Preprocessing remotely-sensed data for efficient analysis and classification

Patrick M. Kelly, James M. White

Los Alamos National Laboratory, Computer Research Group
MS B-265, Los Alamos, NM 87545

ABSTRACT

Interpreting remotely-sensed data typically requires expensive, specialized computing machinery capable of storing and manipulating large amounts of data quickly. In this paper, we present a method for accurately analyzing and categorizing remotely-sensed data on much smaller, less expensive platforms. Data size is reduced in such a way as to retain the integrity of the original data, where the format of the resultant data set lends itself well to providing an efficient, interactive method of data classification.

1. INTRODUCTION

A Landsat Thematic Mapper (TM) quarter scene consists of approximately 12 million pixels, each being represented by seven spectral reflectance values between 0 and 255. Each quarter scene, therefore, occupies 84 megabytes of storage, and performing even simple data manipulations for analysis or display purposes requires a large number of operations. By preprocessing the data by a technique known as vector quantization or clustering, computational requirements necessary for image analysis and manipulation are greatly reduced.

The advantages to clustering large data sets are numerous. Many times when scientists work with multispectral image data, they are interested in grouping together sets of similar data – something that clustering algorithms do automatically. Clustered data also has a number of properties that simplify data analysis and categorization. Data compression is a very desirable by-product of the clustering process, reducing the computational resources necessary to manipulate the data. Additionally, because pixels belonging to the same cluster are intrinsically associated with one another, sets of pixels in an image which share common characteristics can be manipulated simultaneously. Statistics for each cluster can easily be calculated during the clustering process, allowing many properties of the original data to be retained. For many applications, we have found that once clustering has been performed, the original data is no longer needed.

Each pixel in an image is commonly categorized according to its spectral signature. Many methods are used for classifying multispectral data, including both supervised and unsupervised classification methods [1, 2]. When using supervised methods for data classification, a user selects training areas representative of several types of land cover, and a classifier is developed to discriminate between different classes. This classifier is then used to categorize the remaining pixels in the scene. Numerous pattern recognition algorithms of this type exist, including nearest neighbor algorithms, discriminant function techniques, artificial neural networks, and statistical methods. An overview of these techniques can be found in standard pattern recognition textbooks [3, 4]. Statistical methods such as maximum likelihood classifiers [3] have always been popular for this type of problem. In general, although these techniques often work well, they are very time consuming both in computer time and operator effort. Additionally, they do not tend to allow easy classifier adjustments (or "fine-tuning") for the system.

Unlike supervised methods of classification, which require a user to define training sets, unsupervised techniques require no training sets at all. They instead attempt to automatically find the underlying structure of multi-dimensional data, by "clustering" the data into groups sharing similar characteristics. Unsupervised classification is an off-line process, requiring very little time of the system user. A user simply needs to specify a number of clusters to find, and allow the classification program to do the rest. This technique assumes, however, that the number of natural categories present in the data is known *a priori*, with data from different category clusters being well-separated.

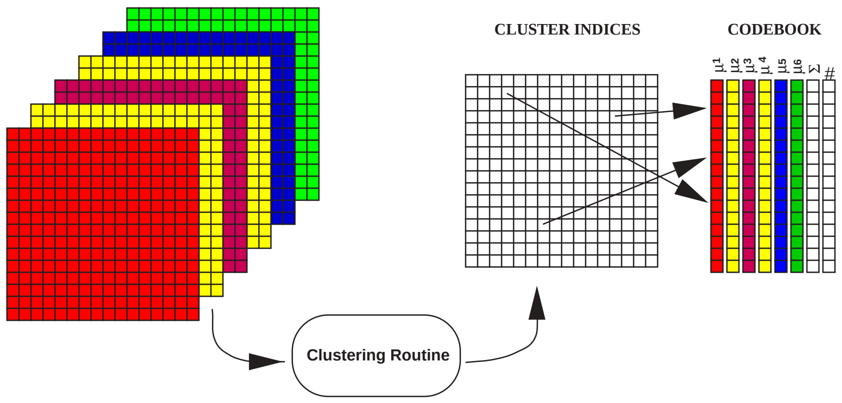


Figure 1: Clustered Representation of Multispectral Image Data

When using clustering methods for analyzing multispectral data, many people attempt to define a relatively small number of clusters – between 5 and 100 clusters, for example. Our technique relies on the fact that many clusters (between 256 and 4096) can be defined for the data. The method of data analysis and classification presented in this paper first preprocesses the data using a fast clustering algorithm. We cluster the data using a relatively large number of clusters (as compared to the number of categories we wish to define for the data), and then use the clustered data for analysis and classification. For many applications, there is no need for the original data after clustering is performed. Using the clustered data, we can efficiently manipulate computer displays as well as analyze and categorize data.

