2021-09

ANALYSIS OF IMAGE ENHANCEMENT ALGORITHMS FOR HYPERSPECTRAL IMAGES

Rivera, Armando A.

Monterey, CA; Naval Postgraduate School

http://hdl.handle.net/10945/68378

NAVAL
POSTGRADUATE
SCHOOL

MONTEREY, CALIFORNIA

THESIS

ANALYSIS OF IMAGE ENHANCEMENT ALGORITHMS FOR HYPERSPECTRAL IMAGES

by

Armando A. Rivera

September 2021

Thesis Advisor:	James W. Scrofani
Co-Advisor:	William Williamson

REPORT DOCUMENTATION PAGE	*Form Approved OMB No. 0704-0188*

1. AGENCY USE ONLY ***(Leave blank)***	**2. REPORT DATE** September 2021	**3. REPORT TYPE AND DATES COVERED** Master's thesis

4. TITLE AND SUBTITLE ANALYSIS OF IMAGE ENHANCEMENT ALGORITHMS FOR HYPERSPECTRAL IMAGES	**5. FUNDING NUMBERS**
6. AUTHOR(S) Armando A. Rivera	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A	**10. SPONSORING / MONITORING AGENCY REPORT NUMBER**

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release. Distribution is unlimited.	**12b. DISTRIBUTION CODE** A

13. ABSTRACT (maximum 200 words)

This thesis presents an application of image enhancement techniques for color and panchromatic imagery to hyperspectral imagery. In this thesis, a combination of previously used algorithms for multi-channel images are used in a novel way to incorporate multiple bands within a single hyperspectral image. The steps of the image enhancement include image degradation, image correlation grouping, low-resolution image fusion, and fused image interpolation. Image degradation is accomplished through a Gaussian noise addition in each band along with image down-sampling. Image grouping is done through the use of two-dimensional correlation coefficients to match bands within the hyperspectral image. For image fusion, a discrete wavelet frame transform (DWFT) is used. For the interpolation, three methods are used to increase the resolution of the image: linear minimum mean squared error (LMMSE), a maximum entropy algorithm, and a regularized algorithm. These algorithms are then used in combination with a principal component analysis (PCA). The use of PCA is used for data compression. This saves time at the expense of increasing the error between the true image and the estimated hyperspectral image after PCA. Finally, a cost function is used to find the optimal level of compression to minimize the error while also decreasing computational time.

14. SUBJECT TERMS hyperspectral, image, enhancement, discrete wavelet frame transform, DWFT, linear minimum mean squared error, LMMSE, principal component analysis, PCA	**15. NUMBER OF PAGES** 105
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UU

THIS PAGE INTENTIONALLY LEFT BLANK

ANALYSIS OF IMAGE ENHANCEMENT ALGORITHMS FOR HYPERSPECTRAL IMAGES

Armando A. Rivera
Lieutenant Junior Grade, United States Navy
BS, U.S. Naval Academy, 2018

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
September 2021

Approved by: James W. Scrofani
Advisor

William Williamson
Co-Advisor

Douglas J. Fouts
Chair, Department of Electrical and Computer Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

This thesis presents an application of image enhancement techniques for color and panchromatic imagery to hyperspectral imagery. In this thesis, a combination of previously used algorithms for multi-channel images are used in a novel way to incorporate multiple bands within a single hyperspectral image. The steps of the image enhancement include image degradation, image correlation grouping, low-resolution image fusion, and fused image interpolation. Image degradation is accomplished through a Gaussian noise addition in each band along with image down-sampling. Image grouping is done through the use of two-dimensional correlation coefficients to match bands within the hyperspectral image. For image fusion, a discrete wavelet frame transform (DWFT) is used. For the interpolation, three methods are used to increase the resolution of the image: linear minimum mean squared error (LMMSE), a maximum entropy algorithm, and a regularized algorithm. These algorithms are then used in combination with a principal component analysis (PCA). The use of PCA is used for data compression. This saves time at the expense of increasing the error between the true image and the estimated hyperspectral image after PCA. Finally, a cost function is used to find the optimal level of compression to minimize the error while also decreasing computational time.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

AVIRIS	Airborne Visual/Infrared Imaging Spectrometer
DFT	Discrete Fourier Transform
DWFT	Discrete Wavelet Frame Transform
LMMSE	Linear Minimum Mean Squared Error
LWIR	Long-Wave Infrared
MSE	Mean Squared Error
MWIR	Mid-Wave Infrared
NIR	Near-Infrared
RGB	Red, Green, and Blue
SLIC	Simple Linear Clustering Algorithm
SNR	Signal-to-Noise Ratio
SWIR	Short-wave Infrared
PCA	Principal Component Analysis
2D	Two-Dimensional
3D	Three-Dimensional

THIS PAGE INTENTIONALLY LEFT BLANK

EXECUTIVE SUMMARY

Hyperspectral sensors capture images that have an enormous amount of spectral data. The spectral resolution of the hyperspectral wavelength bands allows for a better classification of material chemical composition compared to other remote sensors. Material classification is possible because different molecular structures of various chemicals absorb and reflect different wavelengths of light. Hyperspectral imagery has enabled many new remote sensing applications in agriculture, geology, medicine, and many more fields. An example of a hyperspectral image is shown in Figure 1.

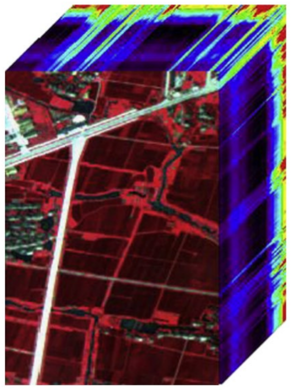


Figure 1. Hyperspectral Image Example. Source: [1].

In Figure 1, the hyperspectral image can be thought of as a series of many images, where each image captures what the ground looks like at a particular wavelength of light. The airborne sensor receives the emitted wavelengths from the ground and counts the number of photons received at that exact wavelength. The series of images form a three-dimensional cube of values, where x and y axes determine the location, and the z-axis defines the wavelength intensity. Lighter colors in this image indicate that the ground material reflects or emits that specific wavelength very well. This can be used to identify materials by comparing the wavelengths reflected with a database of known values.

Hyperspectral images separate the detected photons into different wavelength bands. By separating the detection of a finite number of photons into many different wavelength bands, each band individually does not receive many photons. The smaller number of photons available requires larger pixels and longer integration times to generate enough signal-to-noise-ratio to provide an accurate radiance measurement. In turn, larger pixels require larger optics to get the same spatial resolution. In airborne and space-based remote sensing systems, there is a practical upper bound on the size and weight of optics, which often means that hyperspectral systems have a lower spatial resolution as a result. Lower spatial resolution limits the usefulness of a hyperspectral image when used to classify and segment an image into groups of materials. Without enough spatial resolution, classification methods that sample small areas in clusters, such as the Simple Linear Iterative Clustering algorithm (SLIC), may not accurately define the boundaries of different compounds in an image. These classification methods are improved by having a larger sample size of data.

An approach to overcome the low spatial resolution limitation is image enhancement through image processing techniques. It is possible to fuse information in the data by leveraging the similarities across the low-resolution bands. After fusing the data, the next step is to interpolate the fused data to a new higher resolution image with more useful information for segmentation and classification.

This investigation aims to determine which interpolation methods work best to improve the hyperspectral image signal-to-noise ratio. We have compared six different interpolation methods to determine which methods increase the quality of the image. The first three methods use more complex interpolation algorithms in conjunction with image fusion. The three interpolation algorithms used with image fusion are a linear minimum mean squared error (LMMSE) algorithm, a maximum entropy algorithm, and a regularization algorithm. The following three methods act as a baseline for comparison and are simpler interpolation methods that do not use fused images. The three simple interpolation methods are a nearest neighbor, bilinear, and bicubic interpolation. The system diagram that visually represents this process is shown in Figure 2.

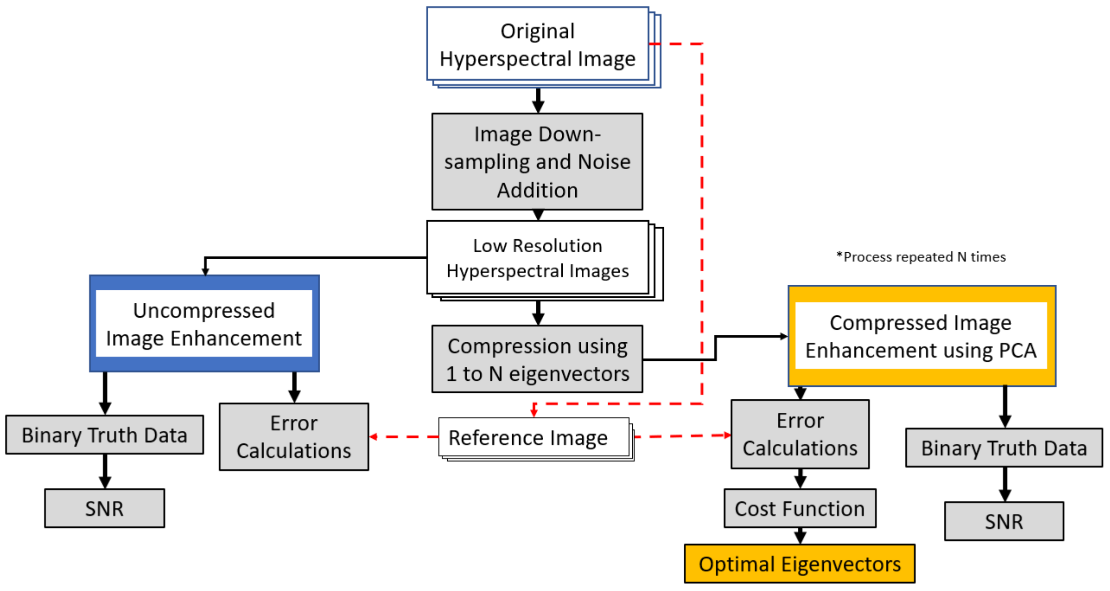

Original Hyperspectral Image
Image Down-sampling and Noise Addition
Low Resolution Hyperspectral Images
*Process repeated N times
Uncompressed Image Enhancement
Compression using 1 to N eigenvectors
Compressed Image Enhancement using PCA
Binary Truth Data
Error Calculations
Reference Image
Error Calculations
Binary Truth Data
SNR
Cost Function
SNR
Optimal Eigenvectors

Figure 2. Image Enhancement System Diagram.

The image enhancement system diagram starts with the original hyperspectral image. Since several wavelength bands are in the hyperspectral image, this first box has several layers representing the multiple bands. Gray boxes represent functions in the algorithm. The next step is to down-sample and add noise to the high-resolution image to form a degraded low-resolution image. Down-sampling and degradation allow for a comparison of the interpolation result to the original unaltered data. After the down-sampling, the low-resolution image is sent to two different parts of the investigation. Both the "blue" and "gold" side use all six interpolation methods. After interpolation, the signal-to-noise ratio (SNR) and error are calculated. The left "blue" side is the uncompressed experiment, where the down-sampled and degraded low-resolution images are fed directly into the complex and simple interpolation algorithms. The right "gold" side is the compressed experiment, where the degraded low-resolution images are further compressed in size using a principal component analysis (PCA). The compressed section of the investigation determines what methods are best for small data sizes and the optimal compression level. On the gold side, time to calculate each interpolated image increases as the level of compression decreases. Time elapsed while calculating the estimated image and the error between the estimate and the original image are used to generate a cost. The method with the least amount of cost is the most effective.

After testing these six different methods of image enhancement, we have concluded that of the methods tested, the best algorithms for increasing signal-to-noise ratio are the Linear Minimum Mean Squared Error (LMMSE) algorithm and regularization algorithm. These two algorithms had the highest SNR performance of all six methods in the uncompressed data testing. For large datasets, image fusion used in conjunction with these LMMSE and regularization proves to be effective in image enhancement.

The second goal of this investigation is to determine a method of selecting an optimal number of principal components (eigenvectors) to use for image enhancement when a principal component analysis was used to compress the hyperspectral data. As more eigenvectors are used, the error between the reconstructed image and the original image decreases. The time taken to conduct the compressed image enhancement is measured for all methods along with the error measurements. The error and the time elapsed are used in a cost function. The minimum of the cost function determines the optimal number of eigenvectors that balances the needs of estimated image accuracy and lower computational time. The visual representation of both the uncompressed and compressed image enhancement experiments can be seen in Figure 3.

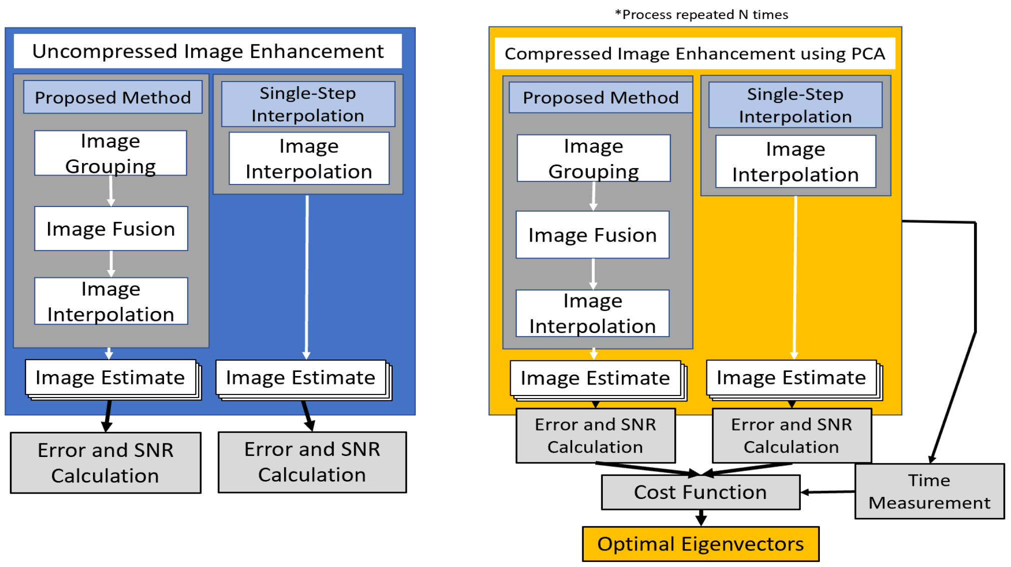


Figure 3. Uncompressed and Compressed Image Enhancement Process.

In Figure 3, and as stated previously, both the blue and gold experiments use all six of the interpolation methods discussed previously. The three more complex methods fall under the box named “proposed method,” and the simple interpolation methods are under “single-step interpolation.” In the proposed method track, image grouping and fusion are used with the complex interpolation, whereas the single-step interpolation methods do not use image grouping and fusion. Separating these two tracks is done to compare the proposed image enhancement to the three simpler methods. If the proposed methods cannot produce better results than the single-step interpolation, then they are not methods with any practical use. As stated previously, the LMMSE and Regularization algorithm had the highest SNR in the uncompressed dataset. In the PCA compressed dataset, the highest performers were the LMMSE method for the Indian Pines image and the bilinear method for the Salinas image. A key detail is that the image of Indian Pines is approximately twice the size of the Salinas image. These results suggest that as the hyperspectral cube becomes larger, the more complex methods improve in producing an enhanced image from the dataset. However, as the image is compressed further, these methods break down and cannot produce accurate estimations of the original image. It is our recommendation that for hyperspectral image enhancement, data compression should not be used. The cost savings due to computational time reduction can be mitigated with stronger computational hardware and code optimization. Additionally, it is recommended that image fusion used in conjunction with LMMSE and regularization methods be further investigated for use on a broader array of images than what is tested in this experiment.

Reference

[1] H. Zhang, L. Zhang, and H. Shen, “A super resolution reconstruction algorithm for hyperspectral images,” *Signal Processin*g, vol. 92, pp. 2082–2096, 201.

ACKNOWLEDGMENTS

I want to express my gratitude to my wife, Sheila, who kept me motivated and reminded me of my purpose.

THIS PAGE INTENTIONALLY LEFT BLANK

I. BACKGROUND

A. INTRODUCTION

Remote sensing allows a user to understand the physical properties of various locations while maintaining a large range to the scene being viewed. Hyperspectral imagery has been used for remote sensing since the early 1990s [1]. Before this, traditional remote sensing consisted of panchromatic images, color images, or multi-spectral images with only a small number of wavelength bands. These were primarily based upon the visible spectrum. Images typically presented to the human eye consist of three bands: red, green, and blue (RGB), which correspond to approximately 0.6, 0.5, and 0.4 microns. Overhead imagery also made extensive use of panchromatic images. In panchromatic imaging, light is collected across the entire visible spectrum, combined and represented by a single intensity value per pixel, resulting in a grayscale image. Panchromatic imagery is advantageous because more photons are available in a broad spectrum of wavelength bands than what would be available in narrower bands with only specific wavelengths. In panchromatic imaging, pixels can be smaller, and thus the image resolution is improved. Image analysis has historically required manual inspection of captured photographs. In times of conflict, the location, material, and personnel of an adversary could be identified by analyzing panchromatic images. Unfortunately, as these methods became known, adversaries created ways of disguising themselves using camouflage to appear as background noise [2].

To counter deception tactics, more advanced sensors on surveillance aircraft were created that could detect anomalies normally invisible to the human eye. During the early 1970s, multi-spectral sensors that could capture wavelengths of light invisible to the human eye became widely available. Remote sensing cameras such as the Landsat satellite have captured multi-spectral images of the surface of the Earth. Multi-spectral images allowed for monitoring crop health, flood, fire risk, and other environmental phenomena [1]. "Multi-spectral" is defined as the visible spectrum along with the near-infrared (NIR), short-wave infrared (SWIR), mid-wave infrared (MWIR), and long-wave infrared (LWIR) [1]. Multi-spectral imagery was beneficial for object detection and surveillance. Multi-spectral imaging, however, had its limitations in material classification due to the small number of bands that were acquired from an airborne sensor. A visual comparison of multi-spectral and hyperspectral imaging can be seen in Figure 1.

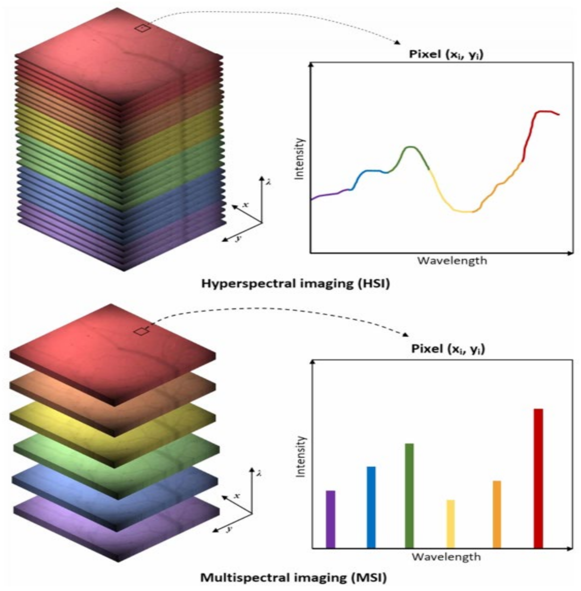


Figure 1. Hyperspectral / Multispectral Comparison. Source: [3].

In Figure 1, the multi-spectral image has only a few discrete wavelength bands. This limits its ability to classify materials detected on the ground since only a few wavelengths are sampled. On the other hand, the hyperspectral image has a greater number of bands, effectively creating a continuous line of intensity values at each wavelength. This allows for a greater amount of wavelength samples.

Hyperspectral imaging sensors allow more detailed spectral data to be acquired from a scene than multi-spectral sensors. This is accomplished through larger optical detector arrays combined with various prisms or interferometers that can capture the spectral content of a ground scene with much higher spectral resolution. The image generated by a hyperspectral instrument can be visualized as a three-dimensional (3D) array of values referred to as a "data cube." The first and second dimensions are the x and y coordinates. The third dimension contains the intensity values at each pixel on each wavelength band. An example of a ground scene captured is shown in Figure 2. This figure is an example of how the Airborne Visual and Infrared Imaging Spectrometer (AVIRIS) sensor captures different spectral plots at each pixel location.

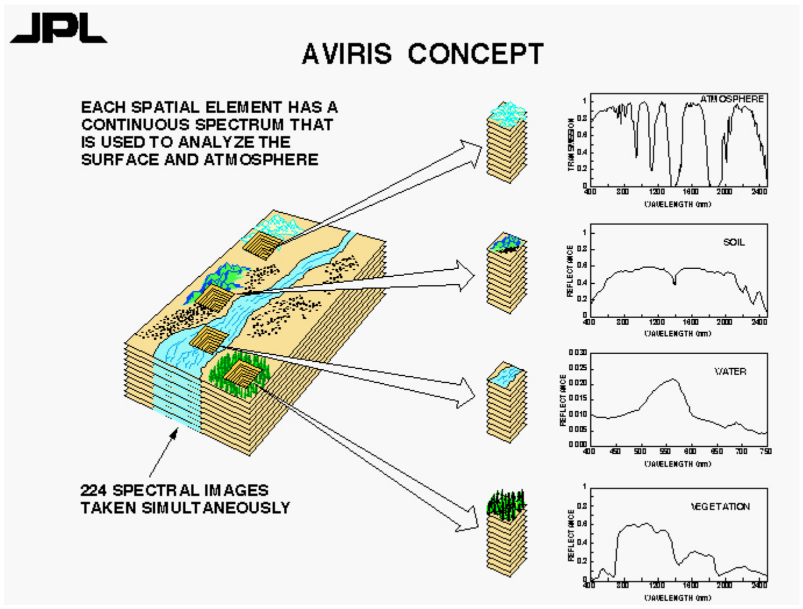

JPL
AVIRIS CONCEPT
EACH SPATIAL ELEMENT HAS A CONTINUOUS SPECTRUM THAT IS USED TO ANALYZE THE SURFACE AND ATMOSPHERE
224 SPECTRAL IMAGES TAKEN SIMULTANEOUSLY
ATMOSPHERE
TRANSMISSION
SOIL
REFLECTANCE
WATER
VEGETATION
WAVELENGTH (nm)

Figure 2. Visualization of Hyperspectral Data. Source: [4].

Figure 2 represents the spectral data that would be captured at different pixel locations if a hyperspectral image was taken of the landscape on the left. The multiple layers of the image represent 224 different wavelength bands captured. This is analogous to how a digital camera image has a red, blue, and green pane stacked on top of each other. At each location, a different spectrum would be captured, representing a different material. The 4 sample locations are cut out, and their plots are displayed on the right. Each material has a distinct wavelength "fingerprint" that allows the chemical composition of the material to be determined.

Figure 3 shows an example pixel spectrum at the top-left pixel of the Indian Pines hyperspectral image [5]. At this location is a large, wooded area.

