Hyperspectral Texture Synthesis Algorithms

by

Nestor Jose Diaz Gonzalez

A thesis submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE
in
COMPUTER ENGINEERING

UNIVERSITY OF PUERTO RICO
MAYAGUEZ CAMPUS
2010

Approved by:

_____	_____
Miguel Velez-Reyes, PhD Member, Graduate Committee	Date
_____	_____
Domingo Rodriguez, PhD Member, Graduate Committee	Date
_____	_____
Vidya Manian, PhD President, Graduate Committee	Date
_____	_____
Paul Castillo, PhD Representative of Graduate Studies	Date
_____	_____
Hamed Parsiani, PhD Chairperson of the Department	Date

ABSTRACT

Hyperspectral images have textured regions and in many cases there are not sufficient samples to train classifiers. By simulating more samples that are self-similar to the original texture efficient classifiers can be trained and used for classification. This kind of tools provides a means to get synthetic data without the expense of data collection. The first step toward this goal is to develop a hyperspectral texture synthesis algorithm that efficiently combines both the spectral information and the spatial variability in the original image. Three algorithms are implemented to fulfill that purpose. To do the synthesis, the first algorithm uses neighboring pixel information, and this is done by a neighborhood search of a multiresolution pyramid constructed from the original sample that encodes the spectrum and spatial intensity gradients. Gaussian decomposition is used for the multiresolution decomposition. The second approach is as simple as putting together blocks with a certain overlap, so that each new block is chosen so that it "agrees" with its neighbors in the region of overlap. A third implementation uses 3D wavelet transform of the original sample. This encodes the spectrum and spatial intensity gradients. The wavelet coefficients are a representation of the hyperspectral image as a compact code. These coefficients are then synthesized by an image quilting algorithm. Results of the synthesis are presented using hyperspectral images from three different remote sensing scenarios. Discriminant analysis and support vector classifier are used to classify the synthesized textures. It was verified that there is a substantial agreement between the input textures and the synthetic ones.

RESUMEN

Las imágenes hiperespectrales suelen tener regiones con texturas. Muchas veces esas regiones no son lo suficientemente grandes para entrenar adecuadamente los algoritmos de clasificación de texturas. Simulando muestras que mantengan características similares a las de la textura original permitiría superar esta limitante. Este tipo de herramientas nos permite obtener datos sintéticos sin incurrir en los costos asociados con la adquisición de datos. Para alcanzar este objetivo es necesario desarrollar un algoritmo de síntesis de textura que combine eficientemente la información espectral y la variabilidad espacial de la imagen original. En este trabajo se muestran tres algoritmos implementados para cumplir con ese propósito. Para hacer la síntesis, el primer algoritmo utiliza información de píxeles vecinos y esto se realiza mediante una búsqueda de la vecindad en una pirámide con varias resoluciones, construida a partir de la muestra original que codifica el espectro y el gradiente de la intensidad espacial. El segundo enfoque es tan simple como unir bloques tomados de la imagen de entrada con una región de superposición. Cada nuevo bloque es elegido de tal manera que la región de superposición encaje con sus vecinos. El tercer enfoque utiliza la transformada wavelet en 3D de la imagen original, y así se construye una representación de la imagen de hiperespectral más compacta. Los resultados de la síntesis se presentan mediante imágenes hiperespectrales desde diferentes escenarios de percepción remota. Clasificador de vectores de soporte y análisis discriminante se utilizan para clasificar las propiedades de las texturas sintetizadas. Se comprobó que hay un acuerdo sustancial entre las texturas de entrada y las sintéticas.

ACKNOWLEDGEMENTS

I want to give thanks to my Lord Jesus Christ for his fidelity at every moment. Thanks to my family, especially to my parents, Nestor Sr. and Lourdes for their unconditional love and support.

Thanks to Dr. Vidya Manian for guidance during these almost two years of research work. I really appreciated the good technical advices provided by Dr. Miguel Velez-Reyes and Dr. Domingo Rodriguez. Thanks to all the LARSIP staff.

During my time in Puerto Rico, there are many people that have made my journey here a more pleasant one. Among them are: Manuel Galvan and Andrea Peralta for adopting me as if I were part of the family; Sandra Montalvo for her good advises and care; Ramon Vazquez for his support. Thanks to Iglesia Bautista Apocalipsis in Santiago, D. R., for their prayers and support, especially to the Jose and Dulce Nochea. Thanks for sharing love to the brothers and sisters at the Iglesia Bautista de Mayaguez, especially to Egui and Marivanesa.