2. CLUSTERING METHODOLOGY

The basic principle of clustering (or vector quantization) is to take an original image (for our example, containing around 12,000,000 pixels with each pixel being represented by a seven-dimensional vector), and represent the same image using only a small number of unique pixel values. A codebook of N "best pixel values" to represent the image must first be generated by some iterative method (the "construction" phase of the clustering algorithm). Once we have generated these values, we step through the original image and assign each pixel to the cluster of the closest match existing in our codebook (the "projection" phase of the clustering algorithm). Figure 1 shows the clustered image representation, as compared to the original image representation.

In processing the data this way, two things have occured. First, we have reduced the volume of data needed to represent the image by a factor of seven. This is reflected by the fact that we now need only a single band of image data which contains indices into the codebook of reference vectors. Second, we have done a preliminary classification of the data; similar pixels in the image are now intrinsically associated with one another.

Since we would like the clustered data to adequately represent the original data, the selection of the codebook vectors is very important. By increasing the number of clusters, the accuracy of image representation can be improved. Depending on the application, we use between 256 and 4096 clusters for a typical TM quarter scene. The time required to cluster the image increases as the number of clusters increases. After clustering has provided a set of clusters, the statistics for each cluster are computed and stored in the codebook along with the cluster reference vectors. This is an important step because from these statistics, the combined statistics of the original data can easily be computed.

As an extra step, the cluster indices are sorted according to values stored in the mean vectors. Before this step is performed, the single two-dimensional band of cluster indices representing the data is meaningless unless it is associated with its codebook. By sorting the clusters according to values in a single dimension, or by the sum of multi-dimensional components in each one, a physical meaning is associated with each index. Bright pixels in the original data set will be associated with larger cluster indices than the darker pixels. The result will be an image which, when not associated with its codebook, can easily be displayed as a black and white image of the current scene.

3. CLUSTERING ALGORITHM

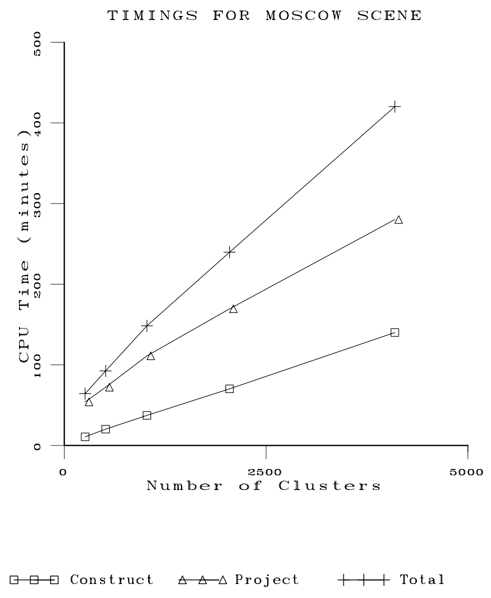
Many types of clustering methods have been developed and analyzed for use with different types of data [3, 5]. In general, many of these algorithms attempt to find a partitioning of a given data set that minimizes a predetermined cost function. The k-means clustering algorithm [4] attempts to minimize a squared error cost function by manipulating a set of k cluster centers. In particular, this algorithm tries to partition the data into k clusters, denoted by C_i, with the representative vector for each cluster ($\hat{x}_i$) being defined as the within-cluster mean:

$$\hat{x}_i = \frac{1}{N_i} \sum_{x_j \in C_i} x_j \tag{1}$$

This algorithm iteratively moves vectors between clusters in such a way as to minimize the total squared error:

$$Error = \sum_{i=1}^{k} \sum_{x_j \in C_i} \| x_j - \hat{x}_i \|^2 \tag{2}$$

This algorithm, however, becomes painfully slow when using very large data sets. One basic problem is that a tremendous number of vector distance calculations must be performed during both the "construction" and "projection" phases of the algorithm. Several methods have been developed to improve this situation [6, 7, 8]. Many of these schemes work very well in lower-dimensional spaces, but still tend to have a difficult time as the dimension of the problem and number of clusters increase.



