

8-1-1993

Automated image-to-image rectification for use in change detection analysis as applied to forest clearcut mapping

Kaleen S. Moriarty

Follow this and additional works at: <http://scholarworks.rit.edu/theses>

Recommended Citation

Moriarty, Kaleen S., "Automated image-to-image rectification for use in change detection analysis as applied to forest clearcut mapping" (1993). Thesis. Rochester Institute of Technology. Accessed from

This Thesis is brought to you for free and open access by the Thesis/Dissertation Collections at RIT Scholar Works. It has been accepted for inclusion in Theses by an authorized administrator of RIT Scholar Works. For more information, please contact ritscholarworks@rit.edu.

**AUTOMATED IMAGE-TO-IMAGE RECTIFICATION
FOR USE IN CHANGE DETECTION ANALYSIS AS
APPLIED TO FOREST CLEARCUT MAPPING**

by

Kaleen S. Moriarty

Rochester Institute of Technology
Center for Imaging Science

August 1993

A thesis submitted in partial fulfillment of the
requirements for the degree of Master of Science in the
Center for Imaging Science in the
College of Imaging Arts and Sciences of the
Rochester Institute of Technology

Signature of the Author Kaleen S. Moriarty

Accepted by Dana G. Marsh Sept. 1, 1993
Coordinator, M.S. Degree Program

Center for Imaging Science
College of Imaging Arts and Sciences
Rochester Institute of Technology
Rochester, New York

CERTIFICATE OF APPROVAL

M.S. DEGREE THESIS

The M.S. Degree Thesis of Kaleen S. Moriarty
has been examined and approved by the thesis
committee as satisfactory for the thesis requirement
for the Master of Science degree

Mr. Carl Salvaggio, Thesis Advisor

Dr. Roger Easton

Mr. Rolando Raqueño

Center for Imaging Science
Rochester Institute of Technology
Rochester, New York

THESIS RELEASE PERMISSION FORM

Title of Thesis: Automated Image-To-Image Rectification For Use In Change
Detection Analysis As Applied To Forest Clearcut Mapping

I, Kaleen Moriarty, grant permission to the Wallace Memorial Library of the
Rochester Institute of Technology to reproduce this thesis in whole or in part
provided any reproduction will not be for commercial use or profit.

Date Sept. 7, 1993

Automated Image-To-Image Rectification
For Use In Change Detection Analysis As
Applied To Forest Clearcut Mapping

by

Kaleen S. Moriarty

A thesis submitted in partial fulfillment of the
requirements for the degree of Master of Science in the
Center for Imaging Science in the
College of Imaging Arts and Sciences of the
Rochester Institute of Technology

Abstract

An automated approach to change detection analysis was developed for use in multi-temporal image comparisons. An algorithm was developed which enables the user to perform automatic image-to-image rectification. Manual registration techniques are utilized to register a reference image to a Universe Transverse Mercator map projection. Control points, or kernel images, are extracted from the rectified reference image and located automatically in the to-be-mapped images via mathematical correlation. A two windowed approach is used that requires an estimation of the location of the control point in the to-be-mapped image. This estimate is used to create a search area which is correlated with the kernel image. The images were rectified to within approximately two pixels.

The images were radiometrically normalized so that actual ground changes can be distinguished from those that occur due to imaging conditions. This was done through a simple histogram matching technique. Next, the images were classified to illustrate the changes in land cover type. An unsupervised classification was used to train the reference image. The rest of the images in the set were classified using the spectral signature data generated from this process. The classification accuracy was dependent on the normalization procedure used.

The process was demonstrated using LANDSAT MSS imagery to show the extent to which the logging technique of clearcutting has devastated the forest stands in the North Cascades of Washington state.

Acknowledgments

I cannot begin to thank Mr. Carl Salvaggio for the moral and academic support given throughout this project. I am eternally grateful for the many hours he spent helping me solve problems. He often provided me with the confidence and will to keep going and his endless patience will never be forgotten.

I would like to thank Dr. Roger Easton for his ability and willingness to explain whatever it was I didn't understand. He was always there to solve seemingly hopeless problems encountered throughout this thesis. I would also like to thank him for reviewing this thesis thoroughly and for being such a dedicated professor.

Mr. Rolando Raqueño has proven to be an invaluable addition to this thesis committee. I especially thank him for his help with computer code problems. Twenty-five cents is a deal!

To John and Kathy , thank you for always believing in my ability to complete this thesis. You might be the only two people who never asked "Aren't you done yet?" Your friendship, love, and support will always be remembered.

Dedication

This thesis is dedicated with love to my family. They may not have understood why this thesis took so long to complete, but they were supportive throughout my life and my years at RIT. Thanks Mom, Dad, Jennifer, Kerrin, and David.

1.0	Introduction.....	1
2.0	Background	3
2.1	Image Registration.....	3
2.1.1	Manual Registration	4
2.1.2	Automated Methods.....	6
2.1.2.1	Image Correlation Techniques.....	6
2.1.2.2	Sequential Similarity Detection Algorithm.....	10
2.1.2.3	A Two-Stage Registration Method using SSDA	11
2.1.2.4	Object Detection Via Clustering.....	13
2.1.2.5	Automated Multisensor Registration.....	13
2.1.2.6	Recognition of Corresponding Structures using Multivalue Logic.....	15
2.1.2.7	Using Feature-Based Mapping	15
2.2	Normalization.....	17
2.2.1	Radiative Transfer Models	18
2.2.2	Histogram Modification.....	22
2.2.2.1	Histogram Equalization and Specification.....	23
2.2.2.2	Linear Histogram Transformation	26
2.2.3	Pseudoinvariant Feature Normalization.....	29
2.3	Classification of Images	31
2.3.1	Supervised Classification.....	32
2.3.1.1	Maximum Likelihood Classifier	33
2.3.1.2	Minimum-Distance-to-Mean Classifier.....	36
2.3.1.3	Parallelepiped Classifier	37
2.3.2	Unsupervised Classification.....	40
2.3.3	Hybrid classification	42
3.0	Experimental Procedure.....	43
3.1	Selection of Imagery.....	44
3.2	Manual Registration	46
3.3	Automated Registration.....	47
3.3.1.	Creating the Correlation Kernels	47
3.3.2	Approximate Registration	48
3.3.3	Locating the Search Windows	50
3.3.4	Correlation	53

3.4	Normalization.....	53
3.4.1	Histogram Matching (Specification).....	54
3.4.2	Use of Invariant Features to Perform Linear Mapping.....	54
3.4.3	Linear Mapping Using Entire Image Statistics	57
3.5	Classification.....	58
4.0	Results.....	61
4.1	Manual Registration of Reference Image.....	61
4.2	Normalization.....	67
4.2.1	Histogram Matching (Specification).....	67
4.2.2	Use of Invariant Features to Perform Linear Mapping.....	68
4.2.3	Linear Mapping Using Entire Image Statistics	75
4.3	Classification.....	76
4.4	Image Plates.....	77
5.0	Conclusions and Recommendations	83
6.0	References.....	86
Appendix A	Program MAKE_KERNEL_UTM.....	90
Appendix B	Program MAKE_GCP_FILE	93
Appendix C	Program VEC_ANGLE.....	112
Appendix D	Coordinate transformation data.....	115
Appendix E	Image set ephemeris data.....	129
Appendix F	ERDAS routines outlined.....	131

List of Tables

Table 1	Images used in study.....	45
Table 2	Control points used in manual registration.....	62
Table 3	Registration error for dependent points.....	63
Table 4	Registration error for independent data points	63
Table 5	1981 ground control points.....	65
Table 6	1988 ground control points.....	66
Table 7	Original image statistics.....	67
Table 8	Histogram matched statistics.....	68
Table 9	Digital count of invariant pixels in each band	69
Table 10a	Output digital counts of invariant objects for band 4	72
Table 10b	Output digital counts of invariant objects for band 3	73
Table 10c	Output digital counts of invariant objects for band 2	73
Table 10d	Output digital counts of invariant objects for band 1	73
Table 11	Digital count RMS error for each band	74
Table 12	Statistics from linear transformation of invariant features.....	74
Table 13	Linear Mapping Using Entire Image Statistics	75

List of Figures

Figure 2.1	Sources of geometric distortion.....	6
Figure 2.2	Histogram equalization.....	24
Figure 2.3	Histogram specification.....	25
Figure 2.4	Transformation of linearly related histograms.....	28
Figure 2.5	Basic steps in supervised classification	32
Figure 2.6	Probability density functions defined by a maximum likelihood	34
Figure 2.7	Pixel observations from selected training sites plotted on a scatter diagram.....	35
Figure 2.8	Equiprobability contours defined by a maximum likelihood classifier	36
Figure 2.9	Minimum distance to mean classification strategy.....	37
Figure 2.10	Parallelepiped classification strategy.....	38
Figure 2.11	Weakness of a parallelepiped classifier for highly correlated data.....	39
Figure 2.12	Parallelepiped classification strategy employing stepped decision region boundaries.....	39
Figure 2.13	An illustration of clustering by iterative optimization (ISODATA method)	41
Figure 3.1	Flowchart of Work	44
Figure 3.2	Correlation result.....	51
Figure 3.3	Example of linear transformation using invariant features.....	56
Figure 3.4	Minimum-distance-to-mean vs. Maximum Likelihood classification.....	59
Figure 4.1a	1981 band 4 look-up table, linear transformation using invariant features	70
Figure 4.1b	1981 band 3 look-up table, linear transformation using invariant features	70
Figure 4.1c	1981 band 2 look-up table, linear transformation using invariant features	71
Figure 4.1d	1981 band 1 look-up table, linear transformation using invariant features	71

1.0 Introduction

Geographic information systems (GIS) were designed to help land managers combine information about land ownership with other spatial information such as topographic maps. The role of GIS is to provide a means by which diverse data types can be efficiently stored and retrieved, manipulated, analyzed, and displayed. Remotely sensed information such as multispectral imagery may be combined with map coordinates or other information in correct geographical and geometrical relation [Richards, 1986]. Prior to computer-based GIS systems, data was combined by using a map overlay method [Lillesand, 1987]. A transparent map was created to represent the information desired. This was then superimposed with other transparent data sheets and the imagery of the corresponding area. However, the creation of sheets to the desired scale and format is very time consuming and not very flexible *i.e.* it is difficult to apply different weighting factors to the data.

The integration of GIS with remotely sensed data was a natural step due to the potential that GIS offers as a tool for managing and analyzing such data [Ehlers *et al.*, 1989].

Georeferencing, or geocoding, of data relates the land data to a specific location. This ties the reflective information from the satellite to GIS information such as a topographic map, thus allowing the user ready access to various planes of information.

Multispectral images obtained on different dates can be useful for mapping the changes over time of an effect which manifests itself visually and can be detected using satellite images. The changes of interest may be naturally occurring, such as changes in forest cover due to a fire, or they may be human made, as in the case of monitoring urban growth in a metropolitan

area. The process of mapping these changes is referred to as "change detection". The steps necessary to implement change detection are:

- 1) Spatial registration of images to each other and/or to GIS information such as topographic data. If images are registered to the same type of topographic data, they will be registered to each other.
- 2) Atmospheric normalization of the images (so it appears as though each has been imaged through the same atmosphere).
- 3) Classification of the images into various land cover types so that the changes over time can be viewed or measured.

Image registration must be performed before comparing two or more multispectral images to allow pixel-to-pixel comparisons. If these images are registered to a map coordinate system, the pixels in each image can be addressed by map coordinates which may be more convenient. For example, latitude and longitude may be used rather than pixel locations [Richards, 1986].

The process of manual registration is time consuming and its success depends on the user's ability to locate control points accurately. This means that a particular control point must be located precisely in all images as well as on a map if the images are to be geocoded. Accuracy also depends on whether or not the control points chosen are unchanged over time. In an application such as change detection, a user may have several images of the same area to study. Manual registration becomes quite inefficient as the

number of images increases. Due to the problems associated with conventional registration techniques, it has become necessary to automate the process of image registration.

The primary focus of this thesis is to develop a usable method of automated image registration. The subsequent steps of image normalization and image classification will be discussed and performed to demonstrate the overall process of change detection.

The change in forest cover due to excessive clearcutting in western Washington state will be analyzed and mapped. An estimated 25 million acres of ancient forest once stood on what is now western Washington, Oregon, and northern California. Much of the forest has been removed in the last century by various logging practices, including clear cutting which is a technique that has been employed extensively in the Douglas-fir region of the Cascade mountains. As the name implies, all vegetation in a given localized area is cut away to remove the desired trees very rapidly as opposed to removing selected trees only and leaving the rest of the vegetation undisturbed. The process of clear cutting severely fragments the remaining ancient forest stands and has detrimental effects on the local ecology [Morrison, 1991].

To bring attention to this excessive logging with the hope of protecting the biological diversity of national forests in the Pacific Northwest, the extent to which these forests have been cut from 1972 to 1988 has been mapped.

2.0 *Background*

2.1 *Image Registration*

Image registration (or rectification) is necessary whenever two or more images of the same scene are to be compared. The intent of image registration

is to correct for spatial distortions which occur during image acquisition [Lillesand, 1987]. The distortion may be due to differences in the sensors used to obtain the images, or as a result of being imaged with the same sensor but at different times. The sources of distortion can be systematic or random. They can result from variations in sensor altitude, attitude, and velocity, as well as other factors such as relief displacement, nonlinearities in sensor sweep, *etc.* The process of image registration is typically composed of four steps:

- 1) Selection of control points.
- 2) Matching of points in the images to be registered.
- 3) Estimation of a mapping function based on the control point pairs chosen between the two images.
- 4) Application of the mapping function to spatially register the images.

Steps three and four can be automated by using available commercial software [Ton and Jain, 1989]. However, in most current methods, the first two steps are performed manually .

2.1.1 *Manual Registration*

The registration process remaps the coordinates of pixels in one image to corresponding coordinates in the second image [Schowengerdt, 1983].

Traditionally, this process is performed manually by precisely locating well distributed and matched ground control points throughout the images. Ideal control points must be stable, meaning their position should not change between acquisition dates [Ton and Jain, 1989]. The map coordinates of the

control points (e.g. UTM, latitude/longitude) must also be known if the user wishes to establish coordinates throughout the image. Features such as crossroads and corners of buildings are good choices for control points. The spatial relationships among control points in the image are matched to the reference image by performing a coordinate transformation that preserves pixel continuity in the output image. The coordinate transformation may be expressed as a power series of the input coordinates:

$$x = \sum_{n=0}^{\infty} \sum_{m=0}^{\infty} a_{nm} (x')^n (y')^m = a_{00} + a_{10}x' + a_{01}y' + a_{11}x'y' + a_{20}x'^2 + a_{02}y'^2 + a_{22}x'^2y'^2 + \dots \quad (1.a)$$

$$y = \sum_{n=0}^{\infty} \sum_{m=0}^{\infty} b_{nm} (x')^n (y')^m = b_{00} + b_{10}x' + b_{01}y' + b_{11}x'y' + b_{20}x'^2 + b_{02}y'^2 + b_{22}x'^2y'^2 + \dots \quad (2.a)$$

where:

x, y (unprimed) correspond to the original image pixel locations

x', y' (primed) correspond to the reference image pixel locations

a_{ij}, b_{ij} are the coefficients generated as a function of a least squares solution, the i and j suffixes indicate the power in x and y , respectively.

This series can be approximated by truncating to fewer coefficients:

$$x = a_{00} + a_{10}x' + a_{01}y' + a_{11}x'y' + a_{20}x'^2 + a_{02}y'^2 \quad (1.b)$$

$$y = b_{00} + b_{10}x' + b_{01}y' + b_{11}x'y' + b_{20}x'^2 + b_{02}y'^2 \quad (2.b)$$

The terms in these equations can account for simple geometric distortions including linear shift, scale change, perspective, and rotation. Figure 2.1 illustrates these effects.

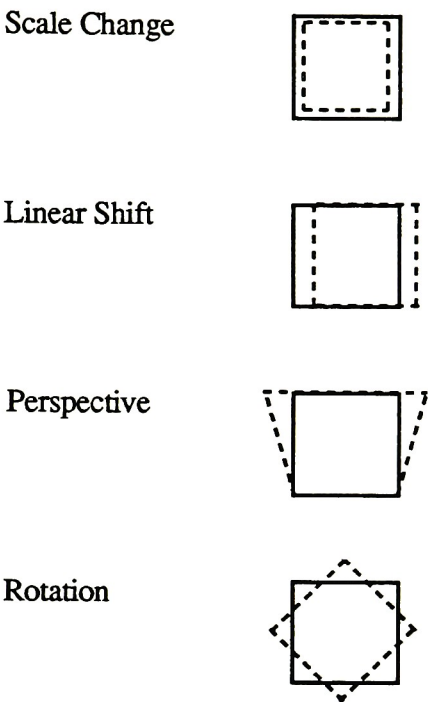


Figure 2.1 Sources of geometric distortion (dashed lines are distorted image)

2.1.2 Automated Methods

2.1.2.1 Image Correlation Techniques

Correlation techniques were proposed as an automated approach to image registration [Pratt, 1974]. A correlation measure is formed between two functions and the location of the maximum correlation is used to locate control points automatically.

Salvaggio and Schott (1987) proposed a method that employs correlation to select the control points necessary to derive the geometric transformation equations. The technique was demonstrated to be successful but was not developed sufficiently to be usable. The correlation process for each point

involves the selection of two windows: the first is the *correlation kernel* and the second is the *search window*. The correlation kernel is smaller and windows a section of the reference image which contains a feature that can be used as a control point. The search area is then segmented from the to-be-registered image by locating approximately the area that is thought to contain the same feature as that in the correlation kernel. This method is not fully automated as it requires the user to either choose control points in one of the images to create the kernel windows or to register an image manually if the data is to be referenced to a map.

To locate the control point more exactly, the kernel image is correlated with the search area according to the following equations:

$$r(i,j) = \frac{\sum_{m=1}^N \sum_{n=1}^N f_1(m,n) f_2(i+m, j+n)}{A_1 A_2} \quad (3)$$

where:

$r(i,j)$ is the value of the correlation at coordinates (i,j) in the search area.

$f_1(m,n)$ is the search area data at point (m,n) in the coordinate system defined by the current position of the correlation kernel.

$f_2(i+m,j+n)$ is the value of the kernel at the position (m,n) , as described above, when the kernel is located at position (i,j) in the search area coordinate system.

A_1 and A_2 are the weights of the search area lying beneath the kernel at any time and the weight of the kernel respectively.

$$A_1 \equiv \left[\sum_{m=1}^N \sum_{n=1}^N f_1^2(m,n) \right]^{1/2} \quad (4)$$

$$A_2 \equiv \left[\sum_{m=1}^N \sum_{n=1}^N f_2^2(i+m, j+n) \right]^{1/2} \quad (5)$$

The usefulness of this method was demonstrated by registering SPOT images to U.S.G.S. topographic data. Since the correlation depends on the reflective data between the images, an immediate problem exists when the brightnesses of a SPOT image and a digitized image of a map are not identical. To circumvent this problem, an aerial photograph was manually registered to the U.S.G.S. data. This was subsequently correlated with the SPOT imagery to generate the appropriate transformations. Therefore, when the SPOT image is registered to the photo data, it will also be registered to the map.

Another problem with the correlation procedure is its sensitivity to scale changes or rotations of the images. An approximate registration must be performed using ephemeris data to account for some of these changes, *e.g.* latitude and longitude and the pixel coordinates of the four corners and the center of the full SPOT scene. These can be converted to the Universal Transverse Mercator (UTM) projection which is the same map projection used for U.S.G.S. topographic maps. The latitude and longitude values must be converted to UTM coordinates, and the SPOT scene can be registered to the corresponding area of the topographic map. This will give a first-order registration so that the correlation process will be more effective.

High-contrast objects with a great deal of high-frequency structure are desirable as control points because the correlation operation is most effective with this type of object. The point in the search window at which the correlation value is largest is chosen as a control point for this image. After the control points have been automatically located in the new image, the geometric transformations can be determined as described in the manual registration method.

Salvaggio and Schott (1987) suggested this as a potential automated method and tested it to prove the concept. They did not develop the

technique to full application. Errors on the order of 2.5 meters on the ground were obtained.

2.1.2.2 *Sequential Similarity Detection Algorithm*

Other methods similar to the above correlation procedure were used previously. One such method is known as a Sequential Similarity Detection Algorithm (SSDA) [Bernstein, 1976] and also uses two windows. A ground control point (GCP) window from the reference image is compared to a search area in the image to be matched. The window area is superimposed over the search area and the sum of the absolute values of the differences between pixel gray values is calculated. The error measure is:

$$\varepsilon(i, j, l_n m_n) = |S_M^{i,j}(l_n m_n) - W(l_n m_n)| \quad (6)$$

For window W of size $M \times M$, the index n runs from 1 to M^2 . The position where the two areas are most similar is selected by thresholding. When the sum exceeds the threshold, the window and search areas are assumed to be dissimilar. If the sum of absolute errors does exceed the threshold, the number of pixel comparisons is recorded for that search point. The window area is then translated over the search area and another comparison is made. This process is repeated until the number of comparisons required before the threshold is exceeded is large, *i.e.* the images are near registration. The pixel located at the center of the window at this location defines the control point. This method was first described using a constant threshold [Barnea and Silverman, 1972] in which the pairs of points to be compared were selected in random order. This is done to ensure that new information is used at each comparison. If there is reason to think that a control point is bad, the

sequential comparison can be terminated before each of the M^2 windowing pairs has been examined. In this manner, the procedure becomes more computationally efficient.

One problem with the SSDA is that it is highly sensitive to brightness differences between the two images. There must not be "major changes of an uncorrelated nature between the scenes in the vicinity of the control point being tested," [Richards, 1986]. This method also will not account for rotations or scale differences.

2.1.2.3 *A Two-Stage Registration Method using SSDA*

At the University of Texas at Austin, an improvement to the SSDA procedure was developed [Kastak and Crawford, 1989]. This method is said to be computationally efficient for translational registration. A two-stage automated algorithm was developed which uses a mean-deviation threshold sequence so that points in the two images that are very dissimilar can be detected without evaluating the similarity function for all points in the image. This reduces the total number of windowing pairs which must be compared [Barnea and Silverman, 1972]. The two-stage algorithm presented here was modified to account for rotational differences as well.

This approach uses the similarity detection method described in section 2.1.2.2 (refer to equation 6 for error measure). If there is adequate evidence that a control point is bad, further comparisons need not be made.

The following adaptive threshold was used as it significantly reduced the

number of necessary windowing pairs. The threshold is defined as:

$$T(k) = \mu k + \sigma \sqrt{k} \quad (7)$$

where

- μ is the mean of the absolute value of the uncorrelated noise at registration,
- σ is chosen to account for the region of safety [Kastak, Crawford, 1989].

The uncorrelated noise refers to the error between the reference and the search images. However, it can be difficult to determine the mean and standard deviation of the noise within the window without manually pre-registering the images.

Consequently, a two-stage algorithm is implemented to eliminate manual pre-registration. By approximating the registration, the noise statistics can be generated which define a threshold sequence. In the first stage, a sampling procedure is used to determine a point in the immediate neighborhood of the best registration point. The similarity assessment is performed on every fifth pixel and the maximum is used as the center of a new 10x10 pixel search area. This is repeated on every pixel within the new search area which gives an approximate point of registration. A new search area is centered at the approximate point to find the optimal point of registration. Also, areas that might appear homogeneous are eliminated within a window as these tend to increase registration errors. The two-stage algorithm reduces the computation time required to locate the control points.

Rotational effects are compensated by rotating the registrant incrementally over some range of angles. The images are co-registered to

determine the best translational shift at each angle. The rotation angle that yields the lowest error is used. Small angular increments will yield more accurate results but will be more computationally intensive.

2.1.2.4 *Object Detection Via Clustering*

Another approach to automated image registration uses a clustering method to locate the same objects in different images. The method derives what has been labeled an RST transformation, which is an abbreviation for rotation, scale, and translation [Stockman *et al.*, 1982]. In this method, possible pairs of image features and model features are matched locally. The "model" features are those of the reference image, and the "image" features are those of the image to be registered. The orientation, position and size of the features must be distinguishable. Edge elements or point features are extracted and vectors are found between suitable pairs.

For each pair of features, the rotation angle (Θ), scale (s), and translation ($\Delta x, \Delta y$) are found and are mapped into a cluster space where each point represents the matching of one image element to one model element based on local features only. The premise is that a cluster of these points in the feature space will identify a good global transformation as it matches several image elements to corresponding model elements.

2.1.2.5 *Automated Multisensor Registration*

A method that addresses the problem of developing a robust automated multisensor registration technique that accommodates a wide variety of data types was proposed by Rignot and Kwok [1991]. The input data set was corrected for geometric distortions by sampling the data on an earth-fixed grid such as the UTM coordinate system, and then resampling to the same pixel

spacing. Sub-images are automatically selected which define local areas of coincident coverage where precise registration is possible. These are selected by locating temporally invariant features in each image. One possible technique mentioned uses binary edge maps to compute a figure of merit for candidate control points [Davis and Kenue, 1978]. These control points would be correlated to retain valid candidates. This method yielded a registration error of approximately 80 meters. For a ground resolution of 25 meters, this corresponds to a misregistration error of about 3 pixels.

Next, a method of registration was chosen. If ancillary data were available as simulated imagery (*e.g.* digital elevation maps, DEM), the images were registered to this simulated imagery, thereby inducing coregistration of the multisensor data on the common grid provided by the DEM. A simulated image was generated for a scene using the elevation data, viewing geometry, and a model of the scene reflectance taking into account the appearance of the scene for any given sun angle and viewing angle [Horn and Bachman, 1978; Woodham, 1980; Little, 1980; Frew, 1984].

In the absence of ancillary data, a method was employed that extracted invariant features across the different images. These were "feature matched" at multiple locations to establish correspondence between the images. Several techniques presented may be used in feature matching. These include binary cross-correlation, distance transform and Chamfer matching [Barrow *et al.*, 1977], dynamic programming [Maitre and Wu, 1989], and structural or symbolic matching. Constraint filtering was performed to eliminate false matches. The authors recommend using a combination of techniques to achieve better results due to the complexity of multisensor registration.

2.1.2.6 *Recognition of Corresponding Structures using Multivalue Logic*

Another method is based on similarity assessment of corresponding structures in the images [Ventura *et al.*, 1990]. This method is said to be insensitive to changes in scale, rotation, and intensity. Corresponding structures in different images provide sets of control points to derive geometric mapping functions. The input images are first segmented into sets of pixels which should correspond to meaningful structures for subsequent analysis. These structures are usually represented in a compact form such as contour points or skeletons. An assessment of similarity is performed which uses linguistic variables and multivalue logic. Pairs of similar structures are used to locate "corresponding pairs" of pixels which become a set of control points. These are points with the same physical structure that have been identified in each image.

2.1.2.7 *Using Feature-Based Mapping*

A method was developed at the Technical Research Centre of Finland which used feature-based mapping and robust estimation to search for ground control points [Holm, 1991]. In this method, appropriate features are selected from each image. Next, a preliminary set of candidate pairs of corresponding features is built. The third step is to create a list of feature pairs that is consistent with an object model.

The extracted features may be points, lines, regions, or shapes. The extraction can be done in a number of ways depending on the type of feature. The features are then assigned descriptions which may help in matching corresponding features. Such information as the texture in the neighborhood of a point feature, the orientation of a line feature, or the perimeter of a

region might be used as descriptors. The centers of gravity of the features may be included in the description as well. Features should be (1) invariant over time, (2) stable (*i.e.* contained in both images), (3) unique to the image, and (4) they should be different from neighboring features.

The next step is that of preliminary matching. Here, a list of candidate corresponding features is built based on the similarity of the descriptions of the features extracted from each image.

The final step, termed consistency matching, has three parts. First, an object model is chosen to reduce the number of good feature combinations. Second, a consistency measure is chosen to determine the closeness of fit of the data with the model. Third, an algorithm is chosen to find the optimal solution. The methods listed as possibilities are probabilistic relaxation, dynamic programming, relational matching, stochastic grammars, clustering, and robust estimation. The authors assumed that the transformation between images is affine *i.e.* that only linear terms in x and y are allowed,

$$x = a_{00} + a_{10}x' + a_{01}y'$$

$$y = b_{00} + b_{10}x' + b_{01}y'$$

The process is iterative until the final result is found.

