Dimensionality Reduction for Useful Display of Hyperspectral Images

A Comparison of K-Nearest Neighbors, Artificial Neural Networks, and A Novel Probabilistic Overlay Technique

Jeffrey B. Cole

***Abstract*— To display hyperspectral images for human analysis, high-dimensional spectral data must be converted into a three band RGB image. Ideally, images for human analysis should be easy to interpret and supply useful visual information about the scene content. In an effort to explore non-linear methods of dimensionality reduction for the useful display of hyperspectral images, I compare and contrast results using k-nearest neighbors, artificial neural networks, and a novel probabilistic overlay technique. The methods are applied to several AVIRIS airborne hyperspectral images and results are compared using both quantitative and qualitative measures. All methods allow a user to specify a desired artificial color for endmembers of interest and the processed images use visual cues to indicate the probabilities that the endmembers exist in the scene. The results indicate that the newly proposed probabilistic overlay technique (when combined with a suitable classifier) outperforms the other methods and is a reasonable, intuitive, and efficient method for converting hyperspectral images into easily interpretable RGB images with useful endmember highlighting.**

I. Introduction

HYPERSPECTRAL images contain far more spectral information than can be displayed with a standard RGB monitor or printer. So the display of these images presents an interesting challenge. A number of different linear methods have already been proposed and implemented for useful dimensionality reduction of hyperspectral images.[1][3][6][8][9] Jacobson and Gupta, for example, used fixed linear spectral weighting envelopes to create natural looking imagery while still maximizing usefulness for human analysis.[1]

In my work, I extended this concept by looking at non-linear methods of dimensionality reduction that could augment the established linear methods and both create natural looking imagery and allow the user to specify desired highlighting colors for spectral endmembers-of-interest.

There are a number of benefits to using non-linear methods of dimensionality reduction for hyperspectral display. First, non-linear methods will allow a designer to indicate desired colors for endmembers of interest. Spectral information for endmembers can be taken from libraries or (as in this paper) hand selected from known regions in an image. In other words, a user could specify that she wants water to be blue, concrete to be red, and everything else in the image to look natural.

Non-linear methods would also have the benefit that the intensity of the artificial endmember colors could be used to indicate the probability that endmembers-of-interest exist in the scene. For example, saturated blue could indicate a high probability of water while faint red could indicate a low probability of concrete.

II. Goals for hyperspectral image visualization

A quantitative and qualitative analysis of hyperspectral display methods requires that we first establish our design goals. I propose the following design goals for displaying hyperspectral imagery with visual cues about endmember probabilities.

1) **Natural colors:** Processed images should look natural to the human eye and should not require extensive training for analysis. For example, trees should not look pink and water should not look yellow. Instead, the contents of a hyperspectral scene should be easy to interpret and color assignments should be intuitive.

2) **Natural contours:** Edges should be preserved in processed images and false contours should not be introduced during processing.

3) **Highlighting of target endmembers:** A designer should be able to specify desired artificial colors for target endmembers, and the saturation level of those artificial colors should indicate the probability of the endmember existing in the scene. Also, endmember highlighting should perform gracefully when numerous classes have high probabilities for the same pixel.

4) **Portability:** The training data should not be image specific. Rather, the transformation should perform well for any hyperspectral image.

5) **Computational ease:** The conversion from hyperspectral space to RGB space should be quick, enabling real-time interactivity. Also, conversion time should not increase when the training sample size increases.

III. Training Data

K-nearest neighbors and artificial neural networks are supervised learning and regression techniques, which require training data to indicate desired color mappings from hyperspectral space to RGB space. Thus a first task for these methods is to generate training data.

For this paper I developed a user interface tool in MATLAB that allows a designer to assign artificial colors to hand-selected regions in an image. The hyperspectral pixels in each region serve as example spectra for each class.