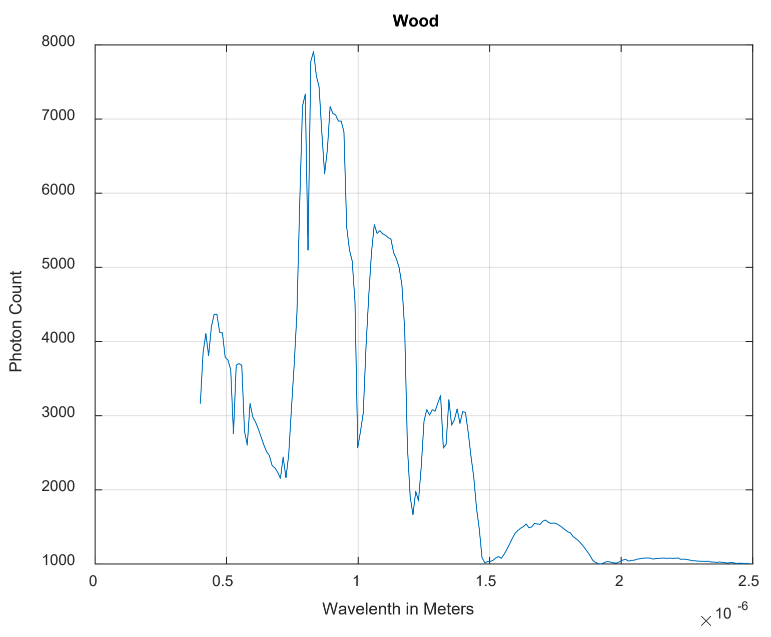


Figure 3. Hyperspectral Image Pixel Spectrum of Wood in Indian Pines HSI.

The example pixel in Figure 3 has a minimum and maximum wavelength of 0.4 µm to 2.5 µm from the AVIRIS sensor construction. The height of the spectrum represents the number of photons at that wavelength captured by the sensor. Photon count represents how well a material on the ground reflects or emits that wavelength. By matching similar wavelength spectra in the hyperspectral image, it is possible to group and classify different parts of the image by material. Image classification, however, does have some limitations.

Because the spectral bands are narrow (often 10nm or less), each pixel in a hyperspectral sensor must be large enough to capture enough photons to measure intensity accurately. Hyperspectral imaging provides extremely detailed spectral information at the cost of lower spatial resolution

When used with classification algorithms, hyperspectral data can allow for very accurate classification of various materials on the ground. Much better than multi-spectral or panchromatic data. Hyperspectral data has many uses. Examples include surveying, locating chemicals, and detecting pollution. The exact material can be determined by comparing the captured spectrum to a spectral library such as the United States Geological Survey (USGS) library [6]. Figure 4 shows a chemical sample along with its reflectance spectrum.

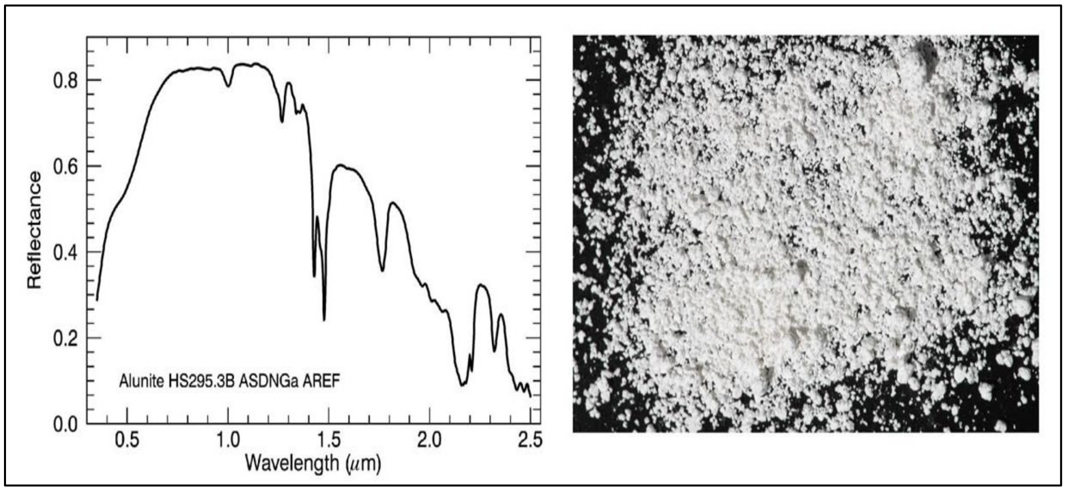


Figure 4. Reflectance Values of Alunite. Source: [6].

In Figure 4, the reflectance spectrum of alunite is plotted. The RGB representation of the material is on the right. The right image is how alunite would look to the naked eye. On the left is the wavelength spectral data of alunite from 0.4µm to 2.5 µm. The Y-axis of the graph represents the ability of the material to reflect photons at a specific wavelength indicated on the X-axis. The left image is how alunite would be represented in a hyperspectral image pixel.

B. PURPOSE

The purpose of this work was to make the classification and segmentation of hyperspectral images more effective by diminishing the effects of random noise additions and increasing the spatial resolution of hyperspectral images. The proposed method aids hyperspectral image analysis by leveraging the similarities within a single hyperspectral image to increase the signal-to-noise ratio (SNR).

C. RELATED WORK

This work extends the image enhancement of multi-channel images explored by El-Khamy, Hadoud, and Dessouky in [7]. They defined the three RGB interpolation algorithms that have been used in this experiment to increase the spatial resolution of the hyperspectral images [7]. They used a four-step process that consisted of image registration, image restoration, image fusion, and image interpolation. The three interpolation algorithms they used in their approach were a Linear Minimum Mean Squared Error (LMMSE) algorithm, a maximum entropy algorithm, and a regularization algorithm. This thesis will focus on applying image fusion and interpolation to grouped bands in a hyperspectral image.

Previous studies have applied a principal component analysis (PCA) in order to save computational time and have a more accurate spatial resolution after the image enhancement has been done on the hyperspectral image. In their paper, Zhang, Zhang, and Shen [8] conducted a PCA on the hyperspectral image before image reconstruction. The rationale being that "primary components contain most of the information of the hyperspectral image, which makes the resolution enhancement of the primary components quite important" [8]. Zhang and Zhang did not address the level of compression necessary to achieve the best results. An approximate percentage of components to use for best results are addressed in Chapter IV.

The method of image degradation in this thesis is accomplished through the use of random gaussian noise with image down-sampling. In their paper, Sun, Xu, Yang, Chen, Fang, and Peng [9] degraded their images with random Gaussian noise. Gaussian-noise was used on each individual band in their low-resolution images. Next, their algorithm was used to remove the noise and improve image resolution. The image enhancement approach in this paper uses this method to generate degraded images that will be used to estimate the original hyperspectral images.

D. SCOPE

Hyperspectral data cubes are generated over long exposure times in order to acquire enough photons to accurately display an image. The high cost of flying hyperspectral sensors and the long integration times needed to gather the image during flight limit the number of data cubes available for public use. In this thesis, we have used two well-known data cubes used in many other studies. The data used in this thesis is publicly available through the Purdue hyperspectral database [5]. These two datasets were used due to their large differences in data size. The differences in data size aid in determining how data size affects the performance of the image enhancement algorithm.

E. THESIS OUTLINE

The following chapters are organized as follows. Chapter II discusses the mathematical theories which define the three more complex methods used for image interpolation, image fusion, and image grouping. Chapter III goes into detail on the algorithm that was used to conduct the experiment. The results of the investigation are discussed in detail in Chapter IV. The conclusions derived from the results are addressed in Chapter V.

THIS PAGE INTENTIONALLY LEFT BLANK

II. MATHEMATICAL PRELIMINARIES

A. INTRODUCTION

The proposed image enhancement approach in this thesis follows several steps. The steps are degraded low-resolution image generation, image grouping, image fusion, and image interpolation. The information in this chapter provides the mathematical principles that accomplish these image enhancement tasks.

B. LOW-RESOLUTION IMAGE GENERATION

A challenge with testing hyperspectral resolution techniques is that often the only sample available of a scene is a single hyperspectral image. Unlike conducting image enhancement with video data, there is often only a single capture of the hyperspectral scene. The single image serves as our high-resolution reference image for evaluating the image enhancement approach. Therefore, low-resolution images must be generated by down-sampling and adding noise to the original hyperspectral image. Hyperspectral image degradation creates a set of low-resolution wavelength bands through the use of random noise addition and down-sampling of each wavelength band. The degradation process artificially creates errors, while image fusion and the advanced interpolation methods use the shared information from each grouped wavelength band in the fusion process to improve the image quality.

The generalized image degradation equation is an iterative process done for each band in the hyperspectral image. The equation to create the degraded low-resolution images, as described in [10], is written as

$$g_k = D_k f_k + v_k \quad \text{for } k = 1, 2, \ldots, P \,, \tag{1}$$

where g_k is the generated low-resolution wavelength band, f_k is the original high-resolution wavelength band, v_k is the degradation noise added to each down-sampled wavelength band, D_k is the down-sampling matrix, and P is the total number of degraded wavelength bands.

Each matrix is a lexicographically ordered vector of the image pixels. Lexicographic ordering means that the image is sorted column-by-column to form a single vector. The original high-resolution image f_k is of size $N^2 x 1$. The down-sampling matrix D_k is of size $M^2 \times N^2$ and shrinks the original image to a smaller size by sampling values at every other location. The result of multiplying f_k by D_k creates the down-sampled matrix g_k of size $M^2 \times 1$, which are reordered to be the low-resolution images of size $M x M$. The noise v_k is size $M^2 \times 1$ and is added to the low-resolution image. N^2 is the total number of pixels in the high-resolution wavelength band, and M^2 is the total number of pixels in a low-resolution wavelength band. The down-sampling ratio between the two images is L, and their relationship is written as $N=LM$. A visual representation of multi-channel degradation can be seen in Figure 5.

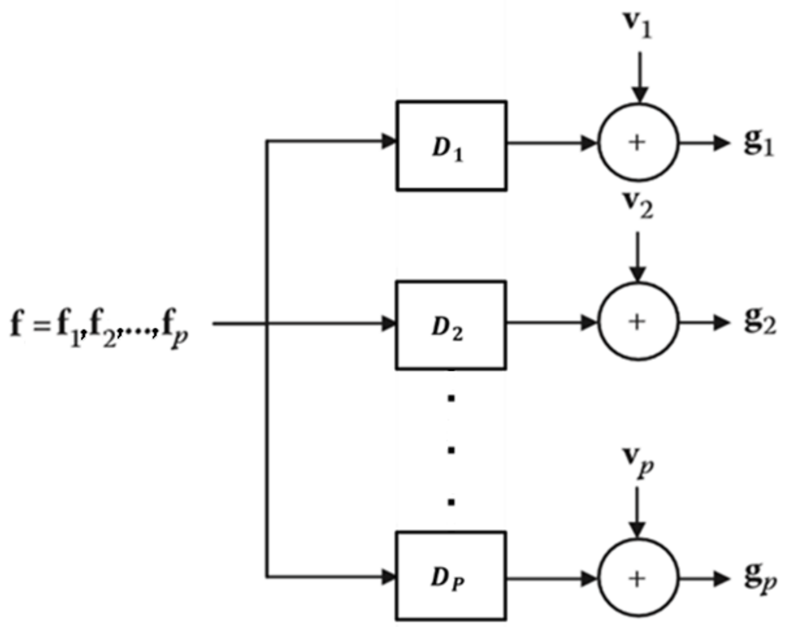


Figure 5. Multi-Channel Image Degradation Model. Source: [10].

In Figure 5, each of the high-resolution images from f_1 to f_p are down-sampled by their corresponding down-sampling matrix D_1 to D_p. After down-sampling, random noise values v_1 to v_p are added to each down-sampled image to create the low-resolution degraded images g_1 *to* g_p. In order to create a simpler implementation in software, it is advantageous to rewrite Equation 1 as a linear multiplication of matrices [10]:

$$g = \overline{D}f + v \ , \tag{2}$$

where [10]:

$$g = \begin{bmatrix} g_1 \\ g_2 \\ \vdots \\ g_p \end{bmatrix}; \ f = \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_p \end{bmatrix}; v = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_p \end{bmatrix}; \ \overline{D} = \begin{bmatrix} D_1 & 0 & \cdots & 0 \\ 0 & D_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & D_p \end{bmatrix} \tag{3}$$

The variables in Equation 3 g, f, v, and $\overline{D}$ represent matrices that contain all of the resulting values from the iterative multi-channel degradation process.

C. IMAGE GROUPING

Image grouping is how groups of the degraded low-resolution bands are chosen for use in the proposed image enhancement approach. In this case, groups of bands from the hyperspectral image are taken and have their spatial information fused with a discrete wavelet frame transform (DWFT) before the final interpolation. Fusing wavelength bands with good spatial correlation is necessary because this will make sure that the fused image low-frequency spatial data is not drastically different from the original low-resolution wavelength band. By fusing data across multiple wavelength bands, both complementary and redundant information is incorporated. The inclusion of redundant information "reduces the uncertainty and increases the accuracy of the features" [7]. Complementary information, however, provides more detailed information regarding features not available from a single wavelength band. The method that was used to group bands for fusion is the 2D correlation coefficient.

In the hyperspectral image, the 2D correlation coefficient is calculated between every possible pairing of the different wavelength bands in the image. The correlation coefficient is calculated by [11]:

$$r = \frac{\sum_m \sum_n (g_{1mn} - E(g_1))(g_{2mn} - E(g_2))}{\sqrt{\left(\sum_m \sum_n (g_{1mn} - E(g_1))^2\right) * \left(\sum_m \sum_n (g_{2mn} - E(g_2))^2\right)}}, \tag{4}$$

where r is the correlation coefficient between the two bands, g_1 is the first low-resolution hyperspectral wavelength band, and g_2 is the second low-resolution hyperspectral wavelength band. $E(g_1)$ is the mean of the low-resolution band g_1, and $E(g_2)$ is the mean of the low-resolution band g_2. The spatial correlation between each wavelength band is what was used to match them in the grouping process. Equation 4 compares the pixel intensity value at all locations in the two images and then returns a value between -1 and 1, which represents how closely correlated the two bands are. An example correlation coefficient stem plot for the first hyperspectral band is provided in Figure 6. This shows how well the first wavelength band in the image correlates with the rest of the wavelength bands. A higher correlation coefficient means those two bands are similar spatially.

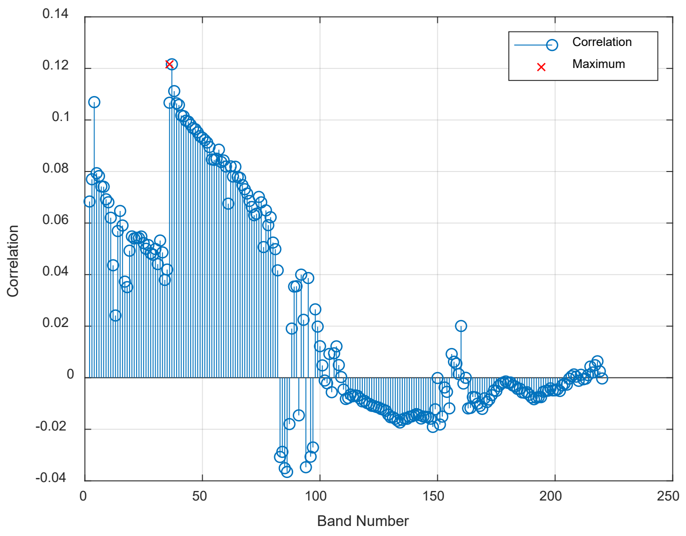


Figure 6. Correlation Coefficient of Band 1 and All Bands.

In Figure 6, the height of each point represents the correlation coefficient between band 1 and the corresponding band number. Each band will have a different correlation plot. Once the correlation coefficient is calculated for each possible pairing, the first and second maximum correlations are used to create the groups for the rest of the image enhancement process. The end of the fusion process will result in a fused low-resolution hyperspectral image with the same number of bands as the degraded low-resolution image. Each band in the fused image corresponds to the wavelet fusion of three bands from the degraded image

D. IMAGE FUSION

Image fusion is the process by which bands in the degraded low-resolution image are "fused" together into a single band that is later interpolated. The assumption is that each band in the image contains complementary information that can be combined and will result in a high-resolution band that has more information than what would be obtained through only a single band. Image fusion has several possible advantages, including image sharpening, feature enhancement, and improved classification [7]. The image fusion in our proposed algorithm is accomplished through a discrete wavelet frame transform (DWFT).

A discrete wavelet frame transform (DWFT) decomposes a band into additive spatial frequency components. Each component contains a different range of spatial frequencies of the band. This separates the spatial frequency components and allows the low, mid, and high spatial frequency data of several bands to be fused together. Given a degraded low-resolution band *P,* it is possible to construct the sequence of approximations [7]:

$$f_1(P) = P_1, f_2(P_1) = P_2, \ldots, f_n(P_{n-1}) = P_n \tag{5}$$

For a registered group of three bands, the method to create the approximation is by conducting successive convolutions on the band with the kernel given by [7]:

$$H = \frac{1}{256}\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix} \tag{6}$$

A wavelet detail plane is calculated by the difference between two consecutive approximations. This difference $\square_l$ will contain the high-frequency details, and P_n is a low-frequency component [7].

$$\square_l = P_{l-1} - P_l \text{ , for } l = 1, 2, \ldots, n \tag{7}$$

Then the fused image reconstruction formula is written as [7]:

$$P = \sum_{l=1}^{n} \square_l + P_n \tag{8}$$

The model for how this is accomplished is shown in Figure 7.

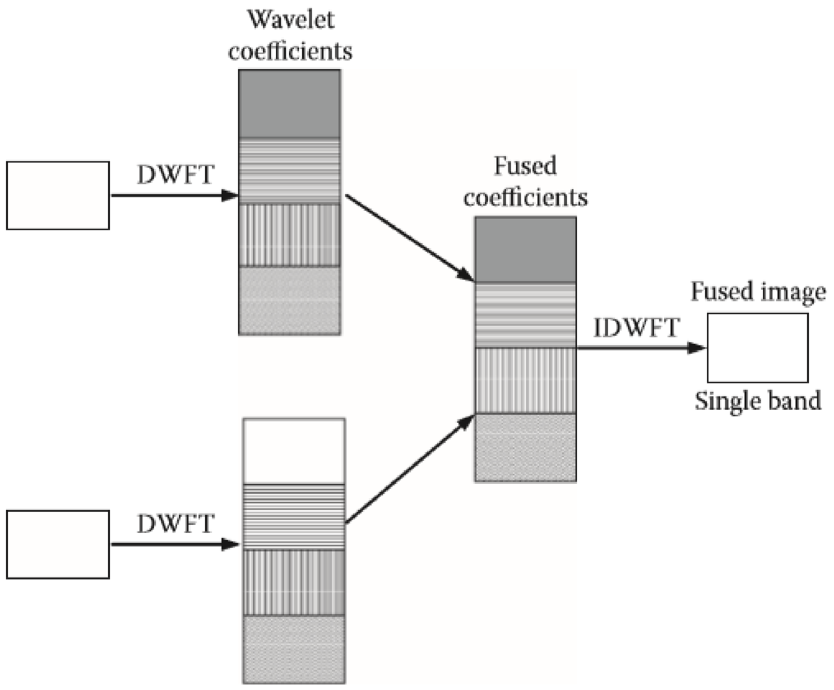


Figure 7. DWFT Image Fusion. Source: [7].

In Figure 7, two images are broken up into spatial frequency components. Those frequency components are then averaged and then the inverse of the DWFT was used to create a fused image with spatial frequency information of both starting images. This process is completed two separate times. First, a fusion of the 1st band and 2nd band in the group of three bands. Then that fused result is fused with the final 3rd band in the group to yield a single fused image that incorporates spatial frequency information from all three bands. A fused band is calculated for each band in the hyperspectral image. The fused hyperspectral image is then sent to the image interpolation step.

E. IMAGE INTERPOLATION

The final step is to interpolate the fused image, which contains the additive components of several bands. These algorithms take the fused image and then interpolate it into a high-resolution image estimate. This is done in order to take advantage of the fused spatial data after the wavelet fusion has been completed. The expectation is that the combined data from multiple bands will aid in creating a final image that will improve the SNR of the final high-resolution image estimate.

1. LMMSE Interpolation

The LMMSE interpolation is done iteratively on each band in the fused low-resolution hyperspectral image. The fused image has incorporated the information from the grouped bands of the degraded low-resolution image. The algorithm is based upon on finding the minimum estimate error possible and is written as follows [12]:

$$\min_{\hat{f}} E\left[e^t e\right] = E\left[Tr(ee^t)\right], \tag{9}$$

where the error is defined as [12]:

$$e = f - \hat{f} \tag{10}$$

The LMMSE estimate of the high-resolution band is given by [12]:

$$\hat{f} = R_f D^t (D R_f D^t + R_v)^{-1} g \tag{11}$$

D is the single band down-sampling and filtering matrix. Variable *g* is the fused low-resolution band. For a down-sampling factor of two, the *N* x *N* high-resolution matrix is transformed into an *M* x *M* matrix. Where *N = 2M*. The down-sampling matrix *D* is of size M^2 x N^2 and is written as [12]:

$$D = D_1 \otimes D_1$$
$$\text{and } D_1 = \frac{1}{2}\begin{bmatrix} 1 & 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 1 & 1 \end{bmatrix} \tag{12}$$

R_f is the high-resolution image autocorrelation matrix, and R_v is the noise variance. Both must be estimated in order to solve for the estimated high-resolution image. Estimating R_f is done by calculating the autocorrelation at different spatial positions of a bilinear or cubic interpolation of the fused degraded image. R_v is estimated by taking a small sample size of the band.

2. Maximum Entropy Interpolation

The maximum entropy model is also done iteratively on each fused band. Maximum entropy image interpolation assumes a unit energy for each band of the high-resolution image estimate. The pixel values are treated as probabilities of photons present at a particular location in the band. The entropy of the high-resolution band estimate becomes [10]:

$$H_e = -\sum_{i=1}^{N^2} f_i \log_2(f_i), \tag{13}$$

where f_i in this section is the desired high-resolution band. H_e is the entropy of the band. The vector form of the equation is written as [10]:

$$H_e = -f^t \log_2(f) \tag{14}$$

To maximize the entropy cost function, ψ, must be minimized such that [10]:

$$\Psi(f) = f^t \log_2(f) - \lambda\left[\|g - Df\|^2 - \|v\|^2\right] \tag{15}$$

λ is a Lagrangian multiplier, g is the fused low-resolution band, and v is random noise. The single-band degradation and down-sampling matrix D defined in Equation 12 was used for this interpolation method as well. By differentiating the left and right side of the equation and setting the derivatives to zero, the equation becomes [10]:

$$\frac{d\Psi(f)}{df} = 0 = \frac{1}{\ln(2)}\left\{1+\ln(\hat{f})\right\} + \lambda\left[2D^t(g - D\hat{f})\right] \tag{16}$$

Solving for $\hat{f}$ results in [10]:

$$\hat{f} = \exp\left[-1-\lambda\ln(2)\left|2D^t(g - D\hat{f}\right|\right] \tag{17}$$

Because $g - D\hat{f}$ must be a small quantity to maximize the accuracy of the estimate, it leads to the form [10]:

$$\hat{f} \cong -\lambda\ln(2)\left[2D^t(g - D\hat{f})\right] \tag{18}$$

Solving for $\hat{f}$ once again leads to [10]:

$$\hat{f} \cong (D^tD + \eta I)^{-1}D^tg\,, \tag{19}$$

where [10]:

$$\eta = -1/(2\lambda\ln(2)) \tag{20}$$

3. Regularized Interpolation

The regularization interpolation algorithm is an iterative approach completed on each fused band. The regularization algorithm minimizes a stabilizing functional in order to determine a high-resolution estimate. This method is beneficial due to the lower computational cost by avoiding a matrix inversion; however, the high number of iterations required to reach a solution is a potential drawback dependent on the data size.[13] The cost function to be minimized is written as [13]:

$$\Psi(\hat{f}) = \left\| g - D\hat{f} \right\|^2 - \lambda \left\| C\hat{f} \right\|^2 \tag{21}$$

C is the 2-dimensional regularization operator and λ is the regularization parameter of the cost function. This cost function controls the trade-off between fidelity to the data and the smoothness of the estimated high-resolution image.[13] The minimization is accomplished by taking the derivative of the function in Equation 22 [13]:

$$\frac{d\Psi(\hat{f})}{df} = 0 = 2D^t(g - D\hat{f}) - 2\lambda C^t C\hat{f} \tag{22}$$

The role of the 2-D operator C is to move small eigenvalues of D away from zero while leaving larger values constant. What this allows for is an incorporation of the required low spatial frequency information of the original band. It also minimizes the second and higher-order difference energy of the estimated high-resolution band. By solving through iteration, the high-resolution band can be estimated [13]:

$$f_{i+1} = f_i + \eta_0 \left\{ D^t g - (D^t D + \lambda C^t C) f_i \right\} \tag{23}$$

The image f_i is the estimated high-resolution band at the i^{th} iteration. η_0 is a convergence parameter that is set at 10^{-3}. The interpolation formula is written as [13]:

$$\hat{f}_{i,j} = (D^t D + \lambda C^t C)^{-1} D^t g_{i,j}\,, \tag{24}$$

where $g_{i,j}$ is the fused low-resolution band of size M^2 x 1 and $\hat{f}_{i,j}$ is size N^2 x 1 both in lexicographic order as discussed in Chapter II, Section B.