Felipe Hernandez and Manuel Lopez thanks for your support and availability. We've shared great moments. Thanks to my colleagues of DSP David Marquez, Gonzalo Vaca, Andres Alarcon and Blas Trigueros, I always enjoyed our discussions.

Thanks to my fellow countrymen Levis Cabrera, Juan Garcia and Andres Pujols for welcoming and helping me to settle down in Mayaguez, as well as everyone that in one way or another made possible this work. May God bless you all!

Table of Contents

Table List

Tables **Page**

Figure List

Figures **Page**

1 INTRODUCTION

Many texture surfaces can be synthesized by simulating their physical generation processes. Biological patterns such as fur, scales, and skin can be modeled using reaction diffusion [1] and cellular texturing [2]. Weathering effects in stones can be reproduced by detailed simulations [3]. These techniques can produce textures directly on 3D meshes so the texture mapping distortion problem is avoided. However, different textures are usually generated by very different physical processes so these approaches are applicable to only limited classes of textures.

Texture synthesis techniques that generate an output texture from an example input can be roughly categorized into three classes. The first class uses a fixed number of parameters within a compact parametric model to describe a variety of textures. The second class of texture synthesis methods is non-parametric, which means that rather than having a fixed number of parameters, they use a collection of exemplars to model the texture. The third, most recent class of techniques generates textures by copying whole patches from the input. Due to the nature of hyperspectral images sometimes it is hard to have a big enough texture image and be able to synthesize from it using patches.

In real world, ‘texture’ is ubiquitous. Every perceivable object or scene has a texture associated with it and it is very easy for humans to perceive what texture is and the human mind is able to perceive and differentiate between two different types of textures with little difficulty. But in the field of machine learning and computer vision, making a computer learn and

understand what is the texture of a particular segment of an image, is a challenging task. The definition of texture is difficult because of its abstract nature and the fact that practically every object seen exhibits some texture. In our case we will define a texture as some visual pattern on an infinite 2-D plane which, at some scale, has a stationary distribution [4]. Since a given texture sample could have been drawn from an infinite number of different textures, our assumption of stationarity conditions helps us deal with this. The usual assumption is that the sample is large enough that it somehow captures the stationarity of the texture and that the approximate scale of the texture elements is known. It could also be defined as the structural pattern of surfaces which is homogeneous in spite of fluctuations in brightness and color. In our case we want to synthesize hyperspectral texture images, which mean that each of the pixels contained in the image will be a vector with the spectral information of the material that has been sensed, as in Figure 1. A region in a hyperspectral image has a constant texture if a set of local statistics or other local properties of the spatial and spectral variability of the image are constant, slowly varying or approximately periodic.

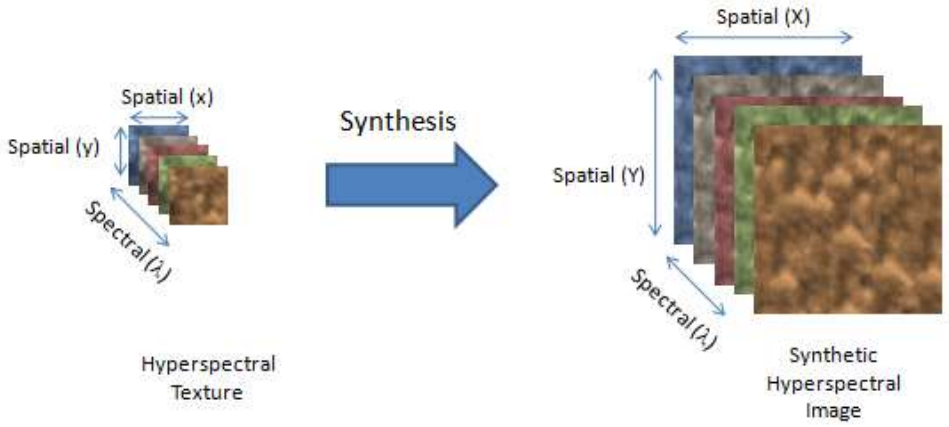


Figure 1: Goal of Hyperspectral Texture synthesis.

Given a finite sample from some texture, an image, the goal is to synthesize other samples from the same texture, as shown in Figure 1. It is important to notice that most of the applications

of texture synthesis are for visual improvement of digital images, movies and video games. In our case it is essential to preserve the spectral signature of the input image in order to have a physical meaning of the sensed object.

The synthesis can be done in a raster scan approach by a deterministic sampling procedure as in [5]. Other texture synthesis methods are Gaussian pyramids, steerable pyramids [6] and wavelet pyramids [7]. The Gaussian pyramid [6] is constructed by successively filtering and down sampling operations. The steerable pyramid is a multiscale, multiresolution linear signal decomposition constructed from k oriented filters. For the wavelet pyramid algorithm uses orthonormal filters.