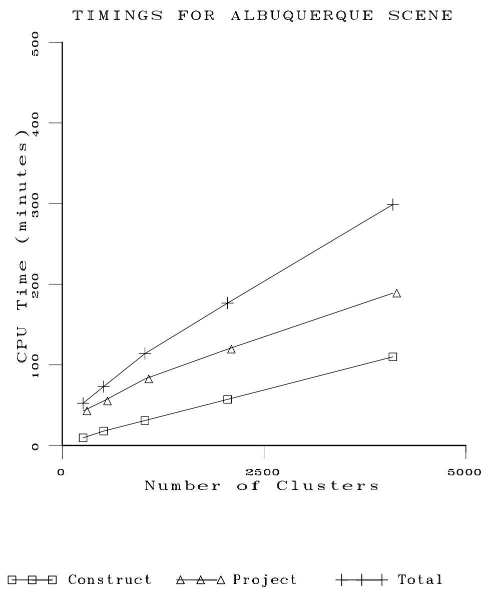


Figure 2: CPU Timings for Moscow and Albuquerque Scenes

We use a version of the nearest neighbor algorithm proposed in [9], where cluster positions are sorted along one of the axes for the data. This algorithm, like many others, does not continue to work effectively as the problem dimension increases. To combat this, we use the first principal component of the data as the axis on which to do the sort. This axis gives the best possible separation of the data.

Another major hindrance with the k-means algorithm is that the "construction" phase can require many passes through our tremendous data set to build the codebook. But this extra work is not necessary; the data has large amounts of redundant information. We use a monte carlo method for passing through the data, and only sample about 10 percent of the actual data.

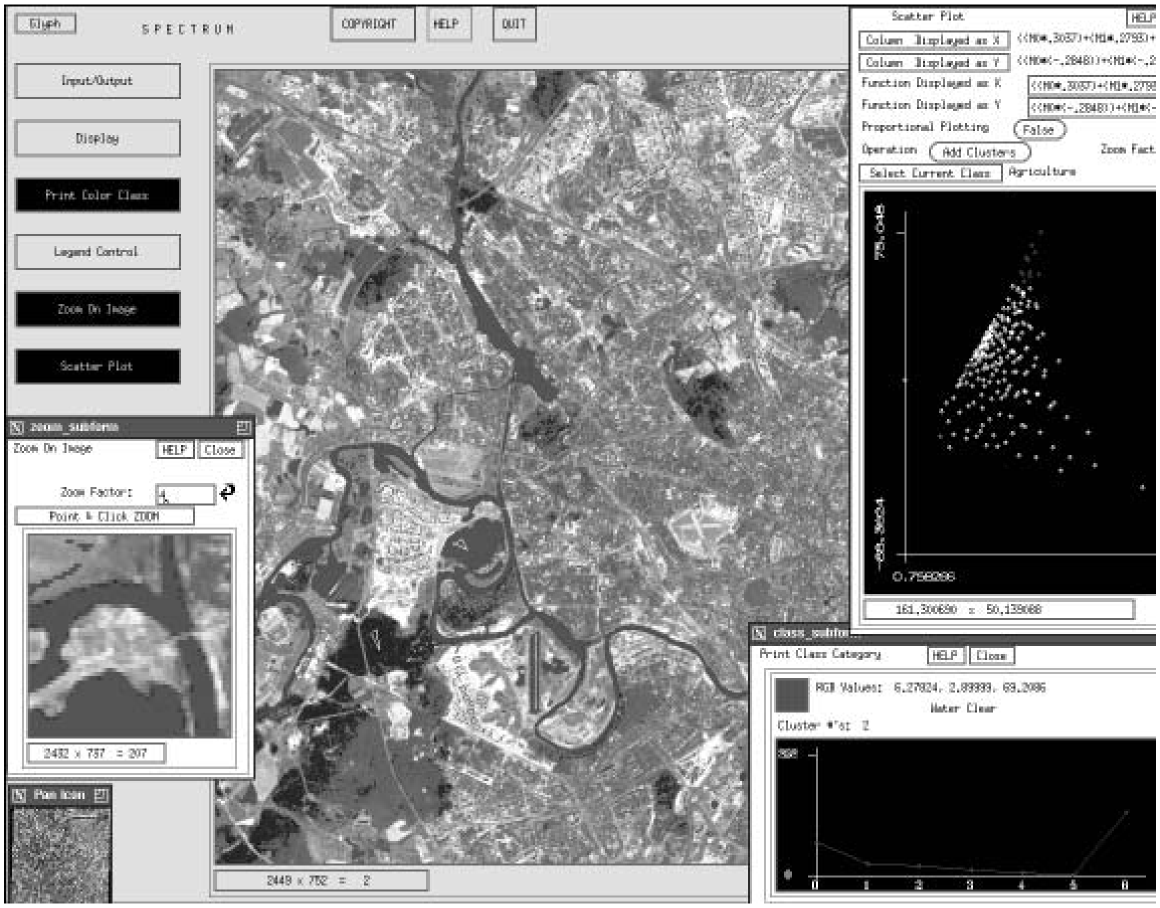
Our overall clustering technique yields the same results as the k-means algorithm, but converges much faster. Clustering times for a TM quarter scene (seven-dimensional data, 3000 rows by 3500 columns) of the Moscow and Albuquerque areas are shown in Figure 2. These were calculated on a desktop SUN SPARCstation IPX with 16 MB of RAM, and show CPU time required for clustering the data into 256, 512, 1024, 2048, and 4096 clusters. It is important to note that the execution time grows linearly as the number of clusters is increased. This is not a property of the algorithm in general, but it has seemed to hold true for the vast majority of real-world multispectral data sets (as well as most others) that the authors have encountered.