1. The values of the similarity measure are used as preliminary weights of the feature pairs.
2. Using weights and least-squares techniques, residuals are computed for all feature pairs.
3. Outliers are removed.
4. New weights for the feature pairs are computed.

5. If a change in transformation parameters from previous iterations was detected, there are enough combinations left, and the iteration count is not too large, goto step 2.

Images could reportedly be registered to less than one pixel using only water (region-based) features.

2.2 *Normalization*

Most of the methods of registration described assume that the two images have been corrected (normalized) for atmospheric and illumination effects. Several techniques can be used to do this.

Satellite sensors measure the total radiance reflected from the many objects contained in a scene. The measurements are quantized to allow easy analysis and manipulation. However, many factors can degrade or distort the data. Teillet [1986] defines the major categories of radiometric effects or degradations as sensor related, (such as calibration and de-striping), and scene related, (such as atmospheric effects, topographic effects, and illumination and view angle). Scattering and absorption from gases and aerosols in the atmosphere will affect the measurement. The principal gases contained in air include nitrogen, oxygen, argon, carbon dioxide, and water vapor. Suspended particles of both solid and liquid matter make up the aerosol content of the atmosphere. Typical atmospheric aerosols are smoke and dust particles and products of vapor condensation [Francis, 1989]. The amount of scattering and absorption depends on the atmospheric conditions at the time the image was taken. Therefore, image normalization is necessary when analyzing and comparing images taken on different dates. Variations in atmospheric conditions, view angles, and sensor parameters occur between images that

have different acquisition dates. This will cause invariant regions to look different. The normalization process makes them appear as though imaged under the same conditions by removing the effects of sun angle, observer angle, sensor response, and atmospheric variations. This ensures that measured changes in reflectance correspond to real changes on the ground. Several techniques have been developed to compensate for these effects, thereby allowing the user to compare the images and obtain useful information from them.

2.2.1 Radiative Transfer Models

As stated, the radiance reaching the sensor is changed as it passes through a turbid medium (atmosphere) by absorption and scattering. Absorption is defined as a permanent loss of radiant energy from the radiation field and scattering as a loss of radiant energy that will reappear in the radiation field from some other direction. The sum of these losses is referred to as atmospheric extinction [Van de hulst, 1957].

An expression that defines the radiative transfer of energy from a ground object to the sensor is [Schott and Henderson-Sellers, 1984]:

$$L(\lambda, \theta, \theta', \phi) = L_r(\lambda, \theta, \theta', \phi) e^{-\tau'(\lambda) \sec(\theta)} + L_u(\lambda, \theta, \theta', \phi) \quad (8)$$

Where L is the radiance field at the sensor,
 L_r is the radiance reflected from the ground element,
 L_u is the path radiance along the line of sight between the ground object and the sensor,
 τ' is the vertical optical depth, *i.e.* the product of the extinction coefficient and the vertical distance traversed by the radiation,
 θ is the angle between the sensor and the normal to the surface,
 θ' is the source elevation angle relative to the normal to the surface, and
 ϕ is the azimuthal angle defined between the source and sensor projected onto the surface.

The reflected radiance L_r can further be broken down into the components of direct solar radiance and downwelled sky radiance. A simplified version that represents the radiance field at the sensor is:

$$L(\lambda, \theta, \theta', \phi) = \tau(\lambda, \theta, \theta', \phi)[L_{sun}(\lambda, \theta, \theta', \phi) + L_{sky}]r(\lambda, \theta, \theta', \phi) + L_u(\lambda, \theta, \theta', \phi) \quad (9)$$

where

L_{sun} is the direct solar radiance reflected from a scene element, and
 L_{sky} is the integrated radiance falling on the same scene element from the sky dome which is reflected toward the sensor by the reflectivity r .

Therefore, "the radiance reaching the sensor from a particular scene element is a linear function of the reflected radiance with the coefficients representing atmospheric transmittance and upwelled radiance [Francis, 1989]."

These atmospheric inhomogeneities can be compensated on a pixel-by-pixel basis as long as the atmospheric parameters can be determined for each pixel. If $\tau(\Delta\lambda, \theta, \theta', \phi)_{i,j}$ and $L_u(\Delta\lambda, \theta, \theta', \phi)_{i,j}$ are the atmospheric transmittance and upwelled radiance in the passband $\Delta\lambda$ for pixel (i,j), the corrected radiance, L_c , is

$$L_c(\Delta\lambda, \theta, \theta', \phi)_{i,j} = \frac{L(\Delta\lambda, \theta, \theta', \phi)_{i,j} - L_u(\Delta\lambda, \theta, \theta', \phi)_{i,j}}{\tau(\Delta\lambda, \theta, \theta', \phi)_{i,j}} \quad (10)$$

A method was developed to compensate for atmospheric effects which utilized the systematic changes in LANDSAT radiance measured over water [Scarpance, 1979]. The change in radiance for imagery from several different dates was attributed to a change in atmospheric haze. The technique used a simple linear atmospheric radiative transfer model with clear lake water as a standard with reflectance $r=0$ in the near-IR. Therefore, any reflectance in this region of the spectrum allows the user to estimate atmospheric path radiance.

Several methods are available to model radiative transfer. The most common is the LOWTRAN method developed by the Air Force Geophysics Laboratory (AFGL) [Kneizys *et al.*, 1983]. The LOWTRAN method uses a band model and empirical data to develop an integrated absorption that is representative of the passband of interest. This is in turn used to compute the average transmittance from each of the four LOWTRAN atmospheric molecular absorption constituents, *i.e.* water vapor, ozone, nitric acid and the uniformly mixed gases. Molecular scattering (Rayleigh scattering), also contributes to atmospheric transmission loss [Francis, 1989]. The assumption that water reflects no infrared (IR) radiation is used to empirically determine the transmission and upwelled radiance terms. A histogram of clear lake water is generated and a radiance value calculated for each digital count. using the following equation:

$$L_{MSS} = \frac{DC_{lake}}{DC_{max}}(L_{max} - L_{min}) + L_{min} \quad (11)$$

The L_{max} and L_{min} are sensor specific values and can be obtained from EOSAT (Earth Observation Satellite data). The values should have the units of (W/cm^2sr) when used in the calculation but may be published in (mW/cm^2sr). The format for input data can be found in the LOWTRAN

manual. Radiosonde data is used along with a constant Henyey-Greenstein scattering phase function of 0.15 while the visibility is altered by trial and error until the calculated radiance value is matched. This gives the transmission and upwelled radiance value for the atmosphere above the lakes for a pixel with that particular digital count. These are used in equation 12 to calculate sensor radiance:

$$L_{ground} = \frac{L_{sensor} - L_u}{\tau} \quad (12)$$

Another method also uses clear lakes to account for atmospheric effects was developed by Gordon (1978). It "assumes an atmosphere composed of spectrally selective Rayleigh scattering and spectrally independent aerosol scattering". The radiance component from the water body that is due solely to the atmosphere is computed from knowledge of the wavelength and the aerosol component by finding the IR radiance from the water body and subtracting the part due to Rayleigh scattering [Francis, 1989].

This method was improved in 1985 by combining the technique of Gordon with measurements made by Ahern [Verdin, 1985]. An atmospheric propagation model is used similar to that of Scarpace. The lake surface radiance is calculated from Aherns' data. This is combined with the lake surface radiance measured from a LANDSAT MSS image and entered into the atmospheric propagation model. Other inputs to the atmospheric propagation model are varied to compute a set of atmospheric parameters that can be used to calibrate the image radiance values.

Other methods used to perform image normalization use ground truth to remove atmospheric and sensor effects from individual images. The images are reduced to a surface-information base which assumes identical sensors.

Such an approach may rely on the use of clear water bodies present in the scenes to extract path radiance information [Ahern *et al.*, 1979] and allow a spatially variant atmospheric correction.

The main difficulty in using these models is the need for atmospheric input data which is not always available. Some of these models require radiosonde data to compute the scattering due to air molecules and aerosols and the absorption due to gaseous components [Caselles, 1989]. At the Canada Centre for Remote Sensing, a dynamic regression algorithm was developed for incorporating atmospheric models into image correction procedures [Teillet *et al.* 1987]. The method uses a discrete ordinate method (DOM) as a model to simplify the radiative transfer equation. To decrease the processing time, the radiative transfer code is run at a limited number of points on a grid. The resulting parameters are combined algebraically in accordance with the radiometric correction equation [Teillet, 1986], fitted with a surface function, and sampled at each pixel location. A simpler model, the so called 5S model, can be run instead.

The dynamic regression algorithm is used so that the atmospheric parameters can be computed without repeatedly running the DOM. The input values for the model are wavelength, solar zenith angle, observer zenith angle, relative azimuth between solar and observer directions, aerosol optical depth through the total atmosphere, and average ground surface reflectance.

2.2.2 *Histogram Modification*

A major problem in each of the above methods is that processing must be performed on all of the images in the temporal set to obtain absolute data normalization. It is desirable to use a normalization technique which operates on a single image, causing it to look like the reference image. In this

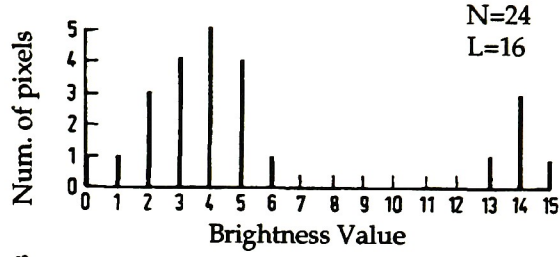
manner, image normalization can be performed in a relative way. Less error is expected due to the simplicity of such an approach and there is no need for information other than that contained in the scene. The following are some methods using this type of approach.

2.2.2.1 *Histogram Equalization and Specification*

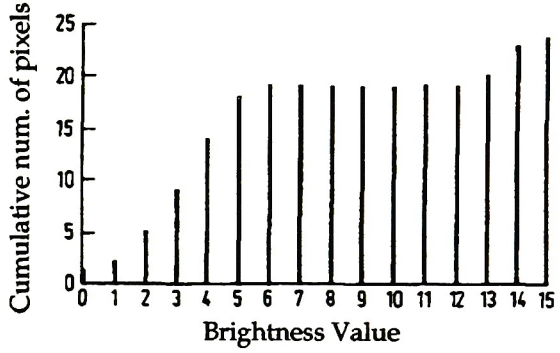
Histogram equalization is a standard image processing technique for contrast enhancement [Gonzalez and Woods, 1992]. Here, a histogram is altered such that it approximates the uniform or flat histogram where the gray levels are equally populated (Figure 2.2). The cumulative distribution function (CDF) is computed and scaled to be used as a look-up table to move histogram bars to new brightness value locations.

The result is a "flattened" normalized output histogram [Richards, 1986].

A. Histogram



B. Cumulative Histogram



C. Resulting Uniform Histogram

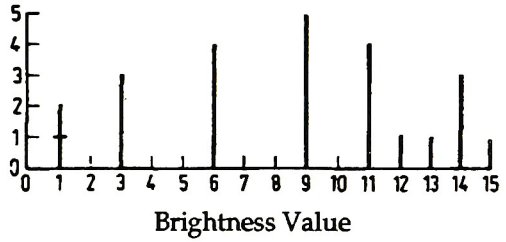


Figure 2.2 Histogram equalization [Salvaggio, 1987]

This hypothetical image has 24 pixels quantized (4 bits), or 16 gray levels (Figure 2.2 A). The corresponding cumulative histogram is shown in figure 2.2.B and is scaled to 4 bits to use as a lookup table. The gray level after equalization is the value of the scaled ordinate at that abscissa [Richards, 1986]. The calculated value is scaled and the closest discrete brightness value is assigned to that pixel. The scale factor for the cumulative histogram is:

$$S.F. = \frac{L-1}{N} \quad (13)$$

where: L is the number of available brightness values, and
N is the number of pixels.

Histogram specification is a generalization of this technique to transform the histogram of an image to match that of a reference image. Rather than normalize all images in a set to the uniform distribution, it may be desirable to transform the images to match the histogram of a reference image. The CDF's associated with each image can be used in combination as a single mapping function. The CDF of the to-be-mapped image is used in the forward direction and the CDF of the reference image is used in the reversed direction. By mapping each pixel through both, the first histogram would be transformed to have the same distribution as the second (Figure 2.3).

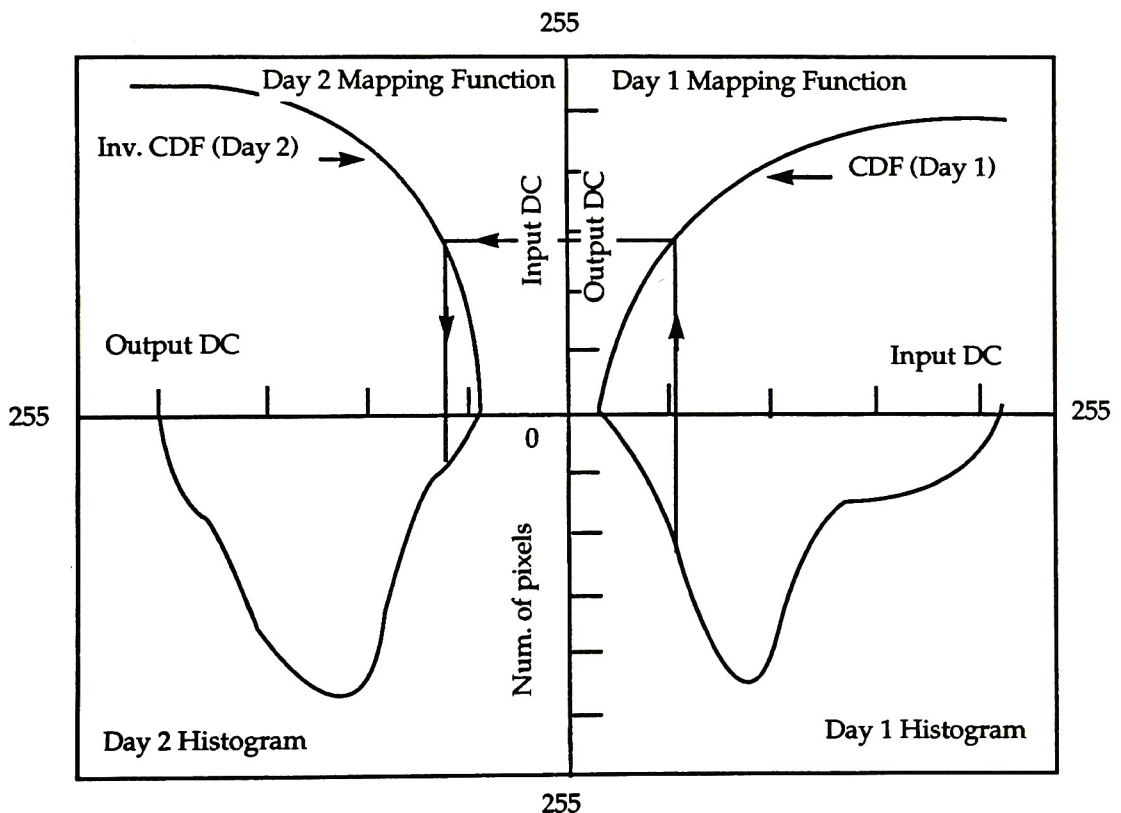


Figure 2.3 Histogram specification, matching Day 1 histogram to Day 2 histogram [Salvaggio, 1987]

2.2.2.2 Linear Histogram Transformation

Linear transformation of a histogram is another technique which modifies the brightness distribution function under certain conditions. [Salvaggio, 1987]. Two assumptions must be met for this method to be valid. The first is that the radiance at the sensor is linearly related to the reflectivity of the scene elements on the ground. The second is that the brightness (or digital count) is a linear function of the radiance reaching the sensor. The measured radiance can be expressed as a linear function of reflectivity:

$$L = K_1 r + K_2 \quad (14)$$

where:

L	is the radiance reaching the sensor (W/m ² sr),
r	is reflectivity,
K ₁	is a constant that includes solar irradiance, downwelled sky radiance, and atmospheric transmission, and
K ₂	is the path radiance.

Since the digital count in each band is a simple linear function of the radiance reaching the sensor, we can say that

$$DC = K_3 L + K_4 \quad (15)$$

where:

- DC is the digital count recorded by the sensor,
- K_3 is a constant which takes into account optical throughput, detector responsivity, and system gain, and
- K_4 includes optical flare of the sensor system and electronic bias.

Objects with the same ground reflectivity on both dates are simple linear functions of the same variables and therefore, are linear functions of each other. One image is the reference image (ref) and one is the mapped image (map).

$$\begin{aligned} DC_{map} &= m_{map} L_{map} + b_{map} \\ DC_{ref} &= m_{ref} L_{ref} + b_{ref} \end{aligned} \quad (16,17)$$

where DC_{ref} and DC_{map} refer to digital count values for each image, and the ' m ' terms are linear coefficients that take into account the sensor response characteristics, and the ' b ' terms are the sensor offset values. By combining the equations, we obtain:

$$\begin{aligned} DC_{map} &= m_{map} \alpha_{map} R_{map} + m_{map} + \beta_{map} + b_{map} \\ DC_{ref} &= m_{ref} \alpha_{ref} R_{ref} + m_{ref} + \beta_{ref} + b_{ref} \end{aligned} \quad (18,19)$$

Two linearly related histograms can be transformed to look like each other as follows. The relative width of the histograms are related by the ratio of their standard deviations [Salvaggio. 1987]. The histogram means are recomputed

and their difference is added to the adjusted mean \bar{x}_{ref} . The histograms now have equivalent spreads and equal mean values (see figure 2.4). The transformations are :

$$DC_1 = m_i DC_2 + b_i \tag{20}$$

$$m_i = \frac{\sigma_2}{\sigma_1} \tag{21}$$

$$b_i = \bar{x}_2 - m_i \bar{x}_1 \tag{22}$$

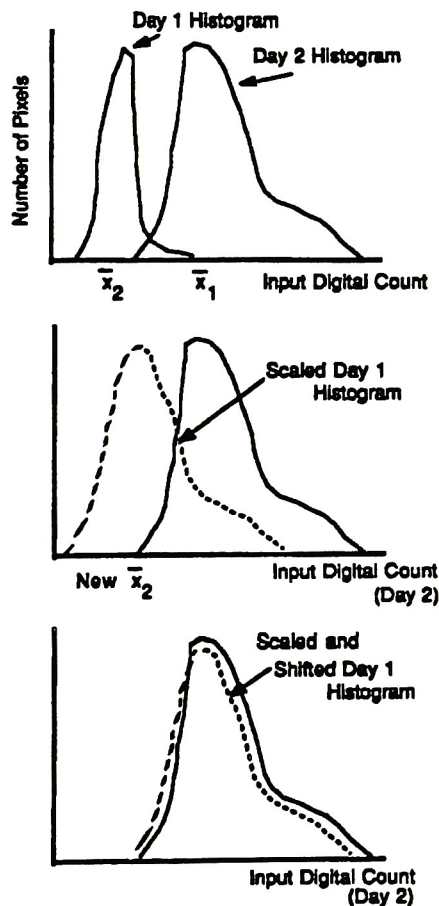


Figure 2.4 Transformation of linearly related histograms
[Salvaggio, 1987]

As long as the assumptions are correct, the linear histogram transformation process will give a better result than histogram specification. However, when simply computing the mean and standard deviation across the entire image rather than using only invariant features, the method may not give acceptable results.

2.2.3 *Pseudoinvariant Feature Normalization*

One method which does address the issue of automated temporal scene normalization uses so-called pseudoinvariant features. This process is a linear histogram transformation of the invariant features of the image. The features used are typically non-natural, *e.g.* roads, urban areas, asphalt, *etc.* and are assumed to have statistically invariant spectral signatures over time [Piech and Schott, 1974]. These invariant features are located in each image and their gray level distributions are determined within each band. To derive the necessary set of linear transformations, two assumptions must be met [Volchok, 1985]. First, that the radiance at the sensor is linearly related to the reflectivity of the scene elements on the ground. Second, that the brightness of the image must be linearly related to the radiance reaching the sensor. A set of transformations can be derived to modify the histograms of the second image to look like the histograms from the first. These transformations can then be applied to the entire second digital image.

The process can use automated image segmentation to isolate the pseudoinvariant features from the two images [Salvaggio, 1987] using equations as 20-22:

$$DC_1 = m_r DC_2 + b_r \quad (20)$$

$$m_r = \frac{\sigma_2}{\sigma_1} \quad (21)$$

$$b_r = \bar{x}_2 - m_r \bar{x}_1 \quad (22)$$

Other researchers have independently developed techniques similar to PIF. Eckhardt and Verdin [1990] describe an "empirical scene normalization approach" which was used in an attempt to match the detector calibration and astronomic, atmospheric, and phase angle conditions present in the reference scene. To do this, one image was chosen as a reference scene to which the others would be normalized. The normalization was performed by applying regression equations to the gray levels which would predict the digital count had it been acquired under the same conditions. These equations were derived by matching the digital counts of targets present in both the scene to be normalized and the reference scene. The reflectance of the targets was assumed to be invariant so that any changes could be associated with atmospheric effects. This is basically the same assumption used to select PIF features as described above.

Caselles [1989] offers "an alternative solution for atmospheric correction. This is achieved by using the apparent reflectance values of ground surfaces which are assumed to have constant ground reflectance with time so that atmospheric input data is not required." Again, this approach is very similar to the PIF approach.

A problem arises when using PIF techniques on MSS images. Because the pixel size (IFOV) of an MSS image is 79×79 m, it is difficult to identify pixels that cover only an invariant feature, *i.e.* few man-made features are at least 160×160 meters in size. When histogram normalization cannot be used for radiometric correction, a similar technique to PIF was proposed by Yokota and Matsumoto [1988]. Two small areas in the Landsat MSS images were selected which are thought to be radiometrically invariant over time. These were a

bright concrete runway and the dark surface of the Pacific Ocean. The wide variation in digital counts allows for development of PIF transformations.

2.3 *Classification of Images*

The intent of image classification is to specify a particular land type (such as soil, water, vegetation, *etc.*) for each pixel [Lillesand, 1987]. The number of classes will vary with the particular scene and application. Classes may be coded by color or symbol for visual interpretation [Schowengerdt, 1983]. Spectral classification uses the radiance values in each spectral band of the image to identify the class to which each pixel belongs. Different surface materials will have different spectral reflectances over the spectrum. Spatial pattern recognition techniques may also be employed which account for the shapes and sizes of objects in the image. However, classification based on spectral criteria is currently more advanced.

A multispectral sensor yields a digital count for each band; this discrete set is the spectral signature of an object. It can also be thought of as a multidimensional vector of the brightness values [Richards, 1986]:

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \quad (23)$$

where x_i are the brightnesses of the pixel in bands i . Different land cover types will exhibit different combinations of digital counts based on their spectral reflectance and emittance properties [Lillesand, 1987]. The pixels are classified by using the available spectral data,. The process is illustrated in Figure 2.5.

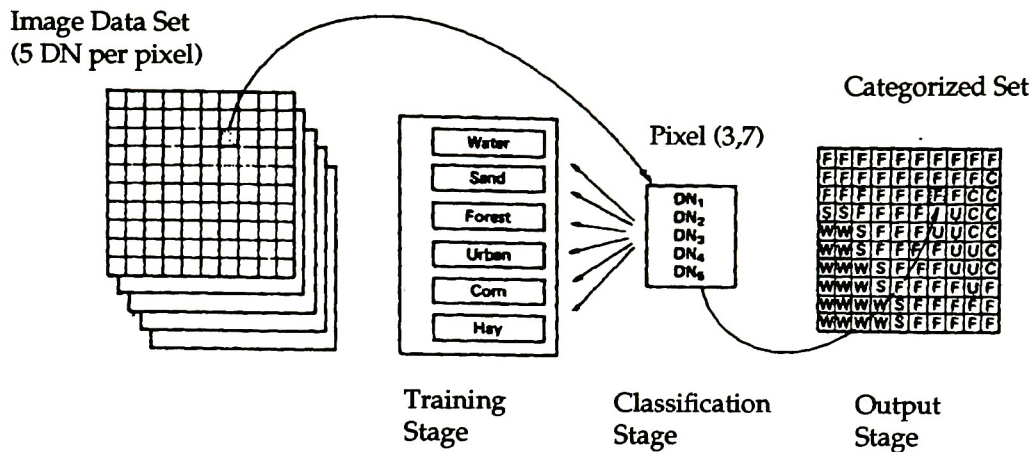


Figure 2.5 Basic steps in supervised classification [Lillesand, 1987]

Generally speaking, classification of images can be performed in a supervised or unsupervised manner. In supervised classification, the categories or classes are defined by the user. These are subsequently analyzed to determine the spectral separability of the categories. In unsupervised classification the spectral separability of the raw data is determined to create the classes which must be interpreted by the user. Often, a combination of these methodologies is used.

An automated approach to the problem of image classification which excludes the use of a supervised approach is desired to eliminate user interactions. Ideally, an unsupervised classification would be performed on the first image and the categories would be identified. All subsequent imagery would be classified by using the information generated from the first image.

2.3.1 Supervised Classification

The essential steps in supervised classification described by Richards are:

1. Determine the set of ground cover types in the image, *e.g.* water, vegetation, soil, urban, *etc.*

2. Choose pixels in the image from each class that are representative of that class. This is called training stage of the data. *A priori* information from other sources, e.g. maps, ground truth, and air photographs may also be used to aid in the training process.
3. The training data set is used to estimate the parameters of the classifier. These parameters are the properties of the probability model used or may be equations that define the partitions in multispectral space.
4. Each image pixel is classified into one of the ground cover classes.
5. Tabular summaries or thematic class maps are created to summarize the results of the classification.

2.3.1.1 *Maximum Likelihood Classifier*

Maximum likelihood is the most common method of supervised classification. This classifier evaluates the variance and the covariance of the category spectral response patterns [Lillesand, 1987]. First, it is assumed that the pixels in the training set are normally distributed. Under this assumption, the spectral response pattern can be described completely by the mean vector and the covariance matrix. Then, the statistical probability that a given pixel belongs to a land cover class is calculated. The following diagram simulates the probability density functions for a maximum likelihood classifier.

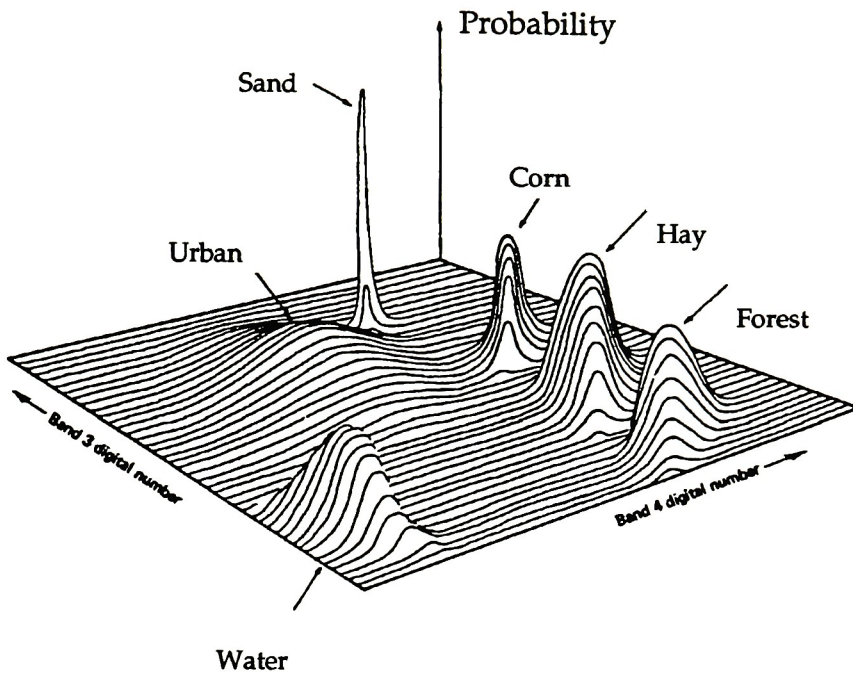


Figure 2.6 Probability density functions defined by a maximum likelihood [Lillesand, 1987]

The probability density functions are used to classify each pixel by computing the probability that a pixel belongs to each class. The pixel would ultimately be assigned to that class with the highest probability. A pixel can also be labeled as unknown if the probability values are below some threshold.