S. Sarkar *et al* [8] presented a hyperspectral texture synthesis algorithm and did the validation of their synthesis analyzing spectral angle deviation from the mean curves that describe spectral properties between the original images and the synthetic ones and demonstrated that a signature-based detection algorithm has similar performance against real and synthesized hyperspectral backgrounds.

1.1 Motivation

This work will provide a tool to generate hyperspectral images, which can be used to test image exploiting algorithms. Generating synthetic images will avoid the high costs associated with the collection of new real hyperspectral data. It can also be used to synthesize target and background regions in images to validate target detection algorithms, providing them with images that would be very hard to acquire in the real world.

We also know that hyperspectral images have textured regions and in many cases there are not sufficient samples to train classifiers. By simulating more samples that are self-similar to the original texture efficient classifiers can be trained and used for classification.

Hyperspectral images have to be restored in many cases due to occluded objects, lost image parts or faulty image acquisition. So another potential application for a texture synthesis algorithm is that it can be used as a complementary part of a hyperspectral image inpainting algorithm.

We would like to have an algorithm that could synthesize an output texture automatically. The resulting texture should not only look like the input hyperspectral texture image, but also share similar characteristics of its spectrum. The user should be able to specify the size of the resulting texture image.

1.2 Objectives

The main objectives of this work are as follows:

- Implement hyperspectral texture synthesis algorithms performing corresponding modifications to the already existing texture synthesis algorithms for gray scale and color images.
- Propose a validation framework using texture statistics; discriminant analysis and support vector classification techniques for synthetic textures, to verify the correspondence between synthetic images and the real ones.

1.3 Contributions of this work

The main contribution of this work can be viewed as the following:

- A modified set of texture synthesis algorithms that can be used for hyperspectral, vector or multichannel images. These algorithms provide hyperspectral texture results that maintain very good spatial correlation.
- A validation method for the proposed algorithms that is composed of a feature extraction, classifier training and classification.
- An evaluation of the three algorithms with different hyperspectral scenarios.

1.4 Summary of Following Chapters

We first develop the necessary background theory in Chapter 2. Chapter 3 explains the implemented algorithms and the validation methodology. Chapter 4 shows the results of the synthesis as well as the confusion matrices of the classification of the synthetic images. Conclusions are presented in Chapter 5.

2 THEORETICAL BACKGROUND

2.1 Hyperspectral Images

In recent years the capability of generating hyperspectral images with several spectral bands has increased. In addition hyperspectral imagers offer high spectral resolution that allows recovering important characteristics of distinct objects placed on the scene of interest. The basic principle is that objects reflect, absorb, and emit electromagnetic radiation in ways characteristic of their molecular composition and shape. The spatially and spectrally sampled information is typically visualized as a cube, whose face is a function of the spatial coordinates and whose depth is a function of spectral band (See Fig. 2).

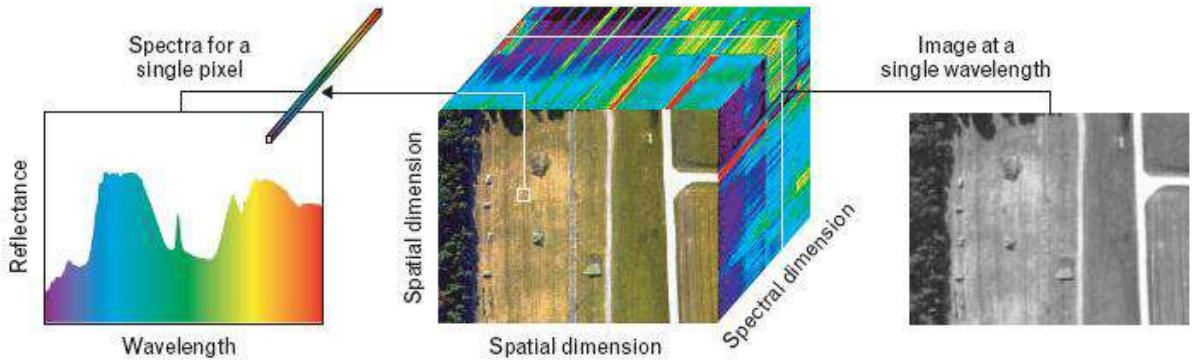


Figure 2: Characteristics of Hyperspectral Image. Image Courtesy of [9].

In the wavelength dimension, each image pixel is a vector that provides a spectrum characterizing the materials within the pixel. Conversely, the data in each band corresponds to a narrow band image of the surface covered by the field of view of the sensor. This has led to numerous applications, especially in the recognition of materials spread out over the Earth's surface and target detection.

