

**IMPROVING THE PERFORMANCE OF A HYBRID CLASSIFICATION
METHOD USING A PARALLEL ALGORITHM AND A
NOVEL DATA REDUCTION TECHNIQUE**

by

Rhonda Phillips

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

in

Computer Science

APPROVED:

Layne T. Watson
Chair of Advisory Committee

Roger W. Ehrich

Randolph H. Wynne

May 9, 2007
Blacksburg, Virginia

Key words: parallel processing, remote sensing, Landsat, forest area, high performance computing, singular value decomposition, classification

Copyright 2007, Rhonda Phillips

IMPROVING THE PERFORMANCE OF A HYBRID CLASSIFICATION METHOD USING A PARALLEL ALGORITHM AND A NOVEL DATA REDUCTION TECHNIQUE

by

Rhonda Phillips

(ABSTRACT)

This thesis presents both a shared memory parallel version of the hybrid classification algorithm IGSCR (iterative guided spectral class rejection) and a novel data reduction technique that can be used in conjunction with pIGSCR (parallel IGSCR). The parallel algorithm is motivated by a demonstrated need for more computing power driven by the increasing size of remote sensing datasets due to higher resolution sensors, larger study regions, and the like. Even with a fast algorithm such as pIGSCR, the reduction of dimension in a dataset is desirable in order to decrease the processing time further and possibly improve overall classification accuracy.

pIGSCR was developed to produce fast and portable code using Fortran 95, OpenMP, and the Hierarchical Data Format version 5 (HDF5) and accompanying data access library. The applicability of the faster pIGSCR algorithm is demonstrated by classifying Landsat data covering most of Virginia, USA into forest and non-forest classes with approximately 90 percent accuracy. Parallel results are given using the SGI Altix 3300 shared memory computer and the SGI Altix 3700 with as many as 64 processors reaching speedups of almost 77. This fast algorithm allows an analyst to perform and assess multiple classifications to refine parameters. As an example, pIGSCR was used for a factorial analysis consisting of 42 classifications of a 1.2 gigabyte image to select the number of initial classes (70) and class purity (70%) used for the remaining two images.

A feature selection or reduction method may be appropriate for a specific classification method depending on the properties and training required for the classification method, or an alternative band selection method may be derived based on the classification method itself. This thesis introduces a feature reduction method based on the singular value decomposition (SVD). This feature reduction technique was applied to training data from two multitemporal datasets of Landsat TM/ETM+ imagery acquired over a forested area in Virginia, USA and Rondônia, Brazil. Subsequent parallel iterative guided spectral class rejection (pIGSCR) forest/non-forest classifications were performed to determine the quality of the feature reduction. The classifications of the Virginia data were five times faster using SVD based feature reduction without affecting the classification accuracy.

Feature reduction using the SVD was also compared to feature reduction using principal components analysis (PCA). The highest average accuracies for the Virginia dataset (88.34%) and for the Amazon dataset (93.31%) were achieved using the SVD. The results presented here indicate that SVD based feature reduction can produce statistically significantly better classifications than PCA.

ACKNOWLEDGEMENTS

I would like to thank my advisor, Dr. Layne Watson, for his guidance and advice. I would like to thank Dr. Roger Ehrich and Dr. Randolph Wynne for serving on my thesis committee and giving me advice. I would also like to thank Dr. Christine Blinn for her assistance and expertise. I would like to thank Virginia Tech Advanced Research Computing for the use of their shared memory computers that were used to obtain results listed in this thesis.

Lastly, I would like to thank my family for inspiring me to stay in school.

This research was supported by NASA (NAG5-10548), Department of Energy (DE-FGO2-O6ER2572O), and the Department of Computer Science, Virginia Polytechnic Institute and State University.

TABLE OF CONTENTS

1. Introduction	1
2. Literature Review	2
2.1 IGSCR	2
2.1.1 Automation	3
2.2 Parallel Computing	3
2.2.1 Comparison of OpenMP and MPI	3
2.3 Feature Selection	4
3. Classification Algorithms	6
3.1 K -means	6
3.1.1 Implementation	8
3.2 K -means Initialization	9
3.2.1 Implementation	10
3.3 Maximum Likelihood	11
3.3.1 Implementation	13
3.4 IGSCR	13
3.4.1 Implementation	17
4. Hierarchical Data Format 5 (HDF5)	18
4.1 Image File	18
4.2 Training Data Files	19
4.3 Signature Files	19
4.4 Utility Programs	19
5. pIGSCR	20
5.1 Implementation Details	28
6. pIGSCR Experimental Results and Discussion	30
6.1 Data Description	30
6.2 Parameter Selection	31
6.3 Parallel Results	33
6.4 Discussion	36
7. Feature Reduction Within pIGSCR	39
8. SVD-Based Feature Reduction	41
9. Data Description and Preparation	43
10. SVD Results and Discussion	49

10.1 SVD versus PCA	50
11 Conclusions and Future Work	57
References	59
Vita	61

LIST OF TABLES

Table 1 HDF5 Objects	18
Table 2 Profiling Results	21
Table 3 Factorial Analysis for VA16	33
Table 4 Execution Time(sec) for VA15, VA16, and VA17, p = number of processors	36
Table 5 Comparison of Averages for PCA and SVD pIGSCR Classifications	51
Table 6 Classification Accuracies of Statistically Different Classifications	51

LIST OF FIGURES

Figure 1. Hexagonal grid used for interpretation of training data.	30
Figure 2. VA15 (Landsat ETM+ path 15).	31
Figure 3. VA16 (Landsat ETM+ path 16).	32
Figure 4. VA17 (Landsat ETM+ path 17).	32
Figure 5. Classification of VA15 (Landsat ETM+ path 15), green is forest and tan is non-forest.	34
Figure 6. Classification of VA16 (Landsat ETM+ path 16), green is forest and tan is non-forest.	35
Figure 7. Classification of VA17 (Landsat ETM+ path 17), green is forest and tan is non-forest.	35
Figure 8. Speedup with respect to one processor.	36
Figure 9. Speedup with respect to two processors.	37
Figure 10. VA17C (Landsat TM/ETM+ path 17/row 34, bands 4, 3, and 2 shown) zoomed to a subset of interest.	43
Figure 11. VA17C PCA (Landsat TM/ETM+ path 17/row 34, bands 1, 2, and 3 shown) zoomed to a subset of interest.	44
Figure 12. VA17C SVD (Landsat TM/ETM+ path 17/row 34, bands 1, 2, and 3 shown) zoomed to a subset of interest.	44
Figure 13. Locations of Virginia, VA17C (Landsat TM/ETM+ path 17/row 34), and zoomed subset shown in Figures 10—12 in relation to each other.	45
Figure 14. AM232 (Landsat TM path 232/row 67, bands 4, 3, and 2 of image acquired in 1998 shown) zoomed to a subset of interest.	46

Figure 15. AM232 PCA (Landsat TM path 232/row 67, bands 1, 2, and 3 shown) zoomed to a subset of interest.	46
Figure 16. AM232 SVD (Landsat TM path 232/row 67, bands 1, 2, and 3 shown) zoomed to a subset of interest.	47
Figure 17. Locations of Rondônia, AM232 (Landsat TM path 232/row 67), and zoomed subset shown in Figures 14—16 in relation to each other.	48
Figure 18. VA17C DR (Landsat TM/ETM+ path 17/row 34) classification, green = forest, tan = non-forest.	50
Figure 19. VA17C SVD DR (Landsat TM/ETM+ path 17/row 34) classification, green = forest, tan = non-forest.	50
Figure 20. IS+ classification accuracy for AM232.	52
Figure 21. AM232 (Landsat TM path 232/row 67) PCA IS+ classification (6 bands), green = forest, tan = non-forest.	53
Figure 22. AM232 (Landsat TM path 232/row 67) SVD IS+ classification (6 bands), green = forest, tan = non-forest.	53
Figure 23. AM232 (Landsat TM path 232/row 67) PCA IS+ classification (9 bands), green = forest, tan = non-forest.	54
Figure 24. AM232 (Landsat TM path 232/row 67) SVD IS+ classification (9 bands), green = forest, tan = non-forest.	54
Figure 25. Execution time for AM232.	55
Figure 26. Number of iterations for AM232.	55
Figure 27. Number of pure classes for AM232.	56

Chapter 1: INTRODUCTION

As remote sensing datasets continue to increase in size, there is a demonstrated need for faster computing resources to decrease processing time. Furthermore, when dealing with certain classification algorithms, more accurate results may be obtained by using slightly different input parameters, such as the number of clusters for K -means [20]. A significantly faster (parallel) implementation of these classification algorithms would allow an analyst to make several runs using different parameters in the equivalent time required to make one serial run, potentially producing more accurate classification results. Although there are increasingly more parallel computers available to the research community, porting existing serial applications to a parallel environment is usually nontrivial.

In the remote sensing and image processing discipline, large data volumes and slow processor speeds have necessitated feature reduction. Since the cost of classifications is dependent upon the number of discriminating features (bands) associated with each pixel in multispectral space, it is desirable to reduce the number of features in a dataset [43]. Even as processing speeds increase due to faster computers and better algorithms, such as the parallel iterative guided spectral class rejection (pIGSCR) classification algorithm [38], the need for feature reduction methods remains as modern sensors increase in sensitivity.

This thesis discusses specific changes that are made to the IGSCR (iterative guided spectral class rejection) classification algorithm to produce a shared memory parallel algorithm (pIGSCR) with accompanying pseudocode for the classification algorithms [49][36]. A further goal of this implementation is to create source code that is both portable and open source. The final pIGSCR code runs on multiple hardware platforms and operating systems, and it does not have the same “black box” that is associated with commercial software libraries. A further goal of this work is to demonstrate the utility of the pIGSCR implementation by accurately and efficiently classifying Landsat data covering the state of Virginia into forest and non-forest land use informational classes.

This thesis also presents a feature reduction method using the singular value decomposition (SVD). The SVD emerges as an ideal candidate for feature reduction of remotely sensed images due to inherent collinearity of the brightness value vectors in geographic data. The SVD is applied in a new way, drastically reducing the computer processing time and memory requirements and therefore making the SVD feasible for feature reduction in large datasets. This work examines various feature reduction methods for the pIGSCR classification algorithm, and demonstrates that the proposed SVD method significantly decreases classification execution times while not negatively affecting classification accuracy, and the SVD outperforms some other commonly used feature reduction methods such as principal components analysis. The SVD has potential for improving classification accuracies as too many features for a given training set can reduce classification accuracy [24], and the SVD can enable the removal of noise (corresponding to small singular values) similar to the Fourier Transform.

Chapter 2: LITERATURE REVIEW

2.1 IGSCR

Unsupervised classification is a process by which all pixels or objects with similar spectral values (spectral classes) are identified (clustering) and then subsequently labeled with respect to informational classes (labeling). Supervised classification, in contrast, requires analyst identification of the spectral classes within each informational class beforehand (training). Remaining pixels or objects are then assigned to a spectral class using a decision rule (classification). As with unsupervised classification, the resulting map must be labeled with respect to informational classes, but for supervised classifications this is trivial since the informational class to which each spectral class belongs was identified in the training stage.

IGSCR is an example of a hybrid classification method, a classification method that exhibits characteristics of both unsupervised and supervised classification [43]. Hybrid classification methods combine multiple classifiers to reduce the workload of the human analyst, most often in the training phase. Bruzzone and Prieto [9] use unsupervised classification (clustering) to modify the spectral signatures generated from a supervised classification so the same training data can be used on images of the same landscape acquired on different dates. Byeungwoo and Landgrebe [10] use a hybrid approach to create a one class classification where the analyst need only train for the class of interest and then unsupervised classification is used to generate signatures for a supervised classification of the original image. Guided clustering requires a user to select training data to represent predefined informational classes, automatically identifies spectral classes (clusters of pixels with similar brightness value vectors) within each informational class (category, such as deciduous forest or row crops) using a clustering algorithm, and then uses the resulting spectral class signatures to perform a supervised classification [5]. This method is advantageous because accurate results are produced while allowing for a greater amount of automation. Guided clustering cannot be entirely automated, however, as user interaction is required after the training process to oversee spectral class creation and refine parameters. IGSCR is more disposed to automation as no user interaction is required after the training phase [49]. IGSCR uses a process called “cluster busting” first introduced by Jensen et al. [25] to refine spectral classes iteratively prior to application of a decision rule. Each spectral class produced by clustering is assigned the value of the majority informational class if that spectral class is statistically determined to be sufficiently pure. In practice, IGSCR-derived area estimates were shown to exceed established precision standards in the USDA Forest Service Forest Inventory and Analysis (FIA) program [36]. IGSCR was also used recently for a study on Sudden Oak Death where Kelly et al. [30] demonstrated that the IGSCR hybrid classification method outperformed both supervised and unsupervised methods alone.

2.1.1 Automation

The IGSCR algorithm requires a clustering algorithm and a means by which brightness value vectors are assigned to clusters (the decision rule), so the choice of the specific algorithms that are used may be dependent on implementations that are available. In 2003, IGSCR was implemented in the C language using a commercial remote sensing image processing package [36]. The decision rule in this prior implementation is maximum likelihood [43] and the clustering algorithm is ISODATA [4].

2.2 Parallel Computing

The first distributed memory parallel computers were both expensive and difficult to use, making them inaccessible to the general scientific community, although vector parallel computers (CDC and Cray) quickly became the norm in scientific computing [41]. The emergence of commodity clusters has reduced the startup cost required to build a very powerful computer containing thousands of processors. Even specialized shared memory multicomputers containing a modest number of processors have become a viable, affordable option for scientific researchers.

To coincide with more publicly available hardware solutions, two software solutions have emerged as standards for programming on their respective architectures. MPI (Message Passing Interface) is a standard describing directives that extend an already existing programming language such as C or Fortran [45]. MPI is the protocol of choice for a distributed memory parallel computer because communication between processors using MPI is done explicitly.

For shared memory programming, OpenMP is the standard that is commonly used [27]. OpenMP uses a fork/join model of parallelization and is an extension to an existing programming language comprised of compiler directives and function calls [27]. OpenMP directives are formed such that the underlying serial code is not perturbed and will therefore still compile without using OpenMP. Using OpenMP to parallelize applications on a shared memory computer allows a programmer to specify serial and parallel portions of code while hiding communication and memory access details.

2.2.1 Comparison of OpenMP and MPI

Although MPI can be used on a shared memory computer, it is more advantageous to develop parallel code using OpenMP on a shared memory platform. First, OpenMP does not require explicit communication between processors because all processors have access to one large global memory. Data transfer is completely contained in the underlying memory management hardware and is not exposed to the programmer. A second advantage resulting from a single memory space is having the luxury of developing parallel code incrementally from the serial code [41]. For example, starting with a serial program it is possible to

separate different parts of code into blocks and parallelize one block at a time. With the fork/join model, the benefit of partial parallelization is realized using multiple processors while the serial portions of code are run using just one processor. The main disadvantage associated with OpenMP is that it is effectively limited to shared memory programming and is not (yet) suitable on a distributed memory platform. If the speedup (execution time on multiple processors divided by the best execution time on one processor) required is much larger than the number of processors available on a shared memory machine, a large distributed memory supercomputer and MPI should be used.