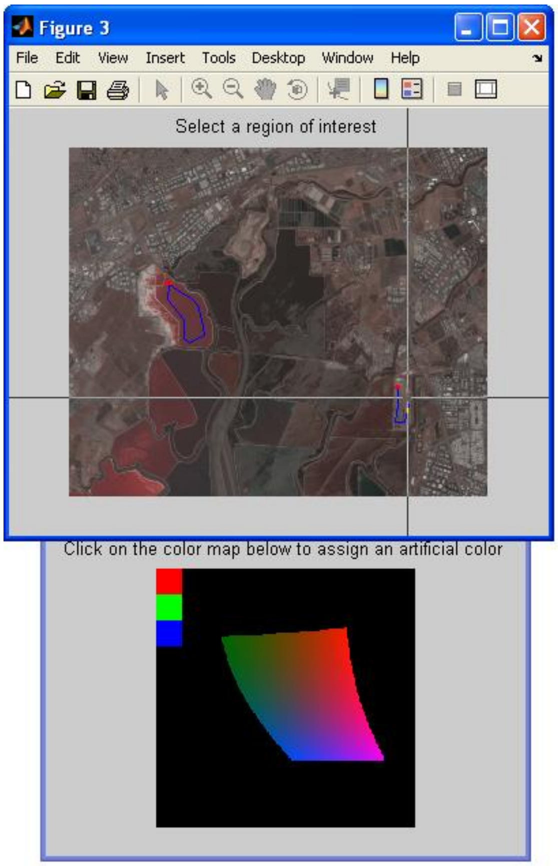


Fig. 1. Screenshot of the GUI used by the designer to specify regions of interest and assign them artificial colors.

Once the user has selected regions and assigned them artificial colors, the program automatically selects a few thousand random pixels to act as training data for natural RGB values. Essentially we want pixels of our known classes to be assigned designer artificial colors and we want all other pixels in the image to look natural. The randomly selected natural RGB training points are only used as training points if they are far enough in angle space from every pixel in the user selected regions.

More formally, a hyperspectral pixel $\vec{x}$ will only be used as a natural RGB training point if

$$\min_i \left(acos \left(\frac{\vec{x} \cdot \vec{y}_i}{\|\vec{x}\| \|\vec{y}_i\|} \right) \right) > threshold, \qquad (1)$$

where $\vec{y}_i$ are all the pixels in the user selected regions.

Each of these natural training points $\vec{x}_i$ are then assigned an RGB value using either the stretched-CMF linear projection[1] or by simply using the three bands that are closest to red, blue and green wavelengths respectively.

Fig. 2. Example of user selected regions with desired artificial colors assigned.

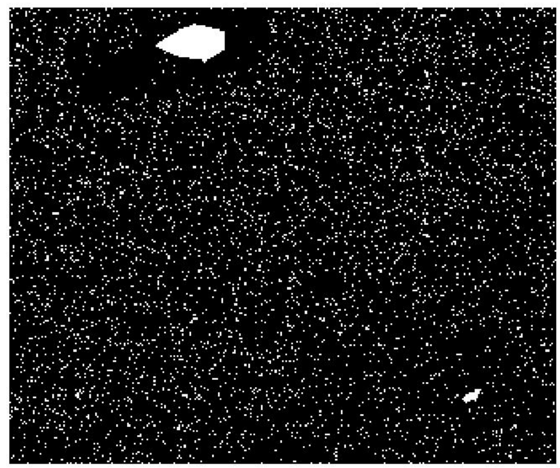


Fig. 3. Map showing the points in the image that were used as training data. Note that there are a couple hundred user selected pixels for each endmember class and five thousand randomly selected pixels for natural RGB training.

IV. Analysis Metrics

Keeping in mind the objectives for hyperspectral image display described in section II, I will use the following metrics for comparing the results of each method.

Quantitative: The mean squared error between the training data and the processed image will be used as a quantitative metric for the performance of each method. Also the processing time required for each method will be used to compare computational ease.

Qualitative: We will judge the quality of our processed images by checking that:

1) colors look natural and are easy to interpret,
2) no artificial contours or colors have appeared,
3) target endmembers are highlighted
4) saturation levels of artificial colors indicate probabilities that endmembers exist in a given pixel, and
5) the method performs well when applied to a diverse set of hyperspectral images.