2.2 Image Classification

Remote sensing image classification is one of the important approaches of information extraction. The traditional classification methods using spectral information alone are difficult to obtain satisfactory results, due to the spatial correlation among pixels and the similar spectral feature of different objects in remote sensing images. One of the effective solutions to the problem is to add spatial information to image classification, that is, to combine spatial and spectral information together in the classification process [10].

Image texture is a kind of spatial information that has been widely used in image classification and has shown higher classification accuracy [11, 12]. Image classification is the process of assigning all pixels in a digital image to particular classes according to their characteristics [13]. As a result we obtained a thematic map in which each pixel belongs to a particular class. Two main classification schemes are the Unsupervised and Supervised Classification. In our case we have a Supervised Classification scheme, since we already know to which class the testing samples belong.

Distance is a numerical description of how far apart objects are. In mathematics, a distance function or metric is a generalization of the concept of physical distance. A metric is a function that behaves according to a specific set of rules, and provides a concrete way of describing what it means for elements of some space to be "close to" or "far away from" each

other. A metric space is a set where a notion of distance between elements of the set is defined. [14]

Overall accuracy, *O*, is calculated as shown in equation 1, where *A* is the number of pixels assigned to the correct class and *B* is the number of pixels that actually belong to that class. It is a good measure of the accuracy of a classification scheme as it is not biased towards the smaller classes.

$$O = \frac{\sum A}{\sum B} \tag{1}$$

Correctly assigned pixels may have been assigned by chance and not based on the classification decision rule. The kappa value κ indicates how accurate the classification output is after this chance, or random, portion has been accounted for. There are a number of ways to show how the kappa value is calculated. Equation 2 defines the kappa value, where r is the number of rows in the confusion matrix, x_{ii} is the number of observations in row i and column i (on the major diagonal), x_{i+} is the total observations in row i, x_{+1} is the total of observations in column i, and N is the total number of observations included in the matrix.

$$\kappa = \frac{N\sum_{i=1}^{r} x_{ii} - \sum_{i=1}^{r}(x_{i+} \times x_{+i})}{N^2 - \sum_{i=1}^{r}(x_{i+} \times x_{+i})} \tag{2}$$

2.2.1 Discriminant analysis

Discriminant function analysis is used to determine which variables discriminate between two or more naturally occurring groups. The discriminant analysis situation is

characterized by the following: one has two types of multivariate observations – the first, called training samples, are those whose group identity (i.e., membership in a specific one of say G given groups is known a priori), and the second type, referred to as test samples, consists of observations for which such a priori information is not available and which have to be assigned to one of the G groups. In this work, the input image is used as the training samples and the synthesized image as the test samples.

A discriminant function that is a linear combination of the components of **x** can be written as

$$g(\boldsymbol{x}) = \boldsymbol{w}^t\boldsymbol{x} + w_0 \quad , \tag{3}$$

where $\boldsymbol{w}$ in the *weight vector* and w_0 the *bias* or *threshold weight*.

For a discriminant function of the form of Eq. 3, a two-category classifier implements the following decision rule: Decide ω_1 if $g(\boldsymbol{x}) > 0$ and ω_2 if $g(\boldsymbol{x}) < 0$. Thus, x is assigned to ω_1 if the inner product $\boldsymbol{w}^t\boldsymbol{x}$ exceeds the threshold $-w_0$ and to ω_2 otherwise.

The equation $g(\boldsymbol{x}) = 0$ defines the decision surface that separates points assigned to ω_1 from points assigned to ω_1 from points assigned to ω_2. When $g(\boldsymbol{x})$ is linear, this decision surface is a *hyperplane*. If $\mathbf{x}_1$ and $\mathbf{x}_2$ are both on the decision surface, then

$$\boldsymbol{w}^t\boldsymbol{x}_1 + w_0 = \boldsymbol{w}^t\boldsymbol{x}_2 + w_0 \tag{4}$$

or

$$\boldsymbol{w}^t(\boldsymbol{x}_1 - \boldsymbol{x}_2) = 0, \tag{5}$$

and this shows that w is normal to any vector lying in the hyperplane. The distance of a point from the hyperplane is given by

$$r - \frac{g(\boldsymbol{x})}{\|\boldsymbol{w}\|} \tag{6}$$

The linear discriminant function $g(\boldsymbol{x})$ can be written as

$$g(\boldsymbol{x}) = w_0 + \sum_{i-1}^{d} w_i x_i \tag{7}$$

where the coefficients w_i are the components of the weight vector $\boldsymbol{w}$. By adding additional terms involving the products of pairs of components of $\boldsymbol{x}$, we obtain the *quadratic discriminant function*

$$g(\boldsymbol{x}) = w_0 + \sum_{i-1}^{d} w_i x_i + \sum_{i-1}^{d}\sum_{j-1}^{d} w_{ij} x_i x_j \tag{8}$$