4. B-spline Variants: Nearest, Bilinear, Bicubic

We have combined the spatial frequency information from multiple bands to increase the image quality of the high-resolution estimate. In order to have a baseline of performance, we will use three simple interpolation methods that will interpolate each band of the degraded low-resolution hyperspectral image independently. These methods will not incorporate information between bands through image fusion.

Splines are piecewise polynomials that can be used to interpolate images. These are known to be the simplest of image interpolation methods and was used to establish a baseline of comparison to the three more complex image interpolation methods. Each polynomial is of degree *n*, where *n* denotes the number of the convolutions of a rectangular pulse β^0. Splines can be characterized in terms of a B-spline expansion given by the equation:

$$\hat{f}(x) = \sum_{k \in Z} c(x_k) \beta^n (x - x_k) \tag{25}$$

Z is a finite neighborhood around *x*. The B-spline basis function $B^n(x)$ is a symmetrical bell curve obtained by *n*+1 convolutions of a rectangular pulse β^0 [7]:

$$\beta^0(x) = \begin{cases} 1, -\frac{1}{2} < x < \frac{1}{2} \\ \frac{1}{2}, |x| = \frac{1}{2} \\ 0, \text{otherwise} \end{cases} \tag{26}$$

The nearest neighbor interpolation is the simplest interpolation method. It is done through a zero-order interpolation, i.e., using the rectangular pulse as the basis function. This rectangular pulse is shown in Figure 8.

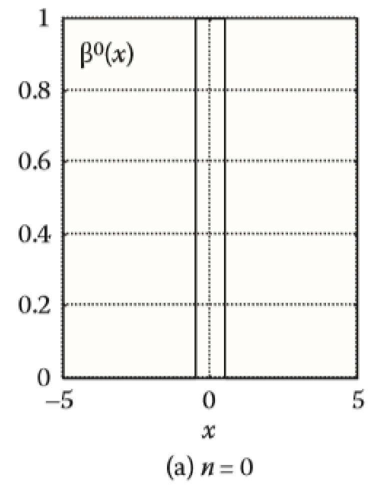


Figure 8. Nearest Neighbor Spline Basis Function. Source: [7].

To calculate the bilinear and bicubic interpolations, the n=1 and n=2 basis functions are used. Their basis functions are shown in Figure 9. By convolving these basis functions throughout the image in two dimensions, the bilinear and bicubic interpolations are completed.

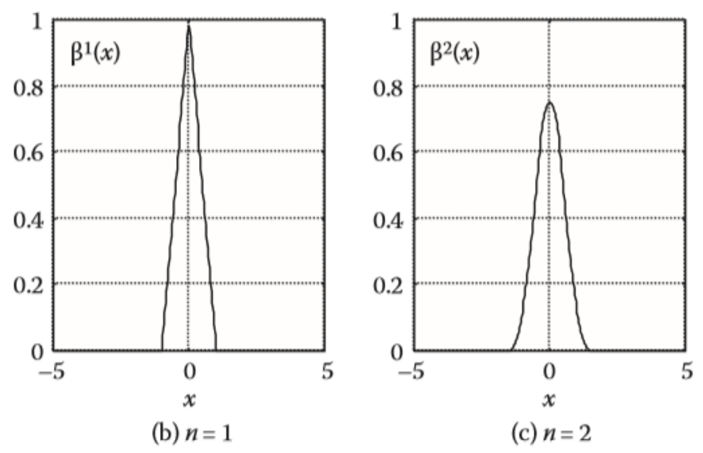


Figure 9. Bilinear And Bicubic Basis Functions. Source: [7].

F. PCA

The fusion and interpolation steps in Sections D and E can be computationally intensive to operate on every band in a hyperspectral data cube. PCA was used because it compresses high-dimensional data into a lower-dimensional space while preserving as much variance as possible from the original space. This aids in calculation time by reducing the number of computations required to generate the estimated high-resolution image in the image enhancement process. A graphical representation of random data undergoing this projection is portrayed in Figure 10, where all of the data is compressed to the axis upon which most of the data information is contained (i.e., the "PC 1" axis).

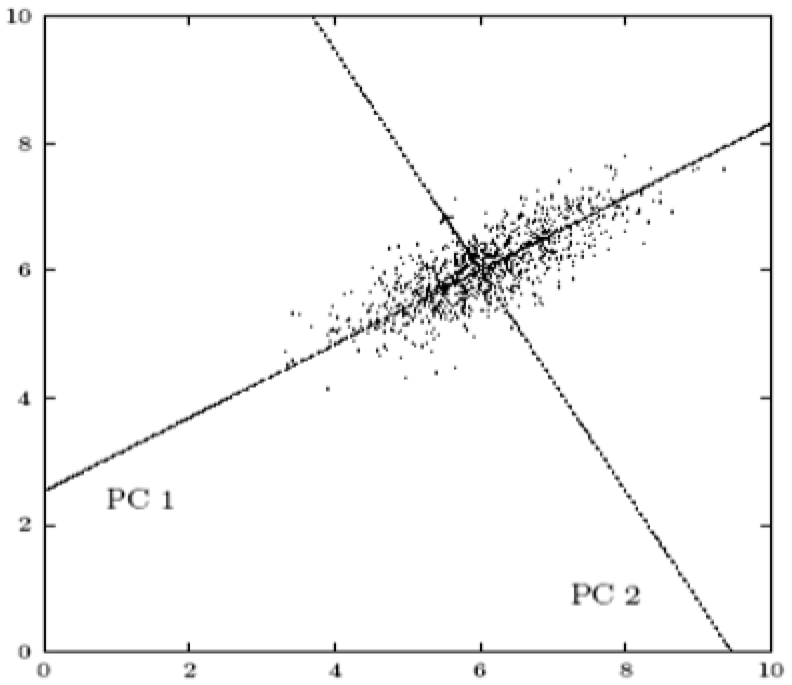


Figure 10. Eigenvalue Priority Determination. Source: [11].

There are several steps to project the space properly onto a lower-dimensional one. First, the covariance matrix of the image $\underline{x}$ must be calculated. The equation for the covariance matrix as written in [11] as

$$C_{\underline{x}} = E\left\{\left(\underline{x}-\underline{m}_x\right)\left(\underline{x}-\underline{m}_x\right)^H\right\} \tag{27}$$

The covariance matrix is related to the eigenvectors and eigenvalues of the data through the relationship [11]:

$$C_x = U\Lambda U^H = \begin{bmatrix} \uparrow & & \uparrow \\ \underline{u}_1 & \cdots & \underline{u}_N \\ \downarrow & & \downarrow \end{bmatrix} \left[\begin{pmatrix} \lambda_1 & & 0 \\ & \ddots & \\ 0 & & \lambda_n \end{pmatrix}\right] \begin{bmatrix} -\underline{u}_1{}^H - \\ \vdots \\ -\underline{u}_N{}^H - \end{bmatrix}, \tag{28}$$

where U is the eigenvector matrix and Λ is the eigenvalue matrix. The next step is to project the data into the lower dimensional space $\underline{y}$. This is done through first sorting the eigenvalues of the covariance matrix by descending value, as seen in Figure 11. Then the eigenvectors are sorted in the same manner based upon their eigenvalue order

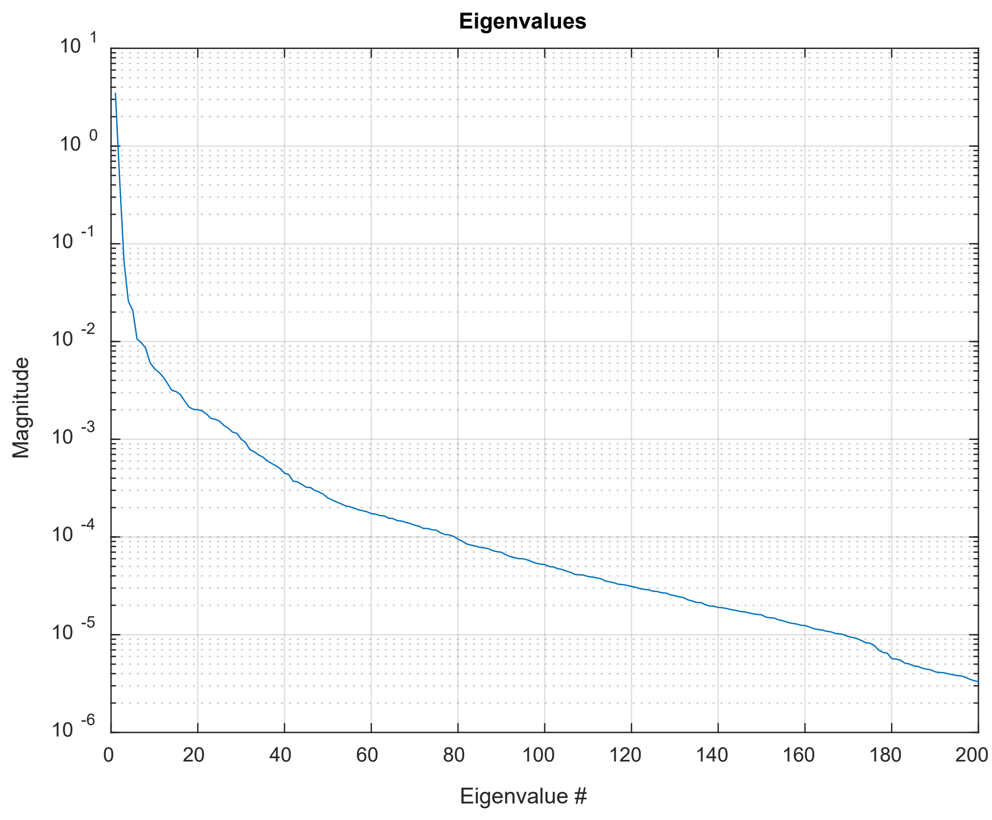


Figure 11. Eigenvalues Of The Indian Pines Hyperspectral Image.

After the eigenvector sorting has been done, the equation

$$\underline{y} = \underline{U}^H \underline{x} \tag{29}$$

is used to project the original space into an uncorrelated space $\underline{y}$. In order to compress the information, it is desired that only a finite number of eigenvectors, $N_{pca,}$ are used to return to the original space after the compression. The equation to compress the original information is written as [11]:

$$\overset{\wedge}{\underline{x}} = \sum_{i=1}^{N_{PCA}} y_i \underline{u}_i,\ N_{PCA} \le N \Leftrightarrow \overset{\wedge}{\underline{x}} = \begin{bmatrix} \vdots & & \vdots \\ \underline{u}_1 & \cdots & \underline{u}_{N_{PCA}} \\ \vdots & & \vdots \end{bmatrix} \begin{bmatrix} y_1 \\ \vdots \\ y_{N_{PCA}} \end{bmatrix}, \tag{30}$$

where $\overset{\wedge}{\underline{x}}$ is the estimation of the original space from the projected space $\underline{y}$. $\underline{U}$ is the sorted eigenvector matrix based upon the descending eigenvalues. N_{pca} is the number of eigenvectors used to project the data. The higher number of eigenvectors, the more of the original space can be reconstructed when returning to the original space. However, it is usually the case that most of the information is contained in the first several eigenvectors, as the eigenvalue plot in Figure 11 shows that most of the data information is contained in the first few eigenvalues. A good visual representation is shown in Figure 12. This is an application of PCA where the original character dataset is shown on the top left along with the projected space for increasing numbers of eigenvectors used. The fewer number of eigenvectors used, the less of the spatial frequency information is taken from the original data. As higher amounts of eigenvectors are used, higher frequency components are added and more of the original data is kept after compression. As more data is retained, a more accurate representation of the final image is created, but a longer computation time is needed. A method to quantitatively compromise between computation time and image error is discussed in Section G.

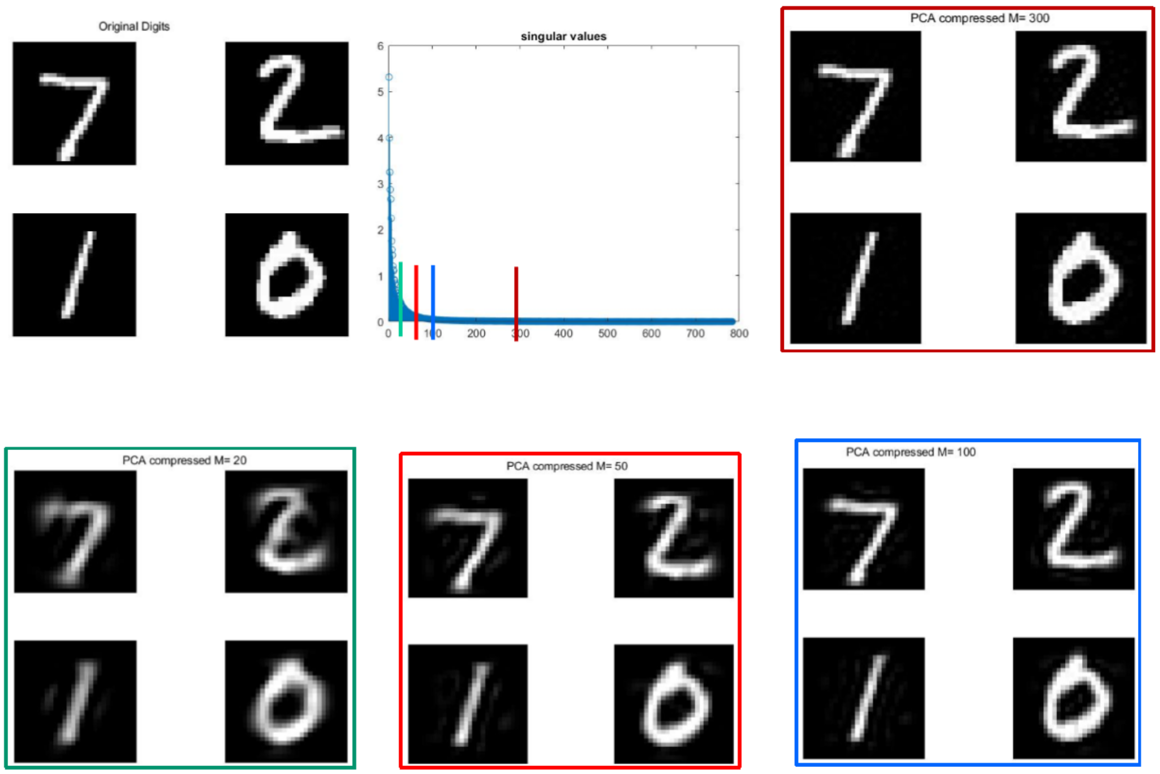


Figure 12. Character Example of PCA. Source: [11].

G. OPTIMIZATION OF IMAGE COMPRESSION

The method by which an optimal value for the PCA is determined by locating the minimum value of a compression cost function. This function incorporates the time taken to conduct the image enhancement on the compressed image and the error between the estimated high-resolution hyperspectral image and the original hyperspectral image. The greater the error and time elapsed, the higher the cost for that number of eigenvectors used for the compression. This was accomplished by executing the image enhancement on every possible compressed space available from using only the first eigenvector to using all the eigenvectors. The cost function used is written as:

$$J = a(t)^2 + b(d)^2 , \tag{31}$$

where J is the cost value, t is the time measured in seconds, and d is the sum of the distance of each estimated pixel value from its original value in the original hyperspectral image. This distance is measured across all the bands in the estimated image. The coefficients a and b are used as weighting factors to represent the relative importance between time and error. An example of what the optimal location will look like when the cost is plotted for each eigenvector is shown in Figure 13.

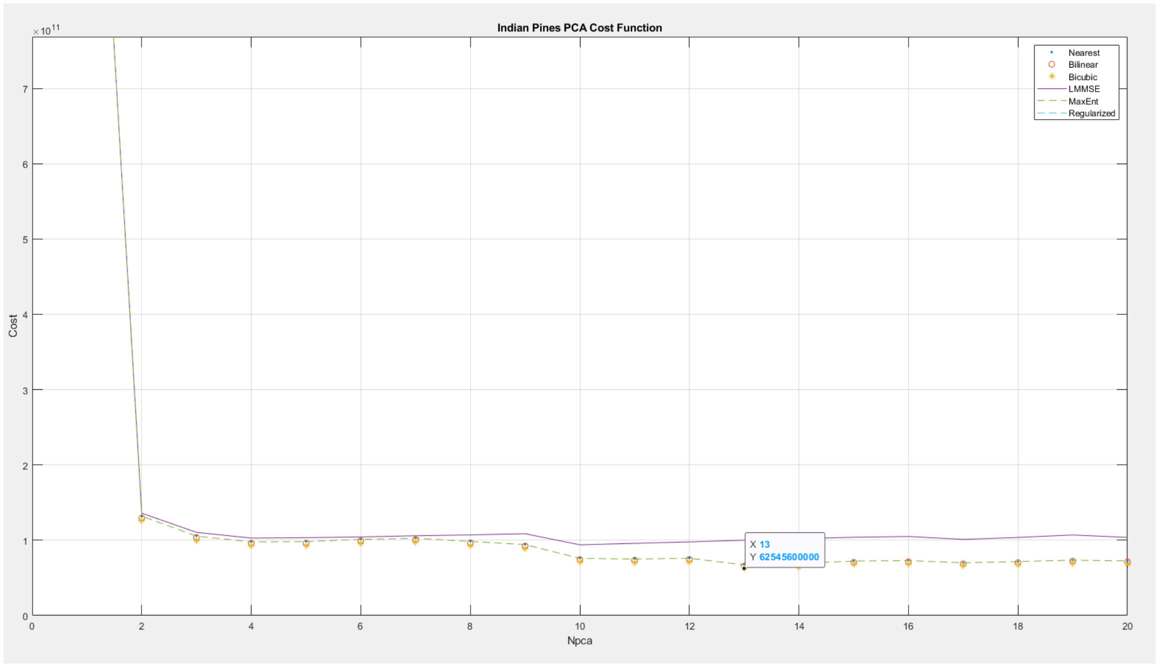


Figure 13. Example Of Optimal Compression Point.

In Figure 13, the optimal compression point is illustrated by the minimum value on the graph. For the Indian Pines image, the minimum value occurred at 13 eigenvectors.

The time taken was measured experimentally through recording the runtime of each image enhancement for each number of eigenvectors used to compress the data. The weighting factors a and b were chosen to represent the relative importance of time and error. The specific values are subjective. In this case, the error between the original image and the estimate was chosen to be much more important than the time elapsed. Therefore, the coefficient b is an order of magnitude larger than coefficient a. The use of a quadratic cost function is to ensure that there is a minimum value for the cost that can be determined as the optimal value for PCA image enhancement.

III. IMAGE ENHANCEMENT APPROACH

This chapter is a detailed description of the image enhancement approach used. The image enhancement algorithm is implemented in MATLAB.

A. GENERAL IMAGE ENHANCEMENT APPROACH

Section A is a summary of the overall enhancement approach. The following sections will go into detail about how each of the system blocks are accomplished. The proposed image enhancement approach is displayed in the system diagram shown in Figure 14.

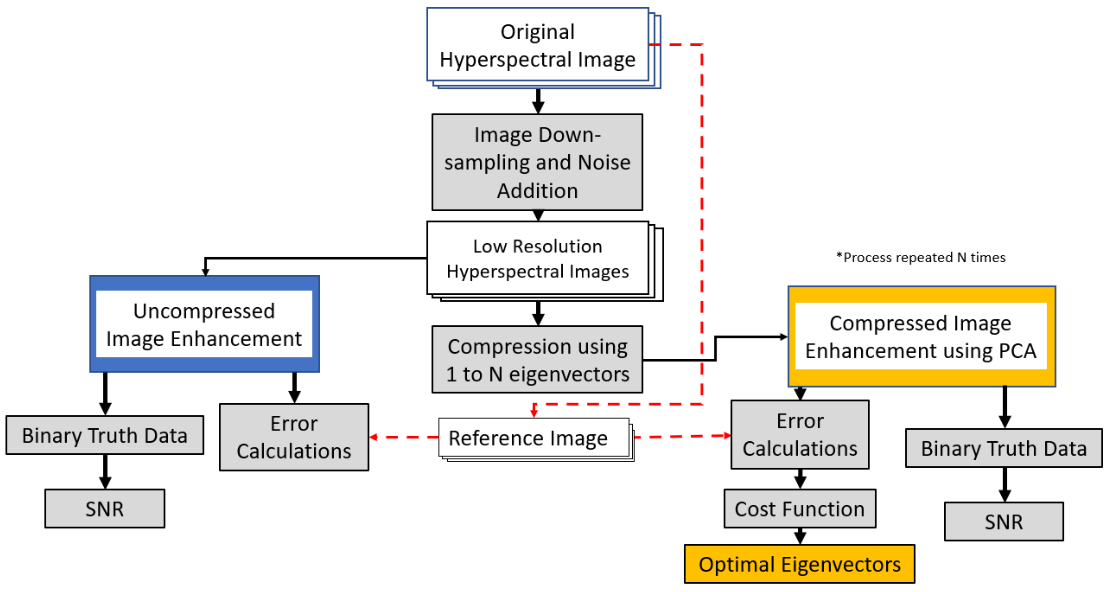

Original Hyperspectral Image
Image Down-sampling and Noise Addition
Low Resolution Hyperspectral Images
*Process repeated N times
Uncompressed Image Enhancement
Compression using 1 to N eigenvectors
Compressed Image Enhancement using PCA
Binary Truth Data
Error Calculations
Reference Image
Error Calculations
Binary Truth Data
SNR
Cost Function
SNR
Optimal Eigenvectors

Figure 14. Image Enhancement System Diagram.

The first goal of the proposed image enhancement process was to use the original hyperspectral images (Indian Pines and Salinas) as our high-resolution ground truth and reconstruct them from a simulated degraded low-resolution image. The original Indian Pines and Salinas images served as our high-resolution reference images. The system diagram starts with the original hyperspectral image.