2.3 Feature Selection

Traditional methods of feature selection include the use of separability indices such as divergence, the Jeffries Matusita Distance, and transformed divergence [43]. A set of spectral classes are analyzed to determine which combinations of bands will result in the greatest separability (greatest distinction between classes), and only those bands are used for the ensuing classification. Another popular approach is that of feature reduction, where the image is transformed to a new coordinate system requiring (hopefully) fewer bands to accurately represent the image. Most feature reduction methods do not require analysis of training data, making this an attractive option for a classification method that does not require training, such as unsupervised classification methods and some hybrid classification methods. Standard feature reduction methods include principal components analysis (PCA) (also called Karhunen-Loeve analysis) [26], maximum noise fraction (MNF, also called minimum noise fraction) [19], canonical analysis [16], and the Kauth-Thomas tasseled cap transformation [29]. The tasseled cap transformation has a fixed axis and is therefore somewhat confined in its application. The other feature reduction methods mentioned are transformations that align the data along axes of decreasing variance, and the resulting low order de-correlated bands are sufficient to perform a classification in many cases. However, Lowitz [34] and Chang [12] have shown that sometimes the higher order components resulting from such transformations are necessary to differentiate between classes in a classification. Also, the axes generated using PCA may not allow for accurate classification of the data using fewer dimensions while a different set of axes exists that will allow for class discrimination using fewer dimensions, the premise upon which canonical analysis is based [43]. Furthermore, rather than directly revealing the rank and basis of the data from the data itself, these methods attempt to reveal these attributes indirectly from a summary of the data, such as the covariance matrix. This explains why the resulting PCA data has full rank (no reduction is possible), but it is still possible that a different alignment of the data will result in a feature reduction.

A mathematical construct that directly reveals the rank and corresponding ideal basis of a dataset is the singular value decomposition (SVD). For a dataset in n -dimensional

space, for any $k < n$, the SVD will show the ideal basis for representing that data using only k dimensions, as will be explained in a later chapter. If the SVD reveals that the dataset is full rank and no feature reduction is possible along the calculated axes, then no axes exist for which a reduction is possible. The SVD reveals the ideal subspace for the data based on the entire dataset, while axes transformations such as PCA attempt to do so with a limited summary of the data (i.e., the covariance matrix). k -dimensional subspaces (for any specific k) revealed by other constructs will never be mathematically closer to the original subspace than the k -dimensional subspace revealed by the SVD.

In remote sensing applications, the SVD is a popular alternative factorization to QR factorization for solving least squares problems [35][13]. The use of the SVD as a feature reduction tool has been limited in remote sensing as the storage and processing are expensive, especially for large datasets such as entire images [14][21]. In the discipline of chemistry, van den Broek et al. [47] used the SVD to reveal the rank and reduce the data dimension of multivariate images.

Chapter 3: CLASSIFICATION ALGORITHMS

3.1 K -means

K -means clusters N data points in E^b (real b -dimensional Euclidean space, all b -tuples of real numbers) into K different clusters such that the sum of the distances between each cluster mean and the data points belonging to the cluster cannot be reduced by reassigning any of the data points to any of the existing clusters (Hartigan, 1975). The algorithm begins with K initialized cluster mean points $m^{(1)}, \dots, m^{(K)} \in E^b$, and each data point $x^{(i)} \in E^b$ is assigned to the nearest cluster mean according to a distance measure such as Euclidean distance,

$$\|x - m\|_2 = \sqrt{\sum_{j=1}^b (x_j - m_j)^2}.$$

Once all data points have been assigned, cluster means are recomputed by

$$m^{(k)} = \frac{1}{N_k} \sum_{i \in I_k} x^{(i)}$$

where $m^{(k)}$ is the new mean for cluster k , I_k is the set of indices of points assigned to cluster k , and $N_k = |I_k|$. The index sets I_1, \dots, I_K constitute a partition of $\{1, \dots, N\}$, and are assumed to be nonempty. The assign and recompute process iterates until there is no significant change in cluster assignment, determined by a user set threshold of number of data values reassigned or total distance the means migrated between iterations.

K -means will converge in a finite number of steps to a local minimum point of the objective function

$$\sum_{k=1}^K \sum_{i \in I_k} \|x^{(i)} - m^{(k)}\|_2^2.$$

There is no guarantee or expectation that this local minimum point is equal to or even close to the global minimum point (Hartigan, 1975). There have been several methods proposed for initializing the cluster means in order to produce a good clustering, which will be discussed in a later chapter. The K -means algorithm is given below.

Algorithm Kmeans(its, x, threshold, K, cluster, sigs)

Input: *its* (maximum number of iterations)

x (three-dimensional image)

threshold (iteration termination criteria)

K (number of clusters)

Output: *cluster* (two-dimensional cluster assignment array)

```

for  $x$ )
sigs (cluster signatures)
begin
  call Initialize_Means( $x, K, m$ ); (where  $m^{(1)}, \dots, m^{(K)}$ 
are the means for clusters  $1, \dots, K$ )
  for  $it := 1$  step 1 until  $its$  do
    begin
      for  $k := 1$  step 1 until  $K$  do
        begin
           $sum^{(k)} := 0$ ;
           $n_k := 0$ ;
        end
       $pixels\_changed := 0$ ; (initialize number of pixels
that change assignment
to zero)
      for  $i := 1$  step 1 until  $rows$  do
        for  $j := 1$  step 1 until  $cols$  do
          begin
             $min\_distance := large\_number$ ; (initialize
minimum distance squared)
             $min\_c := 0$ ; (nearest cluster index)
            for  $k := 1$  step 1 until  $K$  do
              begin
                 $distance := \sum_{p=1}^b (x_p^{(i,j)} - m_p^{(k)})^2$ ;
                if ( $distance < min\_distance$ ) then
                  begin
                     $min\_c := k$ ;
                     $min\_distance := distance$ ;
                  end
                end
               $sum^{(min\_c)} := sum^{(min\_c)} + x^{(i,j)}$ ;
               $n_{min\_c} := n_{min\_c} + 1$ ;
              if ( $min\_c \neq cluster_{i,j}$ ) then
                begin
                   $cluster_{i,j} := min\_c$ ;
                   $pixels\_changed := pixels\_changed + 1$ ;
                end
              end
            end
          end
        end
      end
    end
  end
end

```

```

for  $k := 1$  step 1 until  $K$  do
  if ( $n_k \neq 0$ ) then
     $m^{(k)} := \text{sum}^{(k)} / n_k$ ;
  else
    Delete Cluster  $k$ ;
  if ( $\text{pixels\_changed} / \text{no\_of\_pixels} < \text{threshold}$ ) then
    exit loop;
  end
  compute class statistics for cluster signatures  $\text{sigs}$ ;
end

```

3.1.1 Implementation

Because of the vector operations that are implicit with an implementation of K -means, there are numerous opportunities to exploit the use of Fortran 95 intrinsic functions for both serial and parallel implementations [18]. During an iteration of K -means where each pixel is being assigned to a cluster, the cluster assignment can be stored in an array with dimensions equal to that of the two-dimensional image. Element (i, j) of the cluster assignment array would correspond to the (i, j) element, a brightness vector, of the image array. This classification array is not only the output classification image for K -means, but it is also a mechanism for elegantly computing certain cluster statistics after the iteration terminates. A running tally of the sum and number of data values for each cluster is computed during each iteration in order to compute each cluster mean upon completion of cluster assignment for the entire image. Since the operator $'/'$ is defined for arrays in Fortran 95, all the cluster means are elegantly computed by dividing the sum vector by the previously calculated number of data values $N(k)$ via

$$\text{mean}(1:b, k) = \text{sum}(1:b, k) / N(k),$$

where **mean** is a two-dimensional array containing mean vectors for each cluster, **sum** is the sum of the data values belonging to cluster k , and b is the number of bands in the image.

Once the iteration has converged, there are additional statistics calculated as part of the output signature of each cluster. These include minimum and maximum vectors, and a covariance matrix for each cluster. The covariance matrix calculation will be described in greater detail in a later chapter. In order to calculate the minimum and maximum, an array containing only data values belonging to a specific cluster is created using the Fortran 95 function **PACK**. The cluster assignment array may be used to form a masked array that will be input into the function **PACK** to produce a one-dimensional array that can be input into an array intrinsic function. For example, for a specific cluster and a specific band,

the minimum value is found by masking out all values not associated with the cluster of interest from the two-dimensional array of data values for the band of interest, and then the resulting array is the input to `PACK`. The resulting output is a one-dimensional array containing all data values belonging to the cluster of interest for one band, and this will be the input array for the Fortran 95 intrinsic function `MINVAL`, which will return the minimum value. An example of the use of the above functions is

```
pArray(1:N(k)) = PACK(class(1:r, 1:c, i), class(1:r,1:c, i).eq.k)
min(i) = MINVAL(pArray(1:N(k))),
```

where `min` is a one-dimensional array of length b , `pArray` is the one-dimensional result of the call to `PACK`, `i` is the band of interest, `N(k)` is the number of data values attributed to the cluster k for which the statistics are being computed, and `r` and `c` are the number of rows and columns, respectively. The maximum is found using the Fortran 95 intrinsic function `MAXVAL`, which is analogous to the function `MINVAL` described above.

3.2 *K*-means Initialization

Principal components analysis (PCA) or Karhunen-Loeve analysis transforms a highly correlated dataset into an uncorrelated dataset where the first principal component (PC1) accounts for the most variance in the original multi-dimensional dataset [26]. PCA is often used to reduce the dimensionality of a dataset without losing much information [26].

There are numerous ways to determine initialized mean placement for *K*-means, including randomly seeding the means [20]. As an improvement, Hartigan [20] suggests that the means be seeded along the axis of greatest variance (PC1), and Huang [23] showed that placing the initial means plus or minus one standard deviation from the mean results in higher classification accuracies. The algorithm for *K*-means initialization using the first principal component axis to seed the cluster means is given below. For a matrix A , A_i denotes the i th row, $A_{\cdot i}$ denotes the i th column, and A^t is the transpose.

Algorithm Initialize_Means(x , K , m)

Input: x (three-dimensional image)

K (number of cluster means to initialize)

Input/Output: m (K cluster mean vectors)

begin

$$\Sigma := \frac{1}{rows * cols - 1} \sum_{i=1}^{rows} \sum_{j=1}^{cols} (x^{(i,j)} - m)(x^{(i,j)} - m)^t$$

(where Σ is the covariance matrix and $rows$ and $cols$ are the number of rows and columns in x)

factor $\Sigma = PDP^t$ (where D is a diagonal matrix containing the eigenvalues of Σ and P contains the corresponding orthonormal eigenvectors of Σ)

```

for  $i := 1$  step 1 until  $rows$  do
  for  $j := 1$  step 1 until  $cols$  do
     $PC1_{i,j,1} := P_{\cdot 1}^t x^{(i,j)}$ ; (where  $PC1$  is the principal
    component image)
   $\mu_{PC1} := P_{\cdot 1} 1^t m$ 
   $\sigma_{PC1} := D_{1,1}$  (The variance is the largest eigenvalue
  distribute  $\mu_1, \dots, \mu_K$  evenly between  $\mu_{PC1} - \sigma_{PC1}$ 
  and  $\mu_{PC1} + \sigma_{PC1}$ ;
  for  $i := 1$  step 1 until  $rows$  do
    for  $j := 1$  step 1 until  $cols$  do
       $class_{i,j} := k$  where  $|\mu_k - PC1_{i,j,1}| =$ 
       $\min_{l \in K} |\mu_l - PC1_{i,j,1}|$ ;
    recompute class means;
  end
end

```

3.2.1 Implementation

The PCA calculation first requires a covariance matrix

$$\Sigma = \frac{1}{N-1} \sum_{i=1}^N (x^{(i)} - m)(x^{(i)} - m)^t$$

where $x^{(1)}, \dots, x^{(N)} \in E^b$ are b -dimensional data points, m is the sample mean, and N is the total number of data points. This formula for the covariance matrix requires a previously calculated mean and would require two iterations through the data. Consider the i, j element of $(N-1)\Sigma$,

$$\begin{aligned}
(N-1)\Sigma_{i,j} &= \sum_{k=1}^N (x_i^{(k)} - m_i)(x_j^{(k)} - m_j) \\
&= \sum_{k=1}^N x_i^{(k)} x_j^{(k)} - x_i^{(k)} m_j - x_j^{(k)} m_i + m_i m_j \\
&= A_{i,j} - s_i m_j - s_j m_i + N m_i m_j,
\end{aligned}$$

where

$$s = \sum_{k=1}^N x^{(k)},$$

$$A = \sum_{k=1}^N x^{(k)} x^{(k)t},$$

and

$$m = \frac{s}{N}$$

can be calculated in one iteration through the data points $x^{(1)}, \dots, x^{(N)}$. To save computation time and storage space, only half of Σ and A need be stored and calculated as they are both symmetric. To calculate the eigenvalues $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_b \geq 0$ and corresponding eigenvectors $v^{(1)}, \dots, v^{(b)}$, where b is the number of bands of the data values $x^{(1)}, \dots, x^{(N)}$, only one LAPACK routine **SSYEV** (or its double precision equivalent **DSYEV**) is required [2]. Once these are calculated, the brightness vectors $x^{(1)}, \dots, x^{(N)}$ and the mean m may be transformed to PC1 using the transformation

$$PC1_k = v^{(1)t} x^{(k)}, \quad \mu_{PC1} = v^{(1)t} m, \quad k = 1, \dots, N,$$

where $v^{(1)}$ is the eigenvector corresponding to the largest eigenvalue λ_1 . Once the mean and all brightness values are transformed to PC1, K cluster means are initialized equally spaced in the interval $[\mu_{PC1} - \lambda_1, \mu_{PC1} + \lambda_1]$, and each brightness vector is assigned to the nearest cluster based on its first principal component. Once all vectors have been assigned to a cluster, each cluster mean is calculated using each pixel's non-transformed brightness vector.

3.3 Maximum Likelihood

Maximum likelihood is a highly used decision rule [43]. The algorithm determines the class to which a pixel belongs based on the probability that it belongs to each of the input training classes $\omega_1, \dots, \omega_t$. The following derivation of the maximum likelihood decision rule is taken from [43]. Maximum likelihood is based on the Bayes decision rule where a point $x \in \omega_i$ if $p(\omega_i|x) > p(\omega_j|x)$ for all $j \neq i$. The probabilities $p(\omega_1|x), \dots, p(\omega_t|x)$ are rarely known at classification time, but through training $p(x|\omega_i)$ can be estimated, and $p(\omega_i|x) = p(x|\omega_i)p(\omega_i)/p(x)$. For a pixel x , removing the common term $p(x)$:

$$x \in \omega_i \text{ if } p(x|\omega_i)p(\omega_i) > p(x|\omega_j)p(\omega_j) \text{ for all } j \neq i.$$

Because a logarithm function is monotonically increasing, it follows that

$$\ln(p(x|\omega_i)p(\omega_i)) > \ln(p(x|\omega_j)p(\omega_j))$$

for all $j \neq i$ implies $p(x|\omega_i)p(\omega_i) > p(x|\omega_j)p(\omega_j)$ for all $j \neq i$ (Duda and Hart, 1973). Assume a Gaussian distribution for the training sets,

$$p(x|\omega_i) = (2\pi)^{-\frac{b}{2}} |\Sigma_i|^{-1/2} e^{-\frac{1}{2}(x-m^{(i)})^t \Sigma_i^{-1} (x-m^{(i)})},$$

where b is the number of bands, Σ_i is the covariance matrix for class i , $m^{(i)}$ is the mean for class i , and $|\Sigma_i|$ is the determinant of the covariance matrix. It is assumed that $|\Sigma_i| \neq 0$. Since $\ln(p(x|\omega_i)p(\omega_i))$ is being maximized, any term not dependent on i can be removed, and if no prior knowledge of $p(\omega_i)$ is known, equal prior probabilities are assumed for all classes. After removing non-discriminating terms, the function to be maximized (with respect to i) is

$$g_i(x) = -\ln |\Sigma_i| - (x - m^{(i)})^t \Sigma_i^{-1} (x - m^{(i)}).$$

The maximum likelihood algorithm is given below.