Because $x_i x_j = x_j x_i$, we can assume that $w_{ij} = w_{ji}$ with no loss of generality. Thus, the quadratic discriminant function has an additional $d(d+1)/2$ coefficients at its disposal with which to produce more complicated separating surfaces. The separating surface defined by $g(\mathbf{x}) = 0$ is a second-degree or *hyperquadric* surface. If the symmetric matrix $\mathbf{W} = [w_{ij}]$ is nonsingular, the linear terms in $g(\mathbf{x})$ can be eliminated by translating the axes. The basic character of the separating surface can be described in terms of the scaled matrix $\bar{\mathbf{W}} = \mathbf{W}/(\mathbf{w}^t\mathbf{W}^{-1}\mathbf{w} - 4w_0)$. If $\bar{\mathbf{W}}$ is a positive multiple of the identity matrix, the separating surface is a *hypersphere*. If $\bar{\mathbf{W}}$ is positive definite, the separating surfaces is a *hyperellipsoid*. If some

of the eigenvalues of $\bar{\mathbf{W}}$ are positive and others are negative, the surface is one of the varieties of types of *hyperhyperboloids*. These are the kinds of separating surfaces that arise in the general multivariate Gaussian case. For a more detailed discussion of linear discriminant functions, see [15].

2.2.2 Support Vector Machines

The original idea of Support Vector Machines (SVM) is to use a linear separating hyperplane to separate the training data into two classes [16]. Suppose the training data

$$(\mathbf{x}_1, y_1),\cdots,(\mathbf{x}_l, y_l), \qquad \mathbf{x}_i \in \boldsymbol{R}^n, \qquad y_i \in \{-1,+1\}$$

can be separated by a hyperplane $\boldsymbol{\omega} \bullet \mathbf{x} + b = 0$. We say that this set of vectors is separated by the optimal hyperplane if it is separated without error and the distance between the closest vector and the hyperplane is maximal. To describe the separating hyperplane let us use the following form:

$$\begin{cases} \boldsymbol{\omega} \bullet \mathbf{x}_i + b \geq +1 & \text{if} \quad y_i = +1 \\ \boldsymbol{\omega} \bullet \mathbf{x}_i + b \leq -1 & \text{if} \quad y_i = -1 \end{cases}$$

or equivalently

$$y_i (\boldsymbol{\omega} \bullet \mathbf{x}_i + b) \geq 1, \qquad i = 1,\ldots,l \,. \tag{9}$$

It is easy to check that the optimal hyperplane is the one that satisfies the conditions (9) and minimizes the following function: ½ $\|\boldsymbol{\omega}\|^2$. Vapnik [29] has shown we may perform this minimization by maximizing the following function with respect to the variables α_i :

$$W(\boldsymbol{\alpha}) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i \bullet \mathbf{x}_j)$$

Subject to $0 \le \alpha_i$, $i = 1,...,l$ and $\sum_{i=1}^{l} \alpha_i y_i = 0$. Those $\mathbf{x}_i$s with $0 \le \alpha_i$ are termed Support Vectors (SVs). They are usually a small subset of the training data set, denoted by X_{SVM}. For an unknown vector $\mathbf{x}_i$ classification then corresponds to finding

$$f(\mathbf{x}) = sign\left(\sum_{\mathbf{x}_i \in X_{SVM}} \alpha_i \, y_i (\mathbf{x} \bullet \mathbf{x}_i) + b \right)$$

Where

$$\boldsymbol{\omega} = \sum_{\mathbf{x}_i \in X_{SVM}} \alpha_i y_i \mathbf{x}_i$$

and the sum is over those nonzero SVs with $0 < \alpha i$.