V. K-Nearest Neighbor Method

The first method I applied was an exponentially decaying k-nearest neighbor (KNN) regression technique. I defined the distance between any two hyperspectral pixels as the angle they form in hyper-space:

$$D(\vec{x}, \vec{y_i}) = acos\left(\frac{\vec{x} \cdot \vec{y_i}}{\|\vec{x}\|\|\vec{y_i}\|}\right). \tag{2}$$

For every pixel in an image, I simply find the k nearest training points in the training set and then generate an output color in LAB space by combining the training output values with exponentially decaying weights.

For the testing described in this paper, the number of neighbors k was varied from 1 to 50 and resulting images are shown in tables I and II.

We see from the results that qualitativly, KNN performs poorly. When $K = 1$ we have overfit our data, and as K increases the image becomes washed out. This is because the only allowable output colors are linear combinations of the training data colors. Although we have not introduced any false contours, we have lost a large portion of the distinguishing features in our image. Another problem with KNN is that it does not pass the portability test. When the training data is used to process a new image (see table II) we cannot resolve any new colors because we can only make linear combinations of our training data.

Lastly, KNN has the significant drawback that it is extremely computationally intensive. Analyzing a single image can take hours. And as the training set is increased the processing time will increase exponentially.

VI. Artificial Neural Network Method

Artificial neural networks were also applied to the task of dimensionality reduction. Using the same training data as was used for the KNN method, a neural-net with a single hidden layer (10 nodes) was trained to convert 219 band hyperspectral data into the desired RGB values.

Results for this method are shown in table III. Note that the neural net performs much better than KNN. The target endmember classes are highlighted with the appropriate artificial colors and the rest of the image pixels look natural. Contours are preserved and the image is intuitive and easy to interpret.

Another benefit of the neural network method is its ease of computation. Although training the network can take hours, a single image can be processed in less than a second, making it reasonable to use this method for real-time analysis.

However, one major drawback of this method is that it is difficult to predict how the net will handle new spectral inputs. As can be seen in table III, there is a fair amount of green introduced into our processed image. Since our training colors are only red and blue, the color green is impossible for a human viewer to interpret.

So we see that the neural net performs better than KNN, but it fails one of our quality metrics because it introduces unnatural colors that are not easy to interpret.

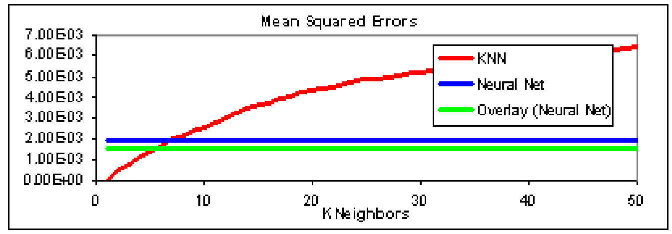


Fig. 4. Mean squared error comparison for each method.

VII. Probabilistic Overlay Method

The false color problem of the neural net method motivated me to consider a new technique for hyperspectral image display. Rather than using a neural net to both make a soft classification and assign output colors, I tried using the neural net to only detect the probability of each class. I then used those probabilities and the user selected artificial colors to generate an overlay map that could finally be combined with the natural looking RGB image.

More formally, for a given hyperspectral pixel $\vec{x}$ we can generate an overlay color $\vec{c}_{overlay}$ using the following equation:

$$\vec{c}_{overlay} = \sum_i \left(\frac{P_i}{\sum P_i}\vec{c}_i\right) \tag{3}$$

where P_i is the probability that $\vec{x}$ is an endmember of class i with the associated artificial color $\vec{c}_i$.

We can then generate our output color $\vec{c}_{out}$ by combining the overlay color with the natural RGB value $\vec{c}_{natural}$ using the following equation:

$$\vec{c}_{out} = (1 - \max_i(P_i))\vec{c}_{natural} + \max_i(P_i)\vec{c}_{overlay} \tag{4}$$

where we generated $\vec{c}_{natural}$ using either the stretched-CMF linear projection[1] or by simply using the three bands that are closest to the red, blue and green wavelengths respectively.