Algorithm Maximum_Likelihood(classes, x, class)

Input: *classes* (class signatures containing covariance matrix and mean)

x (three-dimensional image)

Output: *class* (classification of x)

begin

for $k := 1$ **step** 1 **until** $|classes|$ **do**

begin

 diagonalize Σ_k to produce PDP^t ; (Σ_k is the covariance matrix for class k , D is a diagonal matrix containing the eigenvalues of Σ_k , and P contains orthonormal eigenvectors corresponding to D)

$|\Sigma_k| := \prod_{j=1}^b (D_{jj})$; (calculate determinant of Σ_k where b is the number of bands)

$\Sigma_k^{-1} := PD^{-1}P^t$

end

for $i := 1$ **step** 1 **until** *rows* **do**

for $j := 1$ **step** 1 **until** *cols* **do**

begin

$max_G := -large_number$; ($large_number$ is the largest possible value of max_G)

$max_class := 0$; (the highest probability index)

for $k := 1$ **step** 1 **until** $|classes|$ **do**

begin

$g := -\ln |\Sigma_k| -$
 $(x^{(i,j)} - m^{(k)})^t \Sigma_k^{-1} (x^{(i,j)} - m^{(k)})$;

if ($g > max_G$) **then**

begin

```

        max_G := g;
        max_class := k;
    end
end
classi,j := max_class;
end
end

```

3.3.1 Implementation

In this maximum likelihood implementation, it is assumed that mean vectors and covariance matrices will be input as part of spectral class signatures. The goal of this implementation is to efficiently and portably code the above decision rule.

A quick glance at the above equation reveals that the most costly computations are calculating the determinant and inverse of the covariance matrix. Because the covariance matrix is symmetric, it may be factored as $\Sigma = PDP^{-1}$, where the columns of P are the orthonormal eigenvectors $u^{(1)}, \dots, u^{(b)}$ and D is a diagonal matrix containing eigenvalues $\lambda_1, \dots, \lambda_b$, corresponding to the eigenvectors [33]. The determinant of Σ is the determinant of D , and the determinant of a triangular matrix is the product of the elements of the diagonal [33]. To calculate the inverse of Σ , the inverse of PDP^{-1} may be calculated as

$$(PDP^{-1})^{-1} = (P^{-1})^{-1}(D)^{-1}(P)^{-1} = PD^{-1}P^{-1}.$$

$P^{-1} = P^t$ because the eigenvectors are orthonormal and D^{-1} is trivial to calculate because only the inverse of each scalar term in the matrix diagonal need be calculated.

In order to implement this efficiently, the LAPACK function `SSYEV` (`DSYEV` for double precision) is used to calculate the eigenvectors and eigenvalues of Σ [2]. After determining P and D , Fortran 95 intrinsic functions `MATMUL` and `TRANPOSE` maybe be used to complete the inverse calculation.

3.4 IGSCR

The algorithm begins with the user inputting an image to be classified, training data specific to that image, and various parameters for the unsupervised classification and the spectral class homogeneity test. K -means clustering is performed on the input image, and the resulting spectral classes are analyzed for informational class homogeneity to ensure that only one informational class is present (with high probability) in each spectral class. If more than one informational class is present with significant probability, that spectral class will be rejected, and all pixels belonging to that class will have the opportunity to be reclustered. In order to conduct the homogeneity test, the training data must be analyzed to determine

how many points from each informational class fall into each spectral class. For any training point, t_i , the point's coordinates in the original image are used to determine which spectral class it was assigned to in the unsupervised classification. Each spectral class has an informational class count vector that has a length equal to the number of informational classes, and the component of the vector that corresponds to the informational class that was associated to t_i (by the analyst in the training phase) is incremented. After this process has been applied to each training point, the homogeneity test is conducted for each spectral class. The homogeneity test is given by Musy et al. [36] as

$$N_k(1 - p_0) \geq 5 \text{ (spectral class must have minimum number of pixels) and } Z > Z(\alpha) \text{ (homogeneity test),}$$

where

$$\begin{aligned} Z &= (\hat{p} - p_0 - 0.5/N_k) / \sqrt{p_0(1 - p_0)/N_k}, \\ \alpha &\text{ is the type-I error rate,} \\ Z(\alpha) &\text{ is the value such that } P(Z \geq Z(\alpha)) = \alpha \text{ for a } N(0, 1) \text{ variable } Z, \\ \hat{p} &= N_{maj}/N_k, \\ p_0 &\text{ is the user input threshold,} \\ N_k &\text{ is the total number of training pixels in the spectral class, and} \\ N_{maj} &\text{ is the majority informational class count (count of training pixels belonging to the informational class with the highest count).} \end{aligned}$$

If a spectral class is determined to be pure ($Z > Z(\alpha)$), then the majority informational class is assigned to the spectral class, and each pixel that was assigned to the spectral class by K -means clustering is recoded to the informational class value in the output classification image. Furthermore, each pixel belonging to a pure spectral class is removed from the input image for successive applications of K -means clustering. The pure spectral class and its signature (a set of statistics for the class such as mean and covariance among bands) are kept for later processing. After all spectral classes have been processed, K -means clustering (with the same value of K that was used in each previous iteration) is used again on the remaining pixels and the homogeneity test ensues. This iteration continues until no pixels belonging to impure classes remain in the input image, no pure classes were found during the previous iteration, or a maximum number of iterations has occurred. In this manner, all pixels belonging to impure spectral classes are reclustered during the subsequent iteration and each iteration operates on a proper subset of the pixels used in the previous iteration.

Once the iteration terminates, there is a set of pure spectral classes, an unsupervised classification image recoded to informational class values with zeroes for background and unclassified pixels, a set of unclassified pixels remaining from the original image, and the original image. There are three options for output classification images at this stage, and any or all of them may be selected by the user. First, a maximum likelihood classification may be run on the original image using the pure spectral classes as training classes, producing a Decision Rule (DR) image. Second, all unclassified pixels may be recoded to a reserved value designating unclassified pixels (the number of informational classes plus one) and added to the unsupervised classification image to produce an Iterative Stacked (IS) image. Finally a maximum likelihood classification may be performed on only the unclassified pixels (using the pure spectral classes as training classes), and the resulting classification may be added to the unsupervised classification image to produce a Iterative Stacked plus (IS+) image. The IGSCR algorithm is given below.

Algorithm IGSCR(image, P, DR, IS, IS+, α , p_0 , its, classes, kits, threshold, sigs, DR_image, IS_image, IS+_image)

Input: *image* (three-dimensional image)

T (set of training data containing x,y coords and an informational class value)

DR, IS, IS+ (Boolean values corr. to output)

α (type-I error rate)

p_0 (homogeneity threshold)

its (number of iterations for main loop)

classes (number of classes to create in *K*-means loop)

kits (number of *K*-means iterations)

threshold (loop-ending criteria for *K*-means)

Output: *sigs* (set of pure signatures)

DR_image (image resulting from maximum likelihood on input image)

IS+_image (image resulting from maximum likelihood on impure pixels)

IS_image (image resulting from impure pixels being recoded in unsupervised image)

begin

k_image := *image* (*k_image* is input to each *K*-means call)

for *i* := 1 **step** 1 **until** *its* **do**

begin

```

if (all pixels in k_image are zero) then
    exit loop;
    call Kmeans(kits, k_image, threshold, classes,
    k_classimage, ksigs);
    reset spectral class counts;
    pure_classes := 0; (number of pure classes)
for j := 1 step 1 until |T| do
    begin
        c := k_classimage( $T_x^{(j)}$ ,  $T_y^{(j)}$ );
        if (c ≠ 0) then
            increment the informational class count for
            class of  $P^{(j)}$  and total count within spectral
            class c;
        end
for j = 1 step 1 until |ksigs| do
    begin
        if (totalj(1 − p0)) ≥ 5) then (where totalj
        represents the count of pixels belonging to j)
            begin
                Determine Nmaj and Cmaj where Nmaj is
                the no. of pixels in maj. info. class, Cmaj
                 $\hat{p} := \frac{N_{maj}}{total_j}$ ;
                 $Z := \frac{(\hat{p} - p_0 - .5/total_j)}{\sqrt{p_0(1-p_0)/total_j}}$ ;
                if (Z > Z( $\alpha$ )) then
                    begin
                        increment pure_classes;
                        add ksig(j) to group of sigs;
                        mask j from k_image;
                        recode where k_classimage = j to
                        Cmaj in class_image;
                        (where class_image is the stacked
                        output of the K-means classifications)
                    end
                end
            end
        end
    if (pure_classes = 0) then
        exit loop;

```

```

    end
  if (DR) then
    call MaximumLikelihood(sigs, image, DR_image);
  if (IS+) then
    begin
      call MaximumLikelihood(sigs, k_image,
        IS+_image);
      IS+_image := IS+_image + class_image;
    end
  if (IS) then
    begin
      recode k_image non-zeros to number of informational
      classes + 1, store result in k_image;
      IS_image := k_image + class_image;
    end
  end
end

```

3.4.1 Implementation

The biggest opportunities to exploit the intrinsic functions in Fortran 95 are in the masking and recoding phase. Without vector operations, it would be necessary to loop over each pixel and test the pixel's class value explicitly. Based on this test, the pixel's brightness vector from the input image would either be masked out of the input image or recoded to its informational class value in the output classification image. Using the Fortran 95 intrinsic function `WHERE` with an array mask, an array can be used as a Boolean test to determine whether an operation should be performed on corresponding elements of another array of equal dimensions. An example of using the `WHERE` statement to recode to informational class values is

```

WHERE(class.eq.k)
  recodeImage = classValue(k)
end WHERE,

```

where the dimensions of `class` and `recodeImage` are equal. A similar block is used to mask the pure pixels out of the input image. Another subtle opportunity for improved performance with vector statements arises where it is necessary to use matrix addition to add two images together, such as when the result of a maximum likelihood classification is added to the stacked result of the iterative unsupervised classifications.

Chapter 4: HIERARCHICAL DATA FORMAT 5 (HDF5)

One goal of this implementation is to achieve portability by removing dependencies on specific platforms and software, which would limit the use of the proprietary data format that was used in the previous implementation. The HDF5 data format and API library was chosen because it is flexible, robust, and already widely used in the scientific community [22]. HDF5 can be used on a variety of operating systems with many C and Fortran compilers [22].

HDF5 is a standard that defines a grouping of large scientific datasets and associated properties stored on disk, and it is an API that implements the standard in order to provide efficiency and ease to a programmer. There are various objects defined in HDF5 that can be grouped together in a conceptual tree structure within an HDF5 file. Reading and writing of each object is done through a series of function calls. Table 1 describes all of the HDF5 objects that are used in the implementation of pIGSCR. These objects are all that is necessary to describe the files that are input into and output from pIGSCR.

Table 1 HDF5 Objects

file	container to store file objects
group	acts like unix directory to form hierarchy of objects within file
dataset	large array objects used to store multi-dimensional data
dataspace	describes dimensionality of a dataset
datatype	can be intrinsic or derived type; description of data in a dataset
attribute	generally a small dataset (can be scalar) used to describe metadata for a user-defined property of a group, dataset, or a named datatype

4.1 Image File

The input image file contains a three-dimensional dataset containing the image with a corresponding dataspace and datatype. The three dimensions correspond to rows, columns, and bands of the image. Additionally, there are multiple attributes used to specify information such as corner map coordinates, pixel size, and projection information for the image. This information corresponds to data that would typically be found in a header file accompanying binary data. Although this information is not used by pIGSCR, it may be useful to the analyst in a later application, and therefore it is retained. In order for pIGSCR or any program to extract the image data, it uses the HDF5 API to access the dataspace and datatype to determine the size of the image and precision of each element. This information

is used in one function call to directly read the conceptual three-dimensional image on disk into a three-dimensional array in memory. There are equivalent function calls to write an image to file, which are used to output the final two-dimensional classification images.

4.2 Training Data Files

A single training data file is input into pIGSCR. The training data is represented by a set of x, y coordinates and the corresponding informational class values for the set of points. The x and y coordinates are file indices for specific points in the input image. Each set of training points is a two-dimensional dataset where the first dimension has a length of three (for two coordinates and a class value) and the length of the second dimension is the number of points.

4.3 Signature Files

pIGSCR outputs the pure class signatures that were used for the supervised classification to an HDF5 file. Each class signature is represented by an HDF5 group with the class name as the group name. Each signature group contains the following datasets: minimum, maximum, mean, covariance matrix, and n , a scalar value representing the total number of points in the class. Each signature file has an attribute named “classes” specifying the total number of class signatures contained in the file.

4.4 Utility Programs

As stated above, training data and images are input into pIGSCR in the HDF5 format. Two utilities were written to create these two types of files. One utility converts generic binary images with header files to HDF5 image files as described above. A second utility was written to convert three-column ASCII files containing file or map coordinates and class values to one HDF5 training data file. There is an HDF5 utility called h5dump that will allow a user to examine the contents of any HDF5 file, which is useful for examining the contents of a signature file.

Chapter 5: pIGSCR

The strategy for parallelizing IGSCR is to locate operations in the original algorithm that may be run in parallel and to concentrate on those operations that will result in the greatest speedup. These operations that may be run concurrently will fall into one of two categories of parallelism, functional or data parallelism. Data parallelism, or single instruction multiple data (SIMD) parallelism, is exhibited when multiple processes perform identical operations on different members of one dataset, whereas functional parallelism, or multiple instruction multiple data (MIMD) parallelism, occurs when distinctly different operations are performed concurrently on potentially independent data [41]. An example of functional parallelism inherent in IGSCR is when three different operations may be applied to the input image to produce three independent classifications. Data parallelism is prevalent throughout IGSCR; a simple example of this is the maximum likelihood classification where all pixels are classified using identical operations. Loops over many data values are common indicators of potential data parallelism.

In order to isolate those sections of code that may be run in parallel with a modest shared memory programming adaptation, it is necessary to determine which parts of the sequential algorithm are independent. The following three conditions, known as Bernstein's conditions, guarantee that two statements S_i and S_j are independent:

$$\begin{aligned}O(S_i) \cap O(S_j) &= \emptyset, \\I(S_j) \cap O(S_i) &= \emptyset, \\I(S_i) \cap O(S_j) &= \emptyset,\end{aligned}$$

where the set of all input, output variables to statement S_i is denoted by $I(S_i)$, $O(S_i)$ respectively [7]. Simply put, two statements may be considered independent if the two statements do not write to the same variable and one statement does not have an output variable that is also an input variable to the other statement.

Once the candidate sections for parallelization are selected, it is necessary to determine whether parallelization will result in an appreciable speedup in the overall algorithm. An example of parallelism that does not best take advantage of potential processing power is the functional parallelism inherent in the multiple classifications at the end of IGSCR (maximum likelihood, stacked unsupervised classification, and stacked unsupervised with maximum likelihood), because there are at most three classification tasks and therefore this (functional) parallelism is not scalable beyond three processors. There are other cases when parallelizing a section of code does not result in a faster algorithm, such as when the overall speedup of a parallel section will be negated by overhead associated with parallel communication. Therefore it is important to locate sections of code that may be run in parallel and have a large potential speedup and high likelihood of contributing to large

speedup of the entire algorithm. A profiler is a useful tool to aid an analyst in determining which sections of code might benefit most from parallelization efforts. The output of a profiler will specify the percentage of total time spent in each function within code. By profiling a serial implementation of an algorithm, an analyst may gain insight into which portions of the code are most costly, and the analyst will consequently concentrate most parallelization effort on those sections.

Table 2 shows the approximate results of profiling a serial implementation of IGSCR. The most costly functions are listed.

Table 2 Profiling Results

Function	Percentage of total time
IGSCR	100
K -means	79.5
K -means output calculations	22
Initialize K -means	11.5
Maximum Likelihood	14

These results show that the majority of time spent running IGSCR is spent running K -means followed by maximum likelihood. A significant reduction in time spent in K -means will significantly reduce the overall time required to run IGSCR. Furthermore the profiling provides further insight into the various functions within K -means that are contributing to the large amount of processing time required. It is interesting to note that the time spent in the IGSCR code (determining class homogeneity and subsequently modifying the input and output images) is relatively small in comparison to the total time required to run its clustering and classification functions.