The next step is to down-sample and then add noise to the image to create a degraded low-resolution image. Low-resolution image generation allows the interpolation result to be compared to the original unaltered data. After the down-sampling and degradation, the low-resolution image is fed to two different parts of the algorithm. Both the "blue" and "gold" paths use all six interpolation methods. The "blue" side is the uncompressed experiment, where the degraded low-resolution images are fed directly into the complex and simple interpolation algorithms. The estimated images are used to calculate the SNR and the error. For the SNR, binary truth data was used to define the signal. For error, the difference between the estimate and the original image is calculated.

The second goal of this investigation is to determine a method of selecting an optimal number of principal components (eigenvectors) when PCA was used for image enhancement. As more eigenvectors are used, the error between the reconstructed image and the original image decreases [8]. The "gold" side is the compressed experiment. In the compressed experiment, the low-resolution images are compressed using principal component analysis (PCA). The compressed images are then interpolated and compared to the original high-resolution image. Comparing to the reference determines what methods are best for small data sizes and an optimal compression level. On the gold side, time to calculate each interpolated image increases as the level of compression decreases. Time to calculate the estimated image and the error between the estimate and the original image are used to calculate a cost. The method with the least amount of cost is the most effective. The detailed outline of both the uncompressed and compressed image enhancement experiments can be seen in Figure 15.

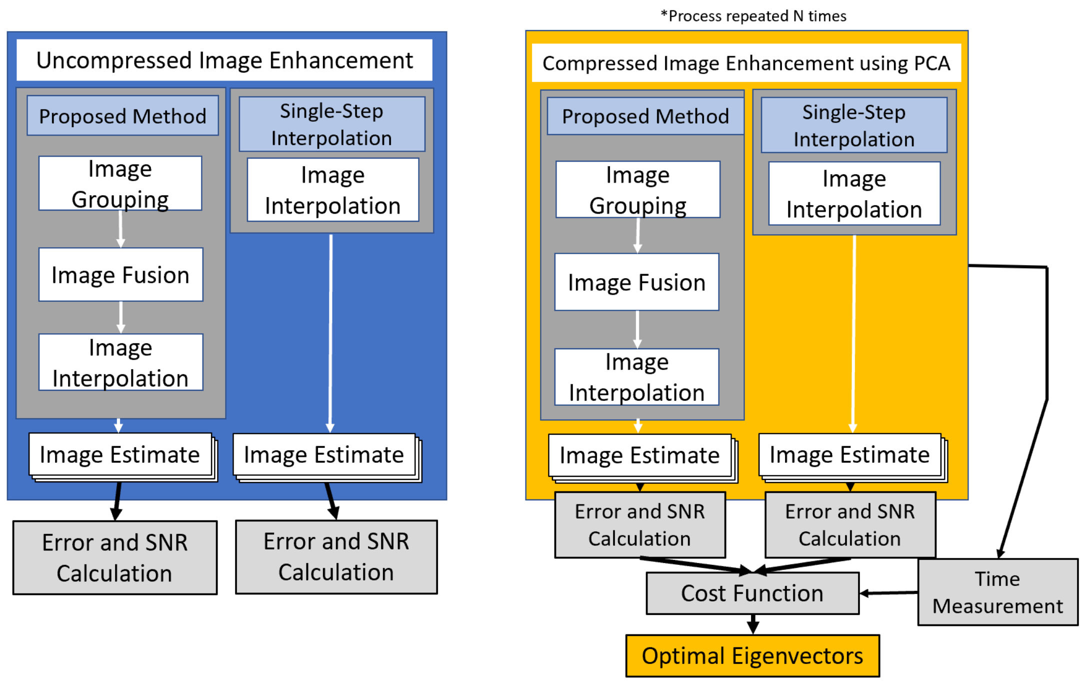
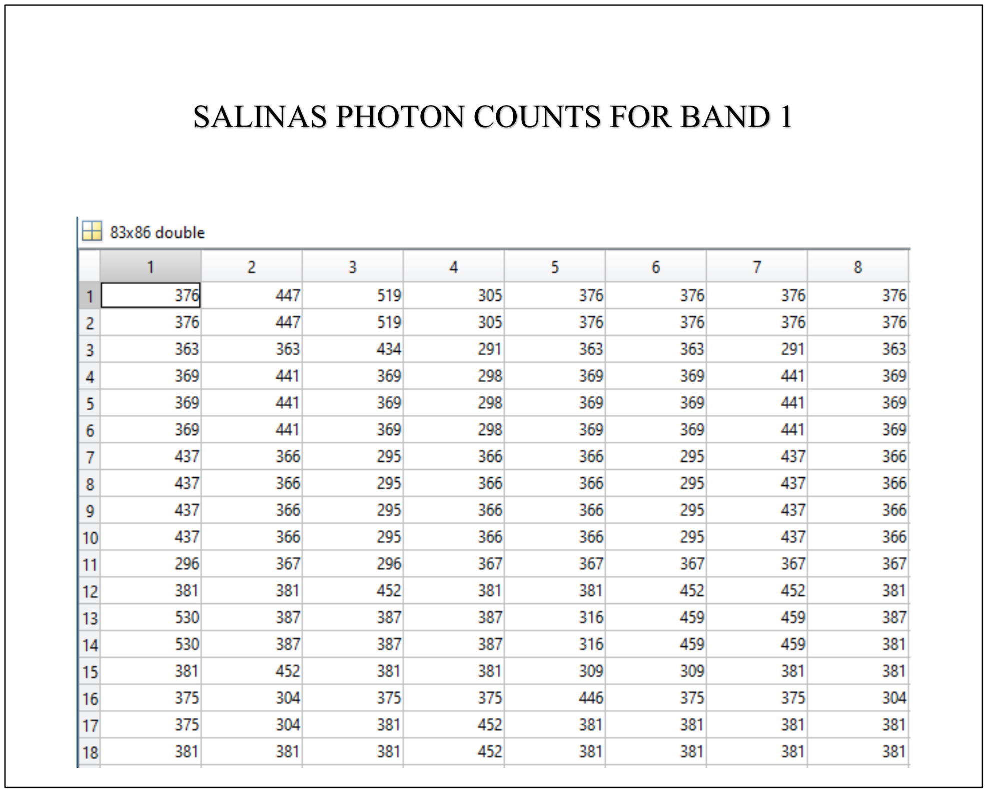


Figure 15. Uncompressed And Compressed Image Enhancement Process.

In Figure 15, and as stated previously, both the blue and gold experiments use all six of the interpolation methods discussed previously. The three more complex methods fall under the box named “proposed method.” The simple interpolation methods are under “single-step interpolation.” In the proposed method track, image grouping and fusion are used with the complex interpolation, whereas the single-step interpolation methods do not. Separating these two tracks is done to compare the proposed image enhancement to the three simpler methods. If the proposed methods cannot produce better results than the single-step interpolation, then they are not methods with any practical use.

B. DATA IMPORTATION

To import the images, the data was first loaded into the workspace with the “load” command. This created a MATLAB “double” variable, which included the photon counts captured at each wavelength and pixel on the AVIRIS sensor. The Indian Pines hyperspectral image is loaded into the workspace as a 145x145x200 double. The Salinas hyperspectral image is loaded into the workspace as an 83x86x204 double. Excerpts of the data matrix for one wavelength band is shown in Figure 16 and 17 for the Indian Pines and Salinas image.



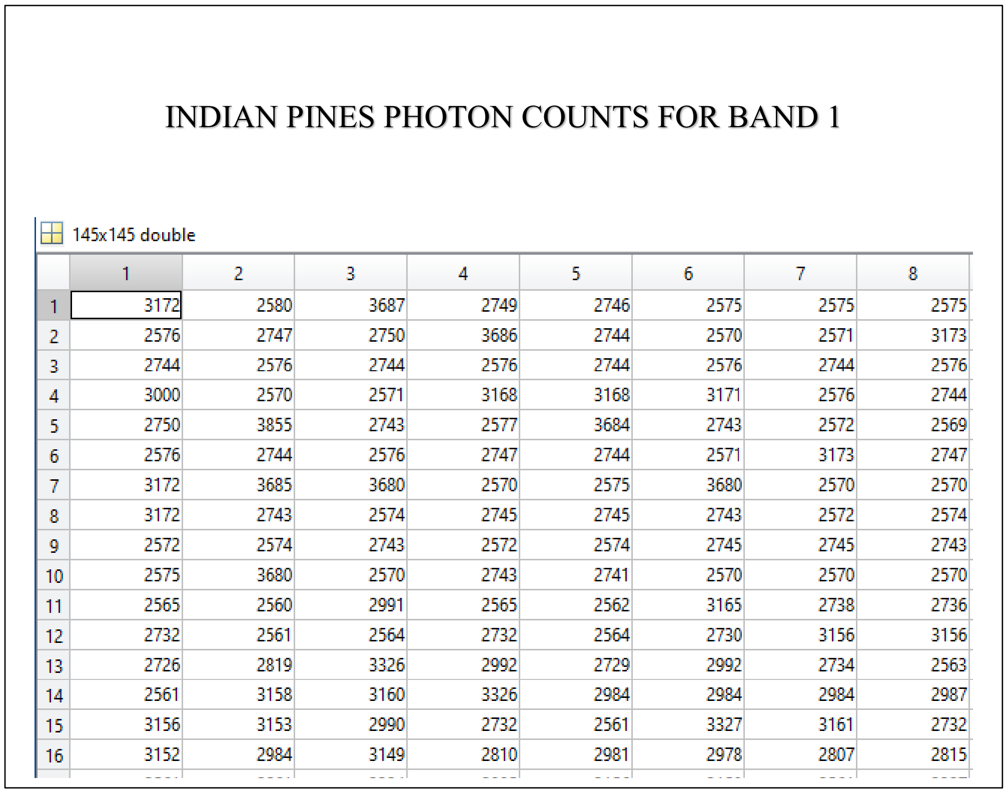
SALINAS PHOTON COUNTS FOR BAND 1

83x86 double

	1	2	3	4	5	6	7	8
1	376	447	519	305	376	376	376	376
2	376	447	519	305	376	376	376	376
3	363	363	434	291	363	363	291	363
4	369	441	369	298	369	369	441	369
5	369	441	369	298	369	369	441	369
6	369	441	369	298	369	369	441	369
7	437	366	295	366	366	295	437	366
8	437	366	295	366	366	295	437	366
9	437	366	295	366	366	295	437	366
10	437	366	295	366	366	295	437	366
11	296	367	296	367	367	367	367	367
12	381	381	452	381	381	452	452	381
13	530	387	387	387	316	459	459	387
14	530	387	387	387	316	459	459	381
15	381	452	381	381	309	309	381	381
16	375	304	375	375	446	375	375	304
17	375	304	381	452	381	381	381	381
18	381	381	381	452	381	381	381	381

Figure 16. Band 1 MATLAB Table for Salinas

In Figure 16, the photon counts for each Salinas pixel location for rows 1–18 and columns 1- 8 from band 1 are displayed.



INDIAN PINES PHOTON COUNTS FOR BAND 1

145x145 double

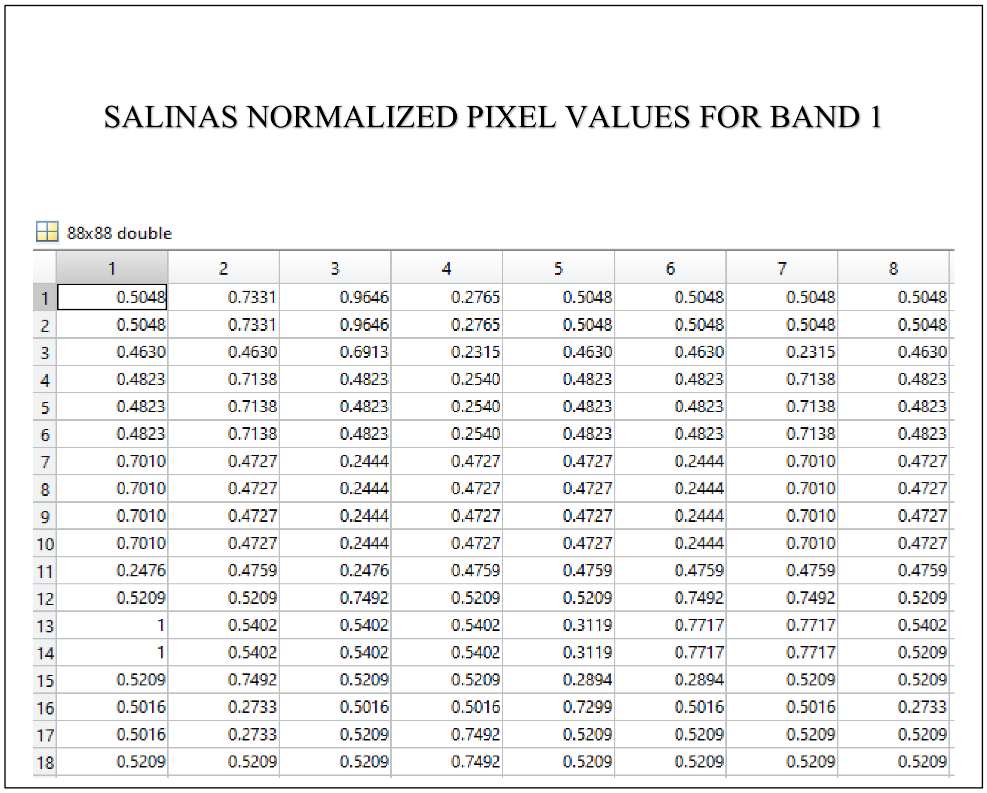
	1	2	3	4	5	6	7	8
1	3172	2580	3687	2749	2746	2575	2575	2575
2	2576	2747	2750	3686	2744	2570	2571	3173
3	2744	2576	2744	2576	2744	2576	2744	2576
4	3000	2570	2571	3168	3168	3171	2576	2744
5	2750	3855	2743	2577	3684	2743	2572	2569
6	2576	2744	2576	2747	2744	2571	3173	2747
7	3172	3685	3680	2570	2575	3680	2570	2570
8	3172	2743	2574	2745	2745	2743	2572	2574
9	2572	2574	2743	2572	2574	2745	2745	2743
10	2575	3680	2570	2743	2741	2570	2570	2570
11	2565	2560	2991	2565	2562	3165	2738	2736
12	2732	2561	2564	2732	2564	2730	3156	3156
13	2726	2819	3326	2992	2729	2992	2734	2563
14	2561	3158	3160	3326	2984	2984	2984	2987
15	3156	3153	2990	2732	2561	3327	3161	2732
16	3152	2984	3149	2810	2981	2978	2807	2815

Figure 17. Band 1 MATLAB Table for Indian Pines

In Figure 17, the photon counts for each Indian Pines pixel location for rows 1–18 and columns 1- 8 from band 1 are displayed.

C. IMAGE PREPARATION

In order to use the images in our proposed method, the data is converted into grayscale images through the "mat2gray()" function. This normalizes the pixel values to be decimal values between 0–1. This ensures that both of the images can be displayed in MATLAB and can be used in the subsequent steps. The converted data for band 1 of the Salinas and Indian Pines data is shown in Figures 18 and 19.



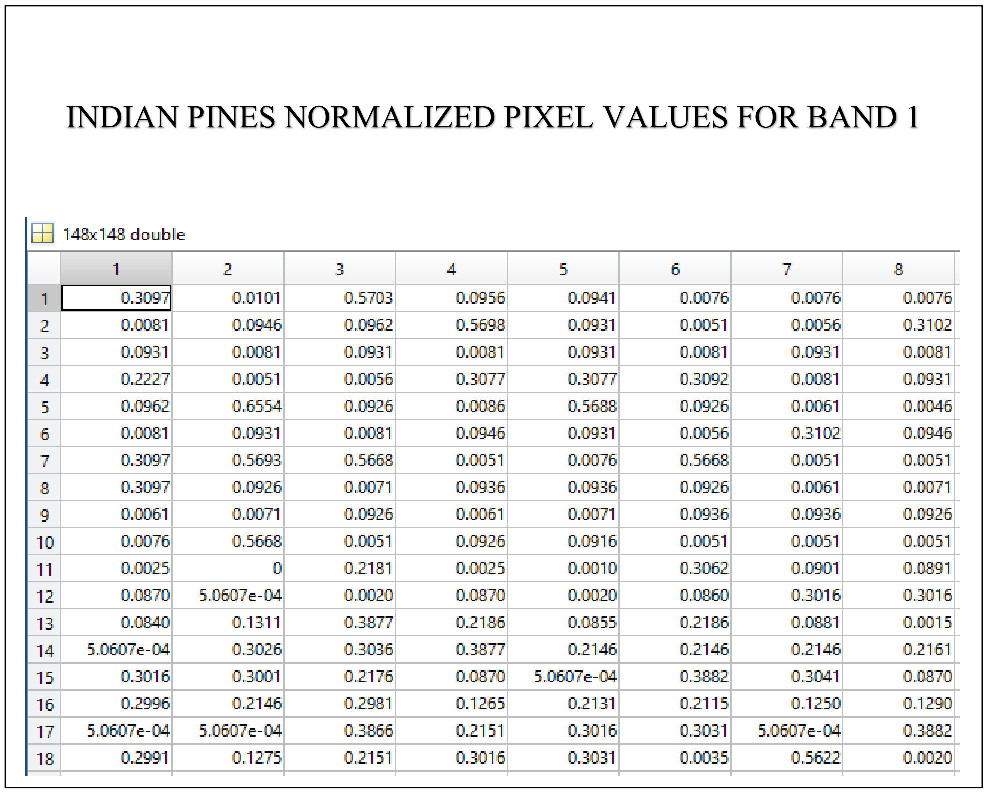
SALINAS NORMALIZED PIXEL VALUES FOR BAND 1

88x88 double

	1	2	3	4	5	6	7	8
1	0.5048	0.7331	0.9646	0.2765	0.5048	0.5048	0.5048	0.5048
2	0.5048	0.7331	0.9646	0.2765	0.5048	0.5048	0.5048	0.5048
3	0.4630	0.4630	0.6913	0.2315	0.4630	0.4630	0.2315	0.4630
4	0.4823	0.7138	0.4823	0.2540	0.4823	0.4823	0.7138	0.4823
5	0.4823	0.7138	0.4823	0.2540	0.4823	0.4823	0.7138	0.4823
6	0.4823	0.7138	0.4823	0.2540	0.4823	0.4823	0.7138	0.4823
7	0.7010	0.4727	0.2444	0.4727	0.4727	0.2444	0.7010	0.4727
8	0.7010	0.4727	0.2444	0.4727	0.4727	0.2444	0.7010	0.4727
9	0.7010	0.4727	0.2444	0.4727	0.4727	0.2444	0.7010	0.4727
10	0.7010	0.4727	0.2444	0.4727	0.4727	0.2444	0.7010	0.4727
11	0.2476	0.4759	0.2476	0.4759	0.4759	0.4759	0.4759	0.4759
12	0.5209	0.5209	0.7492	0.5209	0.5209	0.7492	0.7492	0.5209
13	1	0.5402	0.5402	0.5402	0.3119	0.7717	0.7717	0.5402
14	1	0.5402	0.5402	0.5402	0.3119	0.7717	0.7717	0.5209
15	0.5209	0.7492	0.5209	0.5209	0.2894	0.2894	0.5209	0.5209
16	0.5016	0.2733	0.5016	0.5016	0.7299	0.5016	0.5016	0.2733
17	0.5016	0.2733	0.5209	0.7492	0.5209	0.5209	0.5209	0.5209
18	0.5209	0.5209	0.5209	0.7492	0.5209	0.5209	0.5209	0.5209

Figure 18. Normalized Pixel Values for Salinas

In Figure 18, the photon counts of Figure 16 are converted to decimal values between 0 and 1, where 1 corresponds to the maximum value in the matrix, and 0 corresponds to the minimum value in the matrix.



INDIAN PINES NORMALIZED PIXEL VALUES FOR BAND 1

148x148 double

	1	2	3	4	5	6	7	8
1	0.3097	0.0101	0.5703	0.0956	0.0941	0.0076	0.0076	0.0076
2	0.0081	0.0946	0.0962	0.5698	0.0931	0.0051	0.0056	0.3102
3	0.0931	0.0081	0.0931	0.0081	0.0931	0.0081	0.0931	0.0081
4	0.2227	0.0051	0.0056	0.3077	0.3077	0.3092	0.0081	0.0931
5	0.0962	0.6554	0.0926	0.0086	0.5688	0.0926	0.0061	0.0046
6	0.0081	0.0931	0.0081	0.0946	0.0931	0.0056	0.3102	0.0946
7	0.3097	0.5693	0.5668	0.0051	0.0076	0.5668	0.0051	0.0051
8	0.3097	0.0926	0.0071	0.0936	0.0936	0.0926	0.0061	0.0071
9	0.0061	0.0071	0.0926	0.0061	0.0071	0.0936	0.0936	0.0926
10	0.0076	0.5668	0.0051	0.0926	0.0916	0.0051	0.0051	0.0051
11	0.0025	0	0.2181	0.0025	0.0010	0.3062	0.0901	0.0891
12	0.0870	5.0607e-04	0.0020	0.0870	0.0020	0.0860	0.3016	0.3016
13	0.0840	0.1311	0.3877	0.2186	0.0855	0.2186	0.0881	0.0015
14	5.0607e-04	0.3026	0.3036	0.3877	0.2146	0.2146	0.2146	0.2161
15	0.3016	0.3001	0.2176	0.0870	5.0607e-04	0.3882	0.3041	0.0870
16	0.2996	0.2146	0.2981	0.1265	0.2131	0.2115	0.1250	0.1290
17	5.0607e-04	5.0607e-04	0.3866	0.2151	0.3016	0.3031	5.0607e-04	0.3882
18	0.2991	0.1275	0.2151	0.3016	0.3031	0.0035	0.5622	0.0020

Figure 19. Normalized Pixel Values for Indian Pines

Figure 19 contains the normalized values of the photon counts for the Indian Pines hyperspectral band 1. These values are the normalization of the values in Figure 17.

The next step for image preparation was to extend the images in order to make symmetrical rows and columns that were divisible by 2 for the low-resolution degraded images. For Indian Pines, the hyperspectral image was extended through a zero-order hold to a size of 148x148x200. For the Salinas, the image was extended through a zero-order hold to a size of 88x88x204. The zero-order hold takes the values at the edge of the image and extends them at a constant value.

D. IMAGE DEGRADATION

The next step in the algorithm was to down-sample each band in the hyperspectral image using a 2D down-sampling filter, which compresses the image to one-half of the original size. The method by which this is accomplished is by taking the values at every second pixel location in the band as discussed in Equation 3. The implementation in code was done by indexing the original high-resolution data matrix at even locations.

After down-sampling, random gaussian noise was added to the low-resolution wavelength bands to simulate images captured on a hyperspectral camera. The “imnoise()” function is used to add random gaussian noise to each band in the down-sampled image to create degraded low-resolution images.

E. GROUND TRUTH DATA

The next part of the image enhancement algorithm was to import the ground truth binary data into the MATLAB workspace. The ground truth was used later to calculate the signal-to-noise ratio of the enhanced images. The ground truth data for Indian Pines is shown in Figure 20.