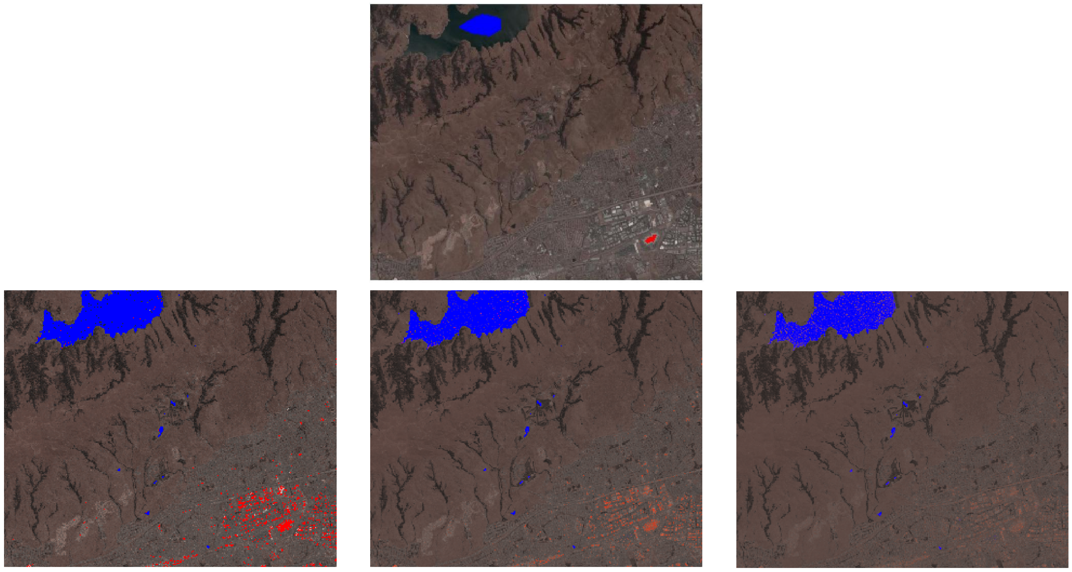
Results for this probabilistic overlay method are shown in tables IV and V. Note that the resulting images have all the benefits of the neural net method with the added benefit that there are never any colors that are not either naturally occurring or a linear combination of the user selected artificial colors.

Table V suggests that the overlay method will perform well even when there are a large number possible endmember classes of interest.

VIII. Analysis and Conclusions

Figure 4 shows the mean squared errors for KNN, neural nets, and the overlay technique. We can see conclusively that the probabilistic overlay method is both qualitatively and quantitatively better than the other two methods.

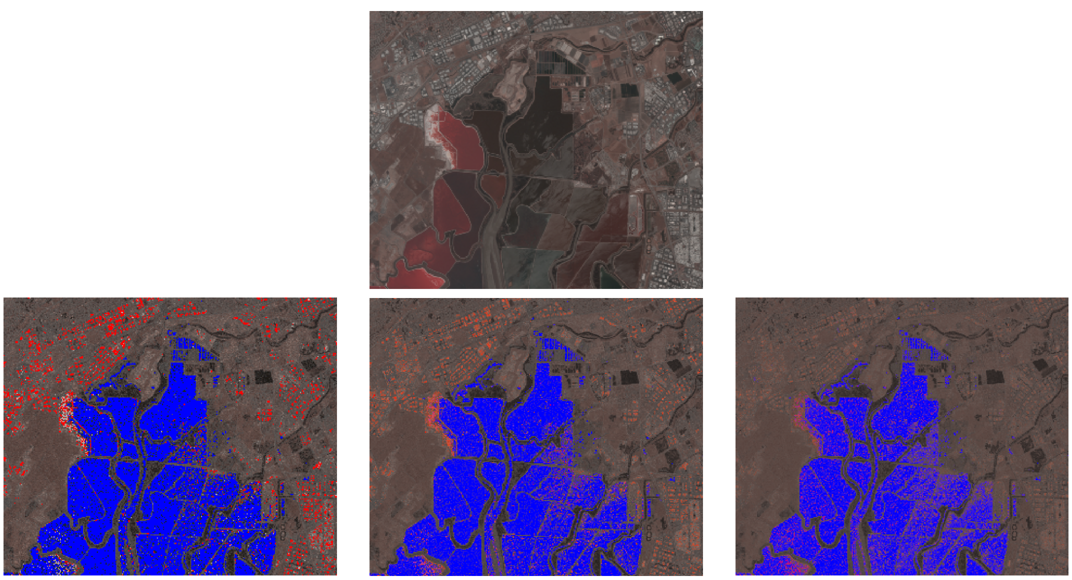
It should also be noted that the overlay method could be used in combination with any hyperspectral classifier [5], not just the basic neural-net that was used in this work.