The most obvious opportunities for parallelization and largest performance gains in IGSCR are the loops that have a large number of predetermined independent iterations. These loops are present throughout IGSCR, most notably in the K -means clustering and maximum likelihood classification algorithms. For clustering, an operation is applied to an individual pixel to determine a cluster assignment, and the calculations for all pixels may be performed concurrently as the operation on each pixel is independent of all other pixels and resulting clustering output. The same is true for the loop to calculate spectral class statistics in maximum likelihood. Because there are far fewer spectral classes than pixels, the speedup may not be as high, but there will still be speedup as a result of calculating class statistics in parallel when many spectral classes are present (enough to compensate for parallel overhead). The shared memory parallel maximum likelihood algorithm therefore differs from the serial version when one process forks to form several processes to calculate class statistics and perform the classification of the image. The parallel maximum likelihood algorithm is given below.

Algorithm SIMD_Maximum_Likelihood(classes, x, class)

Input: *classes* (class signatures containing covariance matrix and mean)

x (three-dimensional image)

Output: *class* (classification of *x*)

begin

for $k := 1$ **step** 1 **until** $|classes|$ **fork** STATLOOP
STATLOOP:

begin

diagonalize Σ_k to produce PDP^t ; (Σ_k is the covariance matrix for class k , D is a diagonal matrix containing the eigenvalues of Σ_k , and P contains orthonormal eigenvectors corresponding to D)
 $|\Sigma_k| := \prod_{j=1}^b (D_{jj})$; (calculate determinant of Σ_k where b is the number of bands)
 $\Sigma_k^{-1} := PD^{-1}P^t$

end

join *no_of_processes*;

private $i, j, max_G, max_class, k, g$

shared *rows, cols, classes, Σ , x , m , Σ^{-1} , class*

for $i := 1$ **step** 1 **until** *rows* **fork** CLASSIFYLOOP
CLASSIFYLOOP:

for $j := 1$ **step** 1 **until** *cols* **do**

begin

$max_G := -large_number$; ($large_number$ is the largest possible value of max_G)

$max_class := 0$; (the highest probability index)

for $k := 1$ **step** 1 **until** $|classes|$ **do**

begin

$g := -\ln |\Sigma_k| -$
 $(x^{(i,j)} - m^{(k)})^t \Sigma_k^{-1} (x^{(i,j)} - m^{(k)})$;

if ($g > max_G$) **then**

begin

$max_G := g$;

$max_class := k$;

end

end

$class_{i,j} := max_class$;

```

    end
  join no_of_processes;
end

```

In K -means, there are several large loops over all pixels to consider for parallelization. The largest consumer of time is the cluster reassignment loop where each pixel is assigned to the nearest cluster and the cluster mean is recalculated for the subsequent iteration.

In the interest of having a large task granularity, it is desirable to spread the outermost loop over all processes. This reduces the overhead associated with forking multiple processes repeatedly in an inner loop. In this algorithm, the outer loop is arbitrarily over all rows, so all rows will be spread over all processes. The cluster assignment loop is ideal for parallelization because it is large and costly, and each iteration is independent. Each pixel is assigned to the closest cluster, and during a single iteration, no pixel's assignment is dependent on another pixel. At the end of the cluster assignment, the cluster means are determined through the use of running sums and counts that are tallied within the assignment loop. These are arrived at by each process maintaining local running sums and counts, and when the loop terminates all local copies are combined through addition to produce global totals. This is known as a reduction operation.

The other functions within K -means that use a significant amount of time are the functions that initialize the cluster means using principal components analysis and calculate final cluster statistics. In the mean initialization function, there are three loops that iterate over all data values in the input image. These large loops are a good focus for parallelization efforts as they are costly and independent of each other.

First the covariance calculation loop may be broken up across the rows of the image. Recall that the covariance matrix can be calculated in one pass of the image by storing various sums. These sums may be calculated in parallel using a reduction as described above. The second large loop converts each pixel to its corresponding first principal component. This is a straightforward linear transformation that is independent of other pixel values and can rather trivially be done in parallel. Finally, the last large loop assigns each pixel to a cluster based on the distance to the cluster's mean value located on the first principal axis. This loop is conceptually identical to the main K -means cluster assignment loop and can be parallelized in the same manner. The parallel algorithms for initializing the cluster means using principal components and K -means clustering are given below.

Algorithm SIMD_Initialize_Means(x , K , m)

Input: x (three-dimensional image)

K (number of cluster means to initialize)

Input/Output: m (K cluster mean vectors)

begin

in parallel: $\Sigma :=$

$$\frac{1}{rows * cols - 1} \sum_{i=1}^{rows} \sum_{j=1}^{cols} (x^{(i,j)} - mean)(x^{(i,j)} - mean)^t$$

(where Σ is the covariance matrix and *rows* and *cols* are the number of rows and columns in x)

factor $\Sigma = PDP^t$ (where D is a diagonal matrix containing the eigenvalues of Σ and P contains the corresponding orthonormal eigenvectors of Σ)

private i, j

shared $rows, cols, PC1, P$

for $i := 1$ **step** 1 **until** $rows$ **fork** PCALOOP

PCALOOP:

for $j := 1$ **step** 1 **until** $cols$ **do**

$PC1_{i,j,1} := P_{\cdot 1}^t x^{(i,j)}$; (where $PC1$ is the principal component image)

join $no_of_processes$;

$\mu_{PC1} := P_{\cdot 1}^t m$

$\sigma_{PC1} := D_{1,1}$ (The variance is the largest eigenvalue distribute μ_1, \dots, μ_K evenly between $\mu_{PC1} - \sigma_{PC1}$ and $\mu_{PC1} + \sigma_{PC1}$;

private i, j, k, l

shared $rows, cols, class, \mu, PC1$

for $i := 1$ **step** 1 **until** $rows$ **fork** ASSIGNLOOP

ASSIGNLOOP:

for $j := 1$ **step** 1 **until** $cols$ **do**

$class_{i,j} := k$ where $|\mu_k - PC1_{i,j,1}| = \min_{l \in K} |\mu_l - PC1_{i,j,1}|$;

join $no_of_processes$;

recompute class means;

end

Algorithm SIMD_Kmeans(its, x, threshold, K, cluster, sigs)

Input: *its* (maximum number of iterations)

x (three-dimensional image)

threshold (iteration termination criteria)

K (number of clusters)

Output: *cluster* (two-dimensional cluster assignment array)

```

for  $x$ )
sigs (cluster signatures)
begin
  call SIMD_Initialize_Means( $x$ ,  $K$ ,  $m$ ); (where
 $m^{(1)}, \dots, m^{(K)}$  are the means for clusters  $1, \dots, K$ )
  for  $it := 1$  step 1 until  $its$  do
    begin
      for  $k := 1$  step 1 until  $K$  do
        begin
           $sum^{(k)} := 0$ ;
           $n_k := 0$ ;
        end
       $pixels\_changed := 0$ ; (initialize number of pixels
that change assignment to zero)
      private  $i, j, min\_distance, min\_cluster, k, distance, p$ 
      shared  $rows, cols, K, b, x, m, cluster$ 
      for  $i := 1$  step 1 until  $rows$  fork ASSIGNLOOP
      ASSIGNLOOP:
        for  $j := 1$  step 1 until  $cols$  do
          begin
             $min\_distance := large\_number$ ; (initialize
minimum distance squared)
             $min\_c := 0$ ; (nearest cluster index)
            for  $k := 1$  step 1 until  $K$  do
              begin
                 $distance := \sum_{p=1}^b (x_p^{(i,j)} - m_p^{(k)})^2$ ;
                if ( $distance < min\_distance$ ) then
                  begin
                     $min\_c := k$ ;
                     $min\_distance := distance$ ;
                  end
              end
            end
            (add to local copies for later reduction)
             $sum^{(min\_c)} := sum^{(min\_c)} + x^{(i,j)}$ ;
             $n_{min\_c} := n_{min\_c} + 1$ ;
            if ( $min\_c \neq cluster_{i,j}$ ) then
              begin
                 $cluster_{i,j} := min\_c$ ;

```



```

        (add the following for later reduction)
         $pixels\_changed := pixels\_changed + 1;$ 
    end
end ( $j$  loop)
reduce  $sum, n, pixels\_changed$  across  $no\_of\_processes$ ;
join  $no\_of\_processes$ ;
for  $k := 1$  step 1 until  $K$  do
    if ( $n_k \neq 0$ ) then
         $m^{(k)} := sum^{(k)} / n_k;$ 
    else
        Delete Cluster  $k$ ;
    if ( $pixels\_changed / no\_of\_pixels < threshold$ ) then
        exit loop;
    end ( $it$  loop)
for  $i := 1$  step 1 until  $rows$  fork OUTPUTLOOP
OUTPUTLOOP:
    for  $j := 1$  step 1 until  $cols$  do
        update  $sigs_k$  for cluster  $k$  to which  $x(i, j)$  belongs
    end
end

```

The final pIGSCR algorithm follows.

Algorithm SIMD-pIGSCR(image, P, DR, IS, IS+, α , p_0 , its, classes, kits, threshold, sigs, DR_image, IS_image, IS+_image)

Input: *image* (three-dimensional image)

T (set of training data containing x,y coords and an informational class value)

DR, IS, IS+ (Boolean values corr. to output)

α (type-I error rate)

p_0 (homogeneity threshold)

its (number of iterations for main loop)

classes (number of classes to create in K -means loop)

kits (number of K -means iterations)

threshold (loop-ending criteria for K -means)

Output: *sigs* (set of pure signatures)

DR_image (image resulting from maximum likelihood on input image)

IS+_image (image resulting from maximum


```

         $C_{maj}$  in class_image in parallel;
        (where class_image is the stacked
        output of the  $K$ -means)
    end
end
end
if (pure_classes = 0) then
    exit loop;
end
if (DR) then
    call SIMD_Maximum_Likelihood(sigs, image,
    DR_image);
if (IS+) then
    begin
        call SIMD_Maximum_Likelihood(sigs, k_image,
        IS+_image);
        IS+_image := IS+_image
        + class_image;
    end
if (IS) then
    begin
        recode k_image non-zeros to number of info
        classes + 1 in parallel, store result in k_image;
        IS_image := k_image + class_image;
    end
end
end

```

5.1 Implementation Details

A good parallel algorithm will not approach its potential parallel speedup without additional consideration of the constraints of both the implementation hardware and the chosen programming paradigm. Theoretically with the shared memory paradigm, all memory is not only accessible by each process, but also equidistant. In practice this is rarely the case as physical limitations prevent such implementations. Therefore the programmer must understand the program's memory access patterns and then use the appropriate language constructs to best utilize the memory access provided by the underlying architecture.

On the SGI Altix specifically, although there is one global memory address space for all processors to access, each processor has one part of that memory that is closest and fastest to access. The methodology for efficiently using this nonuniform memory access (NUMA)

involves (local memory) programming tricks as well as tools that are separate from the program itself.

On the SGI Altix, there is a first touch principle within a program whereby the processor that first “touches” a data value will physically place that data in the memory closest to itself. For example, if after an array is allocated, one processor initializes that array, the physical location of the entire array will reside in the memory closest to that processor. If, however, each processor initializes some portion of the array, that portion will be closest to the processor that initializes it. In an OpenMP application on an SGI Altix architecture, it is important to initialize all shared variables in the same manner that each will be used, and it is important to access those variables consistently throughout the parallel application.

Outside of the application program, there are certain tools available to aid with the memory placement to facilitate good parallel performance. One of these tools specific to the SGI platform is **dplace**, a tool that binds a physical block of memory to a specific process. **Dplace** ensures that processes do not migrate from processor to processor, therefore negating any attempt within the program to keep processes close to the memory each accesses. **Dplace** can be a very useful tool when migrating processes occur during parallel runs, but all tests run on pIGSCR showed that **dplace** had virtually no effect on speed. In fact, using **dplace** would often result in slightly slower run times.

Chapter 6: pIGSCR EXPERIMENTAL RESULTS AND DISCUSSION

6.1 Data Description

pIGSCR was tested using three mosaicked Landsat Enhanced Thematic Mapper Plus (ETM+) satellite images taken from Landsat Worldwide Reference System (WRS) paths 15, 16, and 17, covering the majority of the state of Virginia, USA. These images, which will hereafter be referred to as VA15, VA16, and VA17, respectively, were obtained on April 28, 2004; May 8, 2005; and November 2, 2003, respectively. They are roughly similar in size, with VA15, VA16, VA17 being 1.1, 1.2, 0.9 gigabytes, respectively. The training data was created by the interpretation of point locations from a systematic, hexagonal grid (see Figure 1) over Virginia Base Mapping Program (VBMP) true color digital orthophotographs¹. [48] The source photography data was acquired in the Spring of 2002 at a scale of 1:4,800, and after orthorectification was resampled to a 1 meter spatial resolution. The Landsat data were used to perform a two-class classification, forest or non-forest. All three images are shown in Figures 2, 3, and 4, respectively. The map figures in this document were created using ESRI ArcGIS^(TM) 9.2.

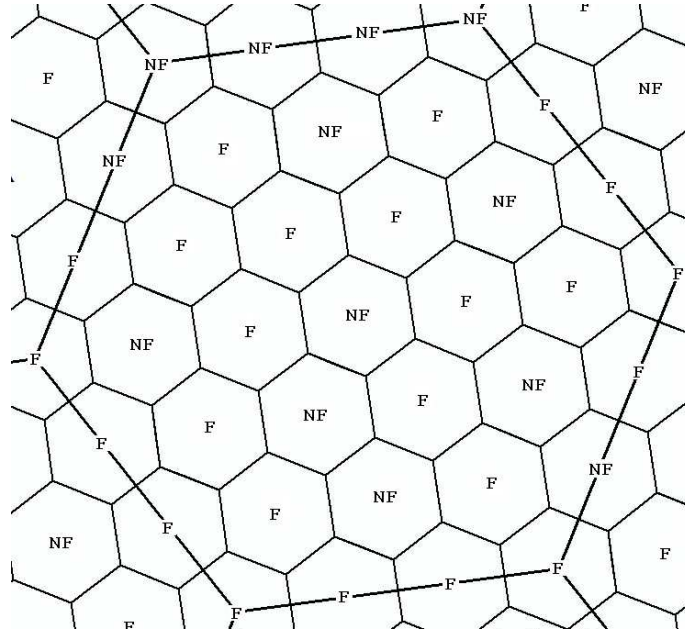


Figure 1. Hexagonal grid used for interpretation of training data.

For the purpose of accuracy assessment, a major component of the parameter selection process, validation data in the form of point locations at the center of USDA Forest Service Forest Inventory and Analysis (FIA) ground plots were used to validate the classifications [6][42]. Only homogeneous FIA plots were used (either 100 percent forest or non-forest),



Figure 2. VA15 (Landsat ETM+ path 15).

and these plots were obtained between 1997 and 2001. The lapse in years between the dates of the FIA plots and those of the classified Landsat imagery likely contributed to a lower classification accuracy in this area of rapid land cover change.

6.2 Parameter Selection

One possible benefit of using a faster algorithm is the ability to make several runs to adjust input parameters to obtain the highest possible classification accuracy. Hartigan [20] mentions that because K -means is highly sensitive to input parameters and initialization of cluster means, several different initial numbers of clusters might be used to find the best clustering. Furthermore, the homogeneity threshold determining pure spectral classes in pIGSCR may be raised or lowered, potentially affecting overall classification accuracy.

Past applications of IGSCR have demonstrated success using homogeneity thresholds of 90 or 95 percent and 100 clustering classes [30][36]. Although it is reasonable to suspect those parameters might result in a good classification in this instance, there are differences between past applications of IGSCR and this application that merit additional consideration.