The following example illustrates maximum likelihood classification as well as other common methods of assigning pixels to classes. The training data from an image is plotted on a scatter diagram that is representative of the different classes (Figure 2.7).

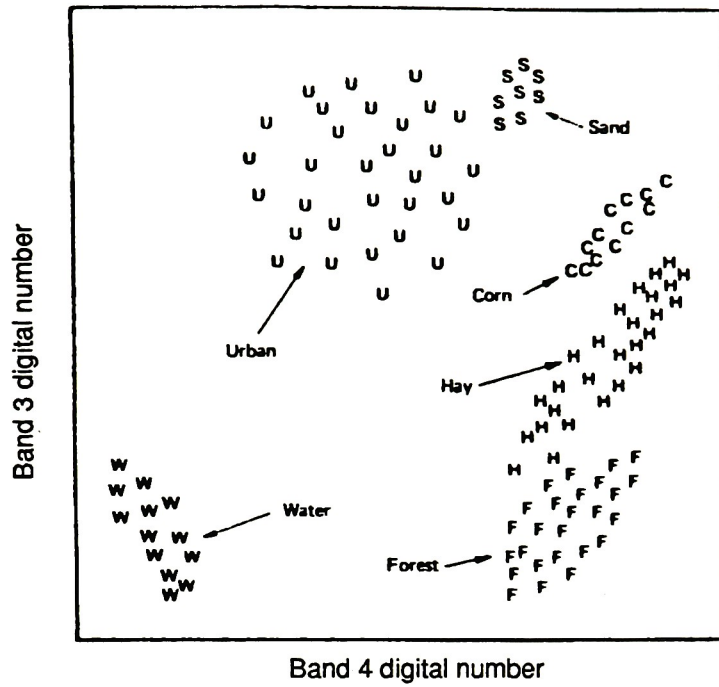


Figure 2.7 Pixel observations from selected training sites plotted on a scatter diagram [Lillesand, 1987]

These pixels are used to generate "equiprobability contours" used to classify all other pixels. The decision regions are shown in Figure 2.8.

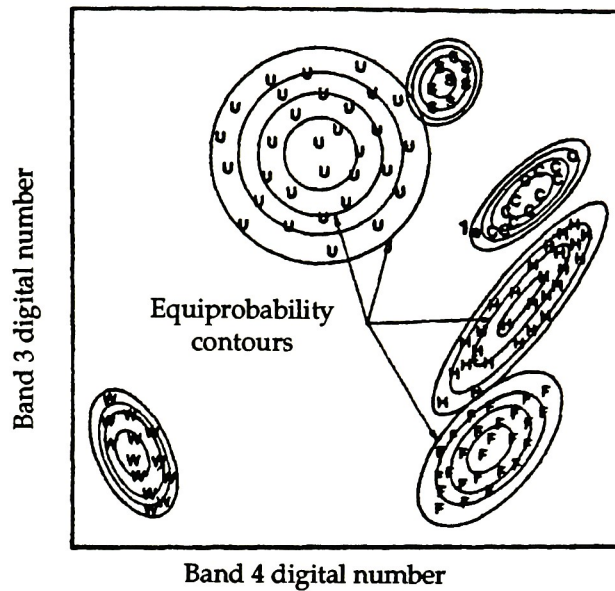


Figure 2.8 Equiprobability contours defined by a maximum likelihood classifier [Lillesand, 1987]

In an extension of this method, weighting factors might be applied to the different classes to create a better classifier. If "sand" is expected to occur rarely, it would be weighted lightly compared to more likely features.

2.3.1.2 Minimum-Distance-to-Mean Classifier

In this method, the mean spectral value of each spectral band is computed for each class chosen in the training step. These values are the components of the mean vector for each class. A pixel in the image can be identified as belonging to one of the classes by calculating a Euclidean distance between the values of each band for that pixel and those for the class means. After making

the necessary computations, the pixel is assigned to the class that is closest to its vector (Figure 2.9).

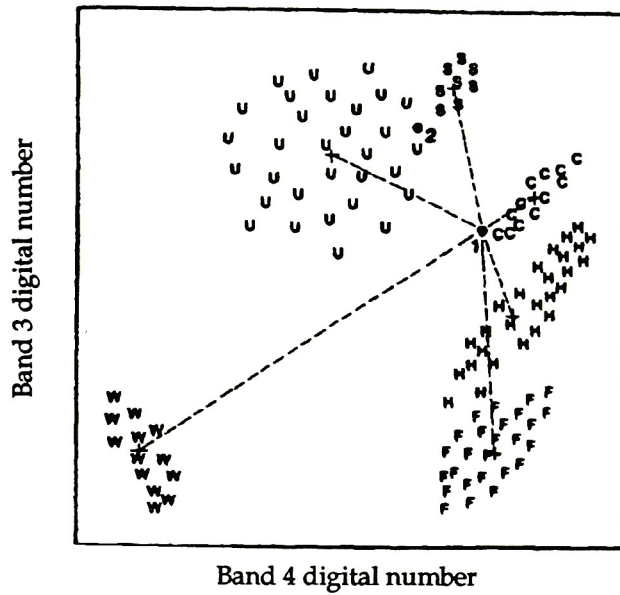


Figure 2.9 Minimum distance to mean classification strategy [Lillesand and Kiefer, 1987]

This method is computationally simple, but it does not account for the variance of any of the classes. A pixel lying on the outskirts of a class with greater variance may be closer to the mean of another class. However, it is entirely possible that the pixel in question belongs to the former group. This could result in misclassified pixels.

2.3.1.3 *Parallelepiped Classifier*

This is a technique which accounts for the variance of the classes by determining the range of values in each band. Boundaries are defined around each class according to the highest and lowest digital counts in each band. A pixel is classified based on this range. A pixel that lies outside all regions is

classified as unknown. In two dimensions, these boundaries can be pictured as rectangles (Figure 2.10). For multidimensional data, the boundaries are parallelepipeds.

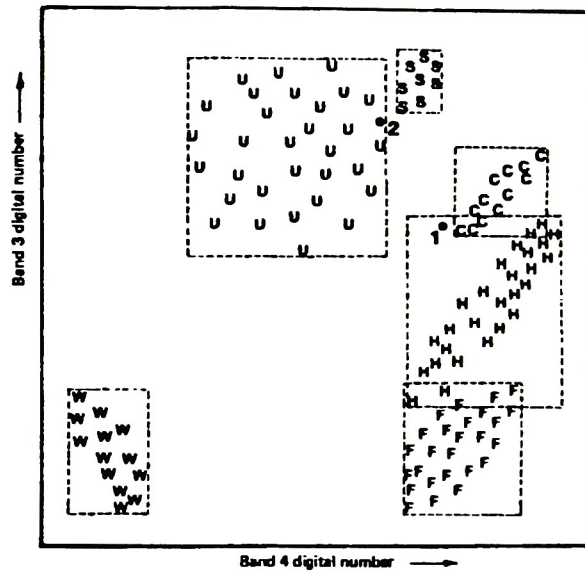


Figure 2.10 Parallelepiped classification strategy [Lillesand, 1987]

This classifier is also fast and computationally simple, but has limitations. Two regions that are close together may have overlapping boundaries. The classifier will randomly assign the pixel to one class or the other under these conditions. This overlap of boundaries is a result of the bands being highly correlated so they are not well described by the rectangular areas. This correlation manifests itself in the scatter diagram as elongated or stretched classes with a positive or negative slope (Figure 2.11). The classes that show high correlation are far too large when a rectangular boundary solution is used as illustrated by the dashed rectangle.

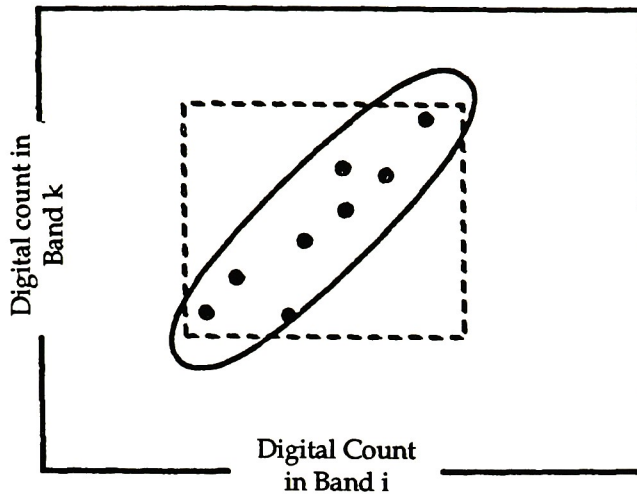


Figure 2.11 Weakness of a parallelepiped classifier for highly correlated data [Salvaggio, 1987]

Because spectral response patterns are often highly correlated, this problem is significant. It can be alleviated somewhat by using stepped boundaries (Figure 2.12), but this method is probably not sufficiently accurate.

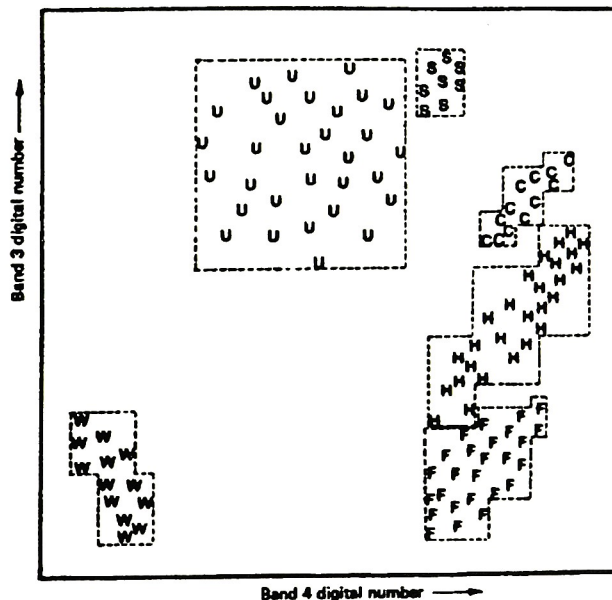


Figure 2.12 Parallelepiped classification strategy employing stepped decision region boundaries [Lillesand and Kiefer, 1987]

2.3.2 *Unsupervised Classification*

In unsupervised classification, the classes or categories are determined by measuring the distance between pixel clusters. The algorithms are used to examine unknown pixels and group them with other pixels having similar spectral characteristics [Lillesand, 1987]. Basically, pixels that belong to a particular spectral class should have spectral characteristics that are close together. After specifying the classes, they must be identified by the user to associate each group with a particular land cover type. Again, maps may be useful in helping the analyst identify each group. An advantage to unsupervised classification is that the spectral classes are found automatically rather than through training. This is advantageous since the user may have trouble distinguishing between certain classes or there may be so many that it is difficult to train the image.

There are many ways that the required clustering can be done. One common clustering algorithm is called the "k-means" approach. The analyst must decide the number of classes. Initial cluster means, called "seeds", are positioned randomly throughout the image and used to start the clustering process. Each seed value is the center of a cluster. Every pixel in the image is assigned to the cluster with the closest mean. After each pixel is assigned to a cluster, the mean vectors are recalculated for each and a new iteration is performed. This procedure continues until the class mean vectors do not change with iteration [Lillesand, 1987]. This approach is computationally intensive due to its iterative nature. This is only one of many possible clustering algorithms that can be used, however, it does not seem pertinent to describe others. The k-means, or ISODATA approach is illustrated in Figure 2.13.

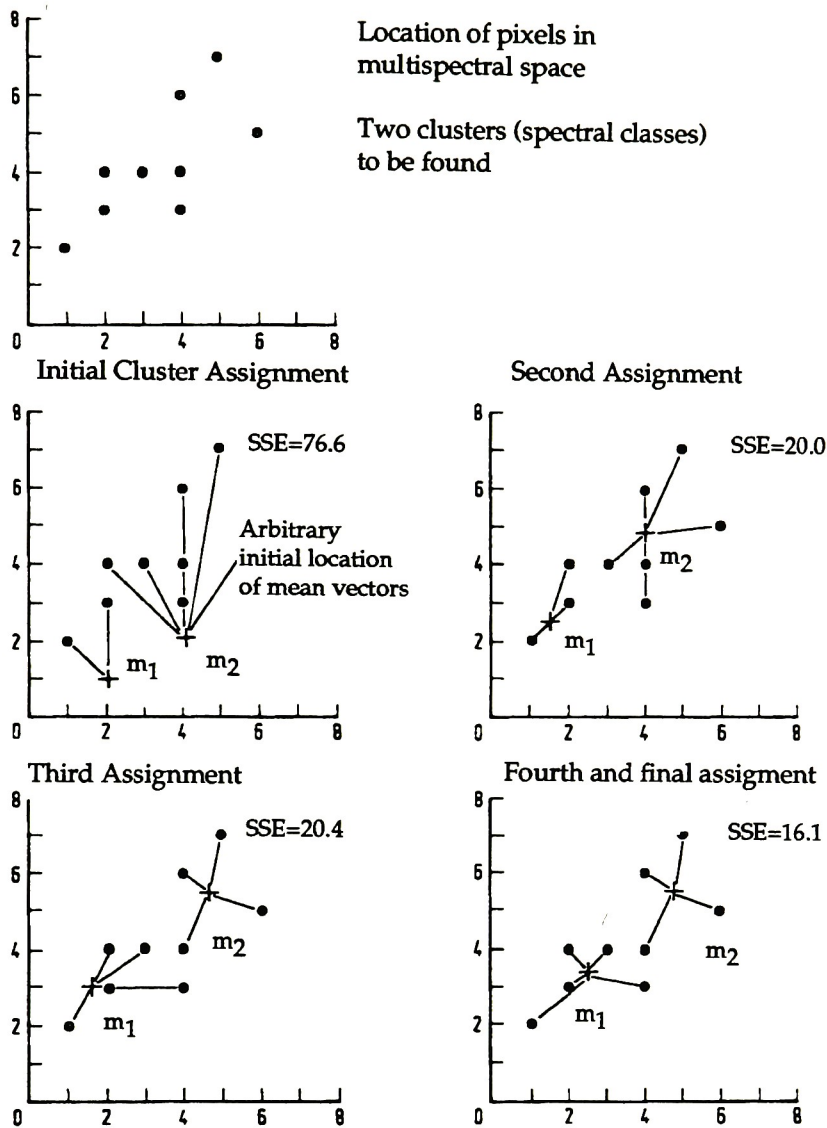


Figure 2.13 An illustration of clustering by iterative optimization (ISODATA method) [Richards, 1986]

To reduce computation time, unsupervised training areas are often chosen so that the clustering algorithm can be applied to a smaller area. The chosen areas must contain all land types that are found in the scene, that is, they must be heterogeneous. It is important to make sure that all spectral classes in the scene are included in these training areas.

2.3.3 *Hybrid classification*

A combination of supervised and unsupervised methods (*i.e.* a hybrid scheme) may be the best means to classify an image.

After clustering is completed, the population of each category can be examined to see if some clusters are underpopulated. Such classes may be eliminated or combined with another class. When complete, training statistics are calculated that are representative of the scene. Next, available ground truth or reference data can be used specify the class types. Richards describes a third step to determine if all features (*i.e.* spectral bands) need to be retained to classify the image reliably. The entire image is then classified into the set of spectral classes. A minimum-distance-to-mean algorithm is often used. Each pixel is then labeled by ground-cover type.

Unsupervised classification is desirable because it reduces the demands on the user. Ideally, training statistics would be created on the reference image and applied to the other images as well. This way, the time needed to perform the classification will be minimized and the process will limit user interaction as well.

3.0 *Experimental Procedure*

The primary objective of this thesis is to develop a method for automated image registration, normalization, and classification in the overall process of change detection. The automated registration is based primarily on the method described by Salvaggio and Schott [1987]. Though they showed that correlation had significant potential, they did not fully develop the method. Recall that this method is not fully automated, but requires an initial manual registration of a reference image so that all subsequent images can be registered to it. Also, some pre-processing of the images to be registered must be performed to make the correlation more efficient and effective. As a result of the degree of user interaction required, the method is most valuable when several images of the same area are to be compared. In the current study, the process of image normalization proved to be difficult. Several methods were tried; the best was a histogram modification technique. The user must match the histograms of each band in the to be registered images to those of a reference image using ERDAS software. This requires minimal processing, but is not automated. Finally, the images were classified which is also a difficult task to automate. It was desired that one image be classified and the spectral signatures generated in this classification be used to classify all subsequent images.

A basic flowchart of the project is shown in Figure 3.1.

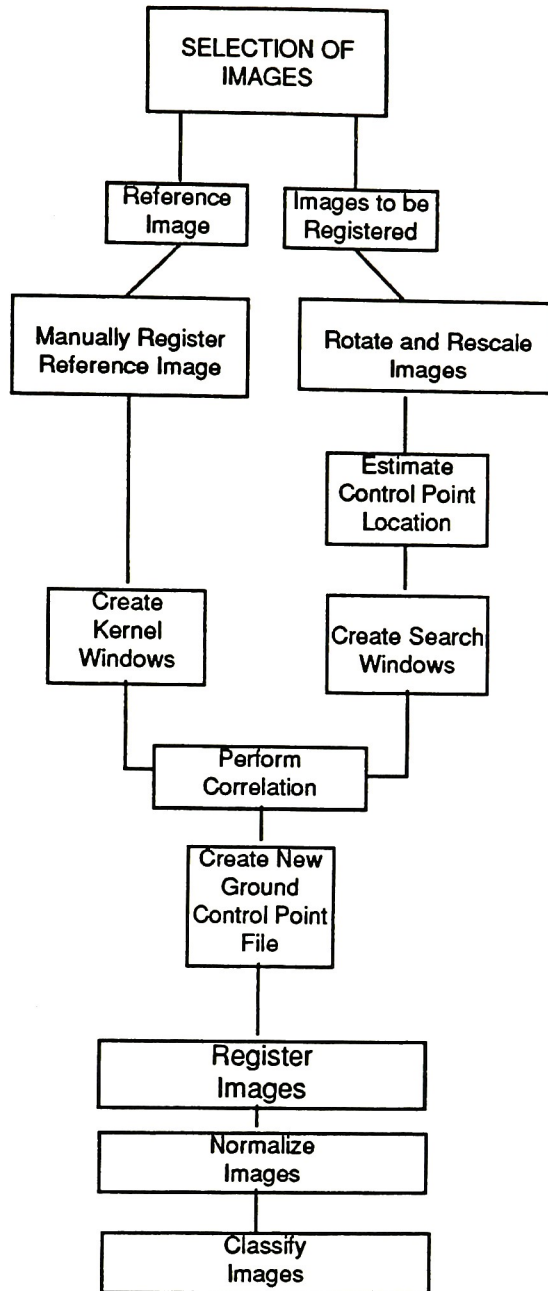


Figure 3.1 Flowchart of Work

3.1 *Selection of Imagery*

The choice of imagery to show changes in forest cover due to clearcutting in the northwest was limited primarily by cost. Because the Landsat MSS (multispectral scanner) scenes used were two years old or older at the time of

purchase, they were available at a discounted rate. The center coordinate of the area of interest is given to Earth Resource Observation Satellite Data Center (EROS) in Sioux Falls, SD to obtain a list of images that contain the coordinate. Care must be taken to select imagery that covers roughly the same area. It was desired that the images have zero percent cloud cover to ensure that areas of interest not be obscured. The final criterion for choosing the images was that the acquisition dates be as close together as possible in their respective year so that snow cover and water levels would be roughly the same and to ensure that they were imaged under similar illumination conditions. The best compromise was found in images that were obtained over a three-week interval. The following images were chosen:

Table 1 **Images used in study**

	Reference	Image 1	Image 2
Date	7-29-72	8-23-81	7-30-88
Scene ID	81006183135	83126718175	85161218262
Spacecraft	Landsat-1	Landsat-3	Landsat-5
Resolution	79 x79 meters	57 x 57 meters	57 x 57 meters

The earliest image was chosen as the reference image and the others as the images to be registered. These will be referred to as images 1 and 2 respectively. The scenes from 1981 and 1988 both had 2983 x 3596 pixels, while the image from 1972 had 2340 x 3240 pixels. Sections of the original Landsat scenes were chosen as the study area to conserve disc space.

3.2 *Manual Registration*

A reference image (typically the oldest image) was manually registered to the U.S.G.S topographic map of Wenatchee, Washington (1:250,000 scale, Zone 10), using conventional registration techniques. A digitizing table was used to locate the coordinates on the map that corresponded to known points on the image. These ground control points must be chosen carefully to obtain satisfactory registration. It is important to calibrate the map on the digitizing table so that the UTM coordinates are accurate. To set up the digitizing table, the map is affixed to the table and known grid marks are digitized. The user must not begin choosing control points until a satisfactory map set-up has been completed. Due to the poor spatial resolution and rural nature of the images, it was difficult to locate "ideal" control points; *i.e.* features that are unlikely to have changed over time. For example, water features were used despite the potential for error due to fluctuation in water levels. The control points were distributed throughout the image to improve the results. The ground control point file (GCP) contained 24 points and used to compute the appropriate transformations in the ERDAS software package (use of all ERDAS programs used are briefly described in Appendix F). A second-order transformation was computed with an arbitrary RMS error tolerance of 2.0 pixels. When performing a least squares regression to determine the transformation, a general rule of thumb is that at least 3 control points per coefficient should be used. In this case, a second-order transformation was used to find six transformation coefficients. Of the 24 points chosen, one was discarded due to its large error to reduce the total error. Once this image is registered to the map, all subsequent images were automatically registered to it.

3.3 *Automated Registration*

The program MAKE_GCP_FILE.C (Appendix B) was created which carries out the steps detailed in the following sections.

3.3.1. *Creating the Correlation Kernels*

The same control points used in the manual registration were located for automated registration by image correlation. Simply stated, the section of the image that contains each control point was extracted from the registered (reference) image. This is the correlation kernel. In the images to be registered, search areas were created based on estimates of the location of that control point in the image.

The geometric transformation obtained by manual registration should warp the image to the same projection as the U.S.G.S. map. Therefore, though the control points in the reference image are located at different positions than in the original image, map coordinates are now associated with them. To create a correlation kernel, the UTM coordinates of the control point from the reference image are supplied to the computer program MAKE_KERNEL_UTM (Appendix A), which locates the input coordinate and builds the kernel by extracting the surrounding pixel pattern and storing it in a file. The control point is always located in the center of the kernel. The size of the kernels can be varied by the user to ensure that the structure of the control point is apparent. The only restriction was that the kernels must have an odd number of rows and columns so that the control point was located precisely in the center. Each kernel was stored in a separate file and contained all four bands from the Landsat scene. Though all bands were extracted, the actual correlation is performed using only the band with the greatest contrast.

3.3.2 *Approximate Registration*

As stated earlier, correlation is highly sensitive to rotation and scale differences. Although all images used were Landsat MSS images, there are still noticeable rotation and scale differences due to the different platforms of each.

MSS scenes have an instantaneous field of view (IFOV) of 79 meters on a side. The sampling rate used in data acquisition resulted in a nominal ground spacing of 56 meters between readings so the pixels are averages over cells of 56×79 meters. Digital MSS data is supplied in computer-compatible tape (CCT) format after 1979, and was resampled into pixels having a nominal dimension of 57×57 meters. This was verified by empirical testing as it is not well outlined in the data supplied with each tape. To test the pixel size in meters, two points were chosen on each image and their UTM coordinates found using the digitizing table. Since UTM coordinates have units of meters, the distance between the points could be determined. By using the pixel location data, the number of pixels between the two points was also determined. The ratio of the distance in meters to the number of pixels is the scale of the image in meters per pixel. The 1981 and 1988 images were verified to have resolution of 57 meters per pixel, while the resolution of the 1972 image was 79 meters per pixel.

The images with the 57×57 meter spatial resolution, were resampled to 79×79 meters per pixel to match the resolution of the 1972 image by reducing the scale by a factor of $57/79 = 0.7215$ in both the x and y directions.

To compensate for rotations, the angle of rotation between the reference image and the images to be registered was estimated. Again, two points were chosen which could be located in all images and have the same UTM coordinates (identical points) but different pixel locations. The locations of

these points must be translated to the origin (0,0) in order for the equations to work. The following equations used:

$$\vec{A} = \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} ; \quad \vec{B} = \begin{bmatrix} x_2 \\ y_2 \end{bmatrix} \quad (24,25)$$

$$|\vec{A}| = \sqrt{\vec{A} \bullet \vec{A}} = \sqrt{x^2 + y^2} \quad (\text{magnitude of vector A}) \quad (26)$$

$$\vec{A} \bullet \vec{B} = (x_1 \times x_2) + (y_1 \times y_2) \quad (27)$$

$$\theta = \cos^{-1} \left(\frac{\vec{A} \bullet \vec{B}}{|\vec{A}| |\vec{B}|} \right) \quad (28)$$

To determine the angle between the two vectors, the coordinates defining the vectors are supplied to the program VEC_ANGLE (Appendix C). A positive angle is a counterclockwise rotation relative to the reference image. The image should be examined visually to determine which way the to-be-mapped image needs to be rotated and ensure that it is rotated correctly, *i.e.* clockwise vs. counterclockwise. For example, given two points in images A and B with the following coordinates:

A1:	(2223,1344)	B1:	(667,1251)
A2:	(2347, 773)	B2:	(678, 666)

Recall that these points must be translated so that one point is located at 0,0. This is done by subtracting the coordinates of one point from both.

x values

2223 - 2223 = 0

2347 - 2223 = 124

y values

1344 - 1344 = 0

773 - 1344 = -571

Such that the new points are:

A1: (0,0)

B1: (0,0)

A2: (124,-571)

B2: (11,-585)

The result is an angle of rotation of -11.175 degrees (clockwise). To increase accuracy, three points were chosen in each image and the average angle of rotation used. The angles were -13.09 and -11.46 for the 1981 and 1988 images, respectively.

Once the angle of rotation and the relative scale were found, the images were *approximately* registered by the ERDAS program LRECTIFY which allows the user to perform both the rotation angle and the scaling at one time, thus minimizing the resampling of the data.

3.3.3 *Locating the Search Windows*

To perform the correlation, a search window must be created within the images to be registered. These are areas in the to-be-registered images that are thought to contain the same feature as the corresponding kernel. They are located using the ephemeris data supplied with each image (Appendix E). The search windows are large enough to ensure that the structure of interest is contained within the area, without being so large that the processing time would be prohibitive. They are designed to be three times larger than the kernel dimensions resulting in a search area nine times as large. This was determined to be large enough by empirically examining the images to be registered. The true and estimated locations of the control points were found in each image so that the required size of the search areas could be

determined. In the worst case, it was found that the desired control point would lie within this window

The correlation is performed only where the kernel fits entirely within the search area. The resultant correlation window will be smaller than the search area by an amount equal to half of the kernel dimensions on all four sides. Therefore, if the search area is 75 x 75 pixels and the kernel is 25 x 25 pixels, the correlation result will be 51 x 51 pixels because it will lose 12 pixels per side, per dimension. This must be remembered when creating the search windows. If they are not large enough to take this into account, the control point may be located in that outer perimeter of the search window and would not be located with the correlation operator. The diminishing of the search area is illustrated in Figure 3.2. The first point where the kernel fits within the search area is shown. The center of this kernel is where the correlation result is placed. Therefore, the area shaded and labeled "correlation" contains all of the values as the kernel is translated through the search area.

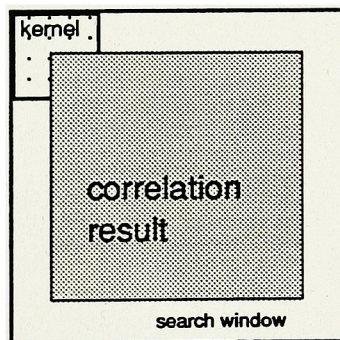


Figure 3.2 Correlation result

The area that is expected to contain the control point is often not easy to locate. It is necessary to know the location on the map of three corners of the image. Three noncolinear points are needed to find a transformation equation with three coefficients. The latitude and longitude coordinates of the

corners are available for every Landsat Scene. These can be easily transformed to UTM map coordinates using the ERDAS CCVRT program. The pixel locations of each corner must be found empirically since the scene has since been rotated and scaled. However, the UTM coordinates associated with each corner remain the same.