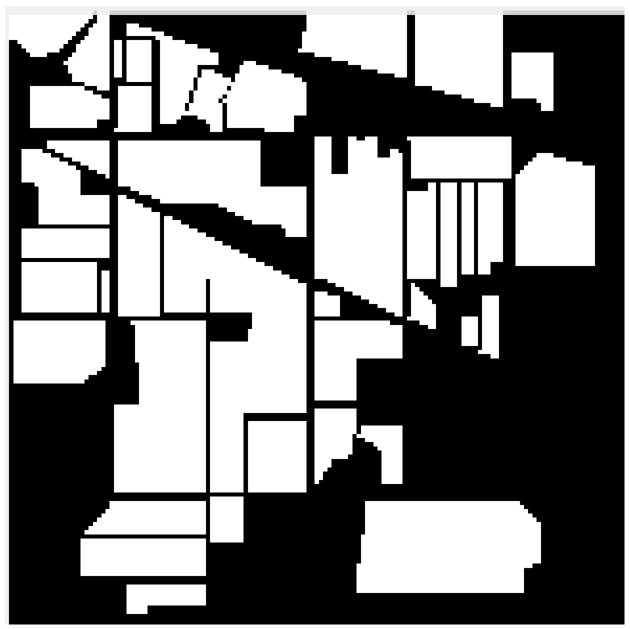


Figure 20. Indian Pines Ground Truth Data.

In Figure 20, white areas are where the desired signal is located in the image. The black areas of the image are defined as the noise for the SNR calculation. The ground truth data for the Salinas hyperspectral image is shown in Figure 21.

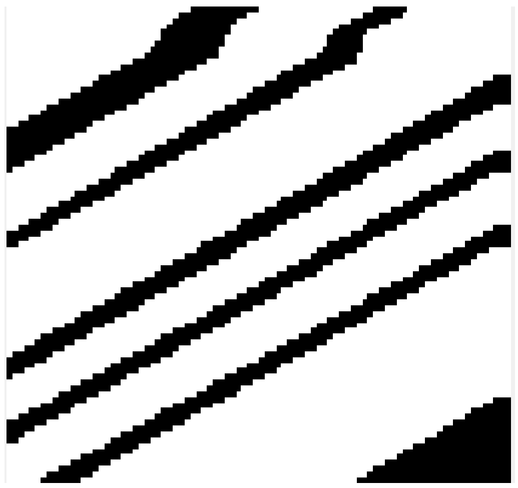


Figure 21. Salinas Ground Truth Data

In Figure 21, the ground truth for the Salinas data is displayed. The white areas correspond to the signal locations, and the black areas correspond to the noise locations.

F. IMAGE BAND GROUPING

The next step in the image enhancement approach was to group bands for image fusion. There are four parts to image grouping. First, calculate the zero-mean low-resolution bands. Second, calculate the correlation between the zero-mean bands. Third, sort the bands into groups of three. Finally, store the resulting groups of three bands in a matrix to use for image fusion. The method of grouping was to calculate the cross-correlation between all wavelength bands within the hyperspectral image. Zero-mean images remove the correlation caused by a similar offset [11]. The zero-mean wavelength bands were calculated by subtracting the mean of the band from itself. This was done to aid in removing a common mean between wavelength bands as the cause for a high correlation coefficient between them. If the mean was not subtracted, a similar mean across wavelength bands could cause drastically different images to have a high correlation [11]. These low-resolution zero-mean wavelength bands were used to create a correlation matrix between bands. An example of the correlation matrix is shown in Figure 22.

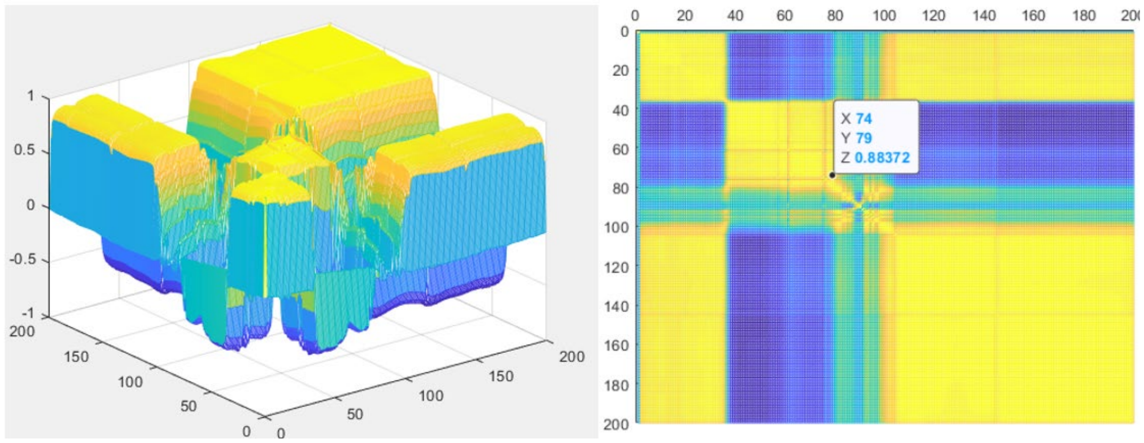


Figure 22. Low-Resolution Hyperspectral Correlation Matrix for Indian Pines.

In Figure 22, the higher correlations are indicated by a lighter color. The darker colors indicate a low or even negative correlation. The x and y coordinates represent pairs of bands, and the z-axis value represents the correlation coefficient. Bands 74 and 79 have a correlation coefficient of 0.88. These correlation values are used to group the wavelength bands. The correlation values are sorted from highest to lowest for each band. A matching function creates groups of three bands based upon the highest correlations. The two highest correlation values are used as the 2nd and 3rd band in the group. An excerpt from the Salinas grouping pairs matrix is shown in Table 1.

Table 1. Salinas Wavelength Band Grouping Table

Group	13	14	15	16	17	18	19	20	21
LR band 1 (correlation coefficient)	13 (1.000)	14 (1.000)	15 (1.000)	16 (1.000)	17 (1.000)	18 (1.000)	19 (1.000)	20 (1.000)	21 (1.000)
LR band 2 (correlation coefficient)	14 (0.9993)	15 (0.9993)	14 (0.9993)	15 (0.9979)	18 (0.9966)	19 (0.9993)	18 (0.9993)	19 (0.9991)	20 (0.9966)
LR band 3 (correlation coefficient)	12 (0.9991)	13 (0.9993)	13 (0.9980)	14 (0.9959)	21 (0.9957)	20 (0.9986)	20 (0.9991)	18 (0.9986)	22 (0.9965)

The grouping matrix was used later in the image fusion step in order to have a stored set of indexes that can be referenced for the groups of bands. In the fusion step, the function increments which group to look at in Table 1. Then the 1st, 2nd, and 3rd bands in the group are fused together.

Some wavelength bands are used more than others due to some bands being spectrally similar while having low spatial correlation. The disparity between spatial and spectral correlation leads to groupings that are not symmetrical. When the top three values are calculated with either using the “maxk()” or “sort()” functions in MATLAB, there are several non-repeating examples in the groupings. These functions return the highest *n* values in an array. In Table 1, the first band taken from the low-resolution image will always be the group number. These values are marked in gray. An example of the non-repeating nature is shown in group 17. Normally, it would be expected that band 17 would be most correlated with the wavelengths closest to it, such as 16 and 18. However, because band 17 is most correlated with 18 and 21, those are the bands chosen for the group. For band 21, the highest correlating bands are those that are neighboring it. The explanation for this is that band 17 has its own highest correlation with 18 and 21, but bands 18 and 21 have higher correlations with other bands.

G. IMAGE FUSION

After the image grouping, the three bands are then fused using a discrete wavelet frame transform (DWFT). The first step is to calculate the wavelet transform and the wavelet coefficients. These calculations are done by implementing the wavelet functions defined in Chapter II, Section D. In MATLAB, this can be done by using the DWFT function “swt2()” shown in Appendix A, Section E. By fusing the grouped bands through the wavelet coefficient addition, the spatial frequency information of each band is incorporated.

Next, an inverse discrete wavelet frame transform was used to create a fused image from the coefficients that were added together. The function in MATLAB is “iswt2().” The resultant output is a fused image with spatial frequency components of all three grouped bands. An example is shown in Figure 23 of the image fusion result after fusing the 50th group. Group 50 uses bands 50, 49, and 51 of the degraded low-resolution image.

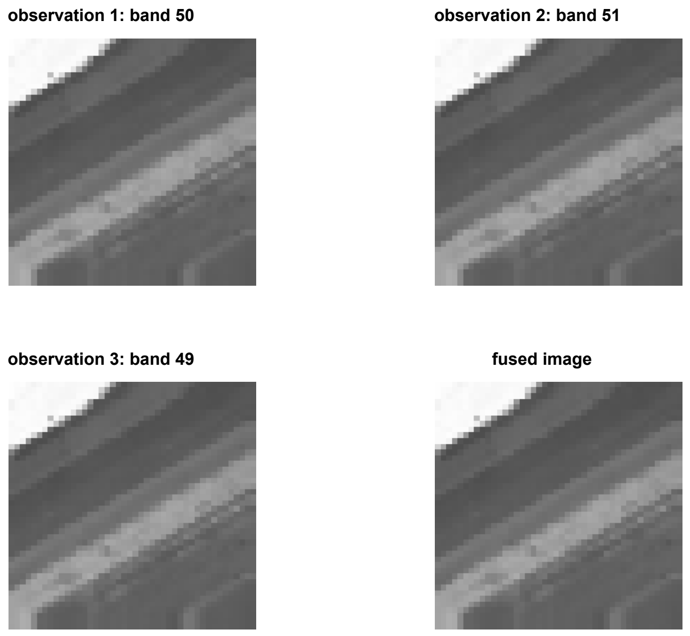


Figure 23. Image Fusion Example Band for Salinas.

In Figure 23, the three bands that are used and the resulting fused band are very similar to the naked eye. Although individual pixel values at each location in the bands vary by very small amounts, these small deviations can lead to large impacts in SNR performance across an entire hyperspectral image with hundreds of bands.

H. IMAGE INTERPOLATION

The next step in the proposed image enhancement algorithm is to interpolate the fused data. The fused image data is interpolated using the LMMSE, maximum entropy, and normalization algorithms, which are attached in Appendix A, Sections B, C, and D. The purpose of the interpolation is to estimate the high-resolution hyperspectral image that was used for error calculation and signal-to-noise ratio.

The unfused low-resolution images are interpolated using the simpler interpolation methods of bilinear, bicubic, and nearest neighbors. These will act as a baseline to measure the effectiveness of image fusion and more complex interpolation methods. Each of these interpolation methods attempts to increase the resolution by the same factor that the image was down-sampled by. An example of the resulting image set is shown in Figure 24.

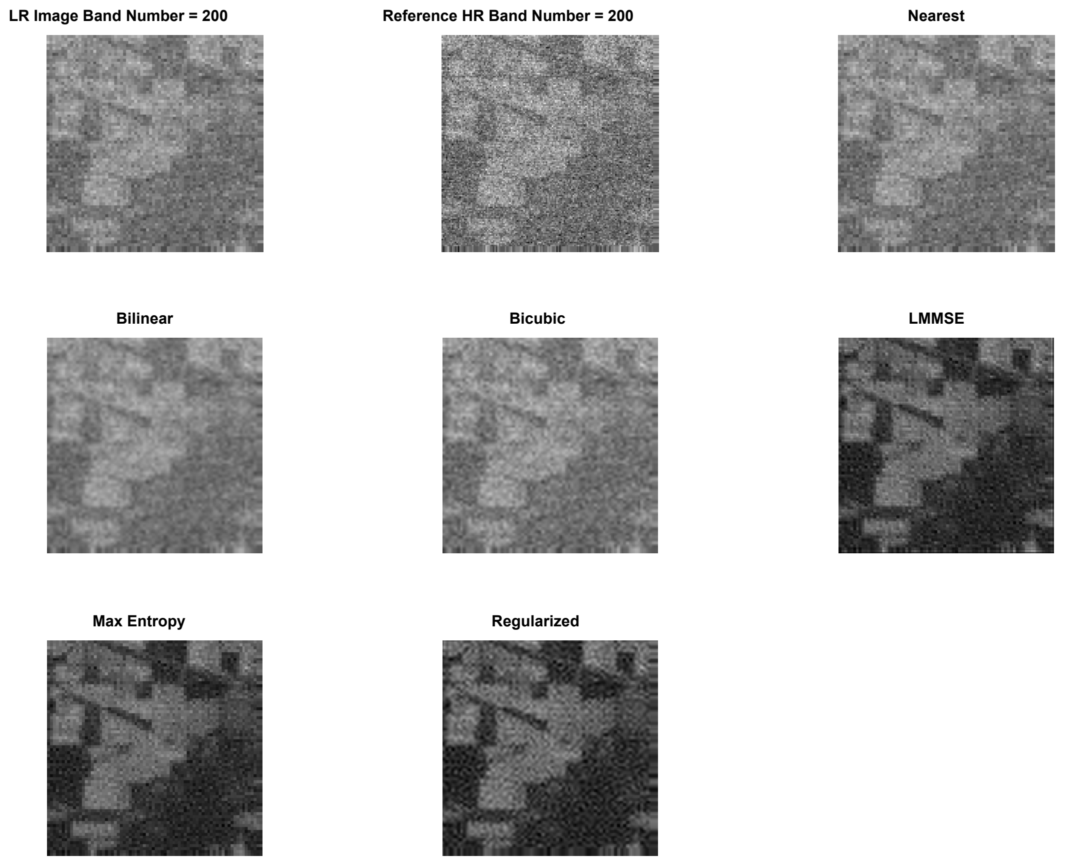


Figure 24. Indian Pines Interpolation Set of a Single Hyperspectral Band.

In Figure 24, the 200th band of the estimated image is shown for each method used on the Indian Pines hyperspectral image. The LMMSE, max entropy, and regularization algorithm reduce background noise and enhance feature edges. This can be very helpful in segmentation and aid in visual classification.

I. IMAGE ERROR

Image error was calculated by adding the distance of each estimated interpolation pixel from the reference high-resolution image pixels across the entire hyperspectral cube. The calculation for this was done by the sum of the absolute value difference matrix in all dimensions, as shown in Equation (32)

$$Error = \sum_{x}\sum_{y}\sum_{z}\left|(A-B)\right|, \tag{32}$$

where *A* is the high-resolution reference image, and *B* is the estimated high-resolution image. This was done through matrix subtraction, then by taking the absolute value of the difference in all dimensions. The sum of all the differences between the original hyperspectral image pixel values and the image estimate pixel values is our error metric.

J. SIGNAL-TO-NOISE RATIO

The signal-to-noise ratio was calculated by using the labeled truth matrix data supplied by the database to assign what was the true signal in the reference image. The signal matrix was calculated by multiplying the binary ground truth through every band in the estimated high-resolution image. This creates a signal image that only includes the parts of the image that are labeled as actual signals by the hyperspectral database. The inverse of the labeled truth is then multiplied by the estimated high-resolution image to calculate the noise component of the image. The SNR of the entire hyperspectral cube was calculated by finding the ratio of the sum of all true signal values in the hyperspectral cube to the sum of all noise components of the hyperspectral cube. This was done through Equation 33.

$$SNR = 20Log\left(\frac{\sum_{x}\sum_{y}\sum_{z} A}{\sum_{x}\sum_{y}\sum_{z} B}\right), \tag{33}$$

where *A* is the signal matrix pixel value sum in all dimensions calculated by using the “sum()” function for all dimensions, and *B* is the noise matrix pixel value sum in all dimensions. An example of Signal and Noise images is shown in Figure 25.

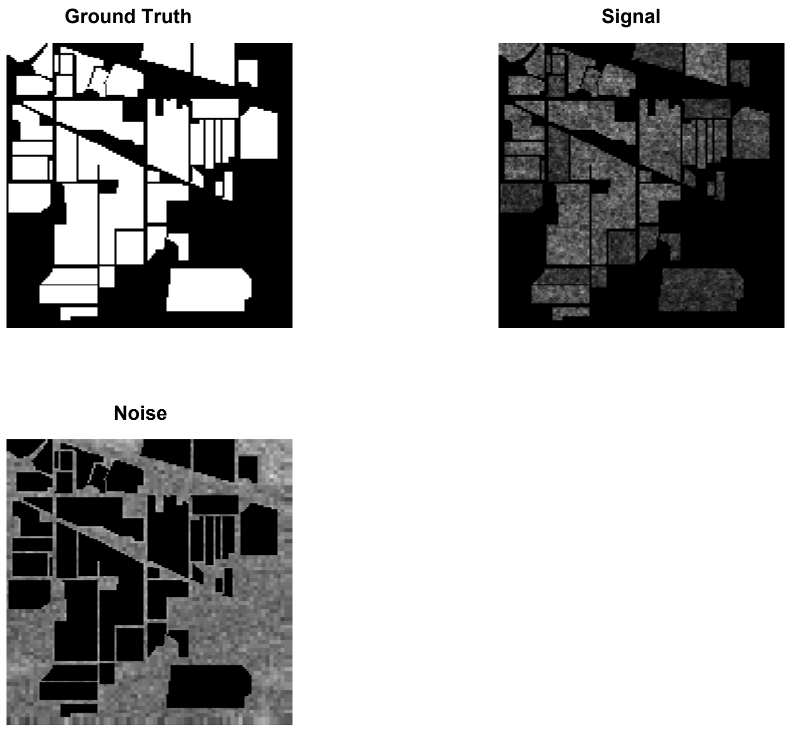


Figure 25. SNR Visualization Images.

In Figure 25, the signal and noise images are made by multiplying a band of the estimated hyperspectral image by the ground truth data. White areas in the ground truth image are areas where the database has defined the signal to be located. Multiplying the ground truth and estimate generates the part of the estimated image that has useful signal components. The inverse of the ground truth data is achieved by essentially reversing which spots are white with the spots that are black in the image. Multiplying the inverse of ground truth to each band in the hyperspectral image results in the noise component of the hyperspectral image.

K. IMAGE COMPRESSION

Image compression was done using a PCA. The mathematical algorithm to conduct PCA compression is described in Chapter II, Section F. The code to implement PCA is in Appendix A, Section A. For each level of compression, from 1 to 20 eigenvectors, the compressed image was interpolated. The compressed interpolation image estimate was then used to calculate the error and SNR. The time elapsed while calculating the estimated image was recorded for each level of compression.

L. IMAGE ESTIMATION TIME AND COST

Image estimation computation time was calculated through the use of the “tic toc” function in MATLAB, which records the time between the two points in code. In each loop of the PCA compression, the time to calculate the image estimations is recorded. The “tic” is used at the beginning of the loop, the “toc” at the end of the loop and the data recorded in the variable “time PCA.” The time to estimate the image along with the error value associated with the level of compression was used to calculate the cost. The equation for cost is Equation (31).

THIS PAGE INTENTIONALLY LEFT BLANK

IV. RESULTS

A. OVERVIEW OF RESULTS

In this chapter we will discuss the results of the image enhancement approach outlined in Chapter III. The image enhancement approach is applied to the Salinas and Indian Pines datasets. In both the compressed and uncompressed steps of the algorithm, the speed, error, SNR, and cost of each method were recorded.

B. DATASET

1. Sensor

The data used in this thesis was acquired from publicly available databases. In this analysis, the corrected data was used. The image was first captured on June 12, 1992, and was downloaded from Purdue's imaging research database [5]. The device that obtained the hyperspectral image was the Airborne Visible/Infrared Imaging Spectrometer (AVIRIS). The AVIRIS is a "whisk broom" scanning sensor that sweeps detector pixels (Silicon (Si) detectors for the visible range, indium gallium arsenide (InGaAr) for the NIR, and indium-antimonide (InSb) detectors for the SWIR) across a 614 pixel-wide swath using a spectrometer to separate the light into 224 unique spectral bands in a range of 0.4 microns to 2.5 microns. Each band has a resolution of 0.01 microns calibrated to within a 1-nanometer error.

2. Images

The images used for this experiment in hyperspectral-image enhancement were hyperspectral images of Indian Pines and Salinas. Table 2 lists the data characteristics of both hyperspectral images.

Table 2. Salinas and Indian Pines Dataset Details

	Indian Pines	Salinas
BANDS	220	204
ROWS	145	83
COLUMNS	145	86
PIXELS	4,625,500	1,456,152

In Table 2, the number of pixels in the Indian Pines image is approximately 3 times that of the Salinas image. Figure 26 is a false-color representation of the Indian Pines scene. This scene has many details in the spatial domain and thin boundaries between farmlands. These details allow for a better visual evaluation of the performance of the image enhancement approach presented in this thesis.

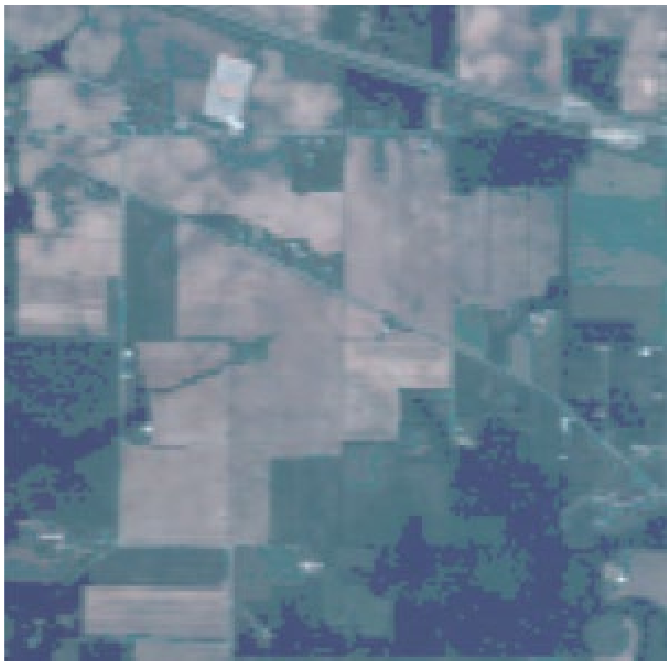


Figure 26. Indian Pines Represented in RGB. Source: [5].

In Figure 26, the red, blue, and green wavelength pixel values of the Indian Pines hyperspectral image are used to generate an RGB image.

C. NON-COMPRESSED RESULTS

The first part of the results will analyze the uncompressed image enhancement of the hyperspectral cube. This will measure the performance of each method before the compression into a lower-dimensional space through the use of PCA.

1. Error Performance

As discussed in Chapter III, Section F, the error is measured as the absolute value difference between the pixel intensity values of the reference image and the estimated high-resolution image. In Table 3, the error is calculated for each interpolation method using the Salinas and Indian Pines data. The reason for the difference in magnitude between the Indian Pines Data and the Salinas data is due to their differences in image size. The lowest error value is in bold.

Table 3. Uncompressed Indian Pines and Salinas Error

	Indian Pines	Salinas
Nearest Neighbor	120400	18412
Bilinear	120231	18579
Bicubic	**108062**	**16411**
LMMSE	172511	50612
Max Entropy	163199	39841
Regularized	153116	40394

In both the Salinas and Indian Pines hyperspectral images, the bicubic interpolation was best at recreating an exact estimation of the original data cube. Its error from the original values across the entire image was far less than any other method used. However, this does not represent the enhancement of the image. SNR will add additional perspective on how the more complex methods aid in identifying meaningful features in the image.

2. Signal-to-Noise Ratio

The SNR was calculated using the method described in Chapter III, Section G. The SNR values listed in Table 4 are used to determine the effect that each method has on the SNR using the ground truth data. It is important to note that different images will have a different maximum SNR. If the ground truth covers most of the image, then the magnitude of SNR will be very high, as most of the image is the signal desired. The difference in SNR values between each method for an image column in Table 3 should be used to compare the effectiveness of each method at enhancing the image signal data, rather than simply looking at the SNR magnitude.