K=1 K=10 K=50

TABLE I

RESULTS WHEN K NEAREST NEIGHBOR METHOD IS APPLIED TO THE SAME HYPERSPECTRAL IMAGE IT WAS TRAINED ON.



K=1 K=10 K=50

TABLE II

RESULTS WHEN K NEAREST NEIGHBOR METHOD IS APPLIED TO AN ALTERNATE IMAGE THAT DOES NOT CONTAIN ANY OF THE TRAINING POINTS.

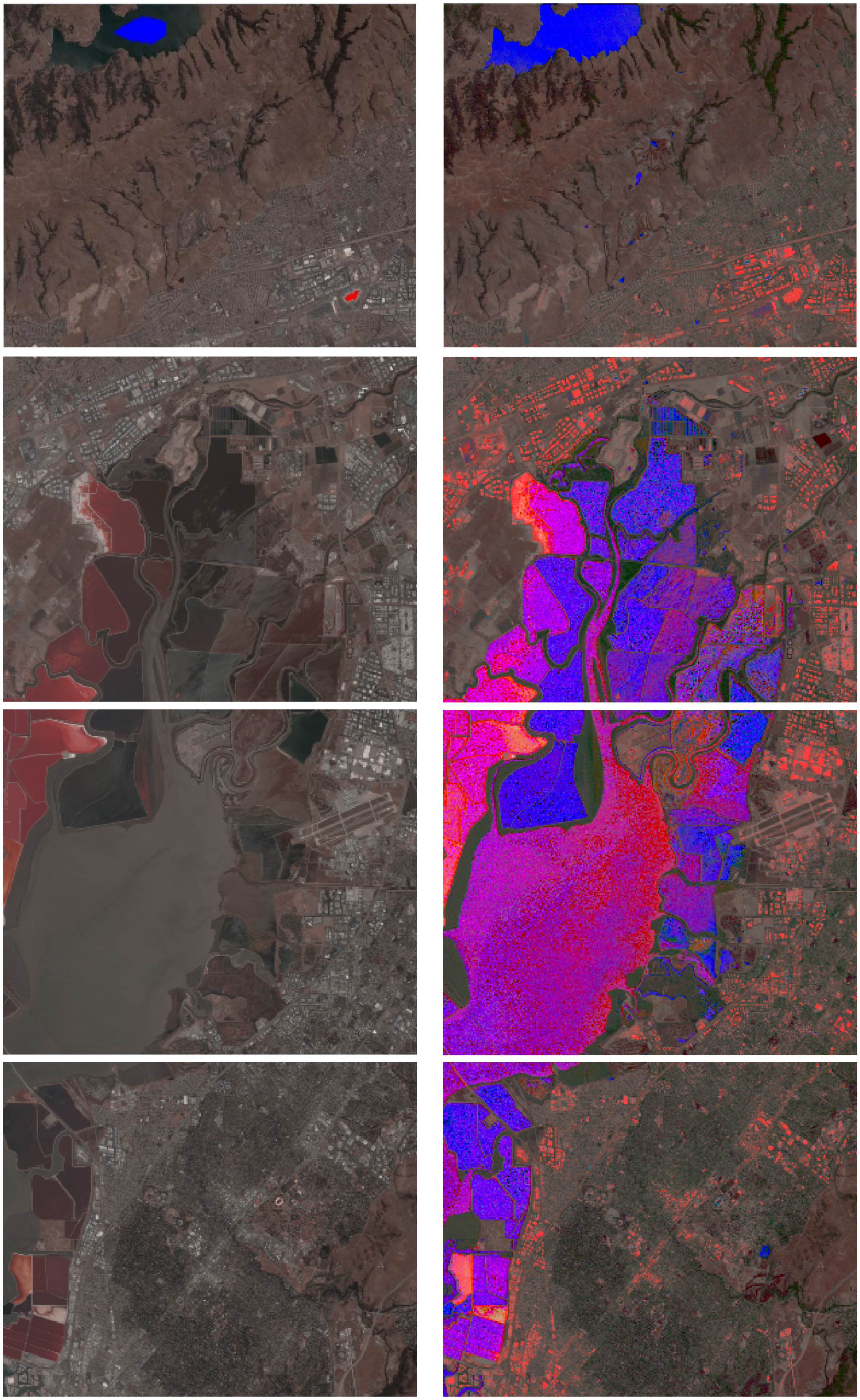


TABLE III

RESULTS WHEN NEURAL NETWORK METHOD IS APPLIED TO A SERIES OF HYPERSPECTRAL IMAGES. NOTE THAT ONLY THE FIRST IMAGE CONTAINS PIXEL VALUES THAT WERE USED DURING THE TRAINING OF THE NET.