4. DATA ANALYSIS AND CLASSIFICATION

Once our TM scene has been clustered, it requires only one-seventh of the storage originally required, and the new clustered representation provides an opportunity to use common computer displays very efficiently. Since there are only N unique "vectors" representing the image, it takes on the order of N operations to manipulate the data as compared to 12 million operations before the clustering was performed. Calculating the vegetation vigor of pixels in a TM scene shows an example of the savings incurred by clustering. One measure of vegetation vigor commonly used by remote sensing specialists is (Band 4 - Band 3) / (Band 4 + Band 3). This transformation results in large values (bright pixels) for pixels representing healthy vegetation, and requires three operations at each pixel, or 36 million operations for the entire scene. If we first cluster the data to 256 clusters, we can use 8-bit computer displays effectively. Since the clustered image contains only 256 unique values, 768 operations are required for calculating the vegetation vigor, and the results can be directly mapped into the computer display look-up-tables (LUTs). While this is a simple type of operation, the same holds true for very complicated transformations such as the Tasseled Cap transformation, Karhunen-Loeve transformation, principal component analysis, etc.

Using a display package called SPECTRUM, developed by Los Alamos National Laboratory and the University of New Mexico, we are able to use any desktop workstation running Unix and Xwindows to analyze and categorize clustered data. Figure 3 shows a clustered TM scene of Moscow as displayed in SPECTRUM. A user can design and manipulate a legend that specifies categories of land cover, labels for each category, and pseudocolor representations to be used when categorizing geographic areas in the clustered image. SPECTRUM can manipulate the color map for the computer display using any transformation of the clustered data, and can display cluster positions on a two-dimensional scatter plot. Using these features, users are able to analyze data in a variety of ways. Data can be categorized by selecting areas with a known type of land cover, causing all associated pixels in the image to be given the same pseudocolor representation. Using the TM data, for example, a user could locate a wheat field, highlight the pixels in that field, and all other wheat fields in the entire image would be highlighted immediately. After categorization, an image can be written out showing the different geographic areas for the scene.

Using the scatter plot, cluster positions can be displayed in a two-dimensional space with axes specified by the user. Scientists can use this feature to interpret and categorize data by looking at different mathematical transformations of the cluster positions, while results of the process are updated in the currently displayed clustered image.



