six cofactors. (There are only six because the matrix is symmetric). Next, the regions and their boundaries are read in. This data is held in the two files REGIONS and BOUNDARIES, which are filled by the region finder. If the type of the region read in is less than 5 it will be added to the STRONG_REGION_LIST, otherwise it is added to the WEAK_REGION_LIST.

READ_IN_BOUNDARY: This procedure reads in the boundary of a region, creating a linked list of the points as it does so. The form of the boundary file has twelve points per row,
each point consisting of a row and column value.

READ_IN_REGIONS: This procedure is used to read in the matrix of region numbers and ground truth values. The region number matrix is found at the end of the file REGIONS; the ground truth in the file GROUND_TRUTH. The constant FIRST_ROW_OF_DATA indicates the row of the actual picture which is the first row of the data in the file AREA. This is done since I had no desire to read from tape every time I wanted to use the program. Instead, another program is used to read in an n by m matrix of pixels from a certain area of the picture. BEGIN_ROW is the beginning row for the area of interest. If it is not the same as FIRST_ROW_OF_DATA, the correct number of lines must be skipped to get to the correct starting line. A similar function is carried out for the columns.

READ_IN_DATA: This routine reads in the actual radiance values from a file called AREA2. Because of ALGOLW-F line length restrictions only 25 points are held per line of the file, each point having three reflectance values which may have up to three digits. The data is placed in the matrix STRING(1) array VALUES.

FOLLOW_BOUNDARY: This procedure and the ones it calls are identical to those in the region finder and will not be described here.

GET_TYPE: This procedure returns the type of the point (or region) whose radiance values are given in the parameters X1, X2, and X3. The function F, which calculates the
probability of a point with feature vector \((X_1, X_2, X_3)\) belonging to class \(i\), is called for all four classes. The values obtained are sorted and \textit{TYPE} is set to the number of the class which gives the point the highest probability of membership.

\textbf{F:} This real procedure calculates the probability that a point with radiance vector \((X_1, X_2, X_3)\) belongs to class \(i\). It is done according to the formula given previously.

\textbf{ADD_TO_BOUNDARY:} This procedure simply adds a point to the boundary list and changes the extreme points of a region if necessary.

\textbf{MERGEABLE:} This logical procedure determines if the two regions \textit{TEMP1} and \textit{TEMP2} are mergeable in theory. In pass one, they are mergeable if the two types are the same and the region \textit{TEMP1} will not change its interpretation if they were merged. In pass two, either the types are the same or the second most likely type of \textit{TEMP2} is the same as the type of \textit{TEMP1}. Also, the average reflectance value difference must be less than the \textsc{strength\_threshold} and once again, region \textit{TEMP1} must not change interpretation if they were merged. In pass three, only the difference in average reflectance for the two regions have to be less than the \textsc{strength\_threshold}.

\textbf{OUTPUT\_MERGED\_REGIONS:} This is a procedure used for debugging. If the user wishes to see descriptors of the final regions, this procedure will place them in a temporary file named \textsc{merged\_regions}. 
**GRAPH_REGIONS:** If a graph is of the picture is called for this procedure draws the outline of every region in the list it is given.

**ADD_DIFFERENT_MARKINGS:** This procedure adds the markings for the different types of regions, as well as adding boxes for the legend. If the variable PLOT has the value one, indicating only a graph of the original picture is desired, all legends are drawn, otherwise only the legends for the four main classes are done. No text is drawn though, since the quality of text from the plotter is less than pleasing.

To place the markings on the graph, the procedure simply looks at each point of the picture and calls ADD_MARKINGS with the type of region to which the point belongs.

**SQUARE:** This will draw a square at \((x, y)\) with a size of SCALE.

**ADD_MARKINGS:** This adds an identifying mark to the point \((x, y)\) which has type value given in TYPE. At present only seven types are available. Type four will have a cross drawn; type three has a line from top left to bottom right; type two is blank; type one has a dot in the centre; type six, which is either type one or three, gets an X shape while type eight, which is either type two or three, has both the above markings.

**PLOT_END:** This must be called to end a plot correctly.

**PLOT:** This is the FORTRAN plot routine. See UBC PLOT for details.
**START_REGION:** This moves to the point \((i,j)\) in the matrix. Those matrix values are not very useful to the plotter though, so they must be transformed into Cartesian co-ordinates.

**GRAPH_REGION:** This procedure calls GRAPH for each point in the boundary list given it.

**GRAPH:** This calculates the Cartesian co-ordinates of the matrix point \((i,j)\) and draws a line to that point.

**COMPARE_WITH_TRUE_DATA:** In order to establish the correctness of the final partition, this routine is called. For example, if point \((i,j)\) belongs to a region of type \(k\) in the final partition, and is of type \(m\) in the ground-truth data, position \((k,m)\) in the matrix CORRECT is incremented. Ideally, \(k\) and \(m\) would be equal, indicating the ground-truth agrees with the calculated results. The matrix is then output in a percentage form.

**The Main Program:** First the data is read in and the parameters for the program established. If plotting is to be done for the original picture it is done before starting the merge. Then the merging process is done for the first two passes for so long as regions continue to be merged. (This is indicated by the logical flag SWITCH). For the third pass the WEAK_REGION_LIST is done away with by joining it to the STRONG_REGION_LIST after changing the type of each region to be the type of the class which gives it the highest probability of membership. This is done since the final pass is rather weak because most of the weak regions should
have been merged by then.