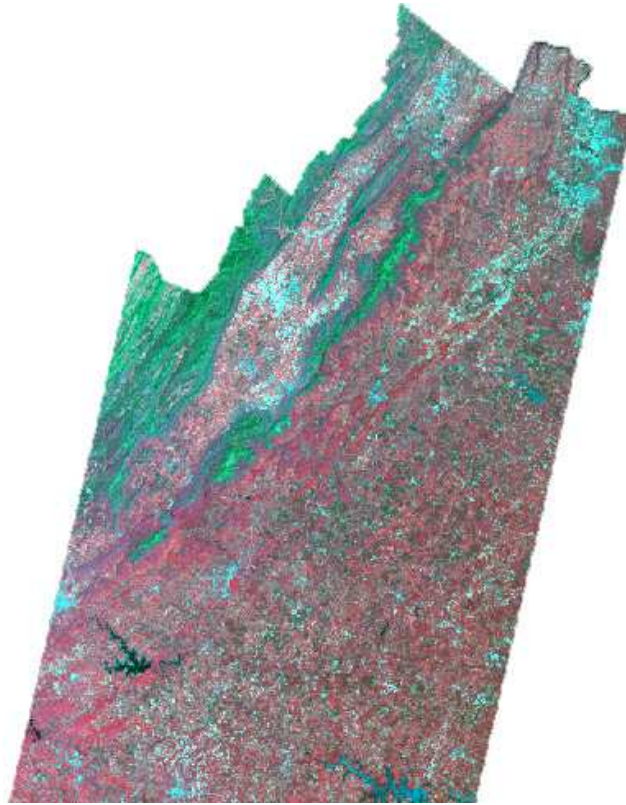


Figure 3. VA16 (Landsat ETM+ path 16).



Figure 4. VA17 (Landsat ETM+ path 17).

First, training was performed using areas of interest with spatially contiguous points in past applications. The training data used in this classification are evenly distributed points that are interpreted over a hexagonal grid placed across the Commonwealth of Virginia using high resolution 1:4,800 orthoimagery. This has resulted in fewer training points that might reduce the overall accuracy of the classification [44] but these points should be a more representative sample of the entire image. Secondly, this implementation of pIGSCR is potentially different from the previous implementation that relied on commercial closed source software libraries to perform classifications. There is no way to verify that the previous implementations of the decision rule and clustering algorithms are mathematically equivalent to the open source implementations, and the description of the clustering algorithm previously used is unclear.

Finally, independent of changes in training methodology and algorithm implementation, it is likely, although not proven, that the ideal set of parameters is dependent on the data used in the classification. If this is indeed the case, there is value in reexamining ideal input parameters for each new pairing of training data and images.

In order to determine a homogeneity threshold and number of initial K -means clusters, several exhaustive combinations within a range were used for an application of pIGSCR on VA16 and resulting accuracies were compared as shown in Table 3. The homogeneity threshold varied between 70 and 95 percent in increments of 5 percent, and the initial number of classes varied between 40 and 100 in increments of 10. The accuracies ranged between 84.4 and 88.9 percent, with the highest, 88.9 percent, resulting from using 70 classes and 70 percent homogeneity. VA16 is representative in landcover and size of VA15 and VA17, and therefore 70 classes and 70 percent were used for the purpose of benchmarking pIGSCR using those images. The resulting classified images are shown in Figures 5, 6, and 7, and the classification accuracies for VA15 and VA17 are 89.5 percent and 90.5 percent, respectively.

Table 3 Factorial Analysis for VA16

p_0	Number of Classes						
	40	50	60	70	80	90	100
95	86.7	86.6	86.5	84.8	86.3	85.4	84.4
90	86.8	87.0	84.4	88.0	85.8	85.8	87.3
85	87.4	88.6	88.1	87.4	87.5	88.5	88.0
80	88.3	87.8	88.2	87.7	87.6	87.8	88.1
75	87.5	87.9	87.0	87.4	87.4	87.8	87.9
70	87.0	88.2	88.2	88.9	88.5	87.7	88.4



Figure 5. Classification of VA15 (Landsat ETM+ path 15), green is forest and tan is non-forest.

6.3 Parallel Results

pIGSCR was tested with the previously described images using one, two, four, eight, and twelve processors of an SGI Altix 3300 with 24 GB of RAM and twelve 1.3 GHz processors and using 16, 32, and 64 processors of an SGI Altix 3700 with 64 1.6 GHz processors and 256 GB of RAM. Figures 8 and 9 show speedup as a function of number of processors when compared with the execution time for one processor and two processors, respectively. The solid black line shows ideal speedup, i.e., speedup equal to the number of processors used. The dashed line represents VA15, the dash-dot line, VA16, and the dotted line, VA17. The lines in Figure 8 are superlinear, which do not indicate much about the effect of increased processing speed on overall execution time. The drastic decrease in processing time is likely the result of additional memory available with additional processors. The lines in Figure 9 take the lack of memory available to one processor into account and therefore most probably best demonstrate parallel speedup. As the number of processors is increased, the speedup of pIGSCR increases with each of the three test images, as shown in both Figure 8 and 9, but the rate of speedup is decreasing. Table 4 is included to show how the two speedups



Figure 6. Classification of VA16 (Landsat ETM+ path 16), green is forest and tan is non-forest.

(based on one and two processors) relate to actual execution times for the three images. Speedup is a good indication of how well the parallel program is using additional computing resources, but decreased execution time is what is important to a user.

All three images are similar in size, so similar processing times and speedups might be expected. Variations in execution time and speedup can be accounted for by different numbers of iterations within pIGSCR, depending on the purity of the clusters that are created by unsupervised classification. VA17 showed the least speedup, as would be expected considering VA17 is the smallest of the three images. Furthermore, VA17 required fifteen iterations of pIGSCR while VA16 required nine and VA15 required six. As more iterations are completed, the parallel speedup slows as K -means is run on successively smaller images. This is because the time spent executing parallel sections of code decreases while the time spent executing serial sections remains constant. This also explains why VA15 with only six iterations had better parallel speedup than VA16 (a larger image) with nine iterations.

6.4 Discussion

Theoretically, an application that parallelizes perfectly would have a parallel speedup equal to the number of processors used, as shown by the solid lines in the figures. In reality,



Figure 7. Classification of VA17 (Landsat ETM+ path 17), green is forest and tan is non-forest.

Table 4 Execution Time(sec) for VA15, VA16, and VA17, p = number of processors

p	VA15	VA16	VA17
1	16,076	27,229	30,311
2	4,056	5,997	4,353
4	1,809	2,949	2,268
8	852	1,451	1,194
12	576	1,050	888
16	426	763	708
32	283	494	495
64	209	355	426

there are a number of factors that influence parallel speedup, including constraints of the underlying architecture, parallel communication and management overhead, and inherent serial portions of the algorithm that is written in parallel. The above speedup graphs provide evidence that many of these different factors influence the parallel speedup of pIGSCR.

Considering only processor number and speed, it would seem impossible for the actual realized speedup to be greater than ideal speedup, as is the case in Figure 8 and 9. In order to explain greater than ideal speedup, or superlinear speedup, computer memory must be considered. Different types of memory are accessed at different speeds. Cache memory is the fastest, followed by random access memory (RAM), and the slowest form of memory (virtual) is the disk drive. If there is more data in an application than can fit in one fast type of memory, then the processor must rely on the slower access of another type of memory

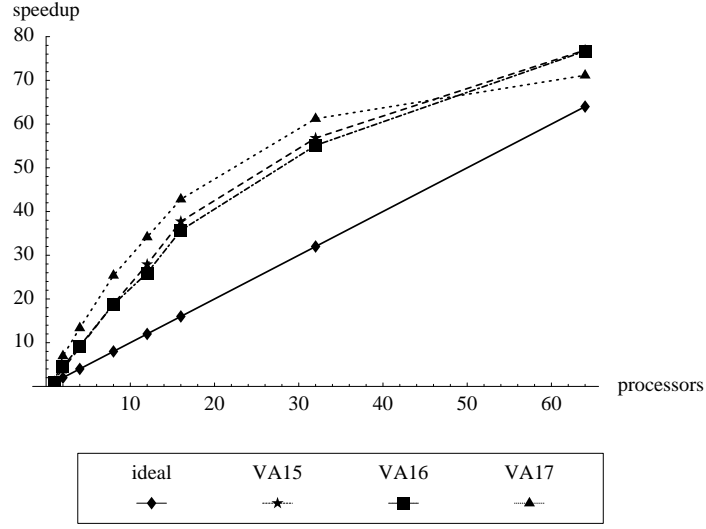


Figure 8. Speedup with respect to one processor.

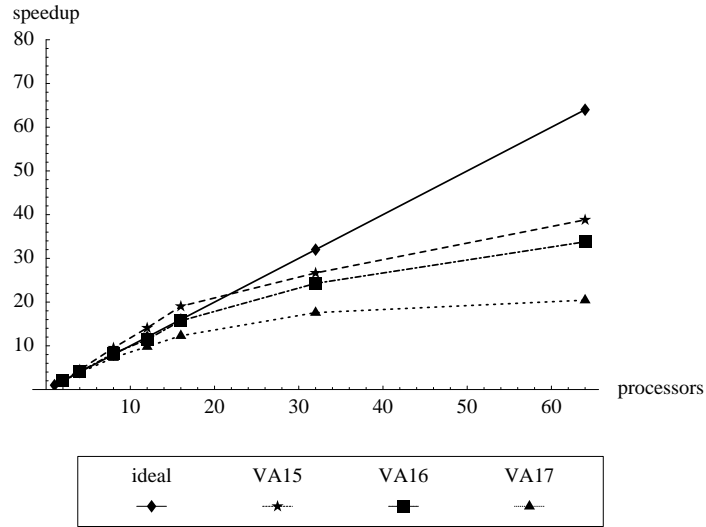


Figure 9. Speedup with respect to two processors.

in order to process all of the data. On the SGI Altix platform specifically, each processor has individual cache memory, and the superlinear speedup observed can be explained by the addition of more cache with the addition of more processors. Furthermore, although adding more processors does not technically add more RAM, because of the Altix's NUMA architecture, adding more processors will decrease the distance (access time) between each processor and the RAM it accesses, if the application is programmed to take advantage of the architecture as described above.

Except for the most trivial parallel applications, parallel codes will not realize perfect

speedup because there is some part of the algorithm that is inherently sequential (Amdahl's Law, [1]). There are several sections of pIGSCR that are still executed serially in the present implementation. Furthermore, because pIGSCR relies on data parallelism, the portion of the computation that is run in parallel depends on the size of the dataset that is processed. As the size of the dataset decreases, the serial section of the algorithm consumes a greater percentage of overall processing time compared with the parallel sections. With each iteration of pIGSCR, a progressively smaller dataset is clustered in parallel with K -means, meaning that the part of K -means that can be run in parallel decreases in size while the amount of serial processing remains the same. This explains the leveling off of the speedup past sixteen processors.

A leveling off of speedup (or decrease in speedup in some cases) may also occur as a result of the underlying architecture's mechanism for communication. In order for each processor to access the global memory space, it must send and receive data using the (hardware) global communication infrastructure. When a processor accesses the memory block that is physically closest on the NUMA architecture, there is less competition for shared communication resources. However, when processors are competing to access the same memory, for example some of the shared variables such as the cluster or class information, slower memory access times could occur. The computation slows down any time two or more processors attempt to access the same memory, and the potential for this to occur is greater as more processors are used. This increased memory contention coupled with a decreasing work load per processor per pIGSCR iteration, results in less than ideal speedup for a large number of processors.

Chapter 7: FEATURE REDUCTION WITHIN pIGSCR

This thesis has shown that pIGSCR is significantly faster than IGSCR, and therefore input parameters may be selected using a factorial analysis of input parameters to determine the specific combination of parameters that results in the highest classification accuracy. This approach has been extended to factorially determine which combination of bands would produce the best classification accuracy for a specific set of input parameters, dataset, and training dataset. Initial experimental runs on images containing six bands demonstrated that the most accurate classifications on all experimental images typically used all six bands. This approach was then further extended to experimentally determine the most accurate combination of bands for each unsupervised classification performed within each iteration of IGSCR. The resulting pIGSCR algorithm is presented below.

Algorithm pIGSCR with feature reduction

User inputs an image, training dataset, validation dataset, maximum number of iterations, α and p_0 for the homogeneity test, and number of clusters to be created in each iteration.

The output that is produced includes a set of pure class signatures that is used in the final supervised classification, and any or all of the following three classification images: DR, IS, IS+

begin

do until maximum number of iterations is reached,
 no pixels remain in original image,
 or no pure classes are found:

do for each possible band combination:

 Cluster remaining pixels using parallel clustering algorithm.

do for each point in the training data set, in parallel:

 determine the spectral class assignment based on the clustering,

 and increment the appropriate informational class count

 and spectral class count

end do

do for each cluster, in parallel:

 use the homogeneity test given above to determine informational class assignment.

end do

do in parallel:

 Recode all pixels in the unsupervised classification to informational classes

end do

 Create accuracy assessment matrix and determine overall classification accuracy
 for band combination

```

end do
Keep unsupervised classification image for most accurate clustering
do for each cluster, in parallel:
    if a particular cluster is pure, then
        Add that cluster's signature to the set of pure class signatures
        and mask all vectors/pixels belonging to that class out of the input image.
        Recode all pixels belonging to that spectral class in the
        unsupervised classification image to the value of the assigned informational class.
    end if
end do
end do
Perform a parallel supervised classification
on the input image using the pure signature set.
do in parallel:
    Recode all impure classes in the unsupervised classification image
    to a value reserved for impure classes and add this to the
    unsupervised classification image for a second output image.
end do
Perform a parallel supervised classification on only the impure pixels and
add them to the unsupervised classification image to produce a third
output image.
end

```

This feature reduction algorithm for pIGSCR produced marginally better accuracies in some classifications performed on images with six bands. Unfortunately, each iteration requires that $2^{bands} - 1$ classifications be performed to determine the most accurate combination of bands, and it is therefore not feasible for images with large numbers of bands. The six band classifications required that 63 individual unsupervised classifications be run per iteration, and an image with as few as ten bands would require over 1000 classifications be performed for each iteration of IGSCR. Additional algorithmic improvements and additional processing power would be required to make this band selection algorithm a feasible option.

Chapter 8: SVD-BASED FEATURE REDUCTION

At a high level, the SVD reveals the minimum number of dimensions required to represent a matrix or linear transformation [32] [46]. Often multi-dimensional data may be represented equivalently (or approximately so) in fewer dimensions due to redundancies in data. If a set of n -dimensional vectors all lie in a k -dimensional subspace, $k < n$, then each n -vector effectively has only k degrees of freedom, and can be uniquely described by k numbers. The natural correlations that occur in nature make the SVD a good candidate for feature reduction. Furthermore, if no reduction is possible, this will be shown by the magnitudes of the singular values revealed by the SVD.