Zoom Factor: 4

2432 x 737 = 207

2449 x 752 = 2

Column Displayed as X <<M0*.3037)+<M1*.2793)+

Column Displayed as Y <<M0*<-.2848))+<M1*<-.2

Function Displayed as X <<M0*.3037)+<M1*.2793

Function Displayed as Y <<M0*<-.2848))+<M1*<-

Proportional Plotting False

Operation Add Clusters

Select Current Class Agriculture

161.300690 x 50.139088

RGB Values: 6.27824, 2.89999, 69.2086

Water Clear

Cluster #'s: 2

Figure 3: Manipulating Moscow Data with SPECTRUM

5. ERROR ANALYSIS

To examine the accuracy of the clustering relative to the number of clusters used, we will look at the average error per pixel introduced by the clustering, the distribution of these errors, and a Chi Square goodness-of-fit measure for different land cover training areas.

An 800 x 800 subsection was extracted from the original 3000 X 3500 original image of Moscow and the 3000 X 3500 clustered version of the image. An error image was created by averaging, for the 7 spectral bands, the absolute difference between the original image and the clustered image data. In the clustered image, each pixel is represented by the mean vector of the cluster to which it is assigned. It should be noted that errors for each of the individual bands is similar in magnitude and distribution to the average between the 7 spectral bands. The first plot in Figure 4 shows a plot of the average error per band per pixel and this error $\pm$ one standard deviation. The average error for 256 clusters is less than 2 digital numbers (DN) and drops to less than 1.25 DN average error for 4096 clusters. The maximum error over the subsection was much larger. There were a few popcorn clouds in the subsection and the error for the center pixel in the clouds ranged from about 70 DN for the 256 clusters image to about 30 for the 4096 clusters image but these outliers in the data set were few and it is an easy process to isolate them as outliers during the clustering process. The second plot in Figure 4 shows a histogram of the per pixel errors. The histograms show that even for the 256 clusters image almost all the pixels have an error within $\pm$ 3 DN.

Finally, we chose three training sites for each of 4 land cover types in the 3000 x 3500 Moscow image representing grass, soil, water, and forest. The training sites were located in the center of large uniform land covers and chosen