Table 4. Uncompressed Indian Pines and Salinas SNR

	Indian Pines	Salinas
Nearest Neighbor	-11.42	.091
Bilinear	-11.59	.063
Bicubic	-11.36	.039
LMMSE	**-11.08**	**.259**
Max Entropy	-11.23	.025
Regularized	**-11.00**	.043

In both the Salinas and Indian Pines data, the two best performers were the LMMSE and Regularization algorithm. It is important to note that the method with the least error, the bicubic interpolation, had the least SNR in both cases. The bicubic interpolation is effective in recreating the original image but does not enhance the desired features of interest. The error performance in most methods was inversely related to its SNR performance. This was caused by an increase of intensity values of the signal portion of the hyperspectral image. Since the pixel intensity values were increased, it increased the error as we have defined it in Chapter III, Section F.

D. PCA IMAGE ENHANCEMENT RESULTS

The next step in the algorithm is to compress the hyperspectral cube using several levels of compression by using a finite number of eigenvectors N_{pca}. This was the "compressed" side of the image enhancement approach seen in Figures 14 and 15. In this case, this meant compressing the data using 1 to 20 eigenvectors of the covariance matrix. For each level of compression, the image enhancement was done on the compressed image.

1. PCA Error Performance

As described in Chapter III, Section F, the error for each compressed image was calculated in the same manner as the uncompressed method. This was done by measuring the difference between the estimated image and the reference image. The error for both the Salinas and Indian Pines methods tended to decrease as more eigenvectors were used, and the level of compression decreased. This was due to more information from the original dataspace being added. This generally would make the estimated image closer to the reference image. However, in some cases, the information added caused the methods to erroneously interpolate the final image. This led to an error increase for some compression levels, indicating that at some point, adding higher-frequency components was detrimental. The error values trend similarly to the uncompressed results, with the bicubic interpolation being the method that created the least difference between the estimated image and the reference. Figures 27 and 28 show the Indian Pines and Salinas error plots.

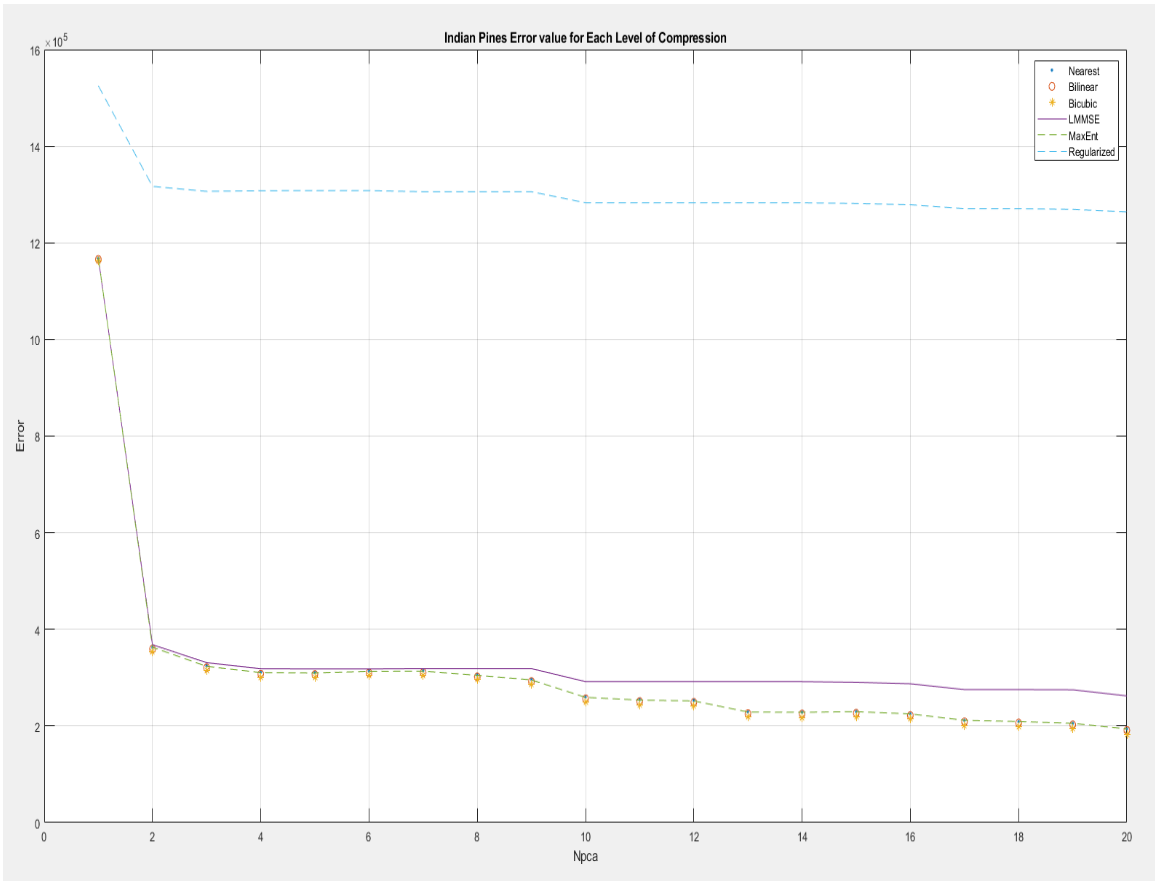


Figure 27. Indian Pines Error for Each Level of Compression.

In Figure 27, the bicubic interpolation again was the best performing method at recreating the original high-resolution image. As more eigenvectors are introduced, the different methods diverge in their error values. The regularized algorithm did not perform well when the data was compressed to a lower-dimensional space, as seen by its higher error value.

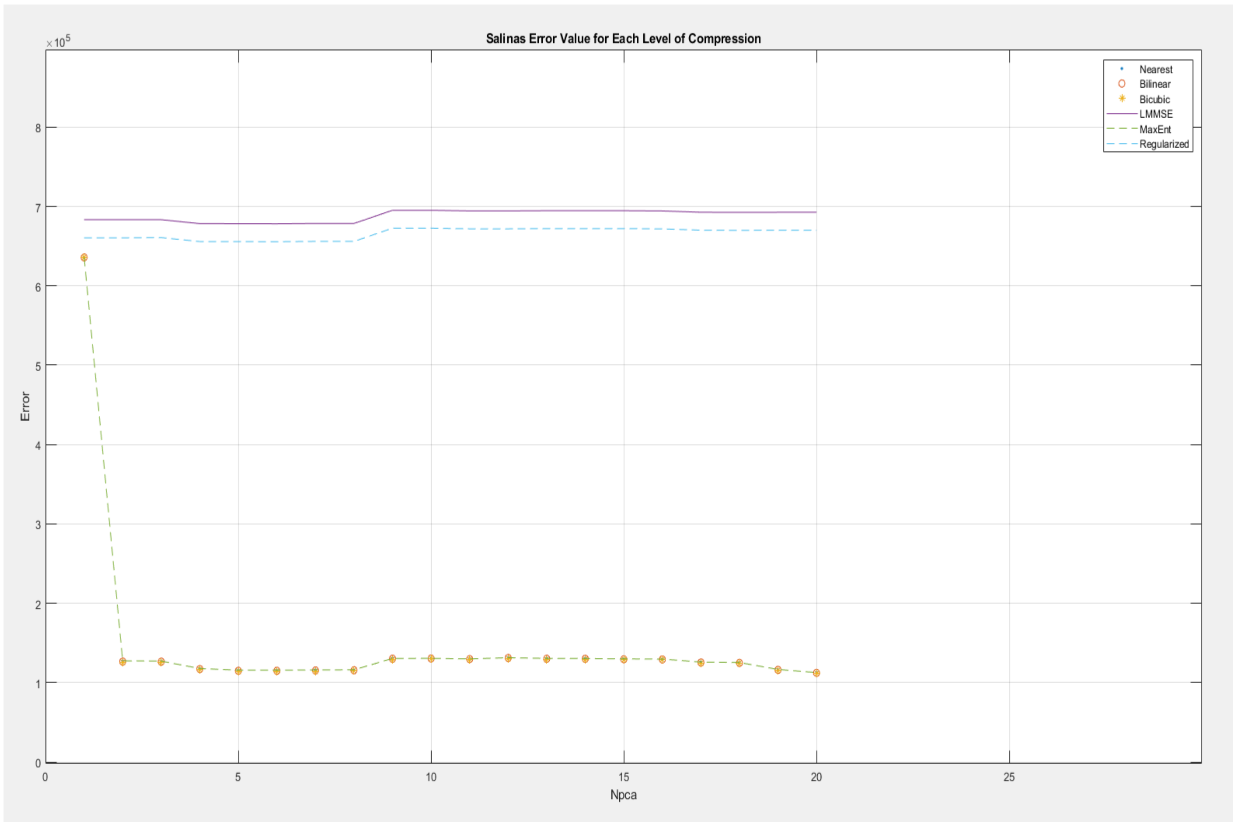


Figure 28. Salinas Error for Each Level of Compression.

In Figure 28, it is apparent that the smaller data size of the Salinas hyperspectral image, along with the data compression from PCA, caused the error values of the LMMSE and regularized interpolation methods to be high compared to the other methods. the bicubic, max entropy, nearest neighbor, and bilinear methods performed much better in compressed data spaces.

2. PCA SNR Performance

The SNR performance of the six methods at each level of compression would be expected to follow the trend similar to the uncompressed SNR plots. The error plots for each dataset followed the uncompressed results closely. For the SNR, however, the results for the compressed data deviate from the uncompressed data results. In Figures 29 and 30, the SNR for both the regularization and LMMSE algorithms is erratic and inconsistent. This is also evident in their high error values in Figures 27 and 28. This drives the conclusion that the compressed data does not give enough information for these methods to be effective in accurately interpolating the fused image. This causes the methods to introduce errors in the estimated image, causing the low SNR and high error values.

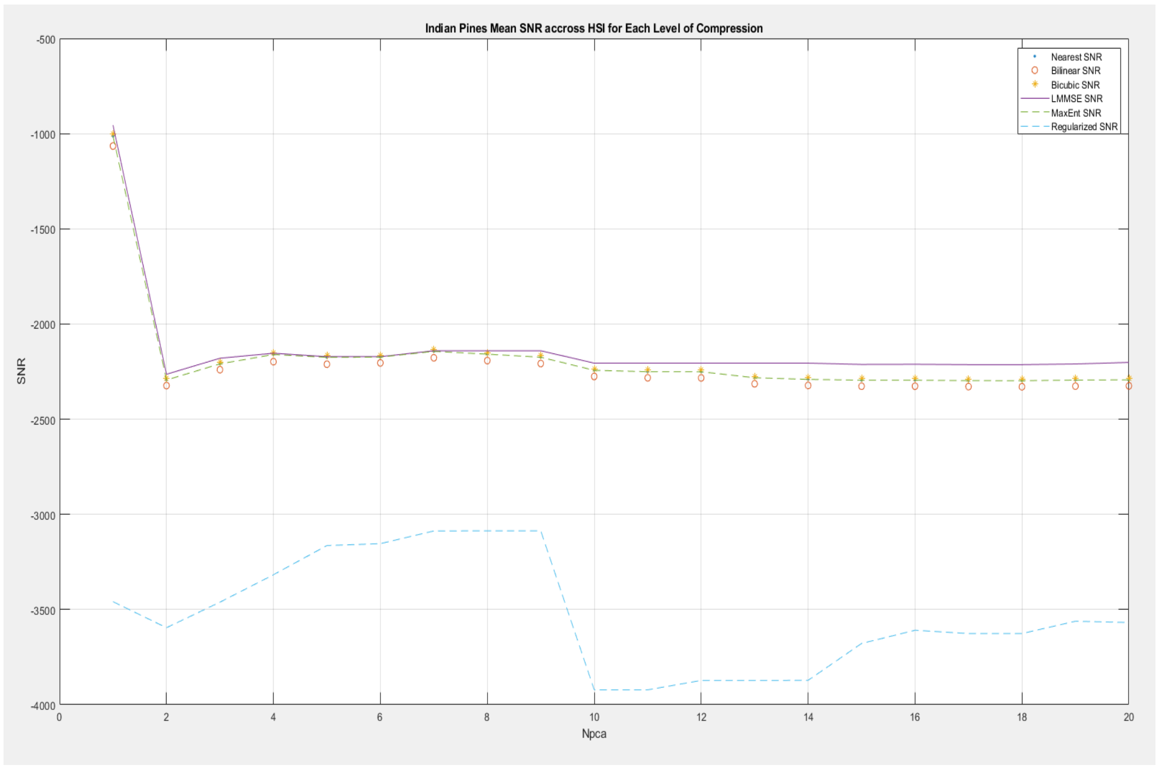


Figure 29. Indian Pines SNR for Each Level of Compression.

In Figure 29, the highest SNR values across all levels of compression were from the LMMSE method. The worst performing method was the regularization method. The conclusion that was drawn from this was that due to the larger data set from the Indian Pines image, the LMMSE interpolation was able to effectively filter noise and enhance the signal portions of the image as defined in Chapter III, Section E.

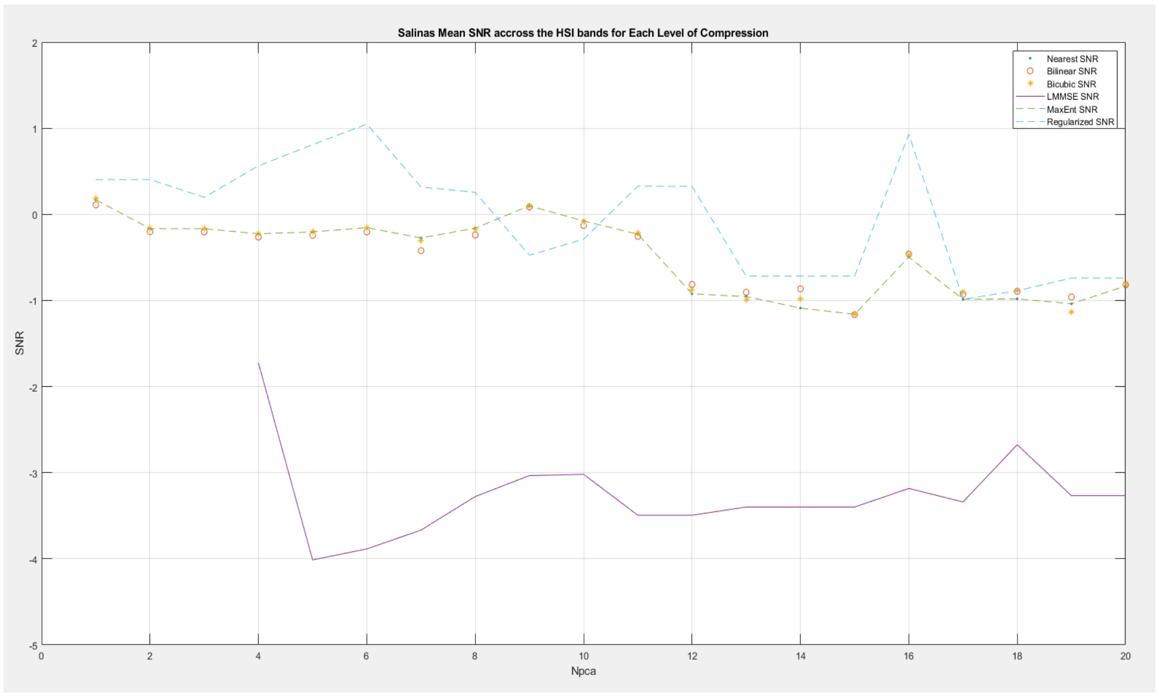


Figure 30. Salinas SNR for Each Level of Compression.

In Figure 30, the highest SNR values across all levels of compression were from the Regularization method. The worst performing method was the LMMSE method. The conclusion that was drawn from this was that due to the smaller data set from the Salinas image, the LMMSE interpolation was unable to effectively filter noise and enhance the signal portions of the image as defined in Chapter III, Section E. The effectiveness of the regularization method is counter-intuitive when taking into consideration its poor performance in the Indian Pines compressed images. Possible causes for this could be the way that data was oriented in the Salinas image.

3. PCA Speed Performance

The PCA time elapsed for both the Salinas and Indian Pines dataset increases linearly with the number of eigenvectors used for compression. The time taken for each compression level is shown in Figures 31 and 32 for both datasets. The total time for the Indian Pines hyperspectral image was approximately 7 times longer than the Salinas total compression time. This is explained through the larger amount of data present in the Indian

Pines hyperspectral image. The total number of pixels in the Indian Pines hyperspectral cube is approximately 3 times that of the Salinas data cube.

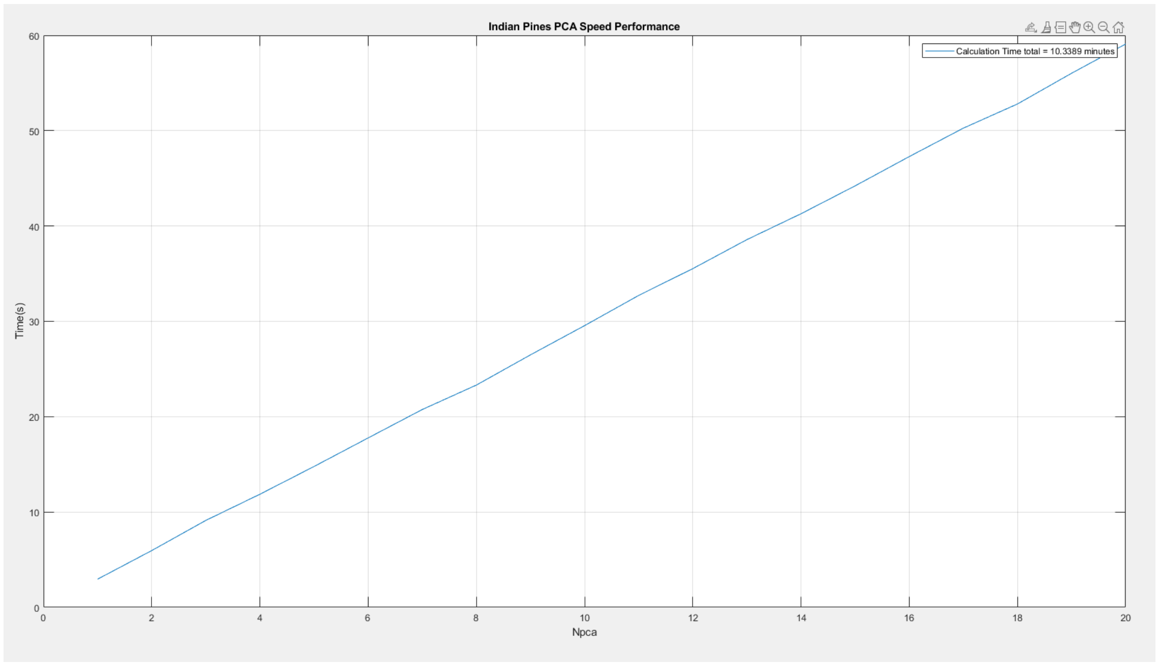


Figure 31. Indian Pines Time Elapsed for Image Estimation for Each Level of Compression.

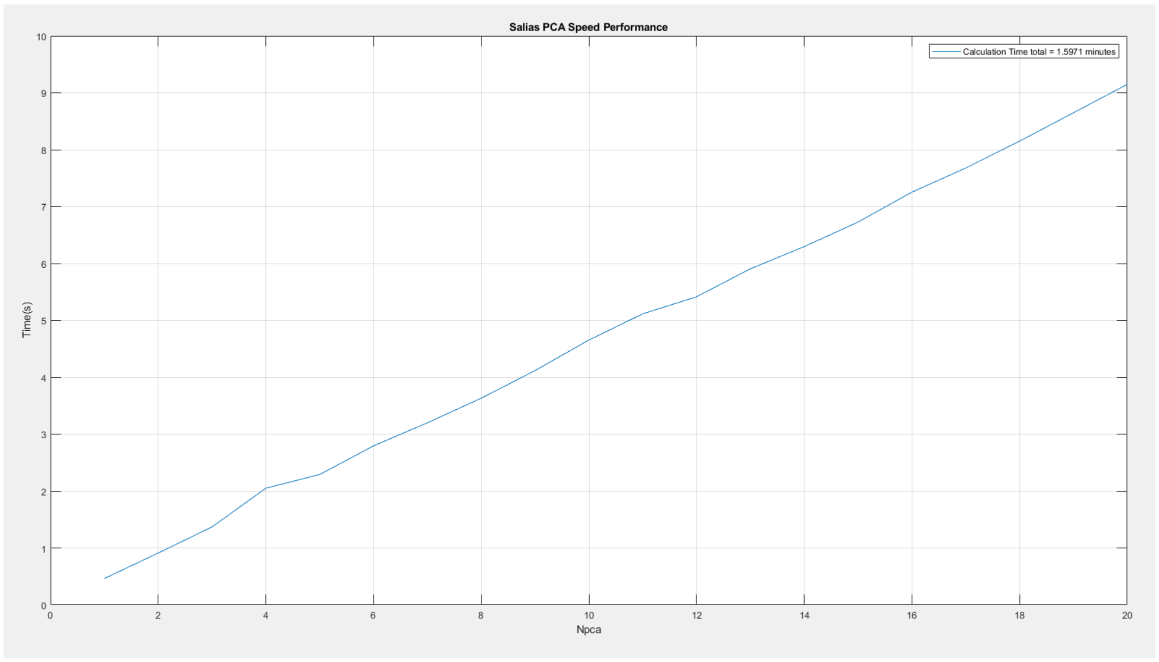


Figure 32. Salinas Time Elapsed for Image Estimation for Each Compression Level.

4. PCA Cost Per Level of Compression

The cost of each method used was calculated by the cost function defined in Chapter II, Section G. By incorporating the error and the time taken to complete the image enhancement, it is possible to determine an optimal number of eigenvectors to use in the image compression. In Figures 33 and 34, the cost for each compression level is displayed. In general, the cost is dominated by the error value due to the weighting factors placing emphasis on the error rather than the time for the calculation. Between the Indian Pines and Salinas datasets, the major difference is the performance of the LMMSE and Regularization algorithms. This performance drop is caused by the difference in the data size of the low-resolution degraded images. Salinas has fewer pixels in its hyperspectral image. When the smaller Salinas hyperspectral data is degraded, down-sampled, and compressed, the LMMSE and regularization algorithms perform poorly.

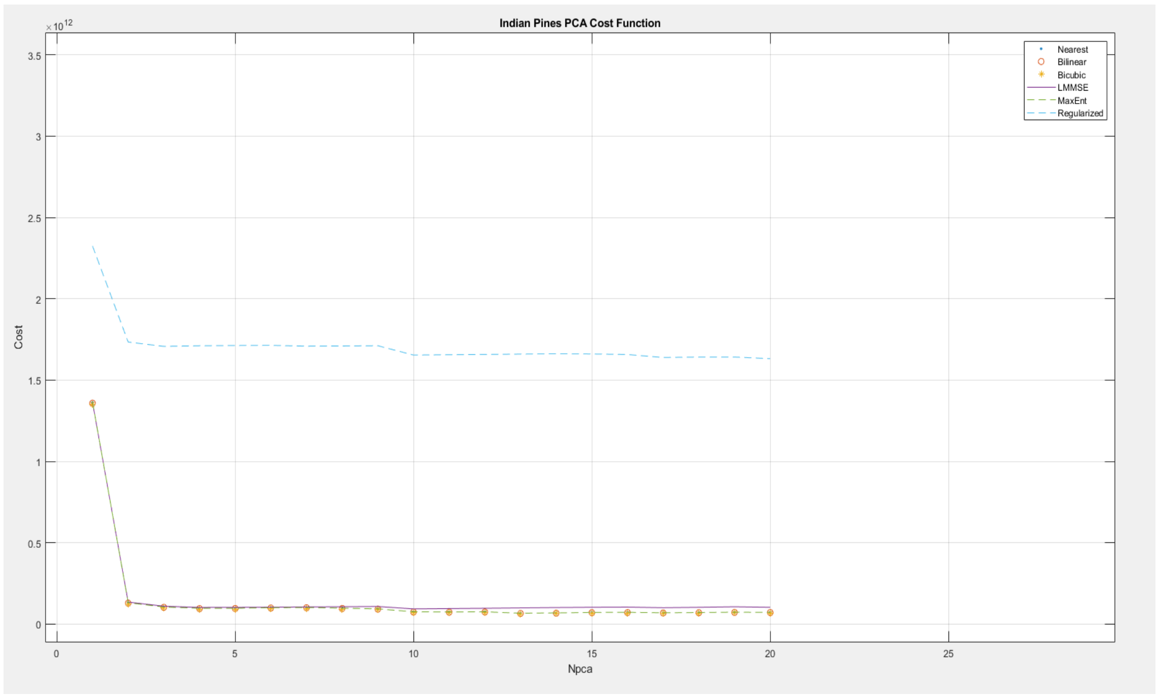


Figure 33. Indian Pines Cost for Each Level of PCA Compression.