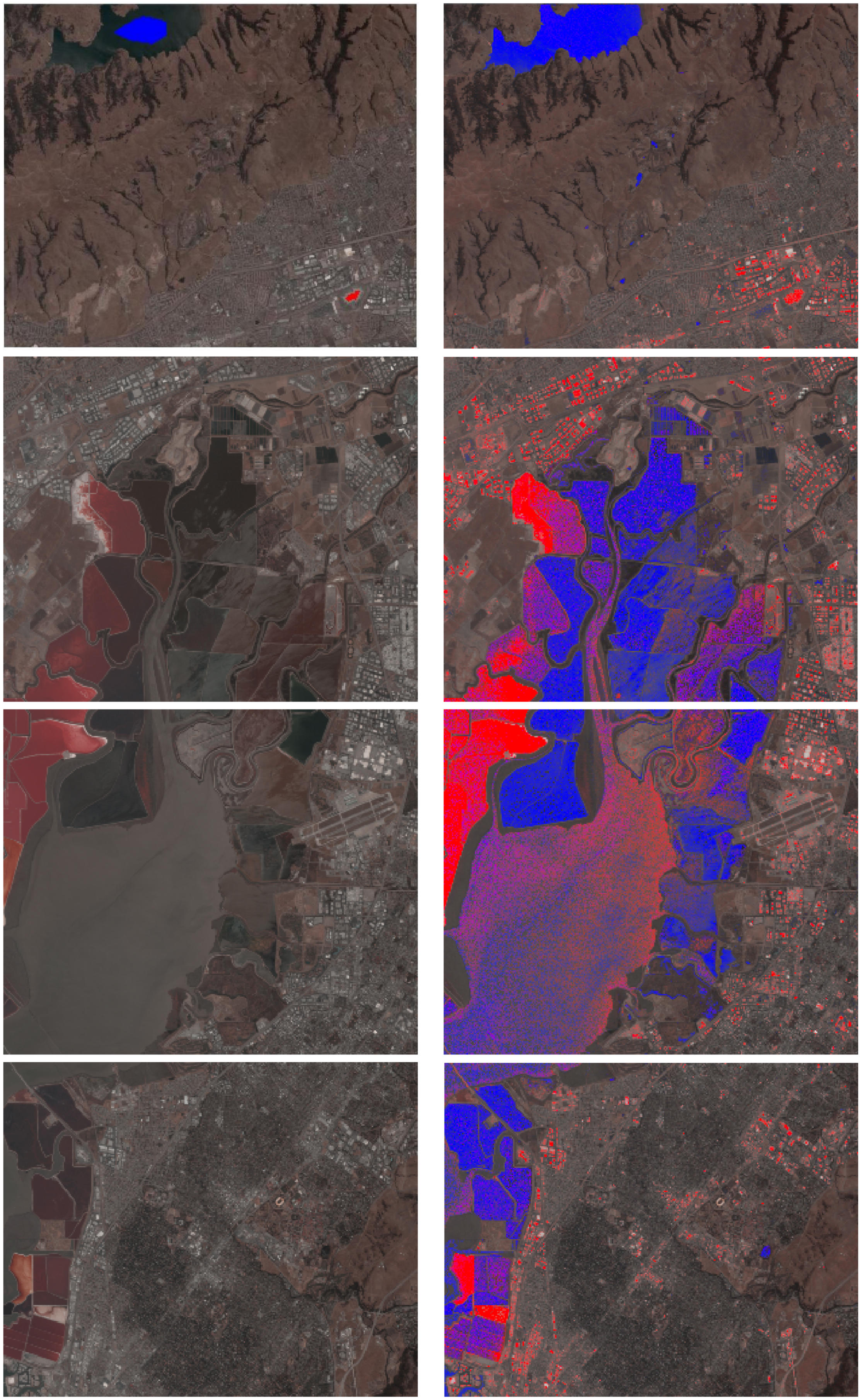


TABLE IV

Results when the overlay method is applied to a series of hyperspectral images with 2 target endmember classes. Note that only the first image contains pixel values that were used during the training of the net.