The SVD of a linear transformation $A : R^n \mapsto R^m$ is

$$A = U\Sigma V^t,$$

where $U : R^m \mapsto R^m$ is orthogonal ($U^t U = I$), $\Sigma : R^n \mapsto R^m$ is a diagonal matrix whose diagonal elements are called the singular values of A , and $V^t : R^n \mapsto R^n$ is orthogonal as well. Each linear transformation in the SVD is expanded and shown as follows, assuming $m < n$:

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix} = \begin{pmatrix} u_{11} & \cdots & u_{1m} \\ \vdots & \ddots & \vdots \\ u_{m1} & \cdots & u_{mm} \end{pmatrix} \begin{pmatrix} \sigma_{11} & \sigma_{12} & \cdots & \sigma_{1n} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{m1} & \sigma_{m2} & \cdots & \sigma_{mn} \end{pmatrix} \begin{pmatrix} v_{11} & v_{12} & \cdots & v_{1n} \\ v_{21} & v_{22} & \cdots & v_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ v_{n1} & v_{n2} & \cdots & v_{nn} \end{pmatrix}.$$

Since entries σ_{ij} when $j > m$ are all zero, the product ΣV^T will produce entries of zero for rows $m + 1$ through n . The SVD may be rewritten as

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix} = \begin{pmatrix} u_{11} & \cdots & u_{1m} \\ \vdots & \ddots & \vdots \\ u_{m1} & \cdots & u_{mm} \end{pmatrix} \begin{pmatrix} \sigma_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \sigma_m \end{pmatrix} \begin{pmatrix} v_{11} & v_{12} & \cdots & v_{1n} \\ \vdots & \vdots & \ddots & \vdots \\ v_{m1} & v_{m2} & \cdots & v_{mn} \end{pmatrix}.$$

This shows that a column $A_{\cdot i}$ of A , an m vector, can be expressed as a linear combination of the m basis vectors in U ($U_{\cdot 1}, U_{\cdot 2}, \dots, U_{\cdot m}$), using the singular values in Σ ($\sigma_1, \sigma_2, \dots, \sigma_m$), and the i th column $V_{\cdot i}^t$ in V^t . The diagonal elements of Σ are nonnegative, and can be ordered such that $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_m$. If some entries on the diagonal of Σ are zero, then for some k , $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_k > \sigma_{k+1} = \dots = \sigma_m = 0$. Using the above logic that allowed

a reduction in the number of rows in V^t from n to m , the number of columns in U can be reduced to k , the number of rows and columns of Σ can be reduced to k , and the number of rows in V^t can be reduced to k , yielding

$$\begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{pmatrix} = \begin{pmatrix} u_{11} & \dots & u_{1k} \\ \vdots & \ddots & \vdots \\ u_{m1} & \dots & u_{mk} \end{pmatrix} \begin{pmatrix} \sigma_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \sigma_k \end{pmatrix} \begin{pmatrix} v_{11} & v_{12} & \dots & v_{1n} \\ \vdots & \vdots & \ddots & \vdots \\ v_{k1} & v_{k2} & \dots & v_{kn} \end{pmatrix}.$$

An operation such as a classification that would be performed on the entire $m \times n$ matrix A can now be equivalently performed on the entire $k \times n$ matrix ΣV^t where $k < m$, resulting in a reduction in the number of bands present in each vector. For practical purposes, singular values may in fact be nonzero yet be sufficiently close to zero to reduce the dimension of the data. The singular values $\sigma_{k+1}, \dots, \sigma_m$ represent distances from the subspace spanned by U_1, \dots, U_k , and very small distances may not affect the operation that will be performed on the reduced data, such as classification. If none of the singular values on the diagonal are close to zero, then the data is already represented using as few dimensions as possible. However, this seems unlikely in the application of remote sensing as geographic data is naturally redundant.

Practically speaking, it would be necessary to think of a three-dimensional (pixel row, pixel column, data bands) image in two dimensions in order to take advantage of the feature reduction made possible by using the SVD. This would also be an expensive computation as the leading dimension of the matrix would be the number of bands, but the second dimension would be equal to the number of pixels in the image. One of the features of an SVD is that it reveals the basis vectors U that can be used to transform any vector from the original vector space (range of A) to the new vector space (range of $U\Sigma$). If a training dataset (columns of T) is truly representative of a particular image (columns of A), then the resulting SVD $T = \tilde{U}\tilde{\Sigma}\tilde{V}^t$ will also reveal a set of basis vectors for the range of A . Using this result, it is possible to perform the SVD on a training data matrix that is $m \times p$ in dimension, where p is the number of points in the training data set, and use the resulting SVD to transform A . In order to do this, it is necessary to project the columns of the matrix A onto the subspace spanned by the first k columns of \tilde{U} . This is accomplished by simply computing $(\tilde{U}_1, \tilde{U}_2, \dots, \tilde{U}_k)^t A$. This result can be arrived at algebraically. Assume that the $m \times p$ matrix T is a submatrix of the $m \times n$ matrix A , and that $\dim \text{range } A = \dim \text{range } T = k$. Then $\text{range } (U_1, \dots, U_k) = \text{range } A = \text{range } T = \text{range } (\tilde{U}_1, \dots, \tilde{U}_k)$, which means each column of A can be reduced to its k coordinates with respect to the orthogonal basis $\tilde{U}_1, \dots, \tilde{U}_k$ of $\text{range } (\tilde{U}_1, \dots, \tilde{U}_k)$. These coordinates are given by multiplying by $(\tilde{U}_1, \dots, \tilde{U}_k)^t$. After the original image has been transformed and the number of bands has been reduced, the pIGSCR classification algorithm is run with no modifications.

Chapter 9: DATA DESCRIPTION AND PREPARATION

pIGSCR with the SVD or PCA was tested using Landsat Thematic Mapper (TM) and Enhanced Mapper (TM/ETM+) images (path 17/row 34) acquired on December 1, 1999, March 6, 2000, June 10, 2000. The images from 1999 and 2000 were layered together to form a composite multi seasonal image, and all images were registered to a rectified 2003 image with an RMSE of 1/5 a pixel or less. Using Leica Geosystems Erdas ImagineTM 8.7 software, registration was performed with 24 control points for each image pair, eight of which were randomly selected as check points. A first order transformation was used, and resampling was performed using nearest neighbor. Each of the three images used for the composite image was converted to reflectance in accordance with the Landsat 7 user's guide [31], and dark object subtraction was performed in ITT ENVITM 4.2 using the band minimum method. The composite image, which will be referred to as VA17C, contained 18 total bands. Figures 10, 11, and 12 show representative subsets of the VA17C image (bands 4, 3, and 2), the VA17C PCA (first three bands) and VA17C SVD (first three bands), respectively. Figure 13 shows the locations of the subset and VA17C in relation to each other and Virginia.

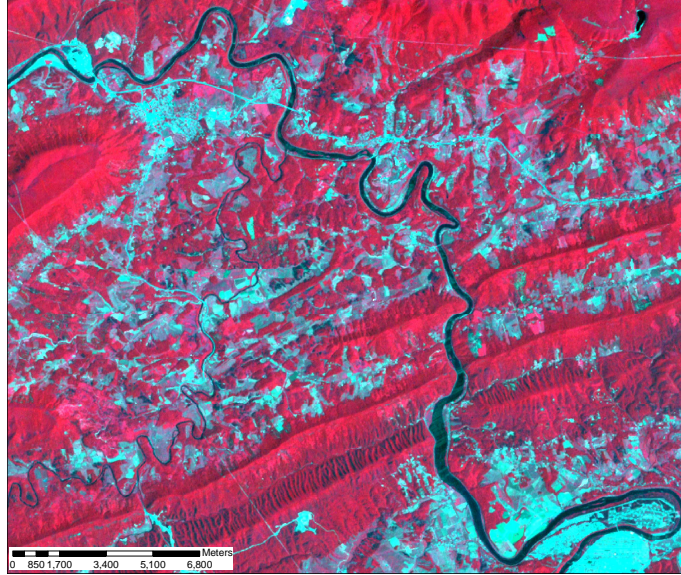


Figure 10. VA17C (Landsat TM/ETM+ path 17/row 34, bands 4, 3, and 2 shown) zoomed to a subset of interest.

The training data for these images was created by the interpretation of point locations from a systematic, hexagonal grid over Virginia Base Mapping Program (VBMP) true color digital orthophotographs [48]. The data were collected in the Spring of 2002 at a scale of 1:4,800, and were subsequently resampled to a 1 meter spatial resolution. The data were

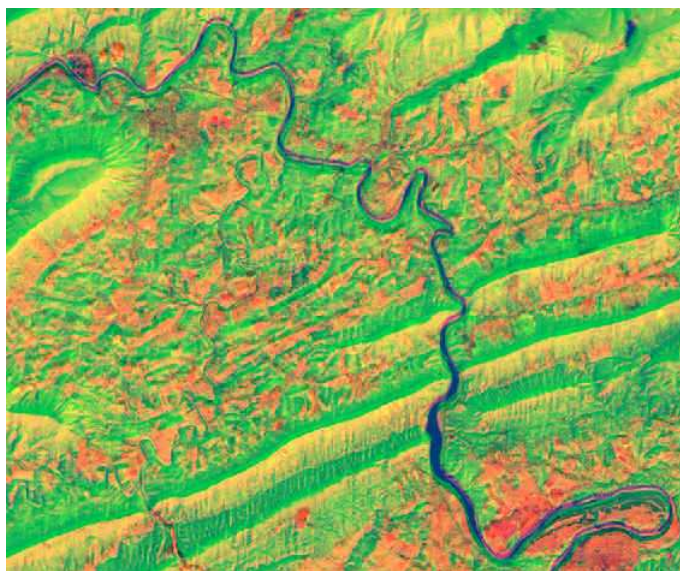


Figure 11. VA17C PCA (Landsat TM/ETM+ path 17/row 34, bands 1, 2, and 3 shown) zoomed to a subset of interest.

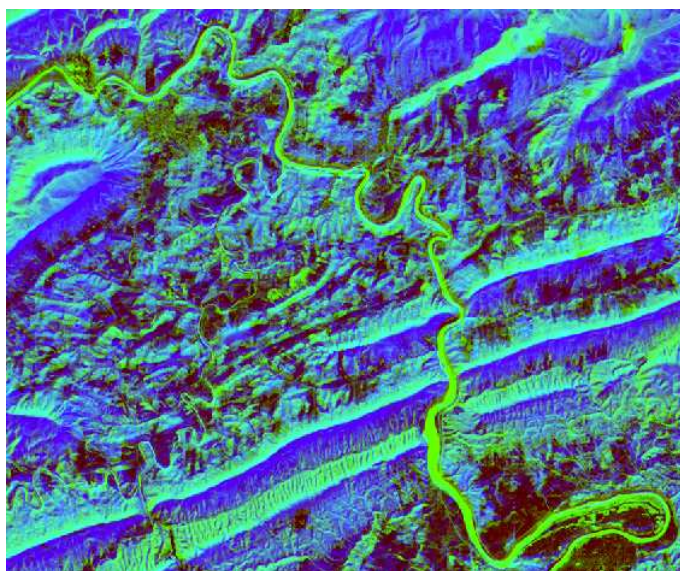


Figure 12. VA17C SVD (Landsat TM/ETM+ path 17/row 34, bands 1, 2, and 3 shown) zoomed to a subset of interest

used to perform a two-class classification, forest or non-forest. For the purpose of accuracy assessment, validation data in the form of point locations at the center of USDA Forest Service Forest Inventory and Analysis (FIA) ground plots [42][6] were used to validate the classifications. Only homogeneous FIA plots were used (either 100 percent forest or non-forest), and these plots were visited between 1997 and 2001.

A second dataset consisting of a multitemporal series of TM and ETM+ images from

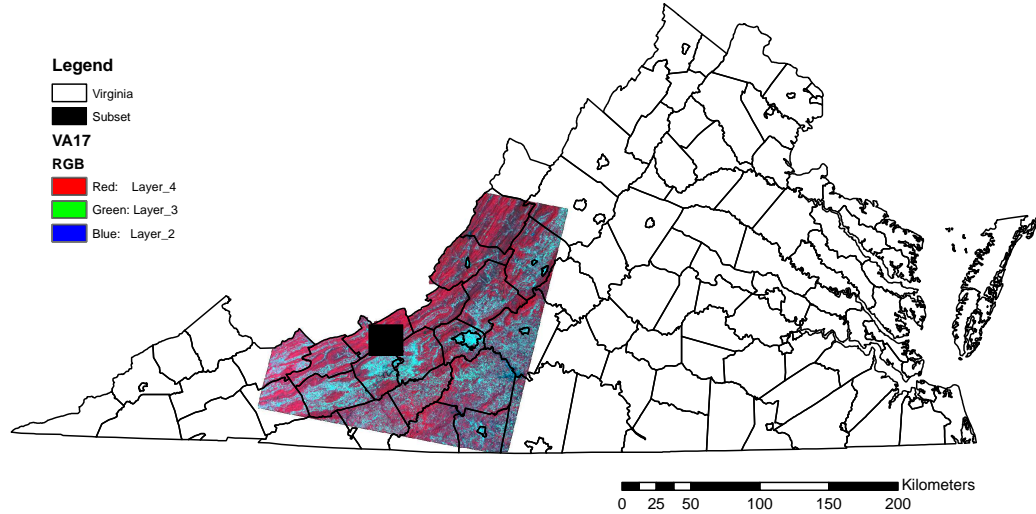


Figure 13. Locations of Virginia, VA17C (Landsat TM/ETM+ path 17/row 34), and zoomed subset shown in Figures 10—12 in relation to each other.

path 232/row 67 (1995-2002) acquired over Rondônia were used, each image registered to the 2001 image. The 2001 image was rectified by the US National Center for Earth Resources Observation & Science, and the remaining registrations were performed using Leica Geosystems Erdas ImagineTM 8.5 with at least 50 control points and 25 randomly selected check points. The root mean squared error was less than 1/3 of a pixel for each image registration, and nearest neighbor resampling was used [28][50]. This multitemporal image will be referred to as AM232. Figures 14, 15, and 16 show representative subsets of AM232 (bands 4, 3, and 2 from the 1998 image, corresponding to training data used for classification), AM232 PCA (first three bands), and AM232 SVD (first three bands). Figure 17 shows the locations of the subset and AM232 in relation to each other and Rondônia, Brazil.

Collecting training data for this region was a challenge, as described by [28][50]. The training data used for a forest/non-forest pIGSCR classification was collected using detailed interviews, Landsat imagery, and detailed maps of various farms shown in the images. At least 67 pixels for each of the two classes (forest and non-forest) were present, and these points were used as seed pixels for a region growing algorithm (Erdas ImagineTM 8.5). Because of the difficulties in acquiring training data, a total of 200 validation pixels were randomly selected from the training data. This ensured a representation of edge and mixed pixels, which is necessary in order to avoid inflated classification accuracies [40].

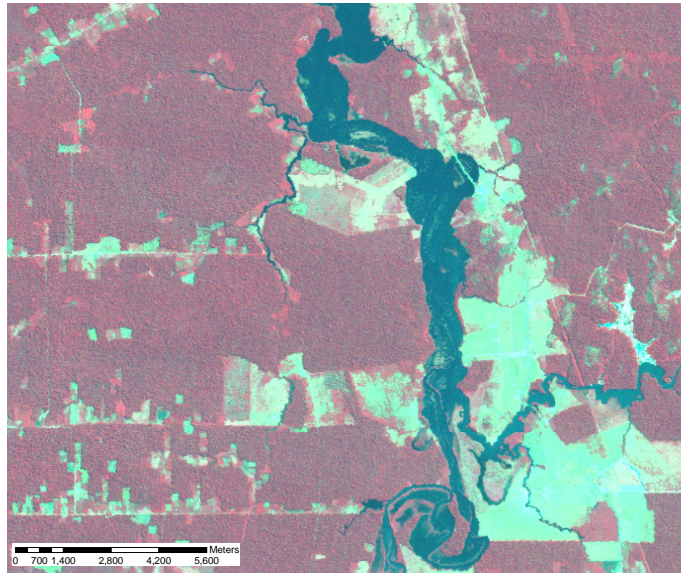


Figure 14. AM232 (Landsat TM path 232/row 67, bands 4, 3, and 2 of image acquired in 1998 shown) zoomed to a subset of interest.

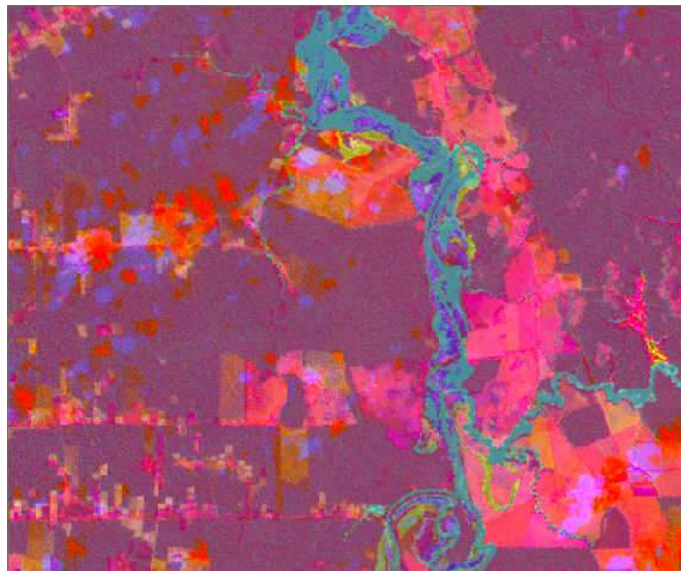


Figure 15. AM232 PCA (Landsat TM path 232/row 67, bands 1, 2, and 3 shown) zoomed to a subset of interest.