To construct the optimal hyperplane in the case when the data are linearly nonseparable, SVM uses two methods to handle this difficulty: First, it allows training errors. Second, it non-linearly transforms the original input space into a higher dimensional feature space by a function $\varphi(x)$. In this higher space, it is more possible that the data can be linearly separated. Then the problem can be described as:

$$\min \frac{1}{2} \|\boldsymbol{\omega}\|^2 + C \sum_{i=1}^{l} \xi_i$$

subject to

$$y_i(\boldsymbol{\omega} \bullet \phi(\mathbf{x}_i) + b) \ge 1 - \xi_i, \qquad \xi_i \ge 0, \quad i = 1,...,l, \qquad C > 0. \tag{10}$$

A penalty term $C \sum_{i=1}^{l} \xi_i$ in the objective function and training errors are allowed. If the penalty parameter C is large enough and the data is linear separable, the problem (10) goes back to (9)

as all ξ_i will be zero. We can equivalently maximize $W(\boldsymbol{\alpha})$ but the constraint is now $0 \le \alpha_i \le C$ instead of $0 \le \alpha_i$:

$$W(\boldsymbol{\alpha}) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j \, (\phi(\mathbf{x}_i) \bullet \phi(\mathbf{x}_j))$$

Subject to $0 \le \alpha_i \le C$, $i = 1, \ldots, l$ and $\sum_{i=1}^{l} \alpha_i y_i = 0$. The inner products $\phi(\mathbf{x}_i) \bullet \phi(\mathbf{x}_j)$ in the high-dimensional space can be replaced by some special kernel functions $K(\boldsymbol{x}_i, \boldsymbol{x}_j)$ that can be easily calculated. By the use of kernels, all necessary computations can be performed directly in input space. Some popular kernels are radial basis function kernel $K(x, x_i) = \exp(-\gamma \|x - x_i\|^2)$ and polynomial kernel $K(x, x_i) = (1 + x \bullet x_i)^d$, where γ and d are parameters. Using different expressions for kernel functions $K(\boldsymbol{x}_i, \boldsymbol{x}_j)$, one can construct different learning machines with arbitrary types of decision surfaces. Hyperspectral image classification methods such as support vector machine that are most commonly used discussed in [17-19].

2.3 Texture Metrics

The SAD between two vectors (r_i, r_j) is defined by the following equation,

$$SAD(r_i, r_j) = \cos^{-1}\left(\frac{r_i \cdot r_j}{|r_i| * |r_j|}\right) = \cos^{-1}\left(\frac{\sum_{t=1}^{N} r_{it} * r_{jt}}{\sqrt{\sum_{t=1}^{N} r_{it}} \sqrt{\sum_{t=1}^{N} r_{jt}}}\right)$$

where N represents the size of the vector.

Edge-based texture descriptors and regional texture descriptors were implemented. Hence, we implemented 6 edge based metrics. The SAD is used in the feature computation. These features are the average (f1), the standard deviation (f2), the average deviation of gradient magnitude (f3), the average residual energy (f4), the average deviation of the horizontal directional residual (f5) and, the average deviation of the vertical directional residual (f6). 12 other texture descriptors have been implemented. The average (f1), the standard deviation (f2), the average deviation of gradient magnitude (f3), and the average residual energy (f4) are given by,

$$f_1(b) = \frac{1}{n_1 n_2} \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} X_b(i, j)$$

$$f_1 = \frac{1}{B} \sum_{b=1}^{B} f_1(b)$$

$$f_2 = \sqrt{\sum_{i=1}^{n_1} \sum_{j=1}^{n_2} (X_b(i, j), f_1(b))^2_{SAD}}$$

where $\{X_b(i, j), i, j \in I, 1 \le i \le n_1, 1 \le \mathbf{j} \le n_2, 1 \le b \le B\}$, represent a hyperspectral image, n_1 and n_2 and B are the number of rows, number of columns and number of bands, respectively.

$$f_3 = \frac{1}{n_1 n_2} \sum_{i=1}^{n_1 - 1} \sum_{j=1}^{n_2 - 1} [X(i, j), X(i+1, j)]_{SAD} + [X(i, j), X(i, j+1)]_{SAD}$$

where $[.]_{SAD}$ is the spectral angle distance between neighboring pixels.

$$f_4 = \frac{1}{n_1 n_2} \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} [X(i, j), f_1(b)]_{SAD}$$

The average deviation of the horizontal directional residual (f_5) and the vertical directional residual (f_6) are computed as below,

$$f_5 = \frac{1}{n_1 n_2} \sum_{i=1}^{n_1 - 1} \sum_{j=1}^{n_2} \left[X(i, j), (X(i-1, j) + X(i+1, j)/2) \right]_{SAD}$$

$$f_6 = \frac{1}{n_1 n_2} \sum_{i=1}^{n_1} \sum_{j=1}^{n_2 - 1} \left[X(i, j), (X(i, j-1) + X(i, j+1)/2) \right]_{SAD}$$

The 12 other features considered are the following:

$$f_7: Entropy = -\sum_i \sum_j M(i, j) * \log(M(i, j))$$

This metric is a measure of randomness present in the neighborhood of current location. A high value shows that the elements of M are equal.

f_8: *Local Homogeneity* (LH)

$$LH = \sum_i \sum_j \frac{M(i, j)}{1 + (i - j)^2}$$

where *M(i,j)* represents the norm of the pixel at location (i,j). LH measures the similarity among pixels.

f_9: *Skewness* is given by:

$$\gamma_3 = \frac{\sum_i \sum_j (X(i, j) - \bar{X})^3}{(N-1)\sigma^3}$$

where $\overline{X}$ represents the mean vector, N is the number of pixels in the window. Skewness is a measure of symmetry, or more precisely, the lack of symmetry. A distribution, or data set, is symmetric if it looks the same to the left and right of the center point.