Stored with each kernel are the UTM coordinates of the control point (kernel center) that must be located in the to-be-registered images. The three control points can be thought of as two maps: one with pixel coordinates and one with UTM coordinates. Based on this information, an equation can be found which takes UTM coordinates to pixel coordinates. Once this is found, it can be used to transform the known UTM coordinates of the control point of interest to the pixel location of that point. A larger window is built around this estimated point. The sequence of calculations to locate the UTM coordinate in the images to be registered are:

$$\begin{bmatrix} a_{00} & a_{10} & a_{01} \\ b_{00} & b_{10} & b_{01} \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \\ x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \end{bmatrix} = \begin{bmatrix} x'_1 & x'_2 & x'_3 \\ y'_1 & y'_2 & y'_3 \end{bmatrix} \Rightarrow \underline{A}\underline{X} = \underline{X}' \quad (29)$$

Matrix \underline{A} contains the coefficients needed for the transformation, matrix \underline{X} contains the UTM coordinates of the corner points, and \underline{X}' contains the pixel locations of those corner points. We want to find \underline{X}^{-1} , i.e. the inverse of the matrix \underline{X} , which can be found since the matrix is square and the chosen three control points are not collinear. Therefore, the coefficients for the matrix \underline{A} can be found:

$$\underline{A}\underline{X}\underline{X}^{-1} = \underline{A} = \underline{X}'\underline{X}^{-1} \quad (30)$$

In the following equation, the unprimed values represent the UTM coordinates and the primed values are the pixel coordinates. Now that the

coefficients are known, the UTM coordinates of the control point of interest are used to calculate the approximate location of the point as follows:

$$\begin{bmatrix} a_{00} & a_{10} & a_{01} \\ b_{00} & b_{10} & b_{01} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x' \\ y' \end{bmatrix} \quad (31)$$

or:

$$x' = a_{00} + a_{10}x + a_{01}y \quad (32)$$

$$y' = b_{00} + b_{10}x + b_{01}y \quad (33)$$

3.3.4 *Correlation*

According to equations 3-5 in section 2.1.2.1, the correlation is performed between the kernel window and corresponding area in the search window. The correlation is performed each time the kernel is translated across the search area and the result stored at the location of the kernel center. The values in the resultant image are examined to locate the maximum. The position of the maximum correlation should correspond to the control point in the new image. The pixel locations and UTM coordinates are stored in a ground control point file (GCP). New transformation equations are generated and the registration is performed.

The program is designed to read each kernel individually from a file containing all of the kernel names. The output is the set of ground control point coordinates. The program automatically reads in the second kernel and proceeds.

3.4 *Normalization*

Three methods to optimize the normalization procedure were tried. A preferred method would use one image as a reference image and alter the others to make them appear as if taken on the same day. This minimizes the

amount of processing required compared to absolute methods that must alter all images in the set. The methods are:

- (1) Histogram Matching
- (2) Linear Mapping Using Invariant Features
- (3) Linear Mapping Using entire image statistics

3.4.1 *Histogram Matching (Specification)*

This was the method that was ultimately used and is a nonlinear histogram matching technique. ERDAS software contains a program called HSTMATCH which allows the user to match the histogram of one image to that of another image. The histogram matching must be performed individually on each band and are then recombined after processing to form the final histogram-matched image. A statistics file was used to provide the input data for the program. Ninety-seven percent of each histogram was used to avoid using outlying data that might skew the results. After running HSTMATCH the user must use the program STRETCH which actually processes the image. This must be done directly after running HSTMATCH as the data is not actually stored in any file.

3.4.2 *Use of Invariant Features to Perform Linear Mapping*

Objects on the ground that were assumed to be statistically invariant were located and used to estimate any changes in digital value between images as due to changes in the atmosphere. Several features were identified in all images and their digital counts recorded. A look-up table was created for each band by placing the desired output digital count on the y-axis and the input digital count on the x-axis. The desired output value is that of the reference image and the input value is that of the image to be changed. The digital

counts of the selected urban features are plotted and a simple linear regression performed. The input value of the images to be altered were simply plugged into the equation to determine the output digital value. This process is very simple and quick. The selected features should be well distributed throughout the scene to average any spatial variations in atmospheric conditions. It is assumed that the features that may have changed between the image acquisition dates, such as vegetation, will have been affected by the atmosphere in the same manner as the urban features. The steps of the process are:

1) Selection of urban features:

This proved to be difficult given the poor resolution of the MSS imagery. It was very difficult to locate pixels which contain only one urban feature. For example, a major interstate that runs through Seattle consisted of mixed pixels and could not be used in the calculation. Another problem with the imagery used was that the scene was primarily composed of rural areas with few invariant features.

Seven pixels were chosen and located in the three images. An attempt was made to choose points that covered the dynamic range of each band.

2) Creation of look-up tables:

An example will be used which uses only 2 points to illustrate the method. Suppose the following data for one band.

	REFERENCE IMAGE:	DAY 2 IMAGE
POINT 1	10	23
POINT 2	55	42

This data is plotted and a linear regression performed to determine the equation of the line created.

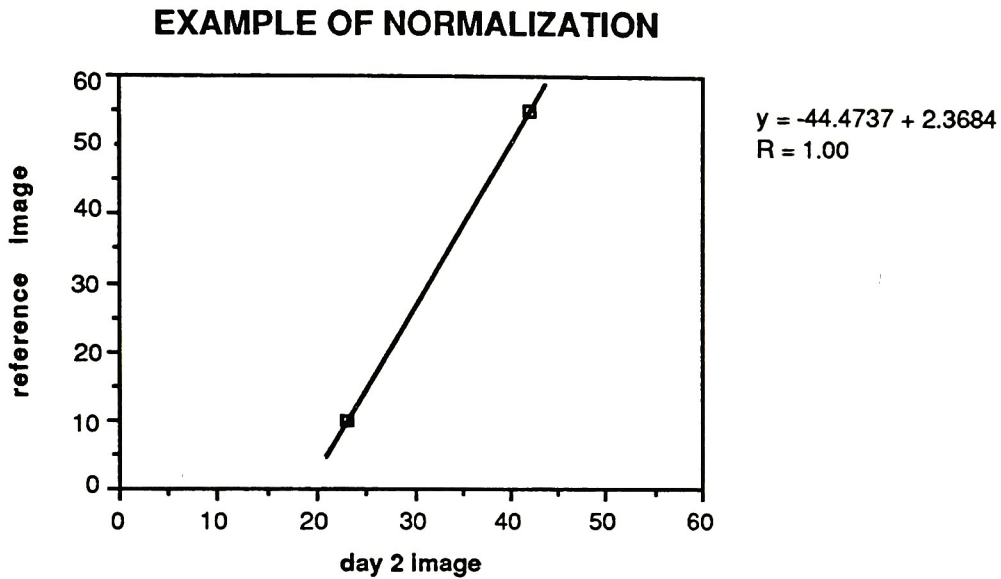


Figure 3.3 Example of linear transformation using invariant features

A simple linear regression was performed to determine the equation of the line. In the above example, the equation

$$DC_{\text{new}} = -44.47 + 2.37 DC_{\text{old}} \quad (34)$$

would be used to transform the day 2 image where DC_{new} and DC_{old} are the digital count values for the new and old images respectively. This can be performed quite easily using ERDAS software with the routine ALGEBRA.

Again, due to the poor resolution of MSS imagery, it was very difficult to locate appropriate urban features. The pixels may be "mixed" pixels, *i.e.* they may contain more than one class. Even if the look-up table is successful in "normalizing" those pixels, it may not necessarily compensate for the atmospheric properties over the entire image.

3.4.3 Linear Mapping Using Entire Image Statistics

This method is typically used on the image statistics of the invariant features. However, due to the poor resolution of this imagery and the fact that it contains very few urban features, this was not possible. Instead, the statistics generated from the entire area of interest were used. In this method the mean and standard deviation of each band was used to calculate a linear transformation as described in section 2.2.2.2, equations 18-20. The following example will help to illustrate the method.

Raw Data:

	Reference Image	To be mapped image
Mean	23.1	38.9
Standard Deviation	16.6	15.1

The slope of the line is calculated using equation 19 which is then plugged into equation 20 to determine the y-intercept of the line:

$$m_t = \frac{\sigma_{ref}}{\sigma_{to-map}} = \frac{16.6}{15.1} = 1.1$$

$$b_t = \bar{x}_{ref} - m_t \bar{x}_{to-map} = 23.1 - 1.1 \times 38.9 = -19.7$$

The linear transformation is obtained using equation 18 as follows:

$$DC_{new} = m_t DC_{old} + b_t = 1.1 DC_{old} + (-19.7)$$

In this manner the histogram of one image can be made to look like the histogram of another image. Once the appropriate transformations are

determined for each band, they can be entered into the ERDAS program ALGEBRA to process the image. This is a linear histogram matching technique as compared to the non-linear histogram specification method.

3.5 *Classification*

The classification process was fairly straightforward, yet the degree of success depends greatly on the quality of the normalization.

The reference image was classified using both supervised and unsupervised methods to compare the results. It was found that the latter did a better job of spectrally separating the classes so it was pursued as a means of "training" the data. The ERDAS program ISODATA was used to perform the unsupervised classification. The user must estimate the number of classes in the image. It is safer to overestimate the number of classes contained in the image as they can always be recombined at a later time. The success of the method is determined by examining the results relative to ground truth. The ERDAS program COLORMOD was used to examine and identify each class. This program can also be used to merge or delete spectral signatures. These signatures are used to classify the other images using MAXCLAS as in supervised classification. In MAXCLAS, a minimum distance classification was found to give better results than a maximum likelihood classifier. Figure 3.4 illustrates this.

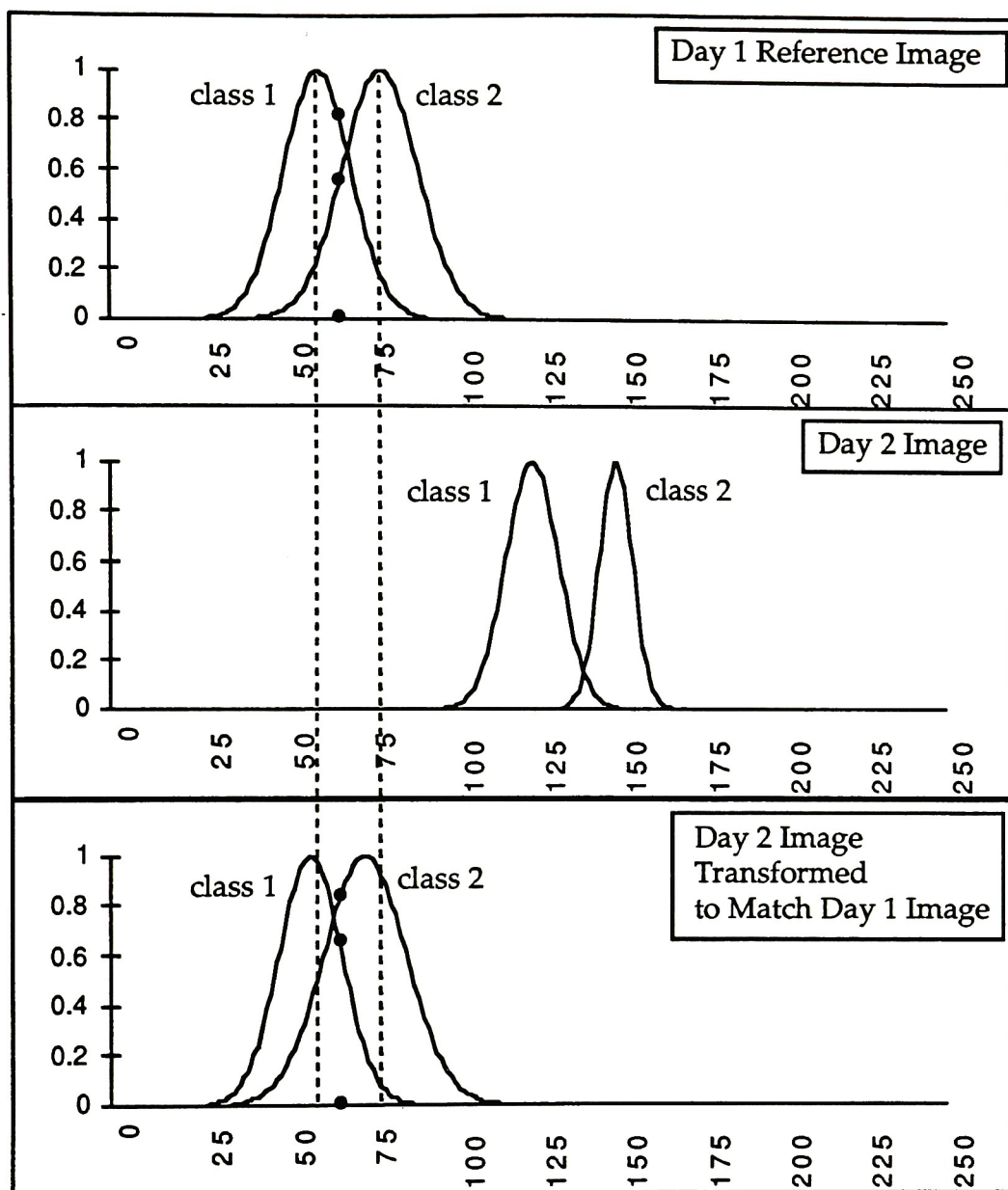


Figure 3.4 Minimum-distance-to-mean vs. Maximum Likelihood classification

Figure 3.4 shows the one-dimensional probability curves for two spectrally similar classes contained in two images that are to be compared. The top section is the reference image and the third section represents the result of the normalization process of histogram matching on the Day 2 image. The dashed lines represent the class means of the two classes in the

reference image. In maximum-likelihood classification a pixel is assigned to the class that it has a higher probability of belonging. In the illustration, digital count 60 is marked by a black dot. The probability of a pixel belonging to a class can be determined by imagining a straight line extending from the x-axis and crossing the class boundaries. In this example, the pixel belongs to class 1 because it has a higher probability as shown in the diagram. In a minimum-distance-to-mean classification, the pixel shown in the reference image would also be classified as a class 1 pixel because its value is closer to the class 1 mean.

A problem can arise in spectral classes that are close together in that they may not be made to match the reference image exactly. As seen in section three of Figure 3.4, the transformed classes are not perfectly centered around the class means. Recall that the reference image statistics are used to classify all images in a set. In this case, as a result of the shift and the closeness of the classes, it is seen that a maximum likelihood classifier will assign the pixel as a class 2 pixel in the Day 2 image. However, the minimum distance to mean classification still accurately classifies this pixel as class 1.

The result of using a maximum likelihood classifier on an image used in this study is shown in Plate 20. This problem is more likely to occur when using LANDSAT MSS imagery since there are only 128 gray levels available in each band.

4.0 *Results*

4.1 *Manual Registration of Reference Image*

As stated previously, the reference image was manually registered to a map. The user can determine the success of the registration in pixels. This is done by dividing the UTM difference between actual control points and those associated with a pixel after registration by the resolution of the images (in meters for UTM coordinates). That is, if the difference in UTM coordinates is 200 meters, the error would be 2.53 pixels using the information that MSS imagery has a spatial resolution of 79 meters per pixel. For some applications, it is necessary to have sub-pixel accuracy. To set up the map, it is important to test several known points to ensure that minimal error is being introduced in this step. Table 2 lists the control points used in the manual registration.

Table 2 Control points used in manual registration

	x Pixel UTM Coordinate	y Pixel UTM Coordinate	x Pixel Location	y Pixel Location
1	592324	5263436	2354	798
2	593596	5257544	2404	867
3	598055	5256323	2482	871
4	596498	5282152	2337	556
5	606502	5296234	2436	357
6	600352	5312486	2256	173
7	604517	5314819	2315	133
8	587360	5304404	2075	311
9	580574	5286484	2046	549
10	624711	5286340	2791	426
11	616392	5263449	2762	728
12	583881	5246069	2293	1036
13	580678	5242794	2256	1086
14	579900	5221448	2344	1349
15	578321	5220669	2321	1363
16	604747	5219883	2772	1295
17	633392	5236586	3182	1011
18	616825	5305188	2569	214
19	649302	5292815	3191	275
20	621244	5250117	2911	879
21	606942	5193946	2929	1606
22	610294	5266797	2643	704
23	623767	5226690	3060	1157

These coordinates were used to derive the appropriate transformations. Several points were chosen to test the accuracy of the registration on both dependent data (points used to generate the transformations) and independent data (points not used in the registration). The points were located in the image using the CURSES program in ERDAS, and on the map using the digitizing table. The UTM coordinates now associated with each

pixel can be compared to those obtained from the digitizing table and the error may be calculated in meters and in pixels. Table 3 contains the errors for these test points for both the x and y directions for the dependent data set; Table 4 lists them for the independent data set.

Table 3 Registration error for dependent points

Δ UTM x	Δ UTM y	Δ Pixel x	Δ Pixel y
78	110	0.99	1.40
-16	-34	0.20	0.43
54	-87	0.68	1.10
80	-35	1.01	0.44
-122	68	1.54	0.86
-55	37	0.69	0.46
168	141	2.12	1.78

RMS pixel error in x direction 1.28

RMS pixel error in y direction 1.12

Table 4 Registration error for independent data points

Δ UTM x	Δ UTM y	Δ Pixel x	Δ Pixel y
34	11	0.43	0.14
137	89	1.73	1.12
-270	66	3.40	0.83
80	19	1.01	0.24
127	-29	1.60	0.37
19	-166	0.24	2.10
75	67	0.94	0.85
129	485	1.63	6.10
135	114	1.71	1.44
184	181	2.32	2.29

RMS pixel error in x direction: 1.83 pixels

RMS pixel error in y direction: 2.40 pixels

Upon completion of manual registration, the kernels were extracted from the newly registered image. The control points are distributed over the area that was used to register the images. The distribution of control points chosen in the 1972 image is shown in Plate 1. The inner rectangle corresponds to the outlined area in Plate 4 to show the relationship between the area used in image registration (Plate 1) and that used in the normalization and classification stages (Plate 4). Plate 4 is the 1972 image (reference) that the 1981 and 1988 images were made to match. Since the correlation procedure is highly sensitive to differences in image scale and rotation, the to-be-mapped images had to be approximately registered to the reference image prior to performing the correlation operation. The 1988 image is shown in Plate 2 before and after approximate registration, thus illustrating the effect of rotational variation between the images. Plate 3 shows the result of correlating a kernel with its corresponding search area. The point in the correlation with the highest gray value is recorded as the located control point. Tables 5 and 6 list the final ground control points that were located in the 1981 and 1988 images, respectively. The first column contains the actual control points which were located manually for comparison purposes. The second column contains the points that were found in the intermediary step used to create the search windows. The errors of the approximations ranged from 0 pixels to as large as 17 pixels in one dimension for the 1981 image and between 1 pixel and 25 pixels for the 1988 image. This was adequate for creating search areas of a reasonable size (75 x 75 pixels). The third column lists the ground control points that were used in the final registration. The last is the error between the final ground control points chosen and the actual points in each image.

Table 5 1981 ground control points

Actual		Approximate		Correlation		Difference	
x	y	x	y	x	y	Δx	Δy
2412	1307	2416	1306	2412	1306	0	-1
2538	734	2542	722	2538	734	0	0
2251	1368	2253	1369	2252	1368	1	0
2210	1411	2210	1412	2211	1411	1	0
2197	1670	2198	1677	2197	1670	0	0
2515	1702	2523	1711	2515	1701	0	-1
2375	1479	2378	1483	2375	1479	0	0
2358	1148	2360	1144	2359	1148	1	0
2429	1240	2433	1239	2429	1240	0	0
2290	1107	2290	1102	2290	1107	0	0
Bad Point		2739	1324	2731	1344	*	*
2876	1492	2887	1497	2876	1492	0	0
2502	884	2509	873	2504	883	2	-1
2308	671	2308	657	2308	672	0	1
2296	629	2296	613	2296	629	0	0
2298	555	2298	538	2297	555	-1	0
2768	860	2775	851	2768	860	0	0
Off of this Image		2435	492	2410	467	*	*
2375	1224	2377	1222	2375	1224	0	0

Table 6 1988 ground control points

Actual		Approximate		Correlation		Difference	
x	y	x	y	x	y	Δx	Δy
1933	1552	1932	1550	1933	1553	0	-1
2060	981	2043	961	2060	981	0	0
1773	1614	1770	1616	1772	1614	1	0
1731	1656	1728	1660	1731	1656	0	0
1718	1916	1723	1927	1718	1916	0	0
2036	1947	2051	1953	2035	1948	1	-1
1895	1726	1899	1728	1895	1726	0	0
1880	1394	1872	1388	1879	1394	1	0
1950	1486	1948	1482	1950	1486	0	0
1811	1353	1800	1347	1811	1353	0	0
2250	1581	2257	1561	2254	1576	-4	5
2395	1740	2411	1731	2395	1740	0	0
2026	1130	2015	1113	2027	1131	-1	-1
1830	918	1807	901	1830	917	0	1
1818	875	1793	857	1818	876	0	-1
1818	799	1793	782	1821	801	-3	-2
2289	1108	2280	1085	2289	1108	0	0
1955	755	1931	732	1954	754	1	1
1896	1471	1891	1466	1896	1471	0	0

Some of the points were not well chosen due to poor structure of the control point, because the control point was not contained in the second image, or because the control point was not located within the search area chosen.

After selecting the control points, the transformations are generated. A set of constraints is entered into the COORDN program to ensure that the best fit is found. If these constraints are not met, the program prompts the user for action to be taken. One choice is to remove the control point with the largest error. This is done until the constraints are met. Points are removed from the

registration in this manner. Therefore, providing that a sufficient number of points are found accurately, it is not problem if a control point cannot be located as it will be removed. Appendix (D) contains the results of the COORDN program. The registered images are shown in Plates 5 and 6 and have been cropped to cover identical areas. Sections of each image were subset and used to create one image to show the continuity after registration (Plate 18).

4.2 Normalization

4.2.1 Histogram Matching (Specification)

This method yielded the best results of the three used. Image statistics from the ERDAS program BSTATS were examined after processing for comparison to the reference image (Plate 4). Each image was visually examined after transformation through the reference-image look-up table. This allowed the user to see any differences present between images. Plate 7 and 8 are the images that were normalized using the histogram matching method. These images most closely resemble the reference image when compared to the other normalization methods. Table 7 and 8 list the statistics of the three images prior to and after normalization, respectively.

Table 7 Original image statistics

	BAND 1		BAND 2		BAND 3		BAND 4	
	Mean	St. dev.	Mean	St. dev.	Mean	St. dev.	Mean	St. dev.
1972	20.79	8.40	14.52	13.03	38.69	17.31	23.10	16.57
1981	17.40	8.99	14.09	10.24	37.57	13.96	38.86	15.07
1988	16.88	4.40	13.38	6.74	42.78	13.73	47.6	15.81

Table 8 **Histogram matched statistics**

	BAND 1		BAND 2		BAND 3		BAND 4	
	Mean	St. dev.	Mean	St. dev.	Mean	St. dev.	Mean	St. dev.
1972	20.79	8.40	14.52	13.03	38.69	17.31	23.10	16.57
1981	19.80	10.00	13.66	11.04	37.50	15.29	22.50	12.17
1988	20.44	4.91	14.25	6.33	37.04	13.86	21.02	9.14

When the statistics are compared, the means of each band are closely matched but the standard deviations have considerable differences. It is believed that this is due to the differences in snow cover between the three images.

4.2.2 *Use of Invariant Features to Perform Linear Mapping*

Seven pixels distributed throughout the scene were chosen that were thought to be urban features that are invariant over time. Ideally, these pixels would have been well distributed in reflectance *i.e.* they would cover the full dynamic range of each band. Unfortunately, it was difficult to locate such points in this imagery. The data points in Table 9 were located in each image and their digital count values recorded. Each is the value of an individual pixel only, not the average of an area. This is because the urban features are too small to obtain an average digital count value.

Table 9 Digital count of invariant pixels in each band

PIXELS 1972	BAND 4	BAND 3	BAND 2	BAND 1
1	16	36	40	39
2	18	44	41	41
3	21	53	53	50
4	14	34	37	38
5	10	26	25	27
6	17	43	37	38
7	19	41	31	31
PIXELS 1981				
1	30	36	32	32
2	36	47	41	36
3	45	63	63	50
4	24	33	34	31
5	19	28	29	27
6	28	37	32	32
7	33	42	33	29
PIXELS 1988				
1	26	38	32	34
2	34	37	37	26
3	53	65	72	52
4	25	33	35	30
5	27	30	29	26
6	31	40	30	31
7	30	37	28	27

From these data, look-up tables were created as described in the experimental procedure. Each band must have its own look-up table. However, after performing the regression, all four equations can be input into ALGEBRA (ERDAS) to process the image without further user interaction. An example of the look-up tables created is shown for the 1981 image.