In Figure 33, the minimum cost level for the Indian Pines image is located at 13 eigenvectors used for the PCA compression. This is common across all of the methods used.

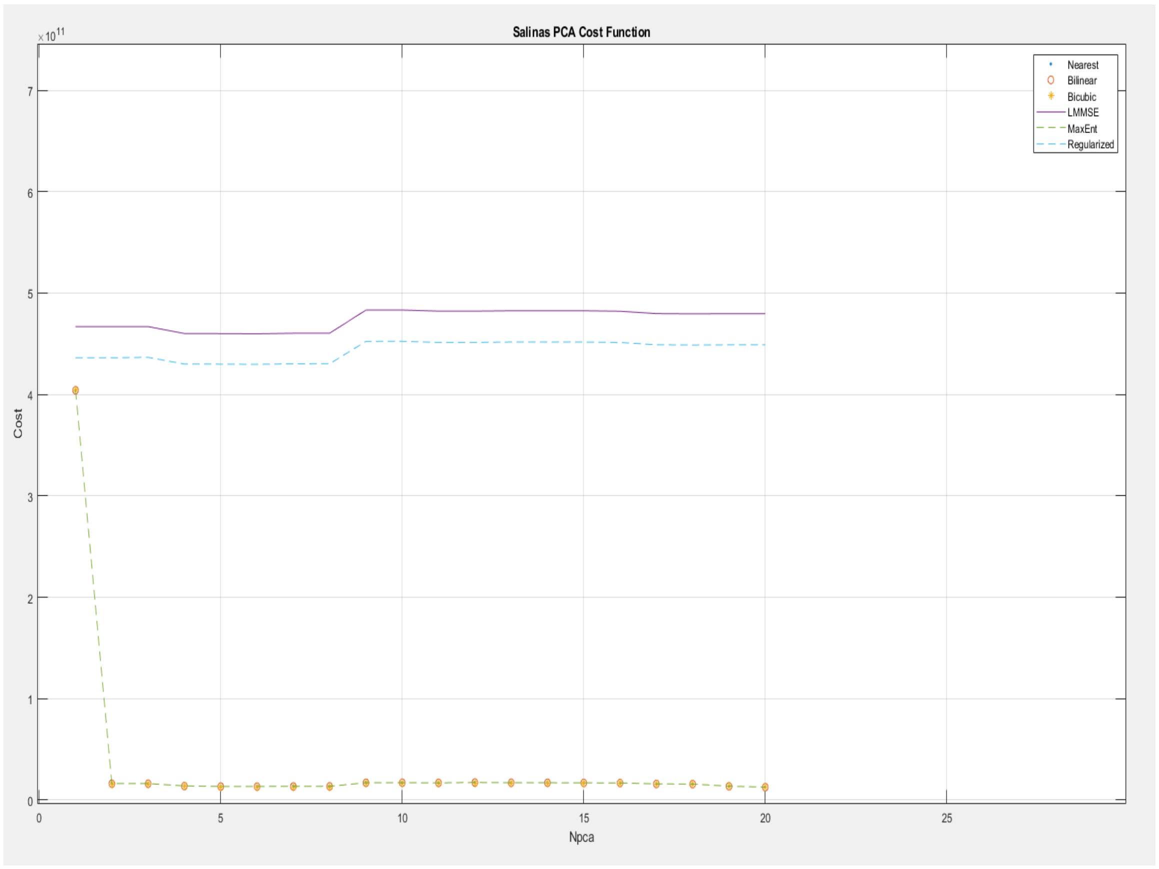


Figure 34. Salinas Cost for Each Level of PCA Compression

In Figure 34, the minimum cost level for the Salinas image is located at 8 eigenvectors used for the PCA compression. This is common across all of the methods used. The cost for LMMSE and Regularization methods were high due to their high error values.

5. Optimal Eigenvalue Determination

The optimal number of eigenvalues for compression can be determined by the compression level with the corresponding lowest cost value. The lowest cost values for each method are shown for both datasets in Table 4. The percentage of eigenvectors used for the Indian Pines image was an average of 5% of the total eigenvectors in the image. For the Salinas data, the average amount of eigenvectors used was 4%.

Table 5. Indian Pines and Salinas Optimal Compression Level.

	Indian Pines	Salinas
Nearest Neighbor	13/20	8/20
Bilinear	13/20	8/20
Bicubic	13/20	8/20
LMMSE	13/20	8/20
Max Entropy	13/20	8/20
Regularized	13/20	8/20

V. CONCLUSIONS

A. CONCLUSIONS

Of the six methods examined in this thesis, the performance of the three complex methods used with image fusion was undoubtedly dependent on the dataset size. The three proposed methods had difficulty maintaining image estimation accuracy when dealing with highly compressed low-resolution data, as seen in Figures 29 and 30. In contrast, the simpler methods were better able to recreate the original hyperspectral image even at higher compression levels. However, the three complex methods performed much better using the metric of SNR, as seen in Table 4. If the goal is to only recreate the original hyperspectral data to the maximum extent, the bicubic method is the most effective out of these six methods. If the goal is to enhance the desired signal in the image, then the LMMSE and regularization algorithms perform best.

The compression level was shown to be a determining factor in the effectiveness of the more complex methods. In the uncompressed datasets, the Bicubic interpolation without image fusion had the least deviation from the reference hyperspectral image. Conversely, The LMMSE and regularization algorithms had the best SNR performance in the uncompressed image enhancement algorithm. In the compressed datasets, the best performance in reference image error was accomplished through the use of the bicubic interpolation. The best SNR performance in the Indian Pines hyperspectral image was the LMMSE algorithm. However, when LMMSE was used in the compressed dataset of the Salinas hyperspectral image, the LMMSE algorithm no longer performs correctly.

It is my conclusion that for image classification, SNR in this investigation is more important than image error. SNR determines the ability to detect the desired signals in the image. The image error for the complex methods can be attributed to the signal portion of the hyperspectral image being enhanced. As the signal portion of the estimated image is enhanced, the greater the difference is from the original image. Therefore, the LMMSE and Regularized algorithms are best for interpolating fused images with uncompressed data. The change in performance due to image size leads to the conclusion that if the degraded low-resolution image does not have enough pixels, the more complex interpolation methods are ineffective and can be detrimental to the final image interpolation. The benefit of this image enhancement approach would be that it would be easier to conduct image segmentation and classification as the true source signals in the hyperspectral image are enhanced over the noise components.

B. FUTURE STUDIES

Future studies could expand the number of algorithms used in combination with the PCA compression in order to explore methods that are best compatible with compressed data and high-order information loss. There are several methods that could be used throughout the image enhancement process. For the image grouping, feature-based, Fourier-based, and area-based methods are possible. For image fusion, curvelet fusion and pixel-level fusion methods can be used. Finally, for interpolation, a Projection onto Convex Sets (POCS) algorithm could be used. Any combination of these methods and many not listed here could be explored in order to maximize the information from degraded low-resolution images.

APPENDIX. MATLAB CODE

A. OVERALL IMAGE ENHANCEMENT CODE

```
%NOTES
%use LR images to fuse, input g and other necessary parameters as
inputs to
%the functions
clc
clear
close all
load SalinasA_corrected.mat
load SalinasA_gt.mat

%AVIRIS SENSOR DATA
wavelength_range = (2.5e-6)-(.4e-6);
wavelength_perband = wavelength_range/224;
wavelengths = .4e-6:wavelength_perband:(.4e-6+((224-
1)*wavelength_perband));

%set original HR images
HRimages = salinasA_corrected;
%set signal to binary for signal error calculation
len=3;
X=salinasA_gt;
gt = wextend('addrow','sp0',X,len,'d');
len=2;
gt = wextend('2D','sp0',gt,len,'d');
Signal = (gt>0);

figure()
imshow(Signal)
%we will use the
%reorganize each band plane into a feature vector of length k1*k2
%where kHR1 is pixel length in x direction ->
%kHR2 is pixel length in down directrion \/
%bands equals number of spectral band in the image
bands = length(HRimages(1,1,:));

%Normalize Values
for band = 1:bands
HRimages(:,:,band) = mat2gray(HRimages(:,:,band));
end

HRimages_temp = zeros(88,88,bands);
%pad to get even row and collumn size for wavelet fusion
for band=1:bands
len=3;
Y=HRimages(:,:,band);
Y = wextend('addrow','sp0',Y,len,'d');
len=2;
HRimages_temp(:,:,band)=wextend('2D','sp0',Y,len,'dr');
```

```
end
HRimages = HRimages_temp;
kHRcol = length(HRimages(1,:,1));
kHRrow = length(HRimages(:,1,1));

trueSignal = zeros(kHRrow,kHRcol,bands);
for band = 1:bands
trueSignal(:,:,band) = Signal.*HRimages(:,:,band);
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%image degredation noise and downsampling
LRimages = zeros(44,44,bands);
%create the low-resolution images through noise and down-sampling
for band = 1:bands
f1 = HRimages(:,:,band);
h = ones(2,2)/4;
[M,N] = size(f1);
g = filter2(h,f1);
g = g(1:2:M,1:2:N);
SNR = 50;
gg = im2col(g,[M/2,N/2],'distinct');
n_var = var(gg)/10^(SNR/10);
g = imnoise(g,'gaussian',0,n_var);
LRimages(:,:,band) = g;
end

%record Low-res image size
kLRrow = length(LRimages(:,1,1));
kLRcol = length(LRimages(1,:,1));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%use correlation to match up different bands within single
hyperspectral
%image
zm_LRimages = zeros(44,44,bands);
%create zero mean low-res images for cross correlation calculation
for band = 1:bands
zm_LRimages(:,:,band) = LRimages(:,:,band)-mean2(LRimages(:,:,band));
end

%find zero-mean cross-correlation between each band
Rxy = supercorr(zm_LRimages);
%look at cross-correlation between bands
figure()
mesh(Rxy)
%use cross correlation to pair bands into groups of 3
pairings = zeros(3,bands);
%each band will have two pairings, first image of 3 is original band
pairings(1,:) = 1:bands;

for band = 1:bands
  x=Rxy(band,:);
  [val,ind] = sort(x,'descend');
```

```
pairings(2,band)=ind(2);
pairings(3,band)=ind(3);
%[maxes,i]=maxk(x,3);
%pairings(2,band)=i(2);
%pairings(3,band)=i(3);
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%image fusion
%now we will treat the group of three as observations
%%% fused LR images
fusedLR = zeros(kLRrow,kLRcol,bands);
for band=1:bands
%index each a seperate observation of the scene
  ob1 = LRimages(:,:,pairings(1,band));
ob2 = LRimages(:,:,pairings(2,band));
  ob3 = LRimages(:,:,pairings(3,band));
%fuse image 1 and 2 then that result with 3
fused12 = wavelet_fusion(ob1,ob2);
fusedLR(:,:,band) = wavelet_fusion(fused12,ob3);
end

% ob1 = LRimages(:,:,pairings(1,50));
% ob2 = LRimages(:,:,pairings(2,50));
% ob3 = LRimages(:,:,pairings(3,50));
% figure()
% subplot(2,2,1)
% imshow(ob1)
% title('observation 1: band 50')
% subplot(2,2,2)
% imshow(ob1)
% title('observation 2: band 51')
% subplot(2,2,3)
% imshow(ob1)
% title('observation 3: band 49')
% subplot(2,2,4)
% imshow(fusedLR(:,:,50))
% title('fused image')



Factor = 2;

% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%image interpolation
%time storage
time = zeros(1,bands);

% %create baseline interpolations for time calculations
%
%preallocate data space
original_nearest2D = zeros(kHRrow,kHRcol,bands);
original_bilinear2D = zeros(kHRrow,kHRcol,bands);
original_bicubic2D = zeros(kHRrow,kHRcol,bands);
```

```
original_LMMSE2D = zeros(kHRrow,kHRcol,bands);
original_MaxEnt2D = zeros(kHRrow,kHRcol,bands);
original_Reg2D = zeros(kHRrow,kHRcol,bands);

for k=1:bands
tic
%interpolate
original_nearest2D(:,:,k) = imresize(LRimages(:,:,k),Factor,'nearest');
original_bilinear2D(:,:,k) =
imresize(LRimages(:,:,k),Factor,'bilinear');
original_bicubic2D(:,:,k) = imresize(LRimages(:,:,k),Factor,'bicubic');
%USE LR AND INPUT PARAMETERS AS PART OF FUNCTION
original_LMMSE2D(:,:,k) = LMMSE_algorithm_redo(fusedLR(:,:,k),n_var);
original_MaxEnt2D(:,:,k)= max_entropy_algorithm_redo(fusedLR(:,:,k));
original_Reg2D(:,:,k)=
salinas_regularized_algorithm_redo(fusedLR(:,:,k));
k
time(k)=toc;
end


%Show first interpolation results
for j=50:50:200
figure()

subplot(3,3,1)
imshow(LRimages(:,:,j))
title(['Salinas LR Image Band Number = ',num2str(j)]);

subplot(3,3,2)
imshow(HRimages(:,:,j))
title(['Salinas Reference HR Band Number = ',num2str(j)]);

subplot(3,3,3)
imshow(original_nearest2D(:,:,j))
title('Nearest')

subplot(3,3,4)
imshow(original_bilinear2D(:,:,j))
title('Bilinear')

subplot(3,3,5)
imshow(original_bicubic2D(:,:,j))
title('Bicubic ')

subplot(3,3,6)
imshow(original_LMMSE2D(:,:,j))
title('LMMSE')

subplot(3,3,7)
imshow(original_MaxEnt2D(:,:,j))
title('Max Entropy')
```

```
subplot(3,3,8)
imshow(original_Reg2D(:,:,j))
title('Regularized')
pause(.01)
end
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
%Signal to noise calculation
%snr storage for each interpolation
snr = zeros(6,bands);
%cost storage
cost = zeros(6,bands);
%signal
sig_nearest=trueSignal.*original_nearest2D;
sig_bilinear=trueSignal.*original_bilinear2D;
sig_bicubic=trueSignal.*original_bicubic2D;
sig_LMMSE=trueSignal.*original_LMMSE2D;
sig_MaxEnt=trueSignal.*original_MaxEnt2D;
sig_Reg=trueSignal.*original_Reg2D;

%noise
noise_nearest=(~trueSignal).*original_nearest2D;
noise_bilinear=(~trueSignal).*original_bilinear2D;
noise_bicubic=(~trueSignal).*original_bicubic2D;
noise_LMMSE=(~trueSignal).*original_LMMSE2D;
noise_MaxEnt=(~trueSignal).*original_MaxEnt2D;
noise_Reg=(~trueSignal).*original_Reg2D;

%difference between final and original images
  diff_nearest = abs(original_nearest2D-HRimages);
diff_bilinear = abs(original_bilinear2D-HRimages);
diff_bicubic = abs(original_bicubic2D-HRimages);
diff_LMMSE = abs(original_LMMSE2D-HRimages);
  diff_MaxEnt = abs(original_MaxEnt2D-HRimages);
diff_Reg = abs(original_Reg2D-HRimages);

diff(1)=sum(sum(sum(diff_nearest)));
diff(2)=sum(sum(sum(diff_bilinear)));
diff(3)=sum(sum(sum(diff_bicubic)));
diff(4)=sum(sum(sum(diff_LMMSE)));
  diff(5)=sum(sum(sum(diff_MaxEnt)));
  diff(6)=sum(sum(sum(diff_Reg)));

%caclulate the total signal to noise for each band
for k = 1:bands
sig_nearest_tot=sum(sum(sig_nearest(:,:,k)));
sig_bilinear_tot=sum(sum(sig_bilinear(:,:,k)));
sig_bicubic_tot=sum(sum(sig_bicubic(:,:,k)));
sig_LMMSE_tot=sum(sum(sig_LMMSE(:,:,k)));
sig_MaxEnt_tot=sum(sum(sig_MaxEnt(:,:,k)));
sig_Reg_tot=sum(sum(sig_Reg(:,:,k)));

noise_nearest_tot=sum(sum(noise_nearest(:,:,k)));
```

```
noise_bilinear_tot=sum(sum(noise_bilinear(:,:,k)));
noise_bicubic_tot=sum(sum(noise_bicubic(:,:,k)));
noise_LMMSE_tot=sum(sum(noise_LMMSE(:,:,k)));
noise_MaxEnt_tot=sum(sum(noise_MaxEnt(:,:,k)));
noise_Reg_tot=sum(sum(noise_Reg(:,:,k)));

snr(1,k)=20*log(sig_nearest_tot/noise_nearest_tot);
snr(2,k)=20*log(sig_bilinear_tot/noise_bilinear_tot);
snr(3,k)=20*log(sig_bicubic_tot/noise_bicubic_tot);
snr(4,k)=20*log(sig_LMMSE_tot/noise_LMMSE_tot);
snr(5,k)=20*log(sig_MaxEnt_tot/noise_MaxEnt_tot);
snr(6,k)=20*log(sig_Reg_tot/noise_Reg_tot);
end

fprintf('Current Elapsed Time is %f minutes\n',sum(time)/60)

Unlike conducting image enhancement with video data, there is o% cost
function parameters
  a=.01; %time
b=1; %diff importance

t_sum = sum(time);
snr_mean = zeros(1,6);
for j=1:6
snr_mean(j) = mean(snr(j,:));
end

for kk = 1:bands
cost(1,kk)= a*t_sum^2+b*(diff(1)).^2;
cost(2,kk)= a*t_sum^2+b*(diff(2)).^2;
cost(3,kk)= a*t_sum^2+b*(diff(3)).^2;
cost(4,kk)= a*t_sum^2+b*(diff(4)).^2;
cost(5,kk)= a*t_sum^2+b*(diff(5)).^2;
cost(6,kk)= a*t_sum^2+b*(diff(6)).^2;
end

[mincost_nearest,i1] = min(cost(1,:));
[mincost_bilinear,i2] = min(cost(2,:));
[mincost_bicubic,i3] = min(cost(3,:));
[mincost_LMMSE,i4] = min(cost(4,:));
[mincost_MaxEnt,i5] = min(cost(5,:));
[mincost_Reg,i6] = min(cost(6,:));

N = 1:bands;

figure()
semilogy(N,diff(1)*ones(1,bands),'.',N,diff(2)*ones(1,bands),'o',N,diff
(3)*ones(1,bands),'*',N,diff(4)*ones(1,bands),'-
',N,diff(5)*ones(1,bands),'--',N,diff(6)*ones(1,bands),'--')
title('Salinas Error Value for Each Method Uncompressed')
legend('Nearest','Bilinear','Bicubic','LMMSE','MaxEnt','Regularized')
grid on
xlabel('Band')
ylabel('Error')
```

```
figure()
semilogy(N,snr_mean(1)*ones(1,bands),'.',N,snr_mean(2)*ones(1,bands),'o
',N,snr_mean(3)*ones(1,bands),'*',N,snr_mean(4)*ones(1,bands),'-
',N,snr_mean(5)*ones(1,bands),'--',N,snr_mean(6)*ones(1,bands),'--')
title('Salinas Mean SNR accross the HSI bands for Each Method
Uncompressed')
legend('Nearest SNR','Bilinear SNR','Bicubic SNR','LMMSE SNR','MaxEnt
SNR','Regularized SNR')
grid on
xlabel('Band')
ylabel('SNR')

figure()
semilogy(N,cost(1,:),'.',N,cost(2,:),'o',N,cost(3,:),'*',N,cost(4,:),'-
',N,cost(5,:),'--',N,cost(6,:),'--')
title('Cost Function using Mean Time Uncompressed')
legend('Nearest','Bilinear','Bicubic','LMMSE','MaxEnt','Regularized')
grid on
ylabel('Cost')

figure()
plot(N,time)
title('Speed Performance Uncompressed')
y=sum(time)/60;
legend(['Calculation Time total = ',num2str(y),' minutes'])
grid on
xlabel('Band')
ylabel('Time(s)')

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%

% PCA SECTION START
% organize 3D matrix into 2D
% use degraded LR images
d = zeros(kLRrow*kLRcol,bands);
for i=1:bands
  d(:,i) = reshape(LRimages(:,:,i),[1,kLRrow*kLRcol]);
end

%covariance matrix
C = cov(d);

%eigenvectors and eigenvalues
[U,Lamda] = eig(C);

%sort eigenvalues
L = sum(Lamda);
[L,x] = sort(L,'descend');
%sort eigenvectors
for i = 1:length(x)
```

```
 Ut(:,i) = U(:,x(i));
end
U = Ut;

%Plot eigenvalues by decreasing magnitude
figure()
loglog(L)
semilogy(L)
grid on
title('Salinas Eigenvalues')
xlim([0,length(L)])
xlabel('Eigenvalue #')
ylabel('Magnitude')

%Plot the cumulative function of the eigenvalues, show response
L_cum = cumsum(L);
L_cum=horzcat([0],L_cum);
figure()
semilogy(L_cum)
loglog(L_cum)
grid on
title('Salinas Cumulative Sum of Eigenvalues')
xlim([0,length(L_cum)])

NPCA = 1:20;
time_PCA = zeros(1,length(NPCA));
%compression of original data and estimated data using NPCA
eigenvectors of
%covariance matrix
for Npca=NPCA
  U_Proj=U(:,1:Npca);  % Identify projection directions
  %project onto lower dimensional space
  d_Projected = U_Proj'*d';
  d_Projected_2D = reshape(d_Projected',[kLRrow,kLRcol,Npca]);

  %interpolate fused compressed data
  tic
  for k = 1:Npca
      %interpolate
      dnearest2D1(:,:,k) =
imresize(d_Projected_2D(:,:,k),Factor,'nearest');
      dbilinear2D1(:,:,k) =
imresize(d_Projected_2D(:,:,k),Factor,'bilinear');
      dbicubic2D1(:,:,k) =
imresize(d_Projected_2D(:,:,k),Factor,'bicubic');
      %USE LR AND INPUT PARAMETERS AS PART OF FUNCTION
dLMMSE2D1(:,:,k) = LMMSE_algorithm_redo(d_Projected_2D(:,:,k),n_var);
      dMaxEnt2D1(:,:,k)=
max_entropy_algorithm_redo(d_Projected_2D(:,:,k));
      dReg2D1(:,:,k)=
salinas_regularized_algorithm_redo(d_Projected_2D(:,:,k));
  end
  time_PCA(Npca)=toc;
  %reshape into proper dimensions
  for j=1:Npca
```

```
      ddnearest1D(j,:) =
reshape(dnearest2D1(:,:,j),[1,(kLRrow*Factor)^2]);
      ddbilinear1D(j,:)=
reshape(dbilinear2D1(:,:,j),[1,(kLRrow*Factor)^2]);
      ddbicubic1D(j,:) =
reshape(dbicubic2D1(:,:,j),[1,(kLRrow*Factor)^2]);
      ddLMMSE1D(j,:) = reshape(dLMMSE2D1(:,:,j),[1,(kLRrow*Factor)^2]);
      ddMaxEnt1D(j,:) =
reshape(dMaxEnt2D1(:,:,j),[1,(kLRrow*Factor)^2]);
      ddReg1D(j,:) = reshape(dReg2D1(:,:,j),[1,(kLRrow*Factor)^2]);
  end

  %Project into original space
  %put projected images back into original space
  d_pca_nearest1D1 = U_Proj*(ddnearest1D);
  d_pca_bilinear1D1 = U_Proj*(ddbilinear1D);
  d_pca_bicubic1D1 = U_Proj*(ddbicubic1D);
  d_pca_LMMSE1D1 = (U_Proj*(ddLMMSE1D));
  d_pca_MaxEnt1D1 = (U_Proj*(ddMaxEnt1D));
  d_pca_Reg1D1 = (U_Proj*(ddReg1D));
  d_pca_original1D1 = (U_Proj*d_Projected);

  %reshape into 2-D
  for k=1:bands
      ddnearest2D(:,:,k) =
reshape(d_pca_nearest1D1(k,:),[(kLRrow*Factor),(kLRcol*Factor)]);
      ddbilinear2D(:,:,k)=
reshape(d_pca_bilinear1D1(k,:),[(kLRrow*Factor),(kLRcol*Factor)]);
      ddbicubic2D(:,:,k) =
reshape(d_pca_bicubic1D1(k,:),[(kLRrow*Factor),(kLRcol*Factor)]);
      ddLMMSE2D(:,:,k) =
reshape(d_pca_LMMSE1D1(k,:),[(kLRrow*Factor),(kLRcol*Factor)]);
      ddMaxEnt2D(:,:,k) =
reshape(d_pca_MaxEnt1D1(k,:),[(kLRrow*Factor),(kLRcol*Factor)]);
      ddReg2D(:,:,k) =
reshape(d_pca_Reg1D1(k,:),[(kLRrow*Factor),(kLRcol*Factor)]);
  end

%     %Show first interpolation results
%     for j=50:50:200
%         figure()
%
%         subplot(3,3,1)
%         imshow(LRimages(:,:,j))
%         title(['LR Image Band Number = ',num2str(j)]);
%
%         subplot(3,3,2)
%         imshow(HRimages(:,:,j))
%         title(['Reference HR Band Number = ',num2str(j)]);
%
%         subplot(3,3,3)
%         imshow(ddnearest2D(:,:,j))
%         title('Nearest')
%
%         subplot(3,3,4)
%         imshow(ddbilinear2D(:,:,j))
%         title('Bilinear')
%
%         subplot(3,3,5)
%         imshow(ddbicubic2D(:,:,j))
%         title('Bicubic ')
%
%         subplot(3,3,6)
%         imshow(ddLMMSE2D(:,:,j))
%         title('LMMSE')
%
%         subplot(3,3,7)
%         imshow(ddMaxEnt2D(:,:,j))
%         title('Max Entropy')
%
%         subplot(3,3,8)
%         imshow(ddReg2D(:,:,j))
%         title('Regularized')
%         pause(.01)
%     end
  %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%

  %Signal to noise calculation
  %signal
  sig_nearest=trueSignal.*ddnearest2D;
  sig_bilinear=trueSignal.*ddbilinear2D;
  sig_bicubic=trueSignal.*ddbicubic2D;
  sig_LMMSE=trueSignal.*ddLMMSE2D;
  sig_MaxEnt=trueSignal.*ddMaxEnt2D;
  sig_Reg=trueSignal.*ddReg2D;

  %noise
  noise_nearest=(~trueSignal).*ddnearest2D;
  noise_bilinear=(~trueSignal).*ddbilinear2D;
  noise_bicubic=(~trueSignal).*ddbicubic2D;
  noise_LMMSE=(~trueSignal).*ddLMMSE2D;
  noise_MaxEnt=(~trueSignal).*ddMaxEnt2D;
  noise_Reg=(~trueSignal).*ddReg2D;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  %difference between final and original image
  diff_PCA_nearest = abs(ddnearest2D-HRimages);
  diff_PCA_bilinear = abs(ddbilinear2D-HRimages);
  diff_PCA_bicubic = abs(ddbicubic2D-HRimages);
  diff_PCA_LMMSE = abs(ddLMMSE2D-HRimages);
  diff_PCA_MaxEnt = abs(ddMaxEnt2D-HRimages);
  diff_PCA_Reg = abs(ddReg2D-HRimages);

  diff_PCA(1,Npca)=sum(sum(sum(diff_PCA_nearest)));
  diff_PCA(2,Npca)=sum(sum(sum(diff_PCA_bilinear)));
```

```
    diff_PCA(3,Npca)=sum(sum(sum(diff_PCA_bicubic)));
    diff_PCA(4,Npca)=sum(sum(sum(diff_PCA_LMMSE)));
    diff_PCA(5,Npca)=sum(sum(sum(diff_PCA_MaxEnt)));
    diff_PCA(6,Npca)=sum(sum(sum(diff_PCA_Reg)));

    %caclulate the total signal to noise for each band
    for k = 1:bands
        sig_nearest_tot=sum(sum(sig_nearest(:,:,k)));
        sig_bilinear_tot=sum(sum(sig_bilinear(:,:,k)));
        sig_bicubic_tot=sum(sum(sig_bicubic(:,:,k)));
        sig_LMMSE_tot=sum(sum(sig_LMMSE(:,:,k)));
        sig_MaxEnt_tot=sum(sum(sig_MaxEnt(:,:,k)));
        sig_Reg_tot=sum(sum(sig_Reg(:,:,k)));

        noise_nearest_tot=sum(sum(noise_nearest(:,:,k)));
        noise_bilinear_tot=sum(sum(noise_bilinear(:,:,k)));
        noise_bicubic_tot=sum(sum(noise_bicubic(:,:,k)));
        noise_LMMSE_tot=sum(sum(noise_LMMSE(:,:,k)));
        noise_MaxEnt_tot=sum(sum(noise_MaxEnt(:,:,k)));
        noise_Reg_tot=sum(sum(noise_Reg(:,:,k)));

        snr(1,k)=20*log(sig_nearest_tot/noise_nearest_tot);
        snr(2,k)=20*log(sig_bilinear_tot/noise_bilinear_tot);
        snr(3,k)=20*log(sig_bicubic_tot/noise_bicubic_tot);
        snr(4,k)=20*log(sig_LMMSE_tot/noise_LMMSE_tot);
        snr(5,k)=20*log(sig_MaxEnt_tot/noise_MaxEnt_tot);
        snr(6,k)=20*log(sig_Reg_tot/noise_Reg_tot);
    end

    fprintf('Current Elapsed Time is %f minutes\n',sum(time_PCA)/60)

    %cost function parameters
    a=.01; %time
    b=1; %diff importance

    for j=1:6
        snr_mean(j,Npca) = mean(snr(j,:));
    end

    cost_PCA(1,Npca)= a*(time_PCA(Npca))^2+b*(diff_PCA(1,Npca)).^2;
    cost_PCA(2,Npca)= a*(time_PCA(Npca))^2+b*(diff_PCA(2,Npca)).^2;
    cost_PCA(3,Npca)= a*(time_PCA(Npca))^2+b*(diff_PCA(3,Npca)).^2;
    cost_PCA(4,Npca)= a*(time_PCA(Npca))^2+b*(diff_PCA(4,Npca)).^2;
    cost_PCA(5,Npca)= a*(time_PCA(Npca))^2+b*(diff_PCA(5,Npca)).^2;
    cost_PCA(6,Npca)= a*(time_PCA(Npca))^2+b*(diff_PCA(6,Npca)).^2;

    Npca
end


figure()
```

```
plot(NPCA,diff_PCA(1,:),'.',NPCA,diff_PCA(2,:),'o',NPCA,diff_PCA(3,:),'
*',NPCA,diff_PCA(4,:),'-',NPCA,diff_PCA(5,:),'--',NPCA,diff_PCA(6,:),'-
-')
title('Salinas Error Value for Each Level of Compression')
legend('Nearest','Bilinear','Bicubic','LMMSE','MaxEnt','Regularized')
grid on
xlabel('Npca')
ylabel('Error')

figure()
plot(NPCA,snr_mean(1,:),'.',NPCA,snr_mean(2,:),'o',NPCA,snr_mean(3,:),'
*',NPCA,snr_mean(4,:),'-',NPCA,snr_mean(5,:),'--',NPCA,snr_mean(6,:),'-
-')
title('Salinas Mean SNR accross the HSI bands for Each Level of
Compression')
legend('Nearest SNR','Bilinear SNR','Bicubic SNR','LMMSE SNR','MaxEnt
SNR','Regularized SNR')
grid on
xlabel('Npca')
ylabel('SNR')

figure()
plot(NPCA,cost_PCA(1,:),'.',NPCA,cost_PCA(2,:),'o',NPCA,cost_PCA(3,:),'
*',NPCA,cost_PCA(4,:),'-',NPCA,cost_PCA(5,:),'--',NPCA,cost_PCA(6,:),'-
-')
title('Salinas PCA Cost Function')
legend('Nearest','Bilinear','Bicubic','LMMSE','MaxEnt','Regularized')
grid on
xlabel('Npca')
ylabel('Cost')

figure()
plot(NPCA,time_PCA)
title('Salias PCA Speed Performance')
y=sum(time_PCA)/60;
legend(['Calculation Time total = ',num2str(y),' minutes'])
grid on
xlabel('Npca')
ylabel('Time(s)')
```