The above equation is modified to be applied to the hyperspectral pixel vectors using the SAD in the following manner:

$$\gamma_3 = \frac{\sum_i \sum_j SAD(X(i,j), \bar{X})^3}{(N-1)\sigma_{SAD}^3}$$

where SAD represents the spectral angle distance between two vectors, and σ_{SAD} is given by:

$$\sigma_{SAD} = \sqrt{\frac{\sum_i \sum_j SAD(X(i,j), \bar{X})^2}{N}}$$

f_{10}: *Kurtosis* is given by,

$$\gamma_4 = \frac{\sum_i \sum_j (X(i,j) - \bar{X})^4}{(N-1)\sigma^4} - 3$$

Kurtosis is a measure of whether the data are peaked or flat relative to a normal distribution. That is, data sets with high kurtosis tend to have a distinct peak near the mean, decline rather rapidly, and have heavy tails. Data sets with low kurtosis tend to have a flat top near the mean rather than a sharp peak. Using SAD the above equation can be rewritten as,

$$\gamma_4 = \frac{\sum_i \sum_j SAD(X(i,j), \bar{X})^4}{(N-1)\sigma_{SAD}^3} - 3$$

f_{11}-f_{18}: Eighth moments are given by,

$$m_{pq} = \sum_i \sum_j M(i, j) * i^p * j^q$$

where $0 \le p \le 2$ and $0 \le q \le 2$, *M(i,j)* represents the norm of the pixel placed in (x,y) coordinates. 2-D feature images results from the feature extraction step using the texture metrics.

2.4 Multidimensional Wavelet Transform

In a sense, wavelet-bases are optimal for a large class of one dimensional signals (including many real signals). However, as has been recognized, the (separable) 2-D wavelet transform does not possess these optimality properties for natural images [20]. The reason for this is that while the separable 2-D wavelet transform represents point-singularities efficiently, it is *inefficient* for line and curve singularities (edges).

Some of the important developments in recent wavelet research have been the implementation of 2-D multiscale transforms that represent edges more efficiently than does the separable wavelet transform. Examples include curvelets, [21, 22] directional filter banks and pyramids, [23, 24] complex filter banks,[25] the steerable pyramid, [26,27] and the complex dual-tree wavelet transform [28,29]. These transforms give superior results for image processing applications compared to the separable wavelet transform. In this paper, we investigate the use of the 3-D version of the dual-tree complex wavelet transform for hyperspectral texture synthesis. The dual-tree wavelet transform is nearly-shift invariant,

isolates edges with different orientations in different subbands, and has a manageable redundancy.

In 3-D, the checkerboard artifact of the separable transform is more serious than in 2-D. Correspondingly, the gain provided by using the oriented wavelet transform in place of the separable one is greater in higher dimensions. The shortcoming of the separable 2-D wavelet transform for image processing is further compounded for hyperspectral images, because with multidimensional separable transforms even more mixing of different orientations occurs.

The principle by which the 2-D dual-tree DWT resolves the problem of the mixing of orientations, can also be used to resolve the mixing of orientations in the 3-D case. This dual-tree wavelet transform is *spectral-selective,* while the separable 3-D wavelet transform is not. (Likewise, the 2-D dual-tree transform is *direction-selective*, while the 2-D transform is not.)

The wavelet associated with *the separable 3-D transform has the same checkerboard phenomenon present in the separable 2-D case,* a consequence of the mixing of orientations. The wavelet associated with *the dual-tree 3-D transform is free of this effect* In addition; the dual-tree 3-D transform has many more subbands than the separable 3-D transform (28 subbands instead of 7). These 28 subbands capture the spectral information having different directions and spectral variability, as can be appreciated in [16].

The introduction of the wavelet decomposition into the synthesis procedure has two advantages. First, it facilitates the measurement of texture statistics at particular scales. The second advantage is the reduction in computational load, since the synthesis is done to the coarser scales, the original information is represented by fewer pixels, and this allows larger features to be represented by smaller neighborhoods. So when we synthesize the wavelet transform coefficients, we are trying to maintain the same statistical relationships that the input image has, while generating more coefficients. We are synthesizing the same image, but in a different domain, the wavelet domain.