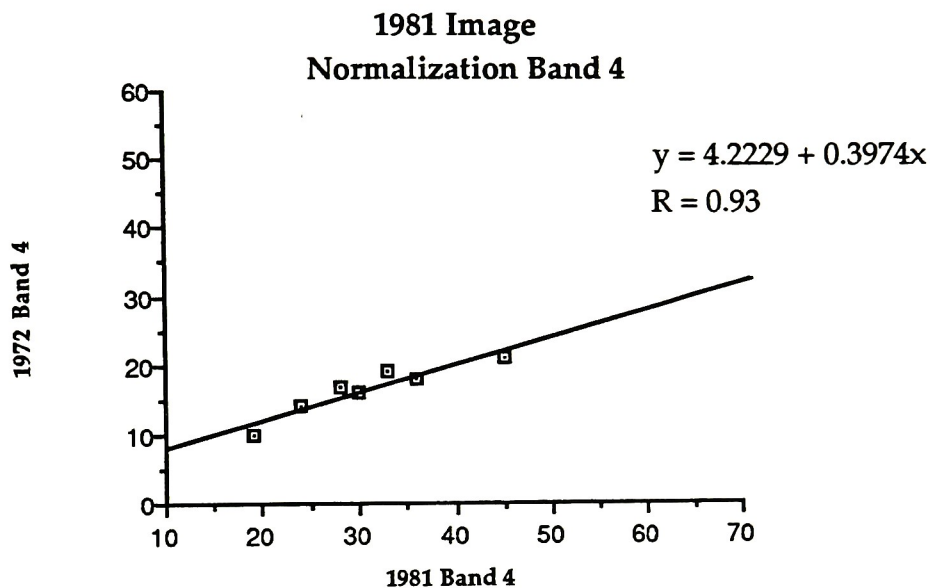


Figure 4.1a 1981 band 4 look-up table, linear transformation using invariant features

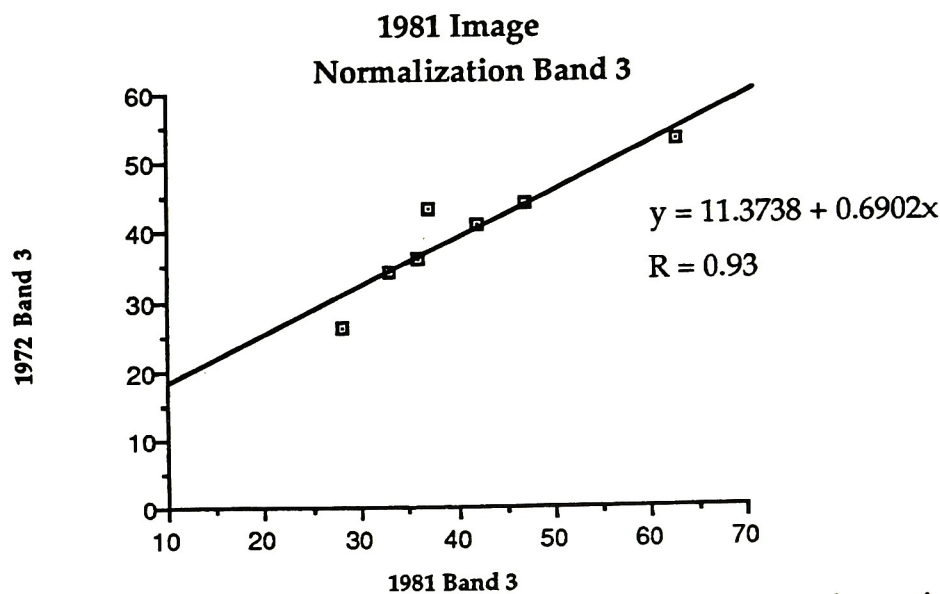


Figure 4.1b 1981 band 3 look-up table, linear transformation using invariant features

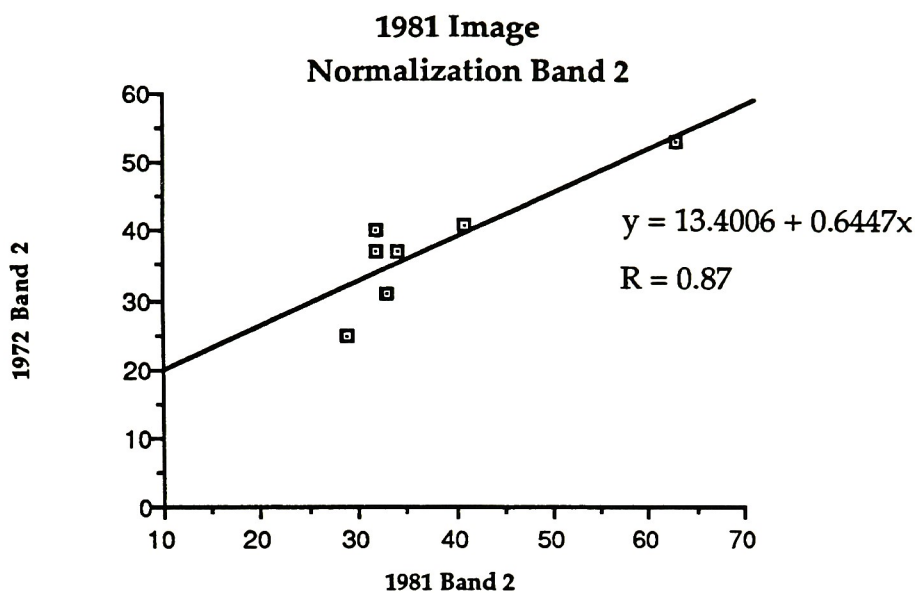


Figure 4.1c 1981 band 2 look-up table, linear transformation using invariant features

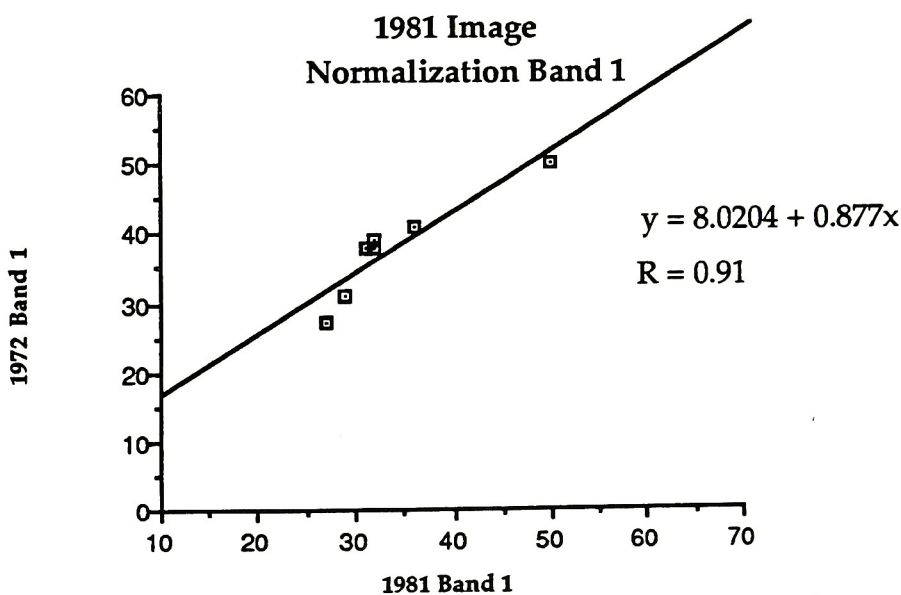


Figure 4.1d 1981 band 1 look-up table, linear transformation using invariant features

The linear equations used to transform the 1988 image are:

Band 1: $DC_{new} = 16.9175 + 0.6441 * DC_{old}$

Band 2: $DC_{new} = 19.7268 + 0.4788 * DC_{old}$

Band 3: $DC_{new} = 14.4458 + 0.6281 * DC_{old}$

Band 4: $DC_{new} = 8.0521 + 0.2584 * DC_{old}$

The digital count values in the original 1988 image are inserted into the above equations to obtain values that resembled those of the 1972 reference image. This is only valid if the objects used did not change over time. For pixels that have changed due either to environmental or manmade changes, the effect compensates for atmospheric differences between acquisition dates.

The following tables are the converted digital count values for each band in each image. Recall that the 1972 image is the reference image and remains unchanged. The 1981 image and the 1988 image are altered to match the spectral characteristics of the reference image.

Table 10a Output digital counts of invariant objects for band 4

BAND 4	1972	1981	1988
PIXEL 1	16	16	14
PIXEL 2	18	18	15
PIXEL 3	21	22	21
PIXEL 4	14	13	14
PIXEL 5	10	11	15
PIXEL 6	17	15	16
PIXEL 7	19	17	15

Table 10b Output digital counts of invariant objects for band 3

	1972	1981	1988
PIXEL 1	36	36	38
PIXEL 2	44	43	37
PIXEL 3	53	54	55
PIXEL 4	34	34	35
PIXEL 5	26	30	33
PIXEL 6	43	36	39
PIXEL 7	41	40	37

Table 10c Output digital counts of invariant objects for band 2

BAND 2	1972	1981	1988
PIXEL 1	40	34	35
PIXEL 2	41	39	36
PIXEL 3	53	54	54
PIXEL 4	37	35	36
PIXEL 5	25	32	33
PIXEL 6	37	34	34
PIXEL 7	31	34	33

Table 10d Output digital counts of invariant objects for band 1

BAND 1	1972	1981	1988
PIXEL 1	39	36	38
PIXEL 2	41	39	35
PIXEL 3	50	51	50
PIXEL 4	38	35	36
PIXEL 5	27	31	33
PIXEL 6	38	36	36
PIXEL 7	31	33	34

The RMS (root-mean-square) error was calculated to estimate the accuracy of the above method. The error must be calculated individually for each band.

$$RMS_{error} = \sqrt{\frac{\sum (x_1 - x_2)^2}{n - 1}} \quad (35)$$

where:

- x_1 is the data from the reference image
- x_2 is the data from the images to be matched
- n the number of pixels

Table 11 Digital count RMS error for each band

	Band 4	Band 3	Band 2	Band 1
RMS 1981	1.35	3.37	4.32	2.79
RMS 1988	3.02	4.81	4.64	3.87

The overall image statistics are found in Table 12 and the images are shown in Plates 9 and 10.

Table 12 Statistics from linear transformation of invariant features

	BAND 1		BAND 2		BAND 3		BAND 4	
	Mean	St. dev.	Mean	St. dev.	Mean	St. dev.	Mean	St. dev.
1972	20.79	8.40	14.52	13.03	38.69	17.31	23.10	16.57
1981	22.73	7.93	21.95	6.65	36.65	9.83	19.04	6.12
1988	27.25	2.89	25.64	3.20	40.78	8.66	19.87	4.14

The overall results from this method were very poor, although they may have been acceptable if just the invariant features are examined.

4.2.3 Linear Mapping Using Entire Image Statistics

The results of this method are shown in Plates 11 and 12. The transformations were calculated using the statistics given in Table 7 by the method described in Section 3.4.3. The equations used were:

1981 mapped to 1972 image:

$$DC_{band1} = .934DC_{old} + 4.45$$

$$DC_{band2} = 1.27DC_{old} - 3.41$$

$$DC_{band3} = 1.24DC_{old} - 7.90$$

$$DC_{band4} = 1.10DC_{old} - 19.65$$

1988 mapped to 1972 image:

$$DC_{band1} = 1.91DC_{old} - 11.53$$

$$DC_{band2} = 1.93DC_{old} - 11.30$$

$$DC_{band3} = 1.26DC_{old} - 15.21$$

$$DC_{band4} = 1.05DC_{old} - 26.88$$

The image statistics were again calculated to compare to the original 1972 image and can be found in Table 13.

Table 13 Linear Mapping Using Entire Image Statistics

	BAND 1		BAND 2		BAND 3		BAND 4	
	Mean	St. dev.	Mean	St. dev.	Mean	St. dev.	Mean	St. dev.
1972	20.79	8.40	14.52	13.03	38.69	17.31	23.10	16.57
1981	20.13	8.50	14.02	12.96	38.68	16.83	24.51	15.15
1988	20.30	8.31	14.35	12.93	39.08	16.15	25.33	14.18

The results of this method were better than those of method 3.4.2; however, they were poorer than those from the histogram specification technique. This method may be useful using other imagery.

The images all covered identical areas since they had previously been registered. However, it is possible that the results were altered due to the differences in snow cover between the image acquisition dates. The reference image had considerably more snow than the others and the 1981 image had even slightly less than the 1988 image.

4.3 *Classification*

Fifteen classes were identified using ISODATA; however, these were modified to include only 11 classes in the final result after examining them using COLORMOD (ERDAS). Plates 13-15 are the classified images. The classifier was able to distinguish between areas in the image that had been clearcut, those that had some degree of regrowth, and those that remain healthy and mature trees. However, it was not possible to determine whether these forest stands were old growth or the result of replanting. Three enlarged areas of clearcuts are shown in Plate 16 to illustrate the changes more effectively. These can be compared to Plate 17, which is a copy of an aerial photograph that dates to 1971. In Plate 17, Lake Keechelus is located in the upper right corner, and is easily identified in Plates 13-15 to give a point of reference. The smaller lake to the left is located in the upper right corner of the Plate 16-Section 2 image. Just to the left of this smaller lake in the aerial photograph is a "puzzle-piece" freshly clearcut area (*i.e.* not rectangular) with a bottle neck that extends to another clearcut. This extended area is split between fresh clearcut (upper) and brush or grassy regrowth (lower). These two areas are identifiable in the Plate 16- Section 2 image and are classified

accurately. When looking at this same area in the 1981 and 1988 images, it can be seen that there has been some regrowth. However, it must also be noted that areas directly adjacent to those have since been cut, thus changing the shape of the original clearcut. This can also be seen on the aerial photograph in the red cross-hatched areas which represent those that have been cut between 1971 and 1989 (Dave Leversee, Wilderness Society, Seattle, Washington).

In the Plate 16-Section 3 image, the line that cuts through the upper left corner is the power line that crosses the lower right corner of the aerial photograph. Again, areas can be identified that had been cut in the 1972 image, experienced regrowth and extended cutting in the later images. To the right of the power lines, in the 1972 images, there are several small, newly clearcut areas as well as some larger rectangular areas. It can be easily seen that these have become grass and brush in the later images and that a large amount of the surrounding forest has been cut since 1972. The area outlined in green was set aside for 1 pair of the endangered (and controversial) spotted owls. Although a quantitative analysis of clearcutting was not performed, it has devastated this area and continues to do so. Two photographic examples of clearcuts can be found in Plates 19A,B.



Plate 1: Distribution of control points

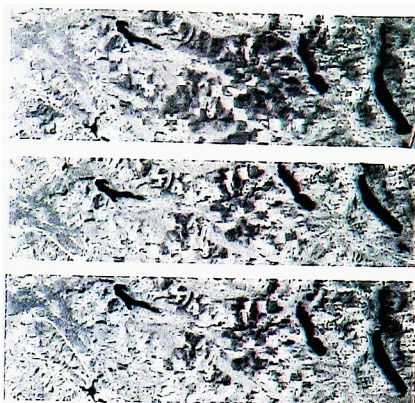


Plate 2:

1972 reference image

1988 before approximate registration

1988 scaled and rotated

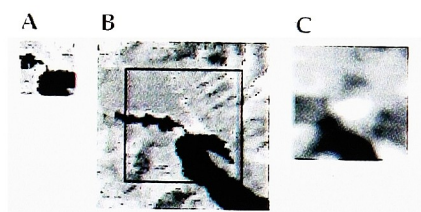


Plate 3: Registration example:

A) kernel area, B) search area, C) correlated result

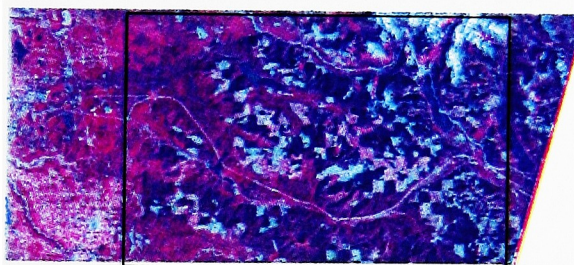


Plate 4:

1972 reference image

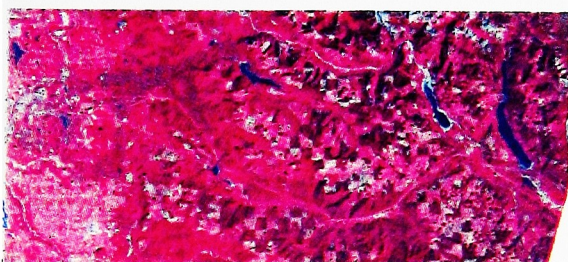


Plate 5:

1981 registered image

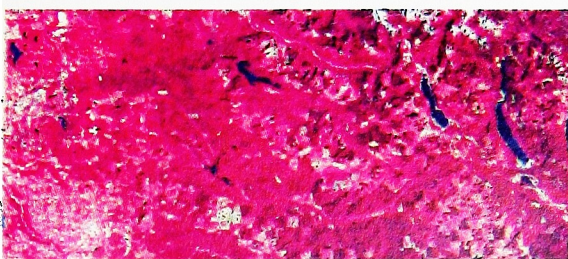


Plate 6:

1988 registered image

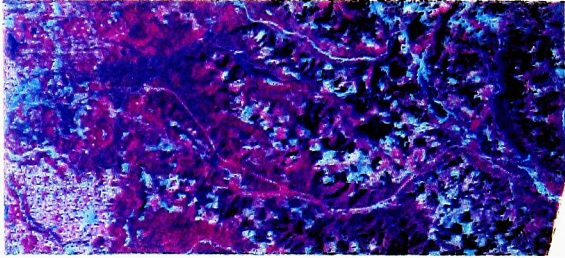


Plate 7: Histogram specification
(1981 image)

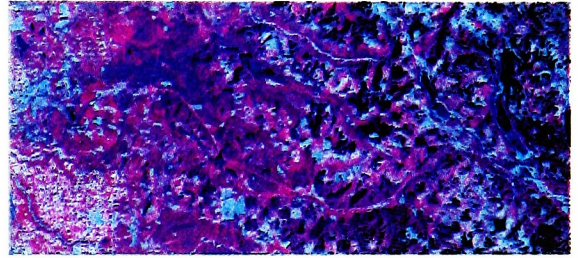


Plate 8: Histogram specification
(1988 image)

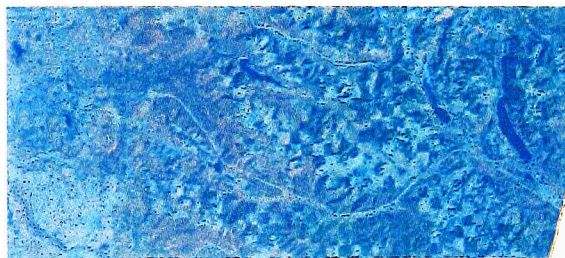


Plate 9: Linear mapping using invariant
features (1981 image)

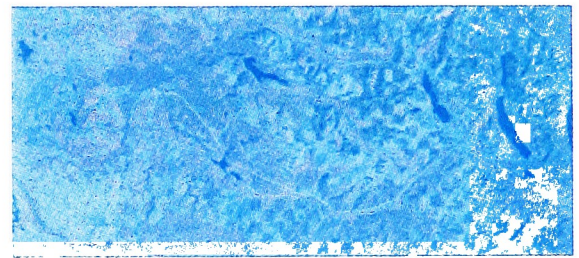


Plate 10: Linear mapping using invariant
features (1988 image)

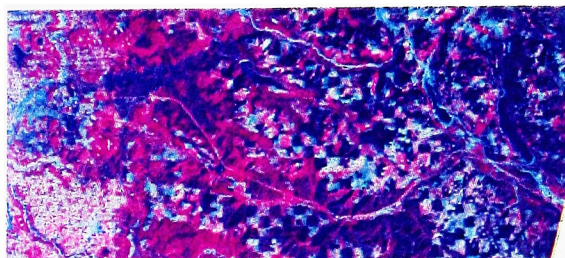


Plate 11: Linear histogram transformation
using entire image statistics
(1981 image)

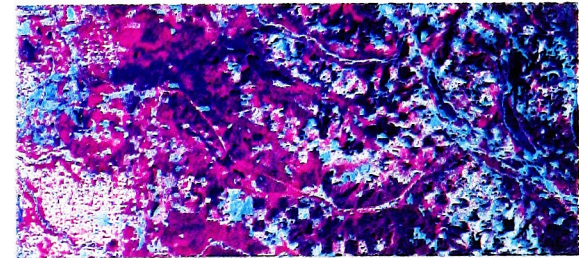


Plate 12: Linear histogram transformation
using entire image statistics
(1988 image)

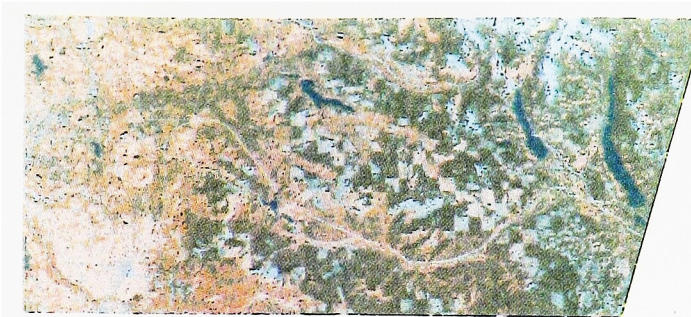


Plate 13:
1972 classified image



Plate 14:
1981 classified image



Plate 15:
1988 classified image

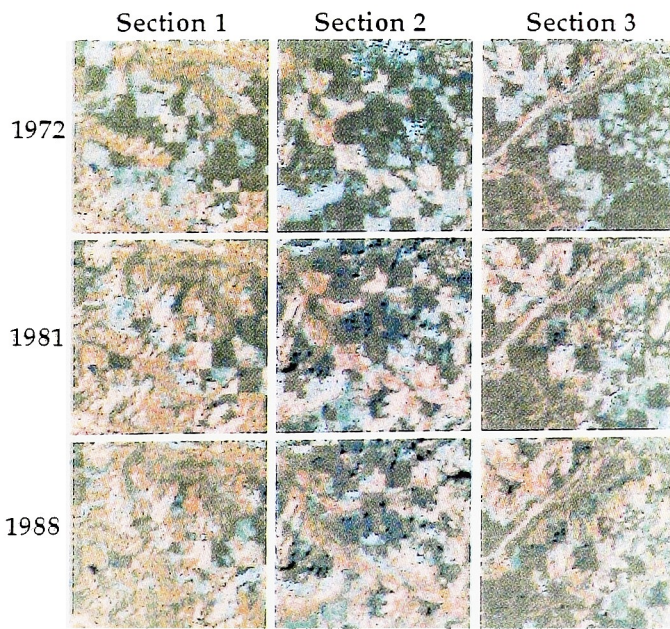


Plate 16:
Comparison of clearcut areas
in classified images



Legend for Plates 13-16

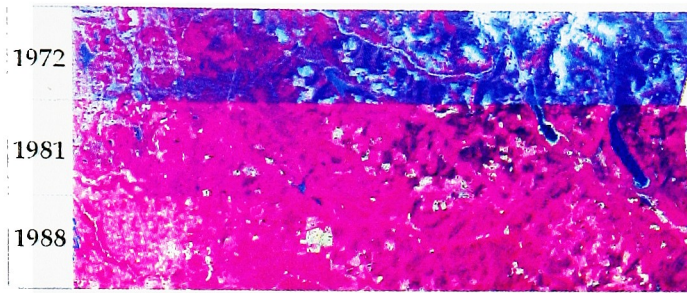


Plate 18 Section of each of the three registered images used to make composite image to show accuracy of registration



Plate 19A Clearcut (example 1)

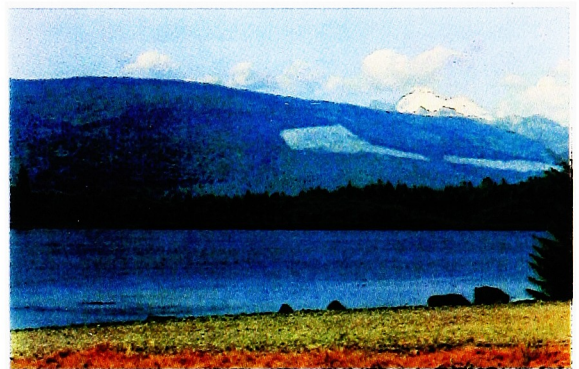


Plate 19B Clearcut (example 2)

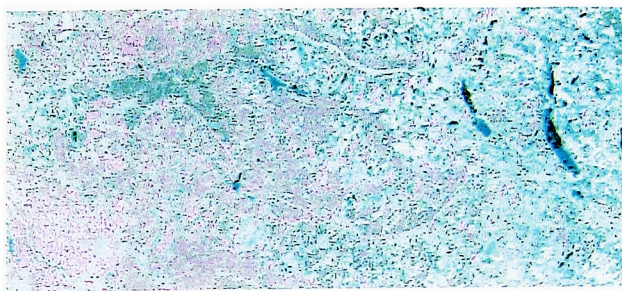


Plate 20 Classification using a maximum likelihood classifier

5.0 *Conclusions and Recommendations*

A technique has been developed to perform automated image-to-image rectification. The method automatically selects control points in an image to-be-registered using mathematical correlation of the scene in two windows. The kernel window is a section of an image that has been manually registered to a map image with the control point located at its center. Each control point used in the manual registration is extracted and stored in a kernel file along with data such as the map coordinates associated with each pixel. The search window is found by locating the control point approximately and extracting the area of the to-be-mapped image that surrounds it. The map coordinates and the pixel locations of three points must be known in the image to be registered. From this information, a transformation can be determined which transforms UTM map coordinates to pixel coordinates. The UTM coordinates associated with the control point in the center of the kernel window are used to locate the search area. The transformation is applied to these UTM coordinates and their approximate location is found. These two windows are correlated and the maximum value is the new control point.

Using LANDSAT MSS images, the process was shown to have a registration error of approximately 1-2 pixels. The poor resolution of MSS imagery as well, as the rural nature of the images may have contributed to the error. Ideal invariant control points were lacking in the imagery as they are more frequently found in urban areas (*e.g.* cross-roads, corners of buildings). The program was designed to be compatible with other types of imagery and is expected to perform better under more ideal conditions.

Image-to-image rectification is useful in the application of temporal change detection. The images were of a section of the Cascade Mountains in Washington State where the process of clearcutting has been employed

excessively. The changes in forest cover due to clearcutting from 1972 to 1988 were examined.

Change detection also involves the steps of image normalization and image classification. Ideally, these steps need to be automated as well. The techniques used were somewhat automated. However, since the focus of this study was automated registration, such methods were not explored thoroughly.

The image normalization process was performed using the technique of histogram specification. This method alters the histogram of one image to resemble that of another. In this case, each of the four MSS bands must be matched to the corresponding band in the reference image. To make the classification process more automated, the data were trained on a reference image and the signatures generated were applied to the entire image set. Training was done in an unsupervised manner which also minimized user interaction. Unsupervised techniques require that the user identify the spectral classes after the classification is complete. Spectral classes can be merged in this step as well.

A qualitative analysis of the normalization and classification stages was performed. Each was adequate for the purposes of detecting the changes in clearcut deforestation. The normalization was affected by differences in snow content between the three images.

One limitation of the technique is that several steps must be taken to pre-process the to-be-registered images prior to correlation. Although all images were from the same sensor, there are still noticeable differences in scale and rotation between them. The correlation is highly sensitive to these changes, so an approximate registration must be performed. In this study, this was

done by empirically determining the angle of rotation needed and the difference in scale. An alternative method would be to locate several control points approximately and use these to register the images. This could be done via the same method for creating the search areas and would result in a rough registration of the images. This method was not implemented but may be more efficient.

Another limitation is that the method was tested using only LANDSAT MSS images. It may be necessary to register two images from different sensors. This is a more difficult task as the resolution may be different if the images were acquired using different sensors. Therefore, a more significant difference in scale must be compensated. Also, the method was only used on an MSS image set and must be made compatible for other types of image sets, such as from SPOT or LANDSAT TM.

Finally, the steps of radiometric normalization and image classification must be made fully automated if an automated approach to the entire process of change detection is desirable.

6.0 *References*

- Ahern, F.J., R.J. Brown, J. Cihlar, R. Gauthier, J. Murphy, R.A. Neville, and P.M. Teillet, "Radiometric Correction of Visible and Infrared Remote Sensing Data at the Canada Centre for Remote Sensing", *International Journal of Remote Sensing*, Vol. 8, No. 9, February 1987, pp. 1349-1376.
- Barnea, D.I. and H.F. Silverman, "A Class of Algorithms for Fast Digital Image Registration", *IEEE Transactions on computers*, Vol. c-21, No. 2, February 1972, pp. 179-186.
- Barrow, H.G., J.M. Tenenbaum, R. Bolles, and H.C. Wolf, "Parametric Correspondence and Chamfer Matching: Two New Techniques for Image Matching", *Proceedings of the 5th Joint Conference on Artificial Intelligence*, Cambridge, Massachusetts, 1977, pp. 659-663.
- Bernstein, R., "Digital Image Processing of Earth Observation Sensor Data", *IBM Journal of Research and Development* (1976) pp. 40-57.
- Caselles, V., and M.J. López García, "An Alternative Simple Approach to Estimate Atmospheric Correction in Multitemporal studies", *International Journal of Remote Sensing*, Vol. 10, No. 6, 1989, pp. 1127-1134.
- Chen, L, and L. Lee, "Progressive Generation of Control Frameworks for Image Registration", *Photogrammetric Engineering and Remote Sensing*, Vol. 58, No. 9, September 1992, pp. 1321-1328.
- Davis, W.A., and S.K. Kenue, "Automatic Selection of Control Points for the Registration of Digital Images", *Proc. of the International Joint Conference on Pattern Recognition*, 4th, Kyoto, Japan, November 1978, pp. 936-938.
- Eckhardt, D.W., and J.P. Verdin, "Automated Update of an Irrigated Lands GIS Using SPOT HRV Imagery", *Photogrammetric Engineering and Remote Sensing*, Vol. 56, No. 11, November 1990, pp. 1515-1522.
- Ehlers, M., "Integration of Remote Sensing with Geographic Information Systems: A Necessary Evolution", *Photogrammetric Engineering and Remote Sensing*, Vol. 55, No. 11, November 1989, pp. 1619-1627

- Francis, J. "Pixel-by-Pixel Reduction of Atmospheric Haze Effects in Multispectral Digital Imagery of Water", *Master's Thesis*, Center for Imaging Science, Rochester Institute of Technology, Rochester, New York, 1989.
- Frew, J., "Registering Thematic Mapper Imagery to Digital Elevation Models," *Machine Processing of Remotely Sensed Data Proceedings*, 1984, pp. 191-196
- Gonzalez, R.C., and P. Woods, Digital Image Processing, Addison-Wesley, Reading, Massachusetts, 1992, pp. 180.
- Holm, M., "Towards Automatic Rectification of Satellite Images Using Feature Based Matching", *International Geoscience and Remote Sensing Symposium (ISGARSS)*, Vol. 4, 1991, pp. 2439-2442.
- Horn, B.K.P., and B.L. Bachman, "Using Synthetic Images to Register Real Images with Surface Models", *Comm. ACM*, Vol. 21, 1978, pp. 914-924.
- Kastak, W.D. and M.M. Crawford, "A Two-Stage Algorithm for Registration of Remotely Sensed Images", *International Geoscience and Remote Sensing Symposium (ISGARSS)*, Vol. 3, 1989, pp. 1283-1286.
- Lillesand, T.M. and R.W. Kiefer, Remote Sensing and Image Interpretation, John Wiley & Sons, Inc., New York, 1987.
- Little, J.J., "Automatic Rectification of Landsat Images using Features derived from Digital Terrain Models", Technical Report 80-10, University of British Columbia, 1980.
- Maitre, H., and Wu, Y., "Dynamic Programming Algorithm for Elastic Registration of Distorted Pictures based on Autoregressive Model", *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol. 37, No. 2, 1989, pp. 288-297.
- Morrison, P.H., D. Kloefer, D.A. Leversee, C. Socha, and D. Ferber, "Ancient Forests in the Pacific Northwest", The Wilderness Society, Washington, D.C., 1991.
- Piech, K.R., and J.R. Schott, "Atmospheric Corrections for Satellite Water Quality Studies", *Proceedings of SPIE*, Vol. 51, 1974, pp. 84-89.