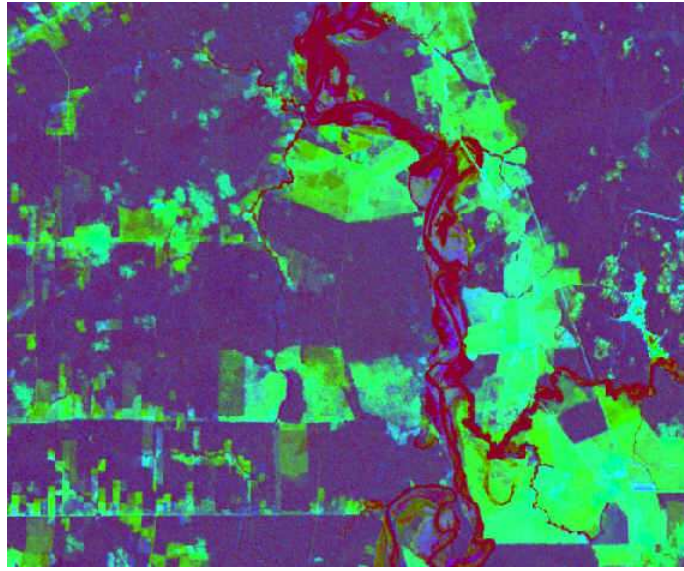


Figure 16. AM232 SVD (Landsat TM path 232/row 67, bands 1, 2, and 3 shown) zoomed to a subset of interest.

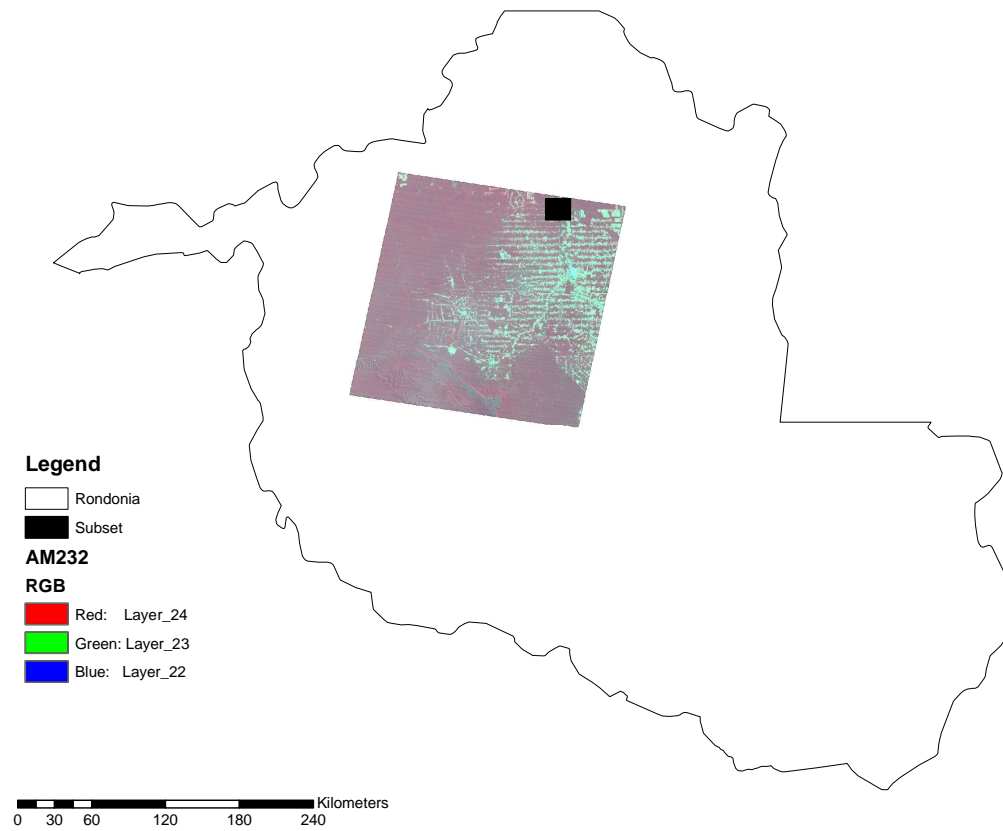


Figure 17. Locations of Rondônia, AM232 (Landsat TM path 232/row 67), and zoomed subset shown in Figures 14—16 in relation to each other.

Chapter 10: SVD RESULTS AND DISCUSSION

A two class classification (forest, non-forest) using pIGSCR with 80 initial classes and a homogeneity threshold of 80 percent was run on VA17C and AM232 and both of the above feature reduction methods were used. pIGSCR was applied to the composite VA17C image using all 18 bands, but applying pIGSCR to AM232 without feature reduction was infeasible considering the number of bands (52) in AM232. The above algorithms were implemented using Fortran 95 [18] and LAPACK [2]. These classifications were run using an SGI Altix 3300 with 24GB of RAM and twelve 1.3 GHz Itanium processors.

pIGSCR was applied to the VA17C SVD image using as few as four and as many as fourteen bands. Each resulting SVD classification was compared to the corresponding classification of the entire eighteen band VA17C image using McNemar's test for statistical significance of the difference between the two classifications as described by Foody [17]. A chi-square distribution with one degree of freedom and $\alpha = .05$ (3.841) was used where

$$\chi^2 = \frac{(x_1 - x_2)^2}{x_1 + x_2},$$

with x_1 being the number of pixels correctly classified by the first classification but incorrectly classified by the second, and vice versa for x_2 . Using this test, all classifications using five or more bands from the SVD image were not statistically different from the classifications using all eighteen bands of VA17C. Out of eleven tests run (four through fourteen bands) four tests using the SVD of VA17C were actually marginally higher in accuracy than the classification accuracy of 88.62% using all eighteen bands. Figures 18 and 19 show the resulting classification images of VA17C and VA17C SVD (using six bands), respectively, where green pixels were determined to be forest and tan pixels were determined to be non-forest. Although these two classifications are statistically the same classification (as determined by McNemar's test), it is clear that using the SVD image resulted in the correct classification of the large river that is prominent in this scene, while the classification using all eighteen bands of VA17C determined most of the river to be forest.

Using all eighteen bands resulted in a pIGSCR execution time of roughly 1650 seconds, more than five times as long as the quickest run of just over 300 seconds using five bands of the SVD image. In general, using more bands increases the execution time, however, execution time is also dependent on the number of iterations performed during the iterative clustering and the number of pure classes used for the decision rule. It is therefore advantageous to use as few bands as possible when execution time is a great concern, and the accuracies resulting from using five through fourteen bands of the SVD image are consistently around 88%, both above and below the classification accuracy of VA17C. These consistent accuracies indicate that the choice of k , the singular value cutoff, is not crucial past five bands.

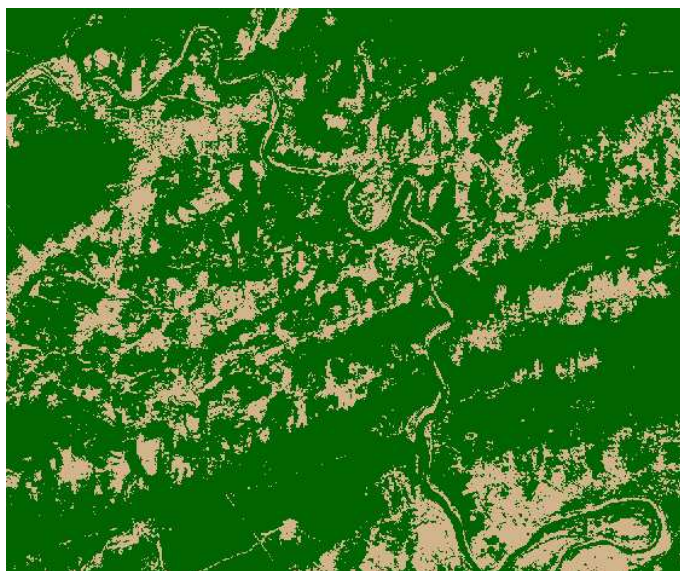


Figure 18. VA17C DR (Landsat TM/ETM+ path 17/row 34) classification, green = forest, tan = non-forest.

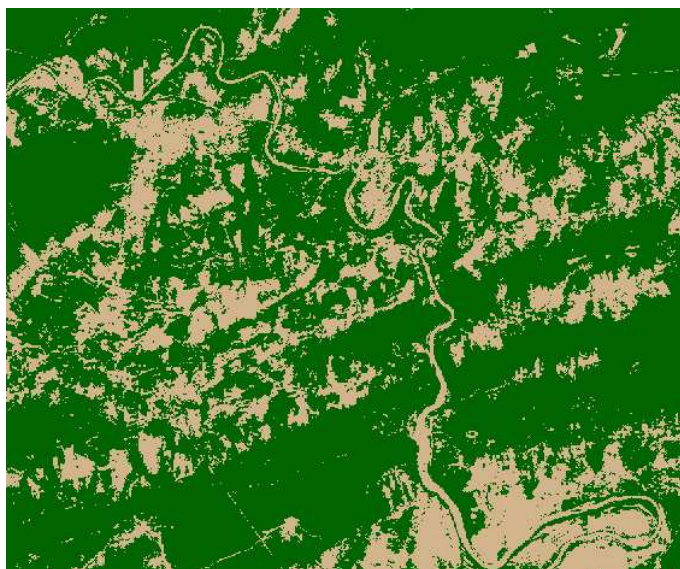


Figure 19. VA17C SVD DR (Landsat TM/ETM+ path 17/row 34) classification, green = forest, tan = non-forest.

10.1 SVD versus PCA

Table 5 summarizes all of the experimental runs of pIGSCR on both datasets, using the SVD or PCA as a means of feature reduction. The values in the table are computed by averaging the results of 11 runs for the Virginia data (four through fourteen bands used) and 18 runs for the Amazon data (three through twenty). Classification accuracies are reported for both pIGSCR output images that contain only two classes (the DR and IS+ images),

Table 5 Comparison of Averages for PCA and SVD pIGSCR Classifications

	SVD	PCA
VA DR accuracy	88.18	88.01
VA IS+ accuracy	88.34	88.16
VA execution time (secs.)	817	818
VA iterations	6.5	6.27
VA classes	78.2	77
AM DR accuracy	92.75	92.5
AM IS+ accuracy	93.31	92.94
AM execution time (secs.)	528	643
AM iterations	2.33	2.5
AM classes	18.33	20.11

corresponding to the validation set containing only two classes. Consider the classification accuracies listed, and notice that using the SVD to reduce the data dimension resulted in higher accuracies than using the PCA for both classification images (DR and IS+) using both datasets.

Considering now just the VA dataset, a comparison between the SVD and PCA using McNemar’s test for statistically different classifications as described above showed that the classifications were consistently the same. All of the classification accuracies of the Virginia data are similar, but the accuracies for the SVD are more often higher (seven out of eleven cases). For this particular dataset and classification (forest/non-forest), the subspace spanned by the first few axes of decreasing variance is likely similar to the subspace revealed by the SVD, accounting for the lack of distinction between the two feature reduction methods.

Table 6 Classification Accuracies of Statistically Different Classifications

bands	class	SVD accuracy	PCA accuracy
6	DR	95	91.5
6	IS+	95.5	92.5
7	DR	93.5	91
7	IS+	94.5	92
9	DR	92.5	95.5
9	IS+	92.5	95.5
10	IS+	94	91.5
12	DR	94	92
12	IS+	94.5	92.5
19	IS+	92	94

Applying the same feature reduction methods and classification to the data from the Amazon region, however, demonstrated more separation between the two feature reduction methods. Table 6 lists all classification accuracies for the SVD and PCA AM232 images where the SVD and PCA reduction to the same number of bands resulted in statistically

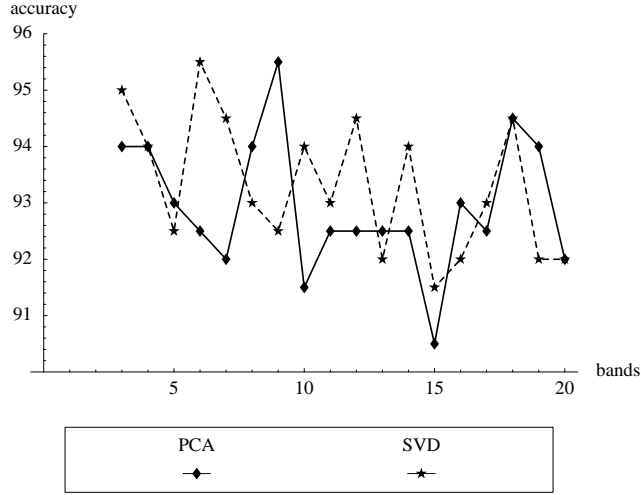


Figure 20. IS+ classification accuracy for AM232.

different classifications. In seven out of ten cases, the SVD feature reduction resulted in a higher classification accuracy. Figure 20 contains a graph comparing the classification accuracies of the two feature reduction methods for different numbers of bands kept (singular value or eigenvalue cutoff). Notice that the SVD accuracies are consistently higher than the PCA accuracies, and increasing the number of bands does not usually result in increased accuracy. Of particular interest are the cases where the number of bands is six or nine, where one method drastically outperformed the other regarding accuracy. Figures 21—24 contain subsets of the classification results of AM232 for the SVD and PCA images, using either six or nine bands. Observing Figures 21 and 24, it appears that these images contain more misclassified forested regions due to cloud cover. Also notice that in Figure 23, the most accurate PCA classification, the river running vertically through the image has been incorrectly classified as forest, showing that this classification is not necessarily better than the corresponding SVD classification.

A more subtle advantage of using the SVD over PCA for feature reduction prior to pIGSCR classification is the unexpected execution time savings. Although the time savings in the Virginia classifications was minimal, the execution times for the classification applied to the SVD AM232 image are consistently shorter than execution times for the classification applied to the PCA AM232 image, as shown in Figure 25. An analysis of pIGSCR reveals that execution time can be affected by the size of the image, the number of iterations required, and the number of pure classes. Since the PCA and SVD produce images of the exact same size, the differences in execution time can be attributed to differences in the number of iterations and classes. Notice that the peaks in the execution time graph (Figure 25), correspond to peaks in Figure 26 (number of iterations) and peaks in Figure 27 (number of classes). A small number of iterations and classes does not appear to negatively impact



Figure 21. AM232 (Landsat TM path 232/row 67) PCA IS+ classification (6 bands), green = forest, tan = non-forest.



Figure 22. AM232 (Landsat TM path 232/row 67) SVD IS+ classification (6 bands), green = forest, tan = non-forest.

overall classification accuracy. For example, at six bands, classification accuracies were higher using the SVD for feature reduction (and were statistically different classifications than when using PCA), yet the classification on the PCA data required more iterations and resulted in more pure classes. Although this advantage seems tailored to pIGSCR, the implications extend to classification in general, both supervised and unsupervised, as pIGSCR is a hybrid classification algorithm and exhibits characteristics of both.



Figure 23. AM232 (Landsat TM path 232/row 67) PCA IS+ classification (9 bands), green = forest, tan = non-forest.



Figure 24. AM232 (Landsat TM path 232/row 67) SVD IS+ classification (9 bands), green = forest, tan = non-forest.

Chapter 11: CONCLUSIONS AND FUTURE WORK

Prior to this work, IGSCR classification was performed using a “black box” proprietary software library, and classification of a full Landsat scene required several hours of processing time. With the creation of a portable and open source serial version of IGSCR, the black box was removed, allowing analysts to verify and modify the algorithms that are used. The

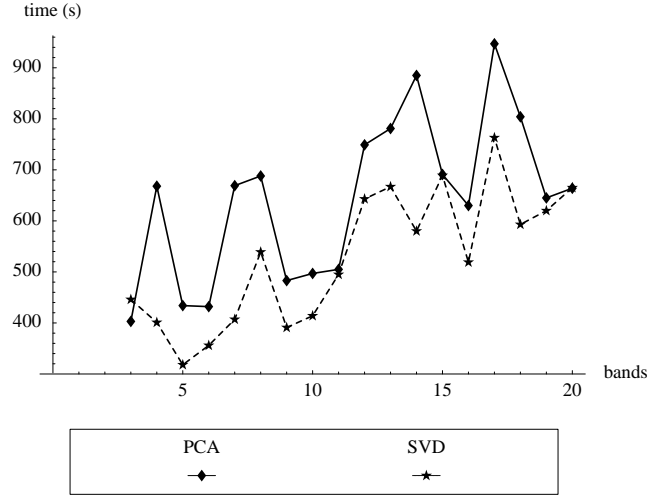


Figure 25. Execution time for AM232.