3 ALGORITHMS

The following flow chart shows the methodology followed to conduct this research work:

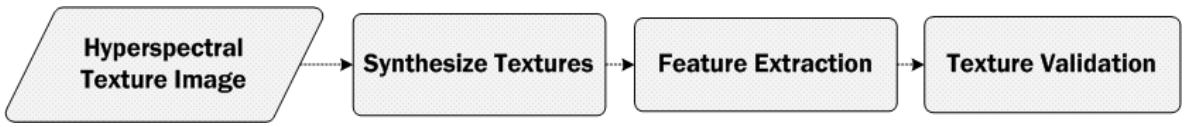


Figure 3: Methodology's Flow Chart

First, hyperspectral texture images were selected to be used as seed images to feed the hyperspectral algorithms. More details about the chosen images and their sensors can be found in chapter 4.

For the texture model, Markov Random Fields (MRF) is used since they have been proven to cover the widest variety of useful texture types. MRF methods, model a texture as a realization of a local and stationary random process. Each pixel of a texture image is characterized by a small set of spatially neighboring pixels, and this characterization is the same for all pixels.

3.1 Pixel Based

In this work Wei and Levoy's approach [30] for texture synthesis for color images was implemented to extend their algorithm for hyperspectral images [31]. The new texture is

generated pixel by pixel, and each pixel is determined so that local similarity is preserved between the example texture and the resulting image.

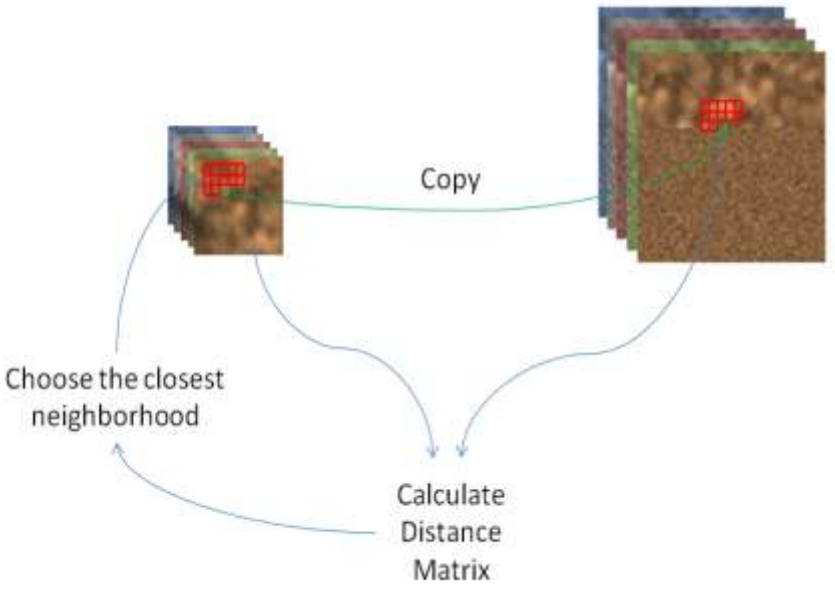


Figure 4: Pixel Based Algorithm

The algorithm takes as inputs an example texture patch image, the size of the desired output image ($X \times Y$) and the window size (w). The choice of the size of the neighborhood will be done in such a way that a texture primitive element is covered by the kernel. Let the input image have $x \times y$ pixels with λ spectral bands. Having this information, the algorithm proceeds as follows:

1. Create an $X \times Y \times \lambda$ image with random pixels. These random pixels are generated band by band using the first-order statistics of the input image, let *Is* be this image. This gives a better initial guess of the desired output image. Even though we do this procedure band by band, we are not considering the bands statistically independent.

Indeed, they dependent, as you can see in step three where each pixel is treated as a vector.

2. Calculate a distance matrix for the pixel to be synthesized. This is done by comparing the neighborhood of the current pixel with all possible pixel neighborhoods of the input image. If *Ia* is the texture primitive image, then $N_a=\{N_a(t)\}$ is the set of neighborhoods in *Ia*. The neighborhood of pixel *r* in *Is* will be compared using a simple L_2 norm (sum of squared difference) to measure the similarity between the neighborhoods. Figure 3 give a general idea of the algorithm Vectorized neighborhoods are used, as in Figure 4. Hence, the pixel *Is(r)* will be updated based on the comparison,

$$I_s(r) = I_a(t^*) where\ t^* = arg \min_{t