- Pratt, W.K., "Correlation Techniques of Image Registration", *IEEE Transactions on Aerospace and Electronic Systems*, Vol. AES-10, No. 3, May 1974, pp. 353-358.
- Richards, J.A., Remote Sensing Digital Image Analysis, Springer-Verlag, 1986.
- Rignot, E.J.M., R. Kwok, J.C. Curlander, and S.S. Pang, "Automated Multisensor Registration: Requirements and Techniques", *Photogrammetric Engineering and Remote Sensing*, Vol. 57, No. 8, August 1991, pp. 1029-1038.
- Salvaggio, C. and J.R. Schott, "Automated Registration of SPOT Level 1B Imagery to U.S.G.S. Topographic Map Data", Technical Report RIT/DIRS 86/87-63-115, Center for Imaging Science, Rochester Institute of Technology, Rochester, New York, 1987.
- Salvaggio, C., "Automated Segmentation of Urban Features From Landsat Thematic Mapper Imagery for Use in Pseudoinvariant Feature Temporal Image Normalization", *Master's Thesis*, Center for Imaging Science, Rochester Institute of Technology, Rochester, New York, 1987.
- Scarpace, F.L., and K.W. Homquist, and L.T. Fisher, "Landsat Analysis of Lake Quality", *Photogrammetric Engineering and Remote Sensing*, Vol. 45, 1979, pp. 623-633.
- Schott, J.R., C. Salvaggio, and W.J. Volchok, "Radiometric Scene Normalization Using Pseudoinvariant Features", *Remote Sensing of the Environment*, 26:1-16, 1988, pp. 1-16.
- Schowengerdt, R.A., Techniques for Image Processing and Classification in Remote sensing, Academic Press, Inc., New York, 1983.
- Stockman, G, S. Kopstein, and S. Benett, "Matching Images to Models for Registration and Object Detection via Clustering", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-4, No. 3, May 1982, pp. 229-241.

Teillet, P.M., "Image Correction for Radiometric Effects in Remote Sensing", *International Journal of Remote Sensing*, Vol. 7, No. 12, May 1986, pp. 1637-1651.

Teillet, P.M., N.T. O'Neill, A. Kalinauskas, D. Sturgeon, and G. Fedosejevs "A Dynamic regression Algorithm for Incorporating Atmospheric Models into Image Correction Procedures", *Proceedings of IGARSS 1987 Symposium*, Ann Arbor, Mi., May 1987, pp. 913-918.

Ton, J., and A.K. Jain, "Registering Landsat Images by Point Matching", *IEEE Transactions on Geoscience and Remote Sensing*, Vol. 27, No. 5 September 1989, pp. 642-651.

Ventura, A.D., A. Rampini, and R. Schettini, "Image Registration by Recognition of Corresponding Structures", *IEEE Transactions on Geoscience and Remote Sensing*, Vol. 28, No. 3, 1990, pp. 305-314.

Volchok, W.J., "A Study of Multispectral Scene Normalization Using Psuedoinvariant Features Applied to Landsat TM Imagery", *Master's Thesis*, Center for Imaging Science, Rochester Institute of Technology, Rochester, New York, 1985.

Woodham, R.J. "Using Digital Terrain Data to Model Image Formation in Remote Sensing", *SPIE, Image Processing for Missile Guidance*, Vol. 238, 1980, pp. 75-78.

Yakota, T., and Y. Matsumoto, "Detection of Seasonal and Long-Term Changes in Land Cover from Multitemporal Landsat MSS Data", *Proceeding of IGARSS 1988 Symposium*, Edinburgh, Scotland, September 1988, pp. 215-216.

Appendix A-Program MAKE_KERNEL_UTM

Program creates kernels windows from a rectified image:

MAKE_KERNEL_UTM

User must specify the UTM coordinates of the control point; from this the kernel is created with the control point at kernel center.

These kernels are used in the program: MAKE_GCP_FILE.C

NOTE: If the user wishes to create kernels using the pixel locations, use the program MAKE_KERNEL

```

#include <stdlib.h>

#include <stdio.h>
#define MX_BANDS 256
#include "image.h"

/*****
Program      Make_kernel_utm.c

description   This program extracts sections from an image that are to
               be used as kernel images in make_gcp_file.c

               These sections contain the control points from manually
               referenced image

               The actual control point is located in the exact center
of
               the kernel. Therefore, to make it simple, the kernels
must
               have an odd number of rows and cols so there is an exact
               center.

               The user is prompted for:
               the name of the input file
               the name of the file in which to store the kernel
               the map coordinates of the center of the control
points
               --these are col_utm_center (x) and
row_utm_center (y)
               the number of pixels in a row to extract (cols, or
               number of elements)
               the number of rows to extract (rows)
*****/

main()
{
    long row_ctr, col_ctr;
    float row_utm_ctr, col_utm_ctr;
    long row_start, col_start;
    long number_of_elements;
    long num_rows;
    long band;
    long row,col;
    long displace_utm_y, displace_utm_x;
    long row_ctr_for_getpixel, col_ctr_for_getpixel;
    long temp_xstart, temp_ystart;
    char in_file[80];
    char out_file[80];

    struct PICTURE_OPTIONS picops_in;
    struct PICTURE_OPTIONS picops_out;
    struct ERDAS_HEADER header_in;
    struct ERDAS_HEADER header_out;

    unsigned char *image_data_location;
    unsigned char band_mask[MX_BANDS];

/*  GETTING INPUT FILENAME, OUTPUT FILE NAME, # OF ELEMENTS TO GRAB */

    printf("\n Enter input filename (*.lan file) >");
    scanf( "%s", in_file );

/*  OPENS THE INPUT FILE */
    if ( ( open_erdas_file(&header_in, &picops_in,in_file, "r",
0)) == (int) NULL ) {

```

```

    exit( 1 );
}

printf ("\n Enter output filename (*.lan file) >");
scanf( "%s", out_file );

printf ("\n Enter center col utm coordinate of kernel (x value)>");
scanf ("%f", &col_utm_ctr);

printf ("\n Enter center row utm coordinate of kernel (y value)>");
scanf ("%f", &row_utm_ctr);

do {
    printf ("\n Enter # of pixels in a row to grab (odd #) >");
    scanf ("%ld", &number_of_elements);
} while (!(number_of_elements%2));

do {
    printf ("\n Enter number of rows to be grabbed (odd #) >");
    scanf ("%ld", &num_rows);
} while (!(num_rows%2));

row_ctr = header_in.ystart +
    ((-row_utm_ctr + header_in.ymap)/header_in.ycell);
col_ctr = header_in.xstart +
    ((col_utm_ctr - header_in.xmap)/header_in.xcell);

/* GETPIXEL NEEDS POINTS TO BE REFERENCED RELATIVE TO 0,0 */
row_ctr_for_getpixel = row_ctr - header_in.ystart;
col_ctr_for_getpixel = col_ctr - header_in.xstart;

/* GETTING ROOM TO STORE DATA */
image_data_location = (unsigned char *) calloc
    ((size_t)(num_rows*number_of_elements*
        picops_in.number_of_bands), (size_t)
        sizeof(unsigned char));

/* FILLING BAND MASK */
for (band = 0; band < MX_BANDS; band++) band_mask[band] = 0;
for (band = 0; band < picops_in.number_of_bands; band++)
band_mask[band] = 1;

/* DETERMINING UPPER LEFT CORNER, STARTING POSITION TO READ IN KERNELS*/
row_start = (row_ctr_for_getpixel - (num_rows/2));
col_start = (col_ctr_for_getpixel - (number_of_elements/2));
col=col_start;

/* OPENING FILE TO PUT KERNEL */
header_out = header_in;
header_out.xstart = col_ctr - number_of_elements/2;
header_out.ystart = row_ctr - num_rows/2;
header_out.irows = num_rows;
header_out.icols = number_of_elements;
header_out.ymap = header_in.ymap - ((header_out.ystart -
    header_in.ystart)*header_in.ycell);
header_out.xmap = header_in.xmap + ((header_out.xstart -
    header_in.xstart)*header_in.xcell);
open_erdas_file(&header_out, &picops_out, out_file, "w", -1);

/* CREATING KERNEL DATA */
for (row = row_start; row < row_start+num_rows; row++) {
    getpixel (picops_in, row, col, band_mask,
number_of_elements,
        image_data_location);
    putpixel (picops_out, row-row_start, col-col_start, band_mask,
        number_of_elements, image_data_location);
}

```

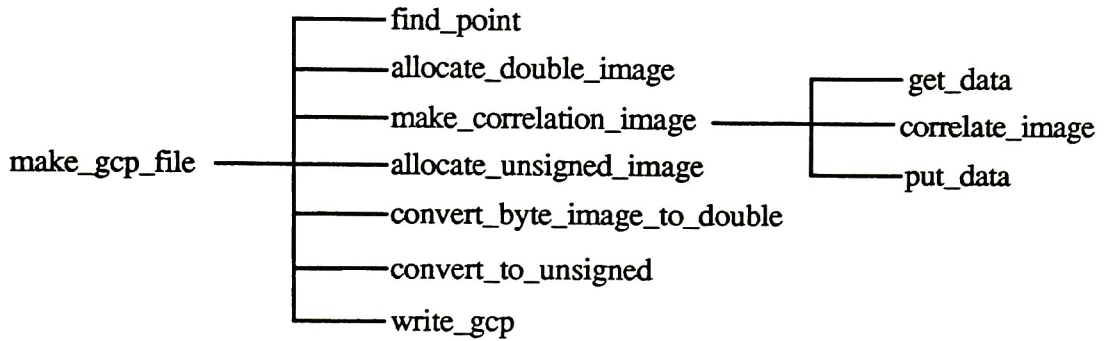
```

}
}

```

Appendix B-Program MAKE_GCP_FILE

Program to run image-to-image rectification:



To use program "make_gcp_file.c"

Kernel windows (images) must be created and supplied to make_gcp_file (created using "make_kernel_utm.c" (Appendix A))

"make_gcp_file.c" expects a filename that contains a list of kernel images---i.e. "kernels_names.nam"---be sure to include full path.

k_thesis.h is on the last page, just after put_data.c

This symbolizes the start of a new function

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "image.h"
#include "k_thesis.h"
```

```
/*****
```

```
MAIN:          make_gcp_file.c
```

description: This program reads in a kernel image from a namefile and creates a search area in an image to be registered. It correlates these two images and returns the maximum value. The kernel image contains a control point extracted from a registered image. It has map coordinates associated with it. The max. value is the newly found control point in the registered image. Both the map coordinates associated with the control point and the location of the new control point are written to a ground control point file (gcp) to be used in image registration using ERDAS software.

functions:

find_point
uses the map coordinates associated with the center point of the kernel image (ie. the control point to be found automatically) and locates that point approximately in the to be registered images. This is done so that a search area can be created.

make_correlation_image
this takes the search area that was created based on find_point and the kernel image and correlates them. The correlation value at each location is stored in a new image called "result".

convert_byte_image_to_double
takes an image and converts data to double

convert_to_unsigned
converts a double image back to unsigned.

allocate_double_image
allocates appropriate amount of space for an image with data that is double

allocate_unsigned_image
allocates space for an image of data type unsigned

write_gcp
writes new ground control points to gcp file with associated UTM (Universe Transverse Mercator, meters) coordinates. This is used to register the images with ERDAS software.

getpixel and putpixel are also used (Carl Salvaggio) and are included in image.h

k_thesis.h
contains structures created for Kaleen's thesis

To use this program:

Must enter:

- the name of the image to be registered
- the name of the gcp file to write data to
- the file name that contains the list of kernels (created using make_kernel)
- the name of the output search window (enable user to view a search area, if there is more than one kernel name in the namefile, the final output search area will be from the last kernel as this is continually written over as the program runs).
- the name of the output correlation image (also gets written over)
- REFERENCE POINTS: these are used to find the approximate locations of the control points. They are three corners of an image that was manually registered. Both the pixel locations and the UTM coordinates are required and are obtained from the ephemeris data associated with each Landsat image.

USAGE: (should be entered in one line, each separated by one space)

```
make_gcp_file  image_to_register  gcp_filename  kernel_filename
output_search_window  correlation_image  x_upper_lft  y_upper_lft
UTM_x_upper_lft  UTM_y_upper_lft  x_upper_rgt  y_upper_rgt  UTM_x_upper_rgt
UTM_y_upper_rgt  x_lower_rgt  y_lower_rgt  UTM_x_lower_rgt  UTM_y_lower_rgt
```

author: Kaleen Moriarty

*****/

```
struct FILECOORD find_point ( struct ERDAS_HEADER header_kernel,
                             struct REFERENCE_POINTS ref_pts);

struct IMAGE_DOUBLE make_correlation_image( struct IMAGE_DOUBLE kernel,
                                           struct IMAGE_DOUBLE search_window );

struct IMAGE_DOUBLE convert_byte_image_to_double ( unsigned char *byte_image,
                                                  long number_of_rows,
                                                  long number_of_columns );

struct IMAGE_UNSIGNED convert_to_unsigned ( double *double_image,
                                           long number_of_rows,
                                           long number_of_columns );

struct IMAGE_DOUBLE allocate_double_image( long num_rows, long num_cols );

struct IMAGE_UNSIGNED allocate_unsigned_image( long num_rows, long num_cols );

void write_gcp ( struct IMAGE_DOUBLE result_image,
                struct ERDAS_HEADER header_result,
                struct ERDAS_HEADER header_kernel,
                FILE *fp_gcpfile );

void print_usage(void);
```

```

main(int argc, char *argv[])
{

struct    ERDAS_HEADER header_image_to_reg;
struct    ERDAS_HEADER header_kernel;
struct    ERDAS_HEADER header_image_out;
struct    ERDAS_HEADER header_kernel_out;
struct    ERDAS_HEADER header_result;
struct    PICTURE_OPTIONS picops_image_to_reg;
struct    PICTURE_OPTIONS picops_kernel;
struct    PICTURE_OPTIONS picops_image_out;
struct    PICTURE_OPTIONS picops_kernel_out;
struct    PICTURE_OPTIONS picops_result;
struct    FILECOORD pts_found;
struct    REFERENCE_POINTS ref_pts;
struct    IMAGE_DOUBLE kernel_converted;
struct    IMAGE_DOUBLE search_area_converted;
struct    IMAGE_DOUBLE result_image;
struct    IMAGE_UNSIGNED result_unsigned_image;

long      row,col;
long      SEARCH_WINDOW_SIZE;
long      search_area_num_rows,search_area_num_of_elements;
long      row_ctr_image_getpixel, col_ctr_image_getpixel;
long      row_start_image, col_start_image;
long      band;
long      pixel;
long      num_control_pts;
char      k_filename[80]; /*      filename that contains names of kernel images,
                           reads in each kernel one at a time      */

char      individual_k_filename[80];

char      gcp_filename[80]; /*      filename that contains gcp coordinates      */
char      image_file[80]; /*      filename containing image to be registered */
char      out_kernel[80]; /*      temporary file to check data input      */
char      out_search[80]; /*      temporary check of search window data input*/
char      view_result[80]; /*      temporary check of correlation result      */
int       status;

FILE *fp_kernelnames;
FILE *fp_gcpfile;

unsigned char *kernel_for_getpixel;
unsigned char *search_area_for_getpixel;
unsigned char band_mask_kernel[MX_BANDS];
unsigned char band_mask_out_kernel[MX_BANDS];
unsigned char band_mask_image[MX_BANDS];
unsigned char band_mask_out_image[MX_BANDS];

    if (argc != 18 ){
        print_usage();
    }

strcpy (image_file, argv[1]);
strcpy (gcp_filename, argv[2]);
strcpy (k_filename, argv[3]);
strcpy (out_search, argv[4]);
strcpy (view_result,argv[5]);
sscanf (argv[6], "%lf", &ref_pts.x_ref);
sscanf (argv[7], "%lf", &ref_pts.y_ref);
sscanf (argv[8], "%lf", &ref_pts.utm_x_ref);

```

```

sscanf (argv[9], "%lf", &ref_pts.utm_y_ref);
sscanf (argv[10], "%lf", &ref_pts.x_one);
sscanf (argv[11], "%lf", &ref_pts.y_one);
sscanf (argv[12], "%lf", &ref_pts.utm_x_one);
sscanf (argv[13], "%lf", &ref_pts.utm_y_one);
sscanf (argv[14], "%lf", &ref_pts.x_two);
sscanf (argv[15], "%lf", &ref_pts.y_two);
sscanf (argv[16], "%lf", &ref_pts.utm_x_two);
sscanf (argv[17], "%lf", &ref_pts.utm_y_two);

if ((fp_gcpfile = fopen (gcp_filename, "w"))==NULL){
    printf ( "ERROR OPENING GCP FILE ");
    exit (0);
}

/***** OPENING IMAGE TO BE REGISTERED *****/

if ( ( open_erdas_file(&header_image_to_reg,&picops_image_to_reg,image_file,"r",
    0)) == NULL ) {
    printf( "ERROR OPENING FILE: IMAGE-TO-REGISTER " );
    exit( 0 );
}

/**** OPENS FILE THAT CONTAINS KERNEL NAMES AND READS IN EACH KERNEL****/

fp_kernelnames = fopen (k_filename, "r");
if ( fp_kernelnames == NULL ) {
    fprintf( stderr, "make_gcp_file: Error opening kernel name file\n" );
    exit( 1 );
}
num_control_pts = 0;

while (fscanf(fp_kernelnames,"%s", individual_k_filename)!=EOF) {
    num_control_pts++;
}
fclose( fp_kernelnames );
fprintf( fp_gcpfile, "%7d\n", num_control_pts );

fp_kernelnames = fopen (k_filename, "r");

if ( fp_kernelnames == NULL ) {
    fprintf( stderr, "make_gcp_file: Error opening kernel name file\n" );
    exit( 1 );
}

/**** OPENS FIRST KERNEL, PROCESSES, THEN SECOND ETC ****/
while (fscanf(fp_kernelnames,"%s", individual_k_filename)!=EOF) {
    status =open_erdas_file ( &header_kernel,&picops_kernel,
        individual_k_filename,"r",0);

/*****
FILLING BAND MASK:
Each Landsat MSS image has 4 bands for BAND MASK, these are numbered 0,1,2,3
The 4th band was used to correlate the images because it had high contrast
To use this band, band_mask [3] is set equal to 1
*****/
for (band = 0; band< MX_BANDS; band++) {
    band_mask_kernel[band] = 0;
    band_mask_out_kernel[band] = 0;
}
band_mask_kernel[3]=1;
band_mask_out_kernel[0]=1;

```

```

for (band = 0; band< MX_BANDS; band++) {
    band_mask_image[band] = 0;
    band_mask_out_image[band] = 0;
}
band_mask_image[3]=1;
band_mask_out_image[0]=1;

/**      DEFINE VARIABLES FOR GETPIXEL,
****this section allows user to write kernel data out to make sure
that it was read in properly. Must remove comment marks to do this in
the future as it is not necessary normally *****/

header_kernel_out = header_kernel;
header_kernel_out.nbands = 1;
picops_kernel_out = picops_kernel;
picops_kernel_out.number_of_bands = 1;

open_erdas_file( &header_kernel_out,&picops_kernel_out,out_kernel, "w",-1);
***/

/**      ALLOCATING SPACE FOR UNSIGNED CHAR KERNEL DATA ****/

kernel_for_getpixel = (unsigned char *)
    calloc ((size_t)
        (picops_kernel.number_of_rows*
        picops_kernel.number_of_columns),
        (size_t)sizeof(unsigned char));

kernel_converted = allocate_double_image
    ( picops_kernel.number_of_rows,
      picops_kernel.number_of_columns );

pixel = 0;
for (row = header_kernel.ystart;
    row<header_kernel.ystart+picops_kernel.number_of_rows;
    row++){

    for(col = header_kernel.xstart;
        col<header_kernel.xstart+picops_kernel.number_of_columns;
        col++){

        status = getpixel( picops_kernel,
            row-header_kernel.ystart,
            col-header_kernel.xstart,
            band_mask_kernel,
            1,
            (kernel_for_getpixel+pixel) );

/***** FOR CHECKING KERNEL DATA, REMOVE TO INCLUDE ***

        status = putpixel( picops_kernel_out,
            row-header_kernel.ystart,
            col-header_kernel.xstart,
            band_mask_out_kernel,
            1,
            (kernel_for_getpixel+pixel) ); ****/

        pixel = pixel + 1;
    }
}

kernel_converted = convert_byte_image_to_double
    ( kernel_for_getpixel,
      picops_kernel.number_of_rows,
      picops_kernel.number_of_columns );

```

```

pts_found = find_point(header_kernel, ref_pts);
/**** GETS SEARCH WINDOW DATA****/
SEARCH_WINDOW_SIZE = picops_kernel.number_of_rows * 3;
header_image_out = header_image_to_reg;
header_image_out.xstart = pts_found.x_out - SEARCH_WINDOW_SIZE/2;
header_image_out.ystart = pts_found.y_out - SEARCH_WINDOW_SIZE/2;
header_image_out.irows = SEARCH_WINDOW_SIZE;
header_image_out.icols = SEARCH_WINDOW_SIZE;
header_image_out.nbands = 1;
header_image_out.ymap = header_image_to_reg.ymap -
    ((header_image_out.ystart -
    header_image_to_reg.ystart)
    *header_image_to_reg.ycell);

header_image_out.xmap = header_image_to_reg.xmap +
    ((header_image_out.xstart -
    header_image_to_reg.xstart)
    *header_image_to_reg.xcell);

picops_image_out = picops_image_to_reg;
picops_image_out.number_of_bands = 1;
picops_image_out.number_of_rows = SEARCH_WINDOW_SIZE;
picops_image_out.number_of_columns = SEARCH_WINDOW_SIZE;

open_erdas_file(&header_image_out, &picops_image_out, out_search, "w", -1);

row_ctr_image_getpixel = pts_found.y_out - header_image_to_reg.ystart;
col_ctr_image_getpixel = pts_found.x_out - header_image_to_reg.xstart;

search_area_for_getpixel = ( unsigned char *) calloc
    (( size_t)
    ( SEARCH_WINDOW_SIZE*SEARCH_WINDOW_SIZE ),
    ( size_t)sizeof(unsigned char) );

search_area_converted = allocate_double_image ( SEARCH_WINDOW_SIZE,
    SEARCH_WINDOW_SIZE );

row_start_image = (row_ctr_image_getpixel - (SEARCH_WINDOW_SIZE/2));
col_start_image = (col_ctr_image_getpixel - (SEARCH_WINDOW_SIZE/2));

pixel = 0;
for (row = row_start_image;
    row < row_start_image+SEARCH_WINDOW_SIZE;
    row++){
    for(col = col_start_image;
        col < col_start_image+SEARCH_WINDOW_SIZE;
        col++){

        getpixel( picops_image_to_reg,
            row,
            col,
            band_mask_image,
            1,
            ( search_area_for_getpixel+pixel) );

        putpixel( picops_image_out,
            row-row_start_image,
            col-col_start_image,
            band_mask_out_image,
            1,
            ( search_area_for_getpixel+pixel) );

        pixel = pixel + 1;
    }
}

```

```

    }
}
search_area_converted = convert_byte_image_to_double
    ( search_area_for_getpixel,
      SEARCH_WINDOW_SIZE,
      SEARCH_WINDOW_SIZE );

header_result = header_image_out;
picops_result = picops_image_out;

open_erdas_file(&header_result,&picops_result,view_result,"w",-1);

result_image = make_correlation_image ( kernel_converted,
                                       search_area_converted );

write_gcp( result_image,header_result,header_kernel,fp_gcpfile );

result_unsigned_image = allocate_unsigned_image
    (picops_result.number_of_rows,
     picops_result.number_of_columns );

result_unsigned_image = convert_to_unsigned
    ( result_image.data,
      picops_result.number_of_columns,
      picops_result.number_of_rows );

status = putpixel( picops_result,
                  0,
                  0,
                  band_mask_out_image,
                  ( picops_result.number_of_columns*
                    picops_result.number_of_rows),
                  result_unsigned_image.data );

cfree( kernel_for_getpixel );
cfree( search_area_for_getpixel );
cfree( kernel_converted );
cfree( search_area_converted );

fclose( picops_kernel.file_pointer );
/**/  fclose( picops_kernel_out.file_pointer );/***/
fclose( picops_image_out.file_pointer );
fclose( picops_result.file_pointer );

}
fclose( fp_gcpfile );
}

void print_usage(void)
{
printf("\n");
printf("USAGE: make_gcp_file image_to_register gcp_filename
kernel_filename\output_search_window correlation_image x_upper_lft y_upper_lft
UTM_x_upper_lft\nUTM_y_upper_lft x_upper_rgt y_upper_rgt UTM_x_upper_rgt
UTM_y_upper_rgt\nx_lower_rgt y_lower_rgt UTM_x_lower_rgt UTM_y_lower_rgt  ");

printf("\n");
exit(0);
}

```

```

#include <stdio.h>
#include <math.h>
#include "k_thesis.h"
#include "image.h"

/*****
function    find_point.c

description    Using three points of reference (three corner points in the
                image to be registered), an estimate of control point location
                is made. The UTM coordinates and the pixel locations of those
                three points must be known. Be sure to double check the
                corner points. If an approximate registration has already
                been performed, these are no longer the same as those supplied
                with the original scene.

author        Kaleen Moriarty
*****/

struct FILECOORD find_point( struct ERDAS_HEADER header_kernel,
                             struct REFERENCE_POINTS ref_pts)
{
    long double    alpha,beta,gamma,delta;
    long double    utm_x_desired, utm_y_desired;
    long double    xmap_center,ymap_center;

    struct FILECOORD pts_found;

    /***
        By subtracting the .ref value from the other 2 points, the calculation
        is made simpler as it is referenced to the origin
    ***/

    ref_pts.x_one = ref_pts.x_one - ref_pts.x_ref;
    ref_pts.y_one = ref_pts.y_one - ref_pts.y_ref;
    ref_pts.utm_x_one = ref_pts.utm_x_one - ref_pts.utm_x_ref;
    ref_pts.utm_y_one = ref_pts.utm_y_one - ref_pts.utm_y_ref;
    ref_pts.x_two = ref_pts.x_two - ref_pts.x_ref;
    ref_pts.y_two = ref_pts.y_two - ref_pts.y_ref;
    ref_pts.utm_x_two = ref_pts.utm_x_two - ref_pts.utm_x_ref;
    ref_pts.utm_y_two = ref_pts.utm_y_two - ref_pts.utm_y_ref;

    /*** xmap_center and ymap_center are the map coordinates of the
        kernel center (the control point to find in new image).

        utm_x_desired, utm_y_desired is that point translated relative to
        the origin.
    ***/

    xmap_center = header_kernel.xmap + (header_kernel.xcell *
                                         ((header_kernel.icols -1)/2));
    ymap_center = header_kernel.ymap - (header_kernel.ycell *
                                         ((header_kernel.irows -1)/2));
    utm_x_desired = xmap_center - ref_pts.utm_x_ref;
    utm_y_desired = ymap_center - ref_pts.utm_y_ref;

    alpha = (1/(ref_pts.x_one*ref_pts.y_two - ref_pts.x_two*ref_pts.y_one))*
            (ref_pts.utm_x_one*ref_pts.y_two + ref_pts.utm_x_two*(-ref_pts.y_one));

```