B. MAXIMUM ENTROPY FUNCTION

The maximum entropy function code is a function that has an input of a low resolution band and the output is a high resolution band. The function code was derived from *Image Super Resolution and Applications* [7].

```
function [band] = max_entropy_algorithm_redo(LR_fused_image_band)
g = LR_fused_image_band;
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
[~,M] = size(g);
N = 2*M;
g = g';
g = im2col(g,[M M],'distinct');
I = speye(N^2);
H1 = sparse(M,N);
counter = 1;
for i = 1:M
  H1(i,counter) = 1;
  H1(i,counter+1) = 1;
  counter = counter+2;
end
H1 = H1/2;
H = kron(H1,H1);
HH = H'*H;
gama = 0.001;
Hopt = inv(HH+gama*I)*H';
f = Hopt*g;
f = col2im(f,[N N],[N N],'distinct');
band=f';
end
```

C. LMMSE FUNCTION

The LMMSE function code is a function that has an input of a low resolution band and the output is a high resolution band. The function code was derived from *Image Super Resolution and Applications* [7].

```
function [band] = LMMSE_algorithm_redo(LR_fused_image_band,n_var)
g = LR_fused_image_band;
f = LR_fused_image_band;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
selectau = 0;
if ~selectau
  key0 = 4;
end
if ~selectau
  a = -1/2;
  s = 0.5;
  [M,N] = size(f);
  ff = zeros(M,N);
  x = f(:,N-1:N);
  x = rot90(x,2);
  y = f(:,1);
  f = [y,f,x];
  for i = 1:M
      for j = 2:N+1
          switch key0
              case 1
                  ff(i,j-1) = f(i,j)*(1-s)+f(i,j+1)*s;
                  %bilinear
              case 2
                  ff(i,j-1) = f(i,j-1)*(a*s^3-
2*a*s^2+a*s)+f(i,j)*((a+2)*s^3 -(3+a)*s^2+1)+f(i,j+1)*(-
(a+2)*s^3+(2*a+3)*s^2-a*s)+f(i,j+2) *(-a*s^3+a*s^2);
                  % Bicubic
              case 3
                  ff(i,j-1) = f(i,j-1)*((3+s)^3-
4*(2+s)^3+6*(1+s)^3-4*s^3)/6+f(i,j)*((2+s)^3-
4*(1+s)^3+6*s^3)/6+f(i,j+1)*((1+s)^3-4*s^3)/6+f(i,j+2)*s^3/6;
                  % Cubic?Spline
              case 4
                  ff(i,j-1) = f(i,j-1)*((-1/6)*(1+s)^3+(1+s)^2+(-
85/42)*(1+s)+(29/21))+f(i,j)*(0.5*s^3-
s^2+(1/14)*s+13/21)+f(i,j+1)*(0.5*(1-s)^3-(1-
s)^2+(1/14)*s+13/21)+f(i,j+2)*((-1/6)*(2-s)^3+(2-s)^2-(85/42)*(2-
s)+29/21);
                  % Cubic o? Moms
          end
      end
  end
  ff = ff(:,1:N);
  fff(1:M,1:2:2*N) = f(1:M,2:N+1);
  fff(1:M,2:2:2*N) = ff(1:M,1:N);
  f = fff';
  clear ff fff
  a = -1/2;
  s = 0.5;
  [M,N] = size(f);
  ff = zeros(M,N);
  x = f(:,N-1:N);
  x = rot90(x,2);
  y = f(:,1);
  f = [y,f,x];
  for i = 1:M
      for j = 2:N+1
          switch key0
              case 1
                  ff(i,j-1) = f(i,j)*(1-s)+f(i,j+1)*s;
                  %bilinear
              case 2
                  ff(i,j-1) = f(i,j-1)*(a*s^3-
2*a*s^2+a*s)+f(i,j)*((a+2)*s^3-(3+a)*s^2+1)+f(i,j+1)*(-
(a+2)*s^3+(2*a+3)*s^2-a*s)+f(i,j+2)*(-a*s^3+a*s^2);
                  % Bicubic
```

```
                case 3
                    ff(i,j-1) = f(i,j-1)*((3+s)^3-
4*(2+s)^3+6*(1+s)^3-4*s^3)/6+f(i,j)*((2+s)^3-
4*(1+s)^3+6*s^3)/6+f(i,j+1)*((1+s)^3-4*s^3)/6+f (i,j+2)*s^3/6;
                    % Cubic?Spline
                case 4
                    ff(i,j-1) = f(i,j-1)*((-1/6)*(1+s)^3+(1+s)^2+(-
85/42)*(1+s)+(29/21))+f(i,j)*(0.5*s^3-
s^2+(1/14)*s+13/21)+f(i,j+1)*(0.5*(1-s)^3-(1-
s)^2+(1/14)*s+13/21)+f(i,j+2)*((-1/6)*(2-s)^3+(2-s)^2-(85/42)*(2-
s)+29/21);
                    % Cubic o? Moms
            end
        end
    end
    ff = ff(:,1:N);
    fff(1:M,1:2:2*N) = f(1:M,2:N+1);
    fff(1:M,2:2:2*N) = ff(1:M,1:N);
    fff = fff';
    fff = (fff>=0).*fff;
else
    fff = f1;
end
wlength = 3;
[~,M1] = size(fff);
fff(M1+wlength,M1+wlength) = 0;
for j = 0:M1-1
    for k = 0:M1-1
        sum = 0;
        for n = 1:wlength
            for m = 1:wlength
                sum = sum+fff(n,m)*fff(n+j,m+k);
            end
        end
        RRR(j+1,k+1) = 1/((wlength)^2)*sum;
    end

end

[~,M] = size(g);
N = 2*M;
R = 2;
kff = zeros(N,N);
kff = RRR';
kff = im2col(kff,[N N],'distinct');
kff = sparse(1:N^2,1:N^2,kff);
g = g';
g = im2col(g,[M M],'distinct');
I = speye(M^2)/12;
H1 = sparse(M,N);
```

```
counter = 1;
for i = 1:M
  H1(i,counter) = 1;
  H1(i,counter+1) = 1;
  counter = counter+2;
end
H1 = H1/2;
H = kron(H1,H1);
Hz1 = speye(M^2)*n_var;
HH = H*kff*H';
I = speye(size(HH));
Hopt = kff*H'*inv(HH+Hz1);
f = Hopt*g;
f = col2im(f,[N N],[N N],'distinct');
f = (f>=0).*f;
f = f';
band=f;
end
```

D. REGULARIZED FUNCTION

The Regularized function code is a function that has an input of a low resolution band and the output is a high resolution band. The function code was derived from *Image Super Resolution and Applications* [7].

```
function [band] = regularized_algorithm_redo(LR_fused_image_band)
g = LR_fused_image_band;
[M,N] = size(LR_fused_image_band);
M = 2*M;
N = 2*N;
lamda = .001;
g = [rot90(g(:,1:4),2),g,rot90(g(:,M/2-8:M/2),2)];
g = [rot90(g(1:4,:),2);g;rot90(g(M/2-8:M/2,:),2)];
[L1,L2] = size(LR_fused_image_band);
L1 = 2*L1;
L2 = 2*L2;
M = 24;
N = 12;
I = speye(M^2);
H1 = sparse(M/2,M);
counter = 1;
for i = 1:M/2
   H1(i,counter) = 1;
   H1(i,counter+1) = 1;
   counter = counter+2;
end;
H1 = H1/2;
H = kron(H1,H1);
```

```
HH = H'*H;
beta = 0.125;
Q1 = sparse(M,M);
for i = 1:M
   Q1(i,i) = -2;
end
for i = 1:M-1
   Q1(i,i+1) = 1;
end
for i = 2:M
   Q1(i,i-1) = 1;
end
Q = kron(Q1,Q1);
QQ = Q'*Q;
L = inv(HH+lamda*QQ);
for ii = 1:L1/8
  for jj = 1:L2/8
      ii;
      f = g(4*ii+1-4:4*(ii+1)+4,4*jj+1-4:4*(jj+1)+4);
      z = f;
   y = im2col(z,[N N],'distinct');
      x1 = L*H'*y;
      x1 = col2im(x1,[M M],[M M],'distinct');
      xx1(8*(ii-1)+1:8*(ii),8*(jj-1)+1:8*(jj)) = x1(9:16,9:16);
end
end
%[a,b] = size(xx1);
xx1 = max(xx1,0);
xx1 = min(xx1,1);
len=4;
xx1=wextend('2D','sp0',xx1,len,'dr');
%%%%%%%%%%%%%%%%%%%%%%%%%
band=xx1;
end
```

E. DWFT FUNCTION

The DWFT function code is a function that has an input of two low-resolution images and the output is a fused low-resolution band. The function code was derived from *Image Super Resolution and Applications* [7].

```
function fused = wavelet_fusion(g1,g2)
%DWFT Fusion
%THE WAVELET FRAME TRANSFORM STEP FOR BOTH IMAGES.
[a1,h1,v1,d1] = swt2(g1,1,'db2');
%[a1,h1,v1,d1] = swt2(g1,1,'db2');
[a2,h2,v2,d2] = swt2(g2,1,'db2');
%SELECTION OF COEFFICIENT USING Linear combination BETWEEN THE
APPROXIAMTION
```

```
A = imlincomb(0.5,a1,0.5,a2);
%INVERSE FRAME WAVELET TRANSFORM TO GENERATE THE FUSED IMAGE.
Z = iswt2(A,h1,v1,d1,'db2');
fused = Z;
%imshow(fused)
end
```

F. CORRELATION FUNCTION

```
function corr_btw_bands = supercorr(HR_image)
[row,col,bands] = size(HR_image);
for i=1:bands;
  for j = 1:bands;
  Rxx(i,j)=corr2(HR_image(:,:,i),HR_image(:,:,j));
  end
end
corr_btw_bands=Rxx;
```

LIST OF REFERENCES

[1] M. T. Eismann, *Hyperspectral Remote Sensing*. Bellingham, WA, USA: SPIE Press, 2012.

[2] R. H. Graham., *The Complete Illustrated History of the Blackbird, the World's Highest, Fastest Plane.* NY, New York, USA*:* Zenith Press, 2017.

[3] L. Giannoni, F. Lange, and I. Tachtsidis, "Hyperspectral imaging solutions for brain tissue metabolic and hemodynamic monitoring: Past, current and future developments," *Journal of Optics,* vol. 20. [Online], Available: doi: 044009. 10.1088/2040-8986/aab3a6

[4] S. Lundeen, "Airborne visible/infrared imaging spectrometer: AVIRIS Overview." January 29, 2019. [Online]. Available: https://aviris.jpl.nasa.gov/aviris/index.html

[5] M. F. Baumgardner, L. L. Biehl, and D. A. Landgrebe, "220 Band AVIRIS Hyperspectral Image Data Set: June 12, 1992, Indian Pine Test Site 3." Purdue University Research Repository. [Online]. Available: doi:10.4231/R7RX991C

[6] R.F. Kokaly, R.N. Clark, G.A. Swayze, K.E. Livo, T.M. Hoefen, N.C. Pearson, R. A. Wise, W. M. Benzel, H. A. Lowers, R. L. Driscoll, and A. J. Klein, 2017: "USGS Spectral Library Version 7," U.S. Geological Survey [Online]. Available: https://doi.org/10.3133/ds1035

[7] F. E. Samie, M. M. Hadhoud, and S. E. El-Khamy, *Image Super Resolution and Applications*. Boca Raton, FL, USA: Taylor & Francis, 2013.

[8] H. Zhang, L. Zhang, and H. Shen, "A super resolution reconstruction algorithm for hyperspectral images," *Signal Processing*, vol. 92, pp. 2082–2096, 201. 6

[9] X. Sun, L. Xu, L. Yang, Y. Chen, Y. Fang and J. Peng, "Super-resolution reconstruction of hyperspectral imagery using a spectral unmixing based representational model," *2016 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, pp. 1607–1610, 2016 [Online]. Available: doi: 10.1109/IGARSS.2016.7729410

[10] S. E. El-Khamy, M. M. Hadhoud, M. I. Dessouky et al., "A wavelet-based entropic approach to high resolution image reconstruction," *Int. J. Machine Graphics Vision,* vol. 17, pp. 235–256, 2008.

[11] M. Fargues, “Random processes” Applications to signal & information processing.” NPS Class Lecture, EC3410: Discrete-time Random Signals, Monterey, CA, October 2018.

[12] S. E. El-Khamy, M. M. Hadhoud, M. I. Dessouky et al., “Wavelet fusion: A tool to break the limits on LMMSE Image super resolution, *Int. J. Wavelets Multiresolution Information Proc.*, vol. 4, pp. 105–118, 2006.

[13] S. E. El-Khamy, M. M. Hadhoud, M. I. Dessouky et al. 2005. “Regularized super resolution reconstruction of images using wavelet fusion,” *J. Optical Eng*., vol. 44.

[14] J. Richards, and X. Jia, *Remote Sensing Digital Image Analysis: An Introduction*, 4th Edition, Berlin: Springer, 2006.

[15] A. Villa, J. Chanussot, J. A. Benediktsson, M. Ulfarsson, and C. Jutten. “Super resolution: An efficient method to improve spatial resolution of hyperspectral images,” *IEEE IGARS*, 2010, pp. 2003–2006.

[16] D. Rajan, S. Chaudhuri, and M. V. Joshi. “Multi-Objective super resolution: Concepts and examples,” *IEEE Signal Processing Magazin*e, vol. 3, pp. 49–61, 2003.

[17] P. Milanfar *Super Resolution Imaging*. Boca Raton, LA, USA: CRC Press, 2011.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California