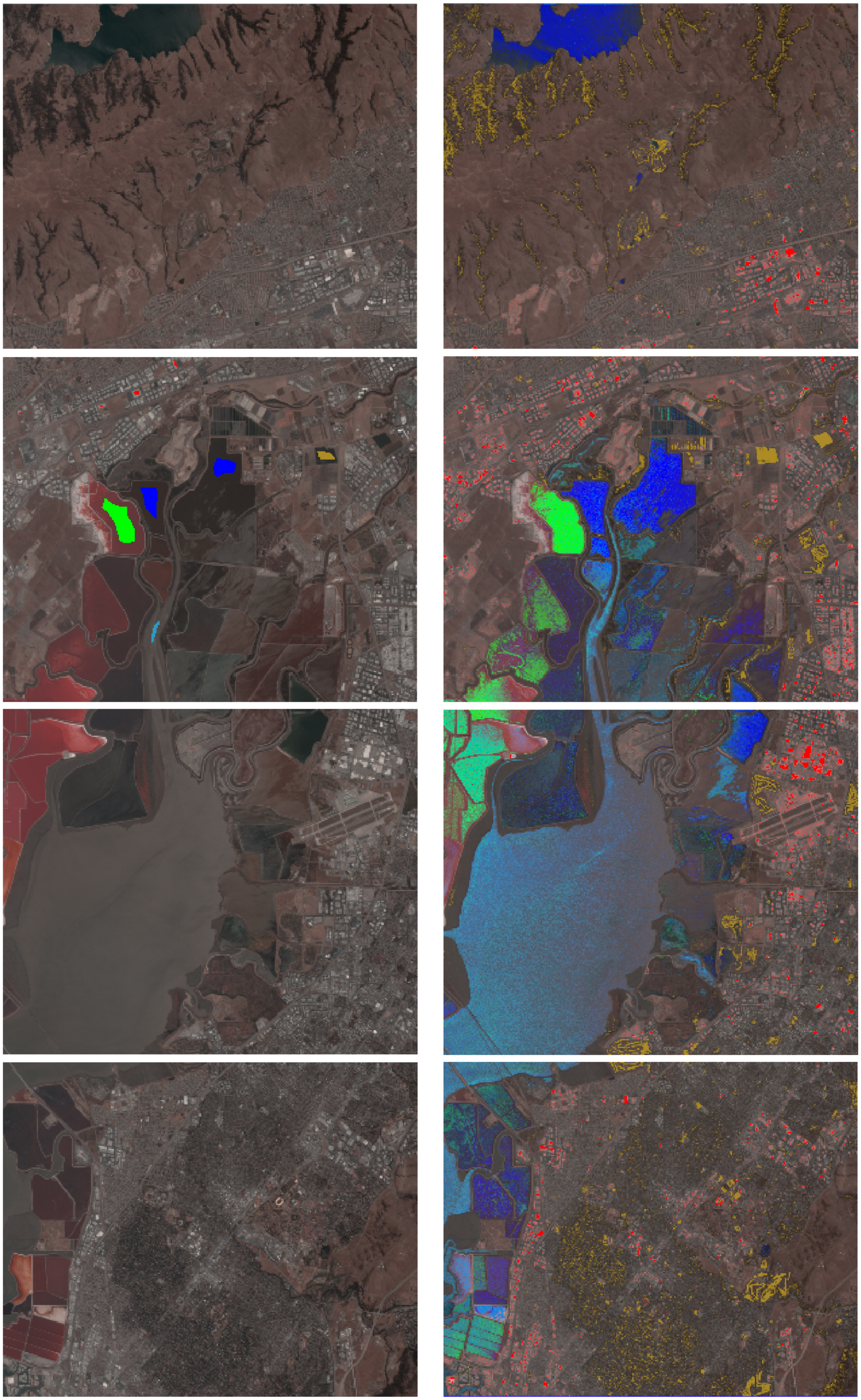


TABLE V

RESULTS WHEN THE OVERLAY METHOD IS APPLIED TO A SERIES OF HYPERSPECTRAL IMAGES WITH 5 TARGET ENDMEMBER CLASSES.

REFERENCES

[1] N. Jacobson and Gupta, "Design goals and solutions for display of hyperspectral images," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 43, pp. 2684–2692, 2005.

[2] M. Gupta, E. Garcia, and A. Stroilov, "Learning icc profiles for custom color enhancements with adaptive neighborhoods," *In review by the IEEE Trans. on Image Proc.*, 2006.

[3] C.-I. C. Harsanyi, "Hyperspectral image classification and dimensionality reduction: An orthogonal subspace projection approach," *IEEE Transactions On Geoscience and Remote Sensing*, vol. 32, pp. 779–785, 1994.

[4] C. Bachmann, T. Ainsworth, and R. Fusina, "Exploiting manifold geometry in hyperspectral imagery," *IEEE Transactions On Geoscience and Remote Sensing*, vol. 43, pp. 441–454, 2005.

[5] J. Qian Du; Szu, H.; Buss, "Lagrange constrained neural network-based approach to hyperspectral remote sensing image classification," *Neural Networks and Signal Processing, 2003. Proceedings of the 2003 International Conference on*, vol. 1, pp. 14–17, 2003.

[6] J. Xiuping Jia; Richards, "Segmented principal components transformation for efficient hyperspectral remote-sensing image display and classification," *Geoscience and Remote Sensing, IEEE Transactions on*, vol. 37, pp. 538–542, 1999.

[7] ——, "Cluster-space representation for hyperspectral data classification," *Geoscience and Remote Sensing, IEEE Transactions on*, vol. 40, pp. 593–598, 2002.

[8] A. D. D. O. R. Tyo, J.S.; Konsolakis, "Principal-components-based display strategy for spectral imagery," *Geoscience and Remote Sensing, IEEE Transactions on*, vol. 41, pp. 708–718, 2003.

[9] R. Tyo, J.S.; Olsen, "Pc-based display strategy for spectral imagery," *Advances in Techniques for Analysis of Remotely Sensed Data, 2003 IEEE Workshop on*, pp. 27–28, 2003.

[10] M. E. A. Wessels, J.; Buchheit, "The development of a high performance, high volume distributed hyperspectral processor and display system," *Geoscience and Remote Sensing Symposium, 2002. IGARSS '02. 2002 IEEE International*, vol. 4, pp. 2519–2521, 2002.