```

beta = (1/(ref_pts.x_one*ref_pts.y_two - ref_pts.x_two*ref_pts.y_one))*
        (ref_pts.utm_x_one*(-ref_pts.x_two) + ref_pts.utm_x_two*(ref_pts.x_one));

gamma = (1/(ref_pts.x_one*ref_pts.y_two - ref_pts.x_two*ref_pts.y_one))*
        (ref_pts.utm_y_one*ref_pts.y_two + ref_pts.utm_y_two*(-ref_pts.y_one));

delta = (1/(ref_pts.x_one*ref_pts.y_two - ref_pts.x_two*ref_pts.y_one))*
        (ref_pts.utm_y_one*(-ref_pts.x_two) + ref_pts.utm_y_two*(ref_pts.x_one));

pts_found.x_out = ((1/(alpha*delta - beta*gamma))*
        ((delta*utm_x_desired) + (-beta*utm_y_desired))) +
        ref_pts.x_ref;

pts_found.y_out = ((1/(alpha*delta - beta*gamma))*
        ((-gamma*utm_x_desired) + (alpha*utm_y_desired))) +
        ref_pts.y_ref;

printf ("\n The x pixel value found is %lf", pts_found.x_out);

printf ("\n The y pixel value found is %lf", pts_found.y_out);
/****
printf ("\n alpha %lf", alpha);
printf ("\n beta %lf", beta);
printf ("\n gamma %lf", gamma);
printf ("\n delta %lf", delta);
*****/

return (pts_found);

}

```

```

#include <stdlib.h>
#include "k_thesis.h"

/*****
function    allocate_double_image

description  this simple allocates space of data type double in which to
              store an image

*****/
struct IMAGE_DOUBLE allocate_double_image(long row,long col)
{
    struct IMAGE_DOUBLE image;
    image.num_of_cols = col;
    image.num_of_rows = row;

    image.data = (double *) calloc (row * col, sizeof(double));
    return(image);
}

```

```

#include "k_thesis.h"
#include "image.h"
/*****
function    make_correlation_image.c

uses        allocate_double_image.c
            correlate_image.c
            put_data.c

description  this routine makes an image that is the result of correlating
            the kernel image and the search window.
            The kernel is translated across the search area. At each
            location, the correlation is performed (correlate_image.c)
            and the value returned and stored in that location in a
            new image.
*****/
extern struct IMAGE_DOUBLE allocate_double_image(long row, long col);
extern double correlate_image ( struct IMAGE_DOUBLE kernel, struct IMAGE_DOUBLE
                               search_window, long row, long col);
extern void put_data ( double value, long row_image_index, long col_image_index,
struct IMAGE_DOUBLE result_image);
struct IMAGE_DOUBLE make_correlation_image( struct IMAGE_DOUBLE kernel,
                                           struct IMAGE_DOUBLE search_window,
                                           struct PICTURE_OPTIONS
                                           picops_kernel_out, struct
                                           PICTURE_OPTIONS picops_image_out)
{
struct IMAGE_DOUBLE result_image= allocate_double_image ( search_window.num_of_rows,
                                                         search_window.num_of_cols);

    long row_image_start;
    long col_image_start;
    long row_image_end;
    long col_image_end;
    double value;
    long row_image_index, col_image_index;
    long row_kernel_index, col_kernel_index;

/* result_image = allocate_double_image( search_window.num_of_rows,
                                         search_window.num_of_cols ); */
/**** THE CORRELATION IS ONLY PERFORMED WHERE THE KERNEL FITS ENTIRELY
      WITHIN THE SEARCH AREA, THE FOLLOWING DETERMINES THE LIMITS
****/
    row_image_start = kernel.num_of_rows/2;
    col_image_start = kernel.num_of_cols/2;
    row_image_end = search_window.num_of_rows - row_image_start;
    col_image_end = search_window.num_of_cols - col_image_start;
    for ( row_image_index = row_image_start;
          row_image_index < row_image_end;
          row_image_index++){
        for ( col_image_index = col_image_start;
              col_image_index < col_image_end;
              col_image_index++){
            value = correlate_image ( kernel, search_window, row_image_index,
                                     col_image_index);

            put_data (value, row_image_index, col_image_index, result_image);
        }
    }
    return (result_image);
}

```

```

#include <stdlib.h>
#include "k_thesis.h"

/*****
function    allocate_unsigned_image.c

description    allocates space for an image of data type unsigned char
*****/

struct IMAGE_UNSIGNED allocate_unsigned_image(long row, long col)
{
    struct IMAGE_UNSIGNED image;
    image.num_of_cols = col;
    image.num_of_rows = row;

    image.data = (unsigned char *) calloc (row * col, sizeof(unsigned char));
    return(image);
}

```

```

#include "k_thesis.h"

/*****
function    convert_byte_image_to_double

description    converts image (unsigned char) to data type double
*****/

struct IMAGE_DOUBLE allocate_double_image(long row, long col);

struct IMAGE_DOUBLE convert_byte_image_to_double ( unsigned char *byte_image,
                                                    long number_of_columns,
                                                    long number_of_rows )
{
    int index;

    struct IMAGE_DOUBLE double_image= allocate_double_image
        ( number_of_rows,
          number_of_columns );

    for (index = 0; index<number_of_rows*number_of_columns;index++){

        double_image.data[index] =(double) byte_image[index];
    }
    return(double_image);
}

```

```

#include "k_thesis.h"

/*****
function      convert_to_unsigned

description    converts image (double) to unsigned char
*****/

struct IMAGE_UNSIGNED allocate_unsigned_image(long row, long col);

struct IMAGE_UNSIGNED convert_to_unsigned ( double *double_image,
                                           long number_of_columns,
                                           long number_of_rows )
{
    int index;

    struct IMAGE_UNSIGNED unsigned_image= allocate_unsigned_image
        ( number_of_rows,
          number_of_columns );

    for (index = 0; index<number_of_rows*number_of_columns;index++){
        double_image[index] = double_image[index]*255;

        unsigned_image.data[index] =(unsigned char) double_image[index];

        if (unsigned_image.data[index] > 255) unsigned_image.data[index]=255;
        if (unsigned_image.data[index] < 0) unsigned_image.data[index] = 0;
    }
    return(unsigned_image);
}

```

```

#include <math.h>
#include "image.h"
#include "k_thesis.h"

/*****
function    write_gcp.c

description    this program steps through the resultant correlation image
                and finds the location of the maximum value. This is then
                written to a .gcp file in the format necessary to be
                used for the ERDAS routine NRECTIFY
*****/

write_gcp ( struct IMAGE_DOUBLE result_image,
            struct ERDAS_HEADER header_result,
            struct ERDAS_HEADER header_kernel,
            FILE *fp_gcpfile )
{
    double max_value;
    double control_pt_x, control_pt_y;
    double utm_val_x, utm_val_y;
    long row,col;

    max_value = -HUGE_VAL;

    for ( row = 0;
          row < result_image.num_of_rows;
          row++ ){
        for ( col = 0;
              col < result_image.num_of_cols;
              col++ ){

            if (( result_image.data[row*result_image.num_of_cols + col]) >
                ( max_value)){
                max_value=result_image.data[row*result_image.num_of_cols+col];
                control_pt_x = header_result.xstart + col;
                control_pt_y = header_result.ystart + row;
            }
        }
    }

    /****UTM value associated with control point is obtained from the kernel data****/
    utm_val_x = header_kernel.xmap +
                (header_kernel.icols/2)*header_kernel.xcell;

    utm_val_y = header_kernel.ymap -
                (header_kernel.irows/2)*header_kernel.ycell;

    printf ("\n Control_pt_x %lf", control_pt_x);
    printf ("\n Control_pt_y %lf", control_pt_y);

    /****
    printf ("\n max_value %lf", max_value);
    printf ("\n Utm x value %lf", utm_val_x);
    printf ("\n Utm y value %lf", utm_val_y);
*****/
    fprintf ( fp_gcpfile," %15.4lf %14.4lf %14.4lf %14.4lf\n",
              utm_val_x,utm_val_y, control_pt_x,control_pt_y);
}

```

```

#include "k_thesis.h"
#include <math.h>
#include "image.h"

/*****
function    correlate_image

description  This performs the correlation for one data point.
              The correlation must be performed as the kernel is
              stepped through the search area. In one position, the
              kernel corresponds to a certain area in the search area (which
              is larger). Every kernel pixel corresponds to a search
              area pixel. The value of each is multiplied with its
              corresponding pixel, summed, and scaled.
              The output correlation value is returned to make_correlation_image
              and stored in result_image at same location in the resultant
              image as the corresponding kernel center
*****/
extern double get_data (long row_kernel_index, long col_kernel_index, struct
                        IMAGE_DOUBLE kernel);

double correlate_image( struct IMAGE_DOUBLE kernel, struct IMAGE_DOUBLE
                        search_window, long row_image_center,
                        long col_image_center)
{
    long row_kernel_index, col_kernel_index;
    long half_kernel_row, half_kernel_col;
    long row_image_index, col_image_index;
    double sum = 0;
    double kernel_weight = 0;
    double search_weight = 0;
    double kernel_value;
    double search_value;
    half_kernel_row = kernel.num_of_rows/2;
    half_kernel_col = kernel.num_of_cols/2;

    for ( row_kernel_index = 0,
          row_image_index = row_image_center - half_kernel_row;
          row_kernel_index < kernel.num_of_rows;
          row_kernel_index++,
          row_image_index++){
        for ( col_kernel_index = 0,
              col_image_index = col_image_center - half_kernel_col;
              col_kernel_index < kernel.num_of_cols;
              col_kernel_index++,
              col_image_index++){
            kernel_value = get_data(row_kernel_index,col_kernel_index,kernel);
            search_value = get_data(row_image_index, col_image_index,search_window);
            kernel_weight += kernel_value * kernel_value;
            search_weight += search_value * search_value;
            sum += kernel_value * search_value;
        }
    }
    kernel_weight = sqrt(kernel_weight);
    search_weight = sqrt(search_weight);
    sum = sum/(kernel_weight*search_weight);
    /* printf ("\n kernel_weight %lf",kernel_weight);
    printf ("\n search_weight %lf",search_weight); */
    return (sum);
}

```

```

}



---




---




---



#include "k_thesis.h"
/***** function get_data.c *****/

double get_data(long row, long col, struct IMAGE_DOUBLE image)
{
    return(image.data[row*image.num_of_cols+col]);
}

/



---




---




---



#include "k_thesis.h"
void put_data(double value, long row, long col, struct IMAGE_DOUBLE image)
{
    long index;
    index = row*image.num_of_cols + col;
    image.data[index] = value;
}



---




---




---



struct REFERENCE_POINTS {

/** these reference points are from the image to be registered
    _ref refers to the upper left hand corner
    _one to the upper right hand corner
    _two to the lower left hand corner
*/

    double x_ref, y_ref, utm_x_ref, utm_y_ref;
    double x_one, y_one, utm_x_one, utm_y_one;
    double x_two, y_two, utm_x_two, utm_y_two;
};

struct FILECOORD {
    double x_out, y_out;
};

struct UTMCOORD {
    double northing, easting;
};

struct IMAGE_DOUBLE {
    int    num_of_rows;
    int    num_of_cols;
    double *data;
};

struct IMAGE_UNSIGNED {
    int    num_of_rows;
    int    num_of_cols;
    unsigned char *data;
};

```

```
/** #define SEARCH_WINDOW_SIZE 100 */  
#define _maxval(x,y) ( (x) ) > ( (y) ) ? (x) : (y)
```

Appendix C-Program VEC_ANGLE

Program to calculate the amount of rotation needed to approximately register images: VEC_ANGLE.C

User: Prompted to enter to points from each image.
Image 1 is the reference image
Image 2 is the image to be altered

Must define a vector in each image, *i.e.* choose any 2 points in the reference image that can also be identified in the "to-be-altered" image. The angle between the 2 is calculated. When entering value into ERDAS routine LRECTIFY, must pay attention to desired direction of rotation as this is not taken into account here. A negative angle refers to a clockwise rotation

```

#include <math.h>
#include <stdio.h>
#include "k_thesis.h"

/*****
program    vec_angle.c

description  This program calculates the approximate rotation of the to-
              be registered image based on two points chosen in the image. The
              points must be referenced to the origin for the calculation to
              work correctly

              The user must simply pick any two points in an image
              that define a vector. These two points must be found in
              all of the images to-be-registered.

Input:      First point in Image 1 (x,y) [defines first vector]
              Second point in Image 1 (x,y)

              First point in Image 2 (x,y) [defines second vector]
              Second point in Image 2 (x,y)
*****/
main()
{
    double imagel_x1,imagel_y1,imagel_x2,imagel_y2;
    double image2_x1, image2_y1, image2_x2, image2_y2;
    double x1,y1,x2,y2;
    double dot, mag_1, mag_2;
    double cos_angle, angle_rad, angle_deg, denominator;

    /*** entering data for first vector ***/
    printf ("\n Enter value for image 1 x1 >");
    scanf ("%lf",&imagel_x1);
    printf ("\n Enter value for image 1 y1 >");
    scanf ("%lf",&imagel_y1);
    printf ("\n Enter value for image 1 x2 >");
    scanf ("%lf",&imagel_x2);
    printf ("\n Enter value for image 1 y2 >");
    scanf ("%lf",&imagel_y2);

    /*** entering data for second vector ***/
    printf ("\n Enter value for image 2 x1 >");
    scanf ("%lf",&image2_x1);
    printf ("\n Enter value for image 2 y1 >");
    scanf ("%lf",&image2_y1);
    printf ("\n Enter value for image 2 x2 >");
    scanf ("%lf",&image2_x2);
    printf ("\n Enter value for image 2 y2 >");
    scanf ("%lf",&image2_y2);

    /*** translates vectors so that the origin is 0,0 ***/
    x1=imagel_x2 - imagel_x1;
    y1=imagel_y2 - imagel_y1;
    x2=image2_x2 - image2_x1;
    y2=image2_y2 - image2_y1;
    dot = x1*x2 + y1*y2;
    mag_1 = sqrt(x1*x1 + y1*y1);
    mag_2 = sqrt(x2*x2 + y2*y2);
    denominator = mag_1*mag_2;
    cos_angle = dot / (mag_1*mag_2);

```

```
angle_rad = acos(cos_angle);
angle_deg = (angle_rad*180)/3.14159;
printf ("\n dot product  = %lf",dot);
printf ("\n length vector A = %lf",mag_1);
printf ("\n length vector B = %lf",mag_2);
printf ("\n denominator = %lf",denominator);
printf ("\n cosine angle = %lf",cos_angle);
printf ("\n angle (radians) = %lf",angle_rad);
printf ("\n angle (degrees) = %lf",angle_deg);
}
```

Appendix D-Coordinate transformation data

Output from ERDAS routine COORDN

Coordinate transformations to register images

*****COORDN OUTPUT FOR 1972 IMAGE*****

Coefficient Filename : SNOQ

Here are the points you are using:

Point Count	Point Number	X	Y	Image Pixels	
=====	=====	=====	=====	X	Y
=====	=====	=====	=====	=====	=====
1	1	592324.3120	5263435.5000	2354.00	798.00
2	2	593596.4370	5257543.5000	2404.00	867.00
3	3	598055.2500	5256323.0000	2482.00	871.00
4	4	596498.0000	5282152.0000	2337.00	556.00
5	5	606502.0000	5296234.0000	2436.00	357.00
6	6	600351.9370	5312486.0000	2256.00	173.00
7	7	604517.2500	5314818.5000	2315.00	133.00
8	8	587360.0000	5304403.5000	2075.00	311.00
9	9	580573.8120	5286483.5000	2046.00	549.00
10	10	624710.6870	5286339.5000	2791.00	426.00
11	11	616392.1870	5263449.0000	2762.00	728.00
12	12	583880.9370	5246068.5000	2293.00	1036.00
13	13	580678.6250	5242794.0000	2256.00	1086.00
14	14	579899.9370	5221448.0000	2344.00	1349.00
15	15	578321.0000	5220668.5000	2321.00	1363.00
16	16	604747.3750	5219883.0000	2772.00	1295.00
17	17	633392.4370	5236585.5000	3182.00	1011.00
18	18	616824.5620	5305187.5000	2569.00	214.00
19	19	649301.6870	5292815.0000	3191.00	275.00
20	20	621243.9370	5250116.5000	2911.00	879.00
21	21	606942.2500	5193946.0000	2929.00	1606.00
22	22	610294.3120	5266797.0000	2643.00	704.00
23	23	623767.6870	5226690.0000	3060.00	1157.00

Order of transformation is 2

0.7188542E+01 0.5605988E+02
0.1254099E-01-0.3786790E-02
-0.3930801E-03-0.7988892E-02
0.5448395E-05 0.9695411E-06
-0.4111700E-06-0.5134207E-07
-0.3907054E-06-0.4024488E-06

These are the computed results of the matrix above:

Point Count	Point Number	Image X Pixel	X Pixel Residual	Image Y Pixel	Y Pixel Residual
1	1	2353.59	-0.4097E+00	798.61	0.6128E+00
2	2	2402.98	-0.1016E+01	867.11	0.1122E+00
3	3	2484.01	0.2006E+01	869.13	-0.1872E+01
4	4	2334.87	-0.2130E+01	556.96	0.9595E+00
5	5	2436.92	0.9248E+00	355.14	-0.1857E+01
6	6	2254.99	-0.1012E+01	173.18	0.1842E+00
7	7	2314.29	-0.7115E+00	132.46	-0.5417E+00
8	8	2075.02	0.1799E-01	310.31	-0.6866E+00
9	9	2047.00	0.1003E+01	550.25	0.1250E+01
10	10	2795.12	0.4115E+01	424.48	-0.1521E+01
11	11	2761.77	-0.2253E+00	729.01	0.1009E+01
12	12	2294.17	0.1173E+01	1035.96	-0.4276E-01
13	13	2256.09	0.8974E-01	1085.41	-0.5898E+00
14	14	2343.83	-0.1689E+00	1348.75	-0.2464E+00
15	15	2321.13	0.1276E+00	1362.91	-0.9075E-01
16	16	2769.86	-0.2138E+01	1295.67	0.6669E+00
17	17	3181.70	-0.2956E+00	1009.64	-0.1361E+01
18	18	2569.83	0.8267E+00	215.48	0.1477E+01
19	19	3189.71	-0.1285E+01	275.54	0.5405E+00
20	20	2908.25	-0.2751E+01	878.50	-0.5041E+00
21	21	2929.37	0.3664E+00	1606.06	0.6459E-01
22	22	2641.81	-0.1187E+01	705.46	0.1455E+01
23	23	3062.68	0.2680E+01	1157.98	0.9821E+00

X RMS error= 1.54439 Y RMS error= 0.98883

Total RMS error= 1.83383

Point Count	Point Number	Error	Error Contribution by Point
1	1	0.7372	0.4020
2	2	1.0219	0.5572
3	3	2.7440	1.4963
4	4	2.3361	1.2739
5	5	2.0747	1.1313
6	6	1.0290	0.5611
7	7	0.8943	0.4877
8	8	0.6869	0.3745
9	9	1.6032	0.8742
10	10	4.3875	2.3925
11	11	1.0334	0.5635
12	12	1.1738	0.6401
13	13	0.5966	0.3253
14	14	0.2987	0.1629
15	15	0.1566	0.0854
16	16	2.2394	1.2211
17	17	1.3932	0.7597
18	18	1.6929	0.9232
19	19	1.3944	0.7604
20	20	2.7970	1.5252
21	21	0.3720	0.2029
22	22	1.8782	1.0242
23	23	2.8543	1.5565

Here are the points that were used:

Point Count	Point Number	X	Y	Image Pixels	
=====	=====	=====	=====	X	Y
=====	=====	=====	=====	=====	=====
1	1	592324.3120	5263435.5000	2354.00	798.00
2	2	593596.4370	5257543.5000	2404.00	867.00
3	3	598055.2500	5256323.0000	2482.00	871.00
4	4	596498.0000	5282152.0000	2337.00	556.00
5	5	606502.0000	5296234.0000	2436.00	357.00
6	6	600351.9370	5312486.0000	2256.00	173.00
7	7	604517.2500	5314818.5000	2315.00	133.00
8	8	587360.0000	5304403.5000	2075.00	311.00
9	9	580573.8120	5286483.5000	2046.00	549.00
10	10	624710.6870	5286339.5000	2791.00	426.00
11	11	616392.1870	5263449.0000	2762.00	728.00
12	12	583880.9370	5246068.5000	2293.00	1036.00
13	13	580678.6250	5242794.0000	2256.00	1086.00
14	14	579899.9370	5221448.0000	2344.00	1349.00
15	15	578321.0000	5220668.5000	2321.00	1363.00
16	16	604747.3750	5219883.0000	2772.00	1295.00
17	17	633392.4370	5236585.5000	3182.00	1011.00
18	18	616824.5620	5305187.5000	2569.00	214.00
19	19	649301.6870	5292815.0000	3191.00	275.00
20	20	621243.9370	5250116.5000	2911.00	879.00
21	21	606942.2500	5193946.0000	2929.00	1606.00
22	22	610294.3120	5266797.0000	2643.00	704.00
23	23	623767.6870	5226690.0000	3060.00	1157.00

Number of Control Points 23

Maximum acceptable RMS error specified by user 2.00000

*****COORDN OUTPUT FOR 1981 IMAGE*****

Coefficient Filename : 81

Here are the points you are using:

Point Count	Point Number	X	Y	Image Pixels	
				X	Y
1	1	596623.3750	5251061.0000	2412.00	1306.00
2	2	606419.3750	5296170.0000	2538.00	734.00
3	3	583983.3750	5246163.0000	2252.00	1368.00
4	4	580665.3750	5242845.0000	2211.00	1411.00
5	5	579717.3750	5222384.0000	2197.00	1670.00
6	6	604918.3750	5219935.0000	2515.00	1701.00
7	7	593700.3750	5237394.0000	2375.00	1479.00
8	8	592357.3750	5263543.0000	2359.00	1148.00
9	9	597966.3750	5256275.0000	2429.00	1240.00
10	10	586906.3750	5266782.0000	2290.00	1107.00
11	11	621666.3750	5249797.0000	2731.00	1344.00
12	12	633121.3750	5236525.0000	2876.00	1492.00
13	13	603891.3750	5284478.0000	2504.00	883.00
14	14	588328.3750	5301068.0000	2308.00	672.00
15	15	587380.3750	5304465.0000	2296.00	629.00
16	16	587538.3750	5310232.0000	2297.00	555.00
17	17	624431.3750	5286295.0000	2768.00	860.00
18	18	598203.3750	5313866.0000	2410.00	467.00
19	19	593621.3750	5257539.0000	2375.00	1224.00

Order of transformation is 2

-0.1116851E+03-0.1312834E+03
 0.1278396E-01 0.2089699E-01
 0.4048165E-01 0.6071567E-01
 0.5951224E-05 0.6885570E-05
 -0.1391608E-05-0.5546356E-05
 -0.3766617E-05-0.6664744E-05

 These are the computed results of the matrix above:

Point Count	Point Number	Image X Pixel	X Pixel Residual	Image Y Pixel	Y Pixel Residual
1	1	2412.96	0.9614E+00	1309.36	0.3363E+01
2	2	2532.70	-0.5298E+01	726.10	-0.7901E+01
3	3	2254.29	0.2292E+01	1371.75	0.3753E+01
4	4	2212.55	0.1552E+01	1413.53	0.2535E+01
5	5	2195.56	-0.1444E+01	1664.32	-0.5683E+01
6	6	2511.49	-0.3509E+01	1696.52	-0.4485E+01
7	7	2374.20	-0.7953E+00	1480.02	0.1020E+01
8	8	2360.07	0.1072E+01	1151.68	0.3677E+01
9	9	2430.24	0.1242E+01	1243.51	0.3510E+01
10	10	2292.06	0.2063E+01	1111.44	0.4435E+01
11	11	2731.60	0.5995E+00	1329.48	-0.1452E+02
12	12	2878.53	0.2532E+01	1502.64	0.1064E+02
13	13	2503.29	-0.7146E+00	880.22	-0.2775E+01
14	14	2304.89	-0.3108E+01	666.11	-0.5894E+01
15	15	2292.17	-0.3827E+01	621.57	-0.7432E+01
16	16	2292.30	-0.4700E+01	544.87	-0.1013E+02
17	17	2764.60	-0.3403E+01	857.17	-0.2832E+01
18	18	2423.75	0.1375E+02	491.97	0.2497E+02
19	19	2375.73	0.7338E+00	1227.75	0.3746E+01

X RMS error= 4.06229 Y RMS error= 8.47028

Total RMS error= 9.39403

Point Count	Point Number	Error	Error Contribution by Point
=====	=====	=====	=====
1	1	3.4982	0.3724
2	2	9.5127	1.0126
3	3	4.3978	0.4682
4	4	2.9718	0.3164
5	5	5.8634	0.6242
6	6	5.6945	0.6062
7	7	1.2934	0.1377
8	8	3.8305	0.4078
9	9	3.7236	0.3964
10	10	4.8914	0.5207
11	11	14.5326	1.5470
12	12	10.9331	1.1638
13	13	2.8657	0.3051
14	14	6.6629	0.7093
15	15	8.3597	0.8899
16	16	11.1656	1.1886
17	17	4.4272	0.4713
18	18	28.5097	3.0349
19	19	3.8171	0.4063

Point number 18 has been deleted.

Here are the points you are using:

Point Count	Point Number	X	Y	Image Pixels	
=====	=====	=====	=====	X	Y
1	1	596623.3750	5251061.0000	2412.00	1306.00
2	2	606419.3750	5296170.0000	2538.00	734.00
3	3	583983.3750	5246163.0000	2252.00	1368.00
4	4	580665.3750	5242845.0000	2211.00	1411.00
5	5	579717.3750	5222384.0000	2197.00	1670.00
6	6	604918.3750	5219935.0000	2515.00	1701.00
7	7	593700.3750	5237394.0000	2375.00	1479.00
8	8	592357.3750	5263543.0000	2359.00	1148.00
9	9	597966.3750	5256275.0000	2429.00	1240.00
10	10	586906.3750	5266782.0000	2290.00	1107.00
11	11	621666.3750	5249797.0000	2731.00	1344.00
12	12	633121.3750	5236525.0000	2876.00	1492.00
13	13	603891.3750	5284478.0000	2504.00	883.00
14	14	588328.3750	5301068.0000	2308.00	672.00
15	15	587380.3750	5304465.0000	2296.00	629.00
16	16	587538.3750	5310232.0000	2297.00	555.00
17	17	624431.3750	5286295.0000	2768.00	860.00
18	19	593621.3750	5257539.0000	2375.00	1224.00

Order of transformation is 2