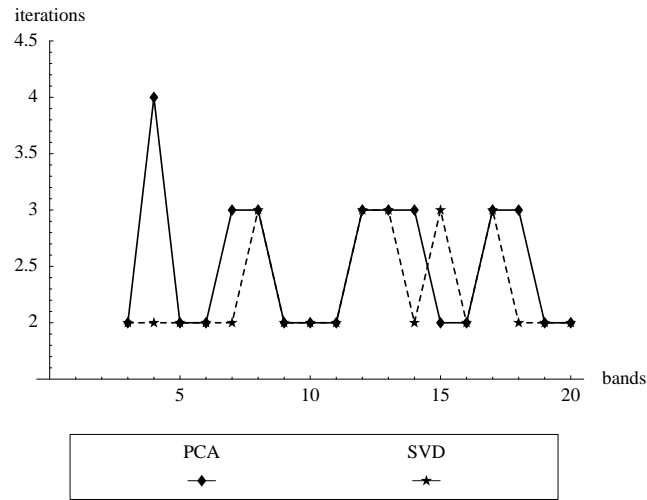


Figure 26. Number of iterations for AM232.

large speedup observed was the result of adding both memory and processors, and therefore does not necessarily indicate good parallel speedup in a technical parallel computing sense; however, these results indicate real time that is saved for a real user. The specifications of one processor used are very similar to a very powerful standalone desktop computer. These results show that execution times can be reduced from several hours to several minutes by using only a modest number of processors in a parallel environment. Hence with parallel computing it is feasible to perform many runs of IGSCR in order to obtain the ideal combination of input parameters leading to the best classification accuracy.

This thesis presents a feature reduction method for remotely-sensed data using the singular value decomposition. This new feature reduction technique was applied to training

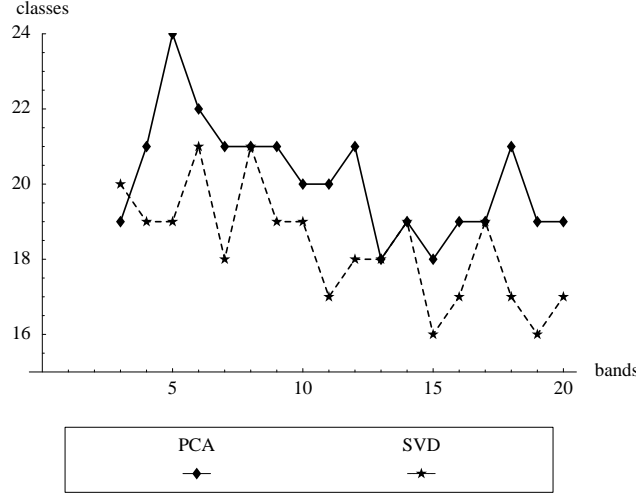


Figure 27. Number of pure classes for AM232.

data from two multitemporal datasets of Landsat TM/ETM+ imagery acquired over a forested area in Virginia, USA and Rondônia, Brazil. pIGSCR forest/non-forest classifications of the Virginia data were five times faster using SVD reduction without affecting the classification accuracy. Feature reduction using the SVD was also compared to feature reduction using principal components analysis (PCA) for both datasets. The highest average accuracies for the Virginia dataset (88.34%) and for the Rondônia dataset (93.31%) were achieved using the SVD. SVD-based feature reduction can yield statistically significantly better classifications than PCA.

pIGSCR showed good parallel speedup through 16 processors, however, speedup showed signs of leveling off at 32 and 64 processors when compared to the execution time on two processors. Further work on a pIGSCR algorithm needs to be done in order to justify using a larger (distributed memory) supercomputer to run pIGSCR. Clustering algorithms and decision rules that are more amenable to a parallel environment might be considered in a distributed memory implementation. Furthermore, adding more underlying classification algorithm options available to a user is now possible. The previous implementation of IGSCR that used proprietary classification libraries allowed less flexibility, limiting a user to specific classification algorithms. Additionally, a more detailed study of the IGSCR framework is possible with this increased flexibility.

This thesis demonstrated the utility of SVD-based feature reduction on images containing fewer than 100 bands, however, SVD-based feature reduction of higher dimensional data (i.e., hyperspectral) should be evaluated in the future. Another area for future exploration with SVD-based feature reduction is classification with increased categorical specificity. With less variability between informational classes, feature reduction based on variance (such as with PCA) is likely to produce a lower quality classification on a reduced dimension image

than feature reduction based on SVD. Finally, this thesis demonstrates that applying the SVD to training data in order to reduce the dimension in an entire image produces good classification results. An ideal implementation would apply the SVD to the entire image to reveal the exact basis vectors, not an approximation derived from the training data. A parallel SVD implementation on multiple processors would allow the SVD to be performed on a large image in a reasonable amount of time, likely resulting in greater classification accuracy.

REFERENCES

- [1] G. M. Amdahl, "Validity of the single processor approach to achieving large scale computing capabilities," In *AFIPS Conference Proceedings*, Vol. 30, pp 483-485, 1967.
- [2] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, and D. Sorensen, *LAPACK Users' Guide*, SIAM, Philadelphia, 1992, 235pp.
- [3] G. H. Ball and D. J. Hall, "A novel method of data analysis and pattern classification," Technical Report AD 699616 Stanford Research Institute, Menlo Park, CA, 9pp, 1965.
- [4] M. E. Bauer, T. E. Burk, A. R. Ek, P. R. Coppin, S. D. Lime, T. A. Walsh, D. K. Walters, W. Befort, and D. F. Heinzen, "Satellite inventory of Minnesota forest resources," *Photogrammetric Engineering & Remote Sensing*, 60(3), 287-298, 1994.
- [5] W. A. Bechtold and C. T. Scott, "The Forest Inventory and Analysis Plot Design," In The Enhanced Forest Inventory and Analysis Program – National Sampling Design and Estimation Procedures, W. A. Bechtold and P. L. Patterson, editors. USDA Forest Service Southern Research Station, General Technical Report SRS-80, pp. 27-42, 2005.
- [6] A. J. Bernstein, "Analysis of programs for parallel processing," *IEEE Transactions on Computers*, 746-757, Oct. 1966.
- [7] J. Bigün, "Unsupervised feature reduction in image segmentation by local transforms," *Pattern Recognition Letters*, 14, 573-583, 1993.
- [8] L. Bruzzone and D. F. Prieto, "Unsupervised Retraining of a Maximum Likelihood Classifier for the Analysis of Multitemporal Remote Sensing Images," *IEEE Trans. on Geosciences and Remote Sensing*, 39(2), 456-460, 2001.
- [9] J. Byeungwoo and D. Landgrebe, "Partially Supervised Classification Using Weighted Unsupervised Clustering," *IEEE Trans. on Geosciences and Remote Sensing*, 37(2), 1073-1079, 1999.
- [10] J. B. Campbell, *Introduction to Remote Sensing*, The Guilford Press, 2002, 621pp.
- [11] W. C. Chang, "On using principal components before separating a mixture of two multivariate normal distributions," *Applied Statistics*, 32, 267-275, 1983.
- [12] C. Clark, A. F. Clark, "Spectral identification by singular value decomposition," *International Journal of Remote Sensing*, 19(12), 2317-2329, 1998.
- [13] S. Danaher, E. O'Mongain, "Singular value decomposition in multispectral radiometry," *International Journal of Remote Sensing*, 13(9), 1771-1777, 1992.
- [14] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*, John Wiley & Sons, New York, 1973, 482pp.
- [15] W. Eppler, "Canonical analysis for increased classification speed and channel selection," *IEEE Transactions on Geoscience Electronics*, 14, 26-33, 1976.
- [16] G. M. Foody, "Thematic map comparison: evaluating the statistical significance of differences in classification accuracy," *Photogrammetric Engineering & Remote Sensing*, 705, 627-633, 2004.
- [17] Fortran, *Information technology — Programming languages — Fortran —*, ISO/IEC 1539-1, 1997.
- [18] A. A. Green, M. Berman, P. Switzer, M. D. Craig, "A transformation for ordering multispectral data in terms of image quality with implications for noise removal," *IEEE Transactions on Geoscience and Remote Sensing*, 26(1), 65-74, 1988.
- [19] J. A. Hartigan, *Clustering Algorithms*, John Wiley & Sons, New York, 1975, 351pp.
- [20] S. Hattori, Y. Myint, "Automatic estimation of initial approximations of parameters for bundle adjustment," *Photogrammetric Engineering and Remote Sensing*, 61(7), 909-915, 1995.
- [21] HDF5, "HDF5 User's Guide," Hierarchical Data Format (HDF) Group, National Center for Supercomputing Applications (NCSA), University of Illinois at Urbana-Champaign (UIUC), Urbana-Champaign, IL, 2005, 408pp.
- [22] K. Y. Huang, "A synergistic automatic clustering technique (SYNERACT) for multispectral image analysis," *Photogrammetric Engineering & Remote Sensing*, 68(1), 33-40, 2002.
- [23] G. F. Hughes, "On the mean accuracy of statistical pattern recognizers," *IEEE Transactions on Information Theory*, 14, 55-63, 1968.
- [24] J. R. Jensen, E. W. Ramsey, H. E. Mackey Jr., E. J. Christensen, and R. R. Shartz, "Inland wetland change detection using aircraft MSS data," *Photogrammetric Engineering & Remote Sensing*, 53(5), 521-529, 1987.
- [25] J. R. Jensen, *Introductory Digital Image Processing, A Remote Sensing Perspective*, Prentice Hall, 2005.

- [26] H. Jordan and G. Alagband, *Fundamentals of Parallel Processing*, Prentice Hall, New Jersey, 2003, 560pp.
- [27] K. A. Joseph, R. H. Wynne, J. O Browder, and J. B. Campbell, "Comparison of segment and pixel-based non-parametric land cover classification in the Brazilian Amazon using multitemporal Landsat TM/ETM+ imagery," *Photogrammetric Engineering & Remote Sensing*, in press.
- [28] R. J. Kauth, G. S. Thomas, "The tasseled cap—a graphic description of the spectral-temporal development of agricultural crops, as seen by landsat," in *Proceedings, Symposium on Machine Processing of Remote Sensed Data*, West Lafayette, IN, 41-51, 1976.
- [29] M. Kelly, D. Sharri, Q. Guo, and D. Liu, "A comparison of standard and hybrid classifier methods for mapping hardwood mortality areas affected by "sudden oak death"," *Photogrammetric Engineering & Remote Sensing*, 70(11), 1229-1239, 2004.
- [30] Landsat 7 Science Data Users Handbook (May 2006), NASA. [ONLINE], Available: <http://landsathandbook.gsfc.nasa.gov/handbook.html>.
- [31] C. Lawson, *Solving Least Squares Problems*, Prentice Hall, 1974.
- [32] D. C. Lay, *Linear Algebra and its Applications*, Addison Wesley, New York, 2000, 486pp.
- [33] G. E. Lowitz, "Stability and dimensionality of Karhunen-Loeve multispectral image expansions," *Pattern Recognition*, 10, 359-363, 1978.
- [34] M. Migliaccio, A. Gambardella, "Microwave radiometer spatial resolution enhancement," *IEEE Transactions on Geoscience and Remote Sensing*, 43(5), 1159-1169, 2005.
- [35] R. F. Musy, R. H. Wynne, C. E. Blinn, J. A. Scrivani, and R. E. McRoberts, "Automated forest area estimation via iterative guided spectral class rejection," *Photogrammetric Engineering & Remote Sensing*, 72(8), 949-960, 2006.
- [36] R. D. Phillips, L. T. Watson and R.H. Wynne, "Hybrid image classification using a shared memory parallel algorithm," in *Proceedings 16th William Pecora, T. Memorial Symposium*, CD-ROM, Sioux Falls, SD, 12 pages, Oct. 23-27, 2005.
- [37] R. D. Phillips, L. T. Watson, and R. H. Wynne, "Hybrid image classification and parameter selection using a shared memory parallel algorithm," *Computers & Geosciences*, doi:10.1016/j.cageo.2006.10.014, 2007.
- [38] R. L. Powell, N. Matzke, C. De Souza, M. Clark, I. Numata, L. L. Hess, D. A. Roberts, and M. Clark, "Sources of error in accuracy assessment of thematic land cover maps in the Brazilian Amazon," *Remote Sensing of Environment*, 90, 221-234, 2004.
- [39] M.J. Quinn, *Parallel Programming in C with MPI and OpenMP*, McGraw Hill, New York, 2004, 529pp.
- [40] G. A. Reams, W. D. Smith, M. H. Hansen, W. A. Bechtold, R. A. Roesch, and G. G. Molsen, "The Forest Inventory and Analysis Sampling Frame," In *The Enhanced Forest Inventory and Analysis Program – National Sampling Design and Estimation Procedures*, W. A. Bechtold and P. L. Patterson, editors. USDA Forest Service Southern Research Station, General Technical Report SRS-80, pp 11-26, 2005.
- [41] J. A. Richards and X. Jia, *Remote Sensing Digital Image Analysis*, Springer, 1999, 363pp.
- [42] J. San Miguel-Ayaz and G. S. Biging, "Comparison of single-stage and multi-stage classification approaches for cover type mapping with TM and SPOT data," *Remote Sensing of Environment*, 59, 92-104, 1997.
- [43] B.M. Shahshahani and D.A. Landgrebe, "Small sample size problem and mitigating the Hughes phenomenon," *IEEE Transactions on Geoscience and Remote Sensing*, 32(5), 1087-1095, 1994.
- [44] M. Snir, S. W. Otto, S. Huss-Lederman, D.W. Walker, and J. Dongarra, *MPI: The Complete Reference*, The MIT Press, Cambridge, 1996, 336pp.
- [45] G. Strang, *Linear Algebra and Its Applications*, Academic Press, 1976.
- [46] W. H. A. M. van den Broek, D. Wienke, W. J. Meissen, C. W. A. de Crom, L. Buydens, "Identification of plastics among nonplastics in mixed waste by remote sensing near-infrared imaging spectroscopy. 1. Image improvement and analysis by singular value decomposition," *Analytical Chemistry*, 67, 3753-3759, 1995.
- [47] VBMP, *Virginia Base Mapping Program (VBMP) Digital Orthophotography Project*, Virginia Geographic Information Network (VGIN) Advisory Board, Richmond, Virginia, 2003.
- [48] J. P. Wayman, R. H. Wynne, J. A. Scrivani, and G. A. Reams, "Landsat TM-based forest area estimation using Iterative Guided Spectral Class Rejection," *Photogrammetric Engineering & Remote Sensing*, 67(10), 1155-1166, 2001.

- [49] R. H. Wynne, K. A. Joseph, J. O Browder, P. M. Summers, “Comparing farmer-based and satellite-derived deforestation estimates in the Amazon basin using a hybrid classifier,” *International Journal of Remote Sensing*, in press.

VITA

Rhonda Phillips was born in Warren, Pennsylvania, USA on May 30th, 1980. She earned a bachelor's degree in computer science and mathematics with a minor in physics from Mansfield University of Pennsylvania in Mansfield, PA, USA in December of 2002. In August 2003, she started graduate study at Virginia Polytechnic Institute and State University (Virginia Tech) in the department of computer science. She received her Master of Science degree in spring 2007 and continues toward a Doctor of Philosophy in computer science at Virginia Tech.