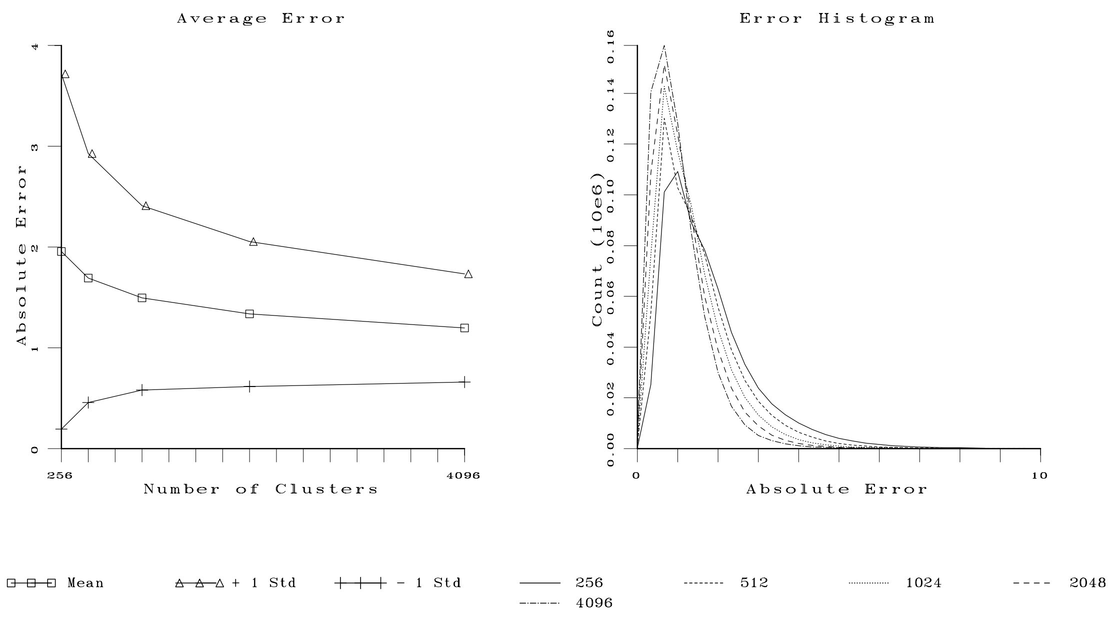


Figure 4: Per Pixel Errors for 800-by-800 Subsection of Moscow Scene

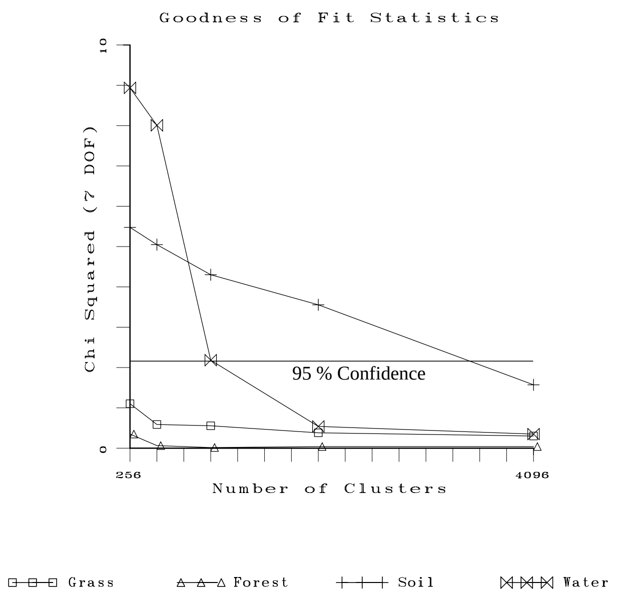


Figure 5: Chi-Squared Goodness of Fit for 7 DOF

as if they were to be used in a traditional supervised classification. We then did a Chi Square goodness-of-fit test to determine what our confidence was that the mean vectors representing the clustered data came from the same process which generated the statistics from the training sets in the original data. The results are shown in Figure 5. A Chi Square test with 7 degrees of freedom has a value of less than 2.83 for greater than 90% confidence and a value of less than 2.17 for a greater than 95% confidence. For an image with 4096 clusters all land covers had greater than 95% confidence. For 256 clusters, the goodness-of-fit values were much worse for the water training sets than for other land covers. The training sets for water were extremely uniform with a variance in each spectral band of less than 1.5. This means that even small differences between mean vectors yield large Chi Square values.

The errors introduced in a fine grain clustering of the multi-spectral data were not large enough to affect a level one land use classification. With 4096 clusters, the clustered image could be used to effectively represent the original data. Each land cover type was identified as easily as with the original image data.

6. CONCLUSIONS

Using a clustering method to do a preliminary classification of multispectral data provides data sets that can be rapidly categorized in an interactive fashion. A desktop workstation can be used to manipulate and analyze the preprocessed data in real time. Unlike present uses of clustering, where scientists attempt to find relatively small numbers of clusters in the data, our techniques define a large number of clusters to use. This data contains a relatively small number of unique representative vectors that must be categorized, as compared to millions of pixels in the raw data.

7. ACKNOWLEDGEMENTS

This work was performed under a U.S. Government contract (W-7405-ENG-36) by the Los Alamos National Laboratory, which is operated by the University of California for the U.S. Department of Energy.

8. REFERENCES

[1] Paul M. Mather. *Computer Processing of Remotely-Sensed Images.* St. Edmundsbury Press Ltd., Bury St. Edmunds, Suffolk, 1987.

[2] Robert A. Schowengerdt. *Techniques for Image Processing and Classification in Remote Sensing.* Academic Press, New York, New York, 1983.

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

[4] J.T. Tou and R.C. Gonzalez. *Pattern Recognition Principles.* Addison-Wesley, Reading, MA, 1974.

[5] A.K. Jain and R.C. Dubes. *Algorithms for Clustering Data.* Prentice Hall, Englewood Cliffs, NJ, 1988.

[6] Jerome H. Friedman, Jon Louis Bentley, and Raphael Ari Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software*, 3(3):209–226, 1977.

[7] J.L. Bentley, B.W. Weide, and A.C. Yao. Optimal expected-time algorithms for closest point problems. *ACM Transactions on Mathematical Software*, 6:563–580, 1980.

[8] M.E. Hodgson. Reducing the computational requirements of the minimum-distance classifier. *Remote Sensing of Environment*, 25:117–128, 1988.

[9] Jerome H. Friedman, Forest Baskett, and Leonard J. Shustek. An algorithm for finding nearest neighbors. *IEEE Transactions on Computers*, pages 1000–1006, October 1975.