```
-0.2244941E+02 0.3077072E+02
0.5402286E-02 0.7491699E-02
0.7367618E-02 0.5798030E-03
0.2610988E-05 0.8196226E-06
0.7877981E-06-0.1588503E-05
-0.7415349E-06-0.1171123E-05
```

These are the computed results of the matrix above:

Point Count	Point Number	Image X Pixel	X Pixel Residual	Image Y Pixel	Y Pixel Residual
=====	=====	=====	=====	=====	=====
1	1	2412.21	0.2143E+00	1308.01	0.2007E+01
2	2	2537.52	-0.4782E+00	734.85	0.8526E+00
3	3	2252.46	0.4635E+00	1368.43	0.4325E+00
4	4	2210.58	-0.4202E+00	1409.95	-0.1047E+01
5	5	2197.36	0.3615E+00	1667.60	-0.2405E+01
6	6	2514.89	-0.1124E+00	1702.68	0.1683E+01
7	7	2374.46	-0.5425E+00	1480.48	0.1479E+01
8	8	2358.74	-0.2568E+00	1149.26	0.1264E+01
9	9	2429.46	0.4600E+00	1242.09	0.2091E+01
10	10	2289.98	-0.2244E-01	1107.65	0.6486E+00
11	11	2730.67	-0.3279E+00	1327.80	-0.1620E+02
12	12	2876.26	0.2599E+00	1498.51	0.6510E+01
13	13	2505.36	0.1358E+01	883.99	0.9892E+00
14	14	2307.74	-0.2595E+00	671.28	-0.7210E+00
15	15	2295.63	-0.3656E+00	627.85	-0.1146E+01
16	16	2297.40	0.3983E+00	554.13	-0.8702E+00
17	17	2767.74	-0.2560E+00	862.88	0.2884E+01
18	19	2374.53	-0.4742E+00	1225.55	0.1552E+01

X RMS error= 0.47445 Y RMS error= 4.36202

Total RMS error= 4.38774

Point Count	Point Number	Error	Error Contribution by Point
=====	=====	=====	=====
1	1	2.0182	0.4600
2	2	0.9776	0.2228
3	3	0.6340	0.1445
4	4	1.1279	0.2571
5	5	2.4319	0.5543
6	6	1.6871	0.3845
7	7	1.5755	0.3591
8	8	1.2897	0.2939
9	9	2.1408	0.4879
10	10	0.6489	0.1479
11	11	16.2077	3.6939
12	12	6.5154	1.4849
13	13	1.6803	0.3830
14	14	0.7662	0.1746
15	15	1.2027	0.2741
16	16	0.9570	0.2181
17	17	2.8950	0.6598
18	19	1.6231	0.3699

Point number 11 has been deleted.

Here are the points you are using:

Point Count	Point Number	X	Y	Image Pixels	
				X	Y
1	1	596623.3750	5251061.0000	2412.00	1306.00
2	2	606419.3750	5296170.0000	2538.00	734.00
3	3	583983.3750	5246163.0000	2252.00	1368.00
4	4	580665.3750	5242845.0000	2211.00	1411.00
5	5	579717.3750	5222384.0000	2197.00	1670.00
6	6	604918.3750	5219935.0000	2515.00	1701.00
7	7	593700.3750	5237394.0000	2375.00	1479.00
8	8	592357.3750	5263543.0000	2359.00	1148.00
9	9	597966.3750	5256275.0000	2429.00	1240.00
10	10	586906.3750	5266782.0000	2290.00	1107.00
11	19	593621.3750	5257539.0000	2375.00	1224.00
12	12	633121.3750	5236525.0000	2876.00	1492.00
13	13	603891.3750	5284478.0000	2504.00	883.00
14	14	588328.3750	5301068.0000	2308.00	672.00
15	15	587380.3750	5304465.0000	2296.00	629.00
16	16	587538.3750	5310232.0000	2297.00	555.00
17	17	624431.3750	5286295.0000	2768.00	860.00

Order of transformation is 2
 -0.2178053E+02 0.6382182E+02
 0.5226336E-02-0.1202823E-02
 0.7133356E-02-0.1099566E-01
 0.2616249E-05 0.1079611E-05
 0.8196241E-06-0.1583728E-07
 -0.7210587E-06-0.1593468E-06

These are the computed results of the matrix above:

Point Count	Point Number	Image X Pixel	X Pixel Residual	Image Y Pixel	Y Pixel Residual
1	1	2412.18	0.1776E+00	1306.19	0.1923E+00
2	2	2537.51	-0.4941E+00	734.07	0.6817E-01
3	3	2252.46	0.4635E+00	1368.43	0.4312E+00
4	4	2210.59	-0.4074E+00	1410.59	-0.4143E+00
5	5	2197.41	0.4083E+00	1669.91	-0.9200E-01
6	6	2514.85	-0.1512E+00	1700.77	-0.2311E+00
7	7	2374.44	-0.5634E+00	1479.44	0.4429E+00
8	8	2358.72	-0.2839E+00	1147.92	-0.7583E-01
9	9	2429.42	0.4194E+00	1240.09	0.8668E-01
10	10	2289.96	-0.3728E-01	1106.92	-0.8479E-01
11	19	2374.50	-0.5044E+00	1224.06	0.5697E-01
12	12	2876.13	0.1278E+00	1491.98	-0.1639E-01
13	13	2505.33	0.1326E+01	882.40	-0.5966E+00
14	14	2307.75	-0.2482E+00	671.84	-0.1624E+00
15	15	2295.65	-0.3479E+00	628.73	-0.2705E+00
16	16	2297.43	0.4261E+00	555.51	0.5065E+00
17	17	2767.69	-0.3111E+00	860.16	0.1591E+00

X RMS error= 0.47971 Y RMS error= 0.28886

Total RMS error= 0.55997

Point Count	Point Number	Error	Error Contribution by Point
=====	=====	=====	=====
1	1	0.2618	0.4675
2	2	0.4988	0.8908
3	3	0.6331	1.1305
4	4	0.5811	1.0377
5	5	0.4185	0.7474
6	6	0.2762	0.4932
7	7	0.7167	1.2798
8	8	0.2939	0.5248
9	9	0.4283	0.7648
10	10	0.0926	0.1654
11	19	0.5076	0.9066
12	12	0.1289	0.2301
13	13	1.4542	2.5970
14	14	0.2966	0.5297
15	15	0.4407	0.7869
16	16	0.6620	1.1821
17	17	0.3494	0.6240

Here are the points that were used:

Point Count	Point Number	X	Y	Image Pixels	
=====	=====	=====	=====	X	Y
1	1	596623.3750	5251061.0000	2412.00	1306.00
2	2	606419.3750	5296170.0000	2538.00	734.00
3	3	583983.3750	5246163.0000	2252.00	1368.00
4	4	580665.3750	5242845.0000	2211.00	1411.00
5	5	579717.3750	5222384.0000	2197.00	1670.00
6	6	604918.3750	5219935.0000	2515.00	1701.00
7	7	593700.3750	5237394.0000	2375.00	1479.00
8	8	592357.3750	5263543.0000	2359.00	1148.00
9	9	597966.3750	5256275.0000	2429.00	1240.00
10	10	586906.3750	5266782.0000	2290.00	1107.00
11	19	593621.3750	5257539.0000	2375.00	1224.00
12	12	633121.3750	5236525.0000	2876.00	1492.00
13	13	603891.3750	5284478.0000	2504.00	883.00
14	14	588328.3750	5301068.0000	2308.00	672.00
15	15	587380.3750	5304465.0000	2296.00	629.00
16	16	587538.3750	5310232.0000	2297.00	555.00
17	17	624431.3750	5286295.0000	2768.00	860.00

Number of Control Points 17

Maximum acceptable RMS error specified by user 2.00000

Here are the points that were NOT used :

Point Count	Point Number	X	Y	Image Pixels	
=====	=====	=====	=====	X	Y
1	18	598203.3750	5313866.0000	2410.00	467.00
2	11	621666.3750	5249797.0000	2731.00	1344.00

*****COORDN OUTPUT FOR 1988 IMAGE*****

Coefficient Filename: 88

Here are the points you are using:

Point Count	Point Number	X	Y	Image Pixels	
				X	Y
1	1	596623.3750	5251061.0000	1933.00	1552.00
2	2	606419.3750	5296170.0000	2060.00	981.00
3	3	583983.3750	5246163.0000	1773.00	1614.00
4	4	580665.3750	5242845.0000	1731.00	1656.00
5	5	579717.3750	5222384.0000	1718.00	1916.00
6	6	604918.3750	5219935.0000	2036.00	1947.00
7	7	593700.3750	5237394.0000	1895.00	1726.00
8	8	592357.3750	5263543.0000	1880.00	1394.00
9	9	597966.3750	5256275.0000	1951.00	1487.00
10	10	586906.3750	5266782.0000	1811.00	1353.00
11	11	621666.3750	5249797.0000	2251.00	1571.00
12	12	633121.3750	5236525.0000	2395.00	1740.00
13	13	603891.3750	5284478.0000	2027.00	1130.00
14	14	588328.3750	5301068.0000	1830.00	918.00
15	15	587380.3750	5304465.0000	1818.00	875.00
16	16	587538.3750	5310232.0000	1768.00	769.00
17	17	624431.3750	5286295.0000	2289.00	1108.00
18	18	598203.3750	5313866.0000	1907.00	742.00
19	19	593621.3750	5257539.0000	1896.00	1471.00

Order of transformation is 2

-0.2719421E+03-0.3419980E+02
 -0.1273514E-01-0.1915097E-01
 0.1042022E+00 0.2845036E-01
 0.1726564E-05-0.8004962E-06
 0.4447227E-05 0.3850458E-05
 -0.1016340E-04-0.4134108E-05

These are the computed results of the matrix above:

Point Count	Point Number	Image X Pixel	X Pixel Residual	Image Y Pixel	Y Pixel Residual
1	1	1937.24	0.4241E+01	1554.62	0.2624E+01
2	2	2048.22	-0.1178E+02	977.44	-0.3556E+01
3	3	1776.72	0.3722E+01	1615.26	0.1263E+01
4	4	1734.29	0.3295E+01	1656.94	0.9362E+00
5	5	1713.81	-0.4188E+01	1914.27	-0.1729E+01
6	6	2027.85	-0.8154E+01	1944.85	-0.2151E+01
7	7	1896.86	0.1862E+01	1726.82	0.8200E+00
8	8	1882.84	0.2845E+01	1395.14	0.1141E+01
9	9	1954.64	0.3641E+01	1488.63	0.1631E+01
10	10	1812.88	0.1882E+01	1352.67	-0.3315E+00
11	11	2255.51	0.4506E+01	1572.83	0.1833E+01
12	12	2396.03	0.1027E+01	1738.92	-0.1080E+01
13	13	2023.16	-0.3844E+01	1128.35	-0.1649E+01
14	14	1810.85	-0.1915E+02	908.15	-0.9855E+01
15	15	1795.34	-0.2266E+02	863.23	-0.1177E+02
16	16	1791.23	0.2323E+02	787.34	0.1834E+02
17	17	2287.02	-0.1975E+01	1109.39	0.1394E+01
18	18	1925.05	0.1805E+02	743.18	0.1182E+01
19	19	1899.45	0.3451E+01	1471.96	0.9582E+00

X RMS error= 10.49183 Y RMS error= 5.70155

Total RMS error= 11.94095

Point Count	Point Number	Error	Error Contribution by Point
=====	=====	=====	=====
1	1	4.9870	0.4176
2	2	12.3091	1.0308
3	3	3.9310	0.3292
4	4	3.4252	0.2868
5	5	4.5310	0.3795
6	6	8.4327	0.7062
7	7	2.0344	0.1704
8	8	3.0650	0.2567
9	9	3.9901	0.3342
10	10	1.9109	0.1600
11	11	4.8643	0.4074
12	12	1.4902	0.1248
13	13	4.1828	0.3503
14	14	21.5365	1.8036
15	15	25.5322	2.1382
16	16	29.5924	2.4782
17	17	2.4176	0.2025
18	18	18.0930	1.5152
19	19	3.5814	0.2999

Point number 16 has been deleted.

Here are the points you are using:

Point Count	Point Number	X	Y	Image Pixels	
=====	=====	=====	=====	X	Y
				=====	=====
1	1	596623.3750	5251061.0000	1933.00	1552.00
2	2	606419.3750	5296170.0000	2060.00	981.00
3	3	583983.3750	5246163.0000	1773.00	1614.00
4	4	580665.3750	5242845.0000	1731.00	1656.00
5	5	579717.3750	5222384.0000	1718.00	1916.00
6	6	604918.3750	5219935.0000	2036.00	1947.00
7	7	593700.3750	5237394.0000	1895.00	1726.00
8	8	592357.3750	5263543.0000	1880.00	1394.00
9	9	597966.3750	5256275.0000	1951.00	1487.00
10	10	586906.3750	5266782.0000	1811.00	1353.00
11	11	621666.3750	5249797.0000	2251.00	1571.00
12	12	633121.3750	5236525.0000	2395.00	1740.00
13	13	603891.3750	5284478.0000	2027.00	1130.00
14	14	588328.3750	5301068.0000	1830.00	918.00
15	15	587380.3750	5304465.0000	1818.00	875.00
16	19	593621.3750	5257539.0000	1896.00	1471.00
17	17	624431.3750	5286295.0000	2289.00	1108.00
18	18	598203.3750	5313866.0000	1907.00	742.00

Order of transformation is 2

```
-0.2271363E+03 0.1167926E+01
 0.1065780E-01-0.6855886E-03
 0.8446600E-01 0.1287149E-01
 0.6018678E-05 0.2587512E-05
-0.1006068E-05-0.4541366E-06
-0.7971865E-05-0.2404207E-05
```

These are the computed results of the matrix above:

Point Count	Point Number	Image X Pixel	X Pixel Residual	Image Y Pixel	Y Pixel Residual
=====	=====	=====	=====	=====	=====
1	1	1935.67	0.2671E+01	1553.38	0.1385E+01
2	2	2049.09	-0.1091E+02	978.14	-0.2863E+01
3	3	1776.96	0.3956E+01	1615.45	0.1448E+01
4	4	1734.97	0.3969E+01	1657.47	0.1469E+01
5	5	1713.93	-0.4072E+01	1914.36	-0.1637E+01
6	6	2028.29	-0.7712E+01	1945.20	-0.1802E+01
7	7	1895.52	0.5182E+00	1725.76	-0.2405E+00
8	8	1882.83	0.2826E+01	1395.13	0.1126E+01
9	9	1953.07	0.2073E+01	1487.39	0.3935E+00
10	10	1814.68	0.3677E+01	1354.09	0.1086E+01
11	11	2253.74	0.2738E+01	1571.44	0.4381E+00
12	12	2398.73	0.3733E+01	1741.06	0.1056E+01
13	13	2022.92	-0.4076E+01	1128.17	-0.1831E+01
14	14	1819.62	-0.1038E+02	915.07	-0.2930E+01
15	15	1805.59	-0.1241E+02	871.32	-0.3680E+01
16	19	1898.65	0.2649E+01	1471.33	0.3256E+00
17	17	2283.70	-0.5299E+01	1106.77	-0.1230E+01
18	18	1933.04	0.2604E+02	749.49	0.7486E+01

X RMS error= 8.40824 Y RMS error= 2.44258

Total RMS error= 8.75584

Point Count	Point Number	Error	Error Contribution by Point
=====	=====	=====	=====
1	1	3.0083	0.3436
2	2	11.2749	1.2877
3	3	4.2127	0.4811
4	4	4.2321	0.4833
5	5	4.3889	0.5013
6	6	7.9192	0.9045
7	7	0.5713	0.0652
8	8	3.0419	0.3474
9	9	2.1105	0.2410
10	10	3.8343	0.4379
11	11	2.7730	0.3167
12	12	3.8796	0.4431
13	13	4.4680	0.5103
14	14	10.7822	1.2314
15	15	12.9453	1.4785
16	19	2.6693	0.3049
17	17	5.4403	0.6213
18	18	27.0954	3.0945

Point number 18 has been deleted.

Here are the points you are using:

Point Count	Point Number	Image Pixels	
		X	Y
=====	=====	=====	=====
1	1	596623.3750	5251061.0000
2	2	606419.3750	5296170.0000
3	3	583983.3750	5246163.0000
4	4	580665.3750	5242845.0000
5	5	579717.3750	5222384.0000
6	6	604918.3750	5219935.0000
7	7	593700.3750	5237394.0000
8	8	592357.3750	5263543.0000
9	9	597966.3750	5256275.0000
10	10	586906.3750	5266782.0000
11	11	621666.3750	5249797.0000
12	12	633121.3750	5236525.0000
13	13	603891.3750	5284478.0000
14	14	588328.3750	5301068.0000
15	15	587380.3750	5304465.0000
16	19	593621.3750	5257539.0000
17	17	624431.3750	5286295.0000

Order of transformation is 2

```
-0.1213194E+02 0.6297542E+02
0.5780400E-02-0.2087703E-02
0.3200037E-02-0.1049011E-01
0.8140073E-06 0.1091321E-05
0.1126233E-05 0.1588384E-06
-0.3628539E-06-0.2168381E-06
```

These are the computed results of the matrix above:

Point Count	Point Number	Image		Image	
		X Pixel	Residual	Y Pixel	Residual
=====	=====	=====	=====	=====	=====
1	1	1933.30	0.3014E+00	1552.70	0.7036E+00
2	2	2059.97	-0.2627E-01	981.26	0.2646E+00
3	3	1773.10	0.9899E-01	1614.34	0.3389E+00
4	4	1731.01	0.1063E-01	1656.33	0.3305E+00
5	5	1717.90	-0.9957E-01	1915.51	-0.4948E+00
6	6	2035.87	-0.1296E+00	1947.38	0.3781E+00
7	7	1895.43	0.4270E+00	1725.73	-0.2667E+00
8	8	1879.93	-0.6748E-01	1394.29	0.2945E+00
9	9	1950.63	-0.3699E+00	1486.69	-0.3089E+00
10	10	1811.01	0.7348E-02	1353.03	0.3068E-01
11	11	2250.88	-0.1164E+00	1570.62	-0.3825E+00
12	12	2395.09	0.9154E-01	1740.01	0.9405E-02
13	13	2027.31	0.3129E+00	1129.43	-0.5697E+00
14	14	1829.98	-0.1621E-01	918.05	0.4859E-01
15	15	1817.98	-0.1789E-01	874.88	-0.1168E+00
16	19	1895.65	-0.3539E+00	1470.46	-0.5377E+00
17	17	2288.95	-0.5266E-01	1108.28	0.2783E+00

X RMS error= 0.20297 Y RMS error= 0.36661

Total RMS error= 0.41905

Appendix D

Point Count	Point Number	Error	Error Contribution by Point
=====	=====	=====	=====
1	1	0.7654	1.8266
2	2	0.2659	0.6346
3	3	0.3531	0.8426
4	4	0.3307	0.7892
5	5	0.5048	1.2045
6	6	0.3997	0.9538
7	7	0.5035	1.2015
8	8	0.3021	0.7210
9	9	0.4819	1.1500
10	10	0.0315	0.0753
11	11	0.3998	0.9542
12	12	0.0920	0.2196
13	13	0.6500	1.5510
14	14	0.0512	0.1222
15	15	0.1182	0.2821
16	19	0.6437	1.5362
17	17	0.2833	0.6760

Here are the points that were used:

Point Count	Point Number	X	Y	Image Pixels	
=====	=====	=====	=====	X	Y
1	1	596623.3750	5251061.0000	1933.00	1552.00
2	2	606419.3750	5296170.0000	2060.00	981.00
3	3	583983.3750	5246163.0000	1773.00	1614.00
4	4	580665.3750	5242845.0000	1731.00	1656.00
5	5	579717.3750	5222384.0000	1718.00	1916.00
6	6	604918.3750	5219935.0000	2036.00	1947.00
7	7	593700.3750	5237394.0000	1895.00	1726.00
8	8	592357.3750	5263543.0000	1880.00	1394.00
9	9	597966.3750	5256275.0000	1951.00	1487.00
10	10	586906.3750	5266782.0000	1811.00	1353.00
11	11	621666.3750	5249797.0000	2251.00	1571.00
12	12	633121.3750	5236525.0000	2395.00	1740.00
13	13	603891.3750	5284478.0000	2027.00	1130.00
14	14	588328.3750	5301068.0000	1830.00	918.00
15	15	587380.3750	5304465.0000	1818.00	875.00
16	19	593621.3750	5257539.0000	1896.00	1471.00
17	17	624431.3750	5286295.0000	2289.00	1108.00

Number of Control Points 17

Maximum acceptable RMS error specified by user 2.00000

Here are the points that were NOT used :

Point Count	Point Number	X	Y	Image Pixels	
=====	=====	=====	=====	X	Y
1	16	587538.3750	5310232.0000	1768.00	769.00
2	18	598203.3750	5313866.0000	1907.00	742.00

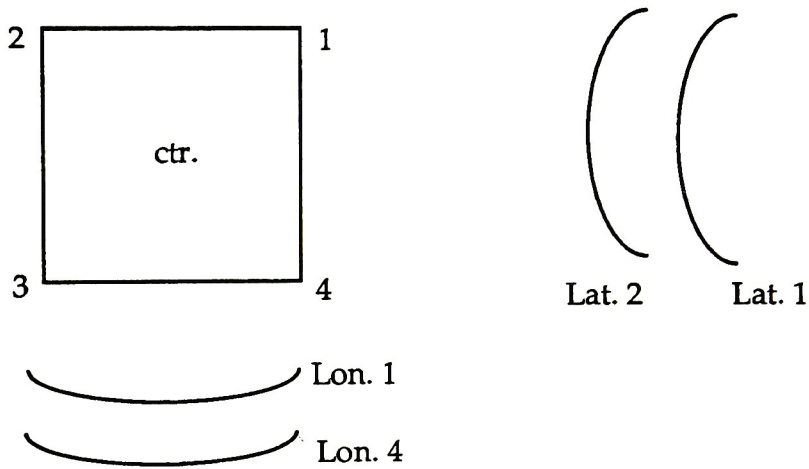
Appendix E-Image set ephemeris data

Corner point latitude and longitude data (obtained from list of scenes)

UTM coordinates obtained using ERDAS routine CCVRT

N= positive latitude (relative to Greenwich, England)

W= negative longitude (relative to Greenwich, England)



REFERENCE IMAGE (1972)

LANDSAT-1

Date: 7-29-72

	Latitude	Longitude	UTM X	UTM Y
Center Point	N 47°18'53"	W 122°23'02"	not needed	not needed
Corner 1	N 47°52'19"	W 120°51'11"	not needed	not needed
Corner 2	N 48°15'18"	W 123°13'33"	not needed	not needed
Corner 3	N 46°44'17"	W 123°53'00"	not needed	not needed
Corner 4	N 46°22'04"	W 121°34'24"	not needed	not needed

1981 IMAGE

LANDSAT-3

Date: 8-23-81

	Latitude	Longitude	UTM X	UTM Y
Center Point	N 47°17'00"	W 122°22'00"	547871.15	5236589.40
Corner 1	N 47°48'25"	W 120°52'51"	658658.26	5296804.66
Corner 2	N 48°21'04"	W 123°13'46"	482978.20	5336557.61
Corner 3	N 46°45'35"	W 123°49'21"	437149.33	5178571.52
Corner 4	N 46°22'56"	W 121°32'01"	612735.18	5137317.76

1988 IMAGE

LANDSAT-5

Date: 7-30-88

	Latitude	Longitude	UTM X	UTM Y
Center Point	N 47°27'00"	W 121°52'00"	585435.96	5255576.83
Corner 1	N 47°58'23"	W 120°22'41"	695690.72	5316407.61
Corner 2	N 48°21'07"	W 122°44'01"	519707.58	5355240.13
Corner 3	N 46°55'37"	W 123°19'31"	475257.53	5196884.49
Corner 4	N 46°32'53"	W 121°01'47"	651054.69	5156604.55

Appendix F-ERDAS routines outlined

ERDAS routines used

ALGEBRA: Data in a LAN image can be enhanced or changed by performing algebraic operations. A different operation can be applied to each band.

Make a new expression (can add your own library of expressions as well.)

Be sure to choose the NO-SCALE option when processing an image in the manner described in this thesis.

BSTATS: Allows user to generate image statistics. Mean, standard deviation, and histogram information.

CALC: Used to convert latitude/longitude data to UTM coordinates, they must be in decimal form. Within the CALC program DD does this conversion.

CALC> -dd (degrees, min, sec)

CCVRT: Allows conversion from lat/lon to UTM coordinates. User must specify input and output coordinate types (lat/lon , UTM). Can input coordinate pairs from a terminal. When choosing UTM coordinates, user will be prompted for information such as the UTM zone number (see ERDAS manual to determine appropriate zone, or look on a USGS map of the area).

All input pairs: x,y / lon,lat / / eastings/northing All pairs entered relative to Greenwich prime meridian and equator.

In this study: Latitude is positive
Longitude is negative

- COLORMOD:** After running ISODATA, the spectral classes can be examined and identified using COLORMOD.
- Select color palette (C)
 [S-Single] to examine one class.
 B= Blinks color on and off (*i.e. one class*)
 You can alter the color scheme using this routine or you can just look at it and be sure to **return the original color**.
- COORDN:** Used to generate the transformation to be input into NRECTIFY. User specifies the order of the transformation.
- CPYSCRN:** User can copy .LAN or .GIS from the display. When using CPYSCR, user is asked to save screen values or function memory values. If it is a .LAN file, choose F (function memory). Then, the values copied will be processed through the look-up tables before being written to the file.
- Can choose entire screen or section of screen.
- CURSES:** Used to look at data values.
- DIGSCRN:** Digitizes data on the screen which can be used to train an images. Asks if you want to digitize polygon, vector, or grid-cell mode.
 Choose Polygon. Choose point carefully with the mouse and follow instructions.
- IMPORTANT:** if there is to be more than a single polygon defining a particular class, choose the polygon and then enter the SAME class number for the next polygon.
- DISPLAY:** Displays a GIS file
- DISPOL:** Displays the vector or polygon data stored in a DIG file.
- FIXHED:** Adds or modifies information contained in the header records of a LAN or GIS file

- GCP:** Creates a ground control point (GCP) file to be used for image rectification. A control point consists of two sets of coordinates for the same location, one being map coordinates and the other pixel locations.
- Choose points that are easily identifiable on both the map and the image. Be careful in the map set-up as well as this could introduce error.
- HSTMATCH:** Used to match one histogram to another.
HISTOGRAM ONE is the image being changed
HISTOGRAM TWO is the source or reference histogram.
- Use statistics of an image file to perform histogram matching.
- Must enter the number of the band for each image. Be very careful because it is easy to make a mistake and ERDAS will not notice. For example, make sure you choose band 3 in both images.
- Enter function memory plane to use as a histogram (RGB) ---choose as appropriate. In this study, red was chosen each time for consistency. Only one band was run at a time although you can recycle through to alter 3 bands at a time. This can be confusing and mistakes can be made easily
- Use 97% of the histogram to remove extremes.
- Use STRETCH immediately following as the data is now stored in function memory and will be replaced.
- ISODATA:** (Iterative Self-Organizing Data Analysis Technique)
Unsupervised classification. User chooses number of classes and program separates data accordingly. ISODATA creates an SBD file but use SIGMAN to manipulate it and name the output signatures.

LDDATA:	<p>Used to load images from tape if Band Interleaved by Line (BIL)</p> <p>Run MTcount to determine number of rows and columns in image.</p> <p>Record # = num. of rows; Num. Bytes = num. of columns</p> <p>Note: may be given for all 4 bands so if Record # = 5964, there are only 5964/4 lines (1491).</p> <p>Can be confusing so examine images after reading them in to ensure they have been read in correctly.</p>
LISTIT:	Lists statistics of a file.
LOADX:	Was used to read in the 1972 image from tape because it was Band Sequential (BSQ).
LRECTIFY	<p>Used to transform a coordinate system using only 1st-order transformation.</p> <p>In this study, LRECTIFY was used to rotate and scale the images. Its important to scale and rotate at once so the data is only resampled once.</p>
MAXCLAS:	<p>Multi-spectral classification program. Classifies an input LAN file using : Minimum distance, Mahalanobis distance, or maximum likelihood.</p> <p>In this study, minimum distance to mean was found to give the best results.</p> <p>The user must choose no for parallelepiped optimization for best results.</p> <p>MAXCLAS requires a signature file (SBD). Use SIGMAN (signature manipulation) to do this.</p>
NRECTIFY	<p>Allows user to transform a LAN file to a new coordinate system). Must run COORDN first to generate the appropriate transformations. Need a ground control point file generated using GCP</p>

READ:	Used to read in a LAN file. If the file is not visible after reading it in, run BSTATS to scale the output data.
SIGEXT:	Generates training samples from a LAN file according to polygons in an input DIG file (from DIGSCRN). SIGEXT calculates signature statistics from these training samples and creates a new signature file with accompanying name file. MAXCLAS needs a NAM file to run as well as an SBD file.
SIGMAN:	Allows you to manipulate signatures. You can merge, delete, and rename signatures.
STRETCH:	Use directly after HSTMATCH to actually process the images. Read all questions carefully when using STRETCH as it is easy to make a mistake as to which band you want to manipulate <i>etc.</i>
SUBSET:	Allows user to remove a section of the image or subset out a band of data.
WRTIFF:	Writes an LAN or GIS file to a .tif file so that it is compatible with other software. Choose INTEL if working with a PC Choose MOTOROLA if using a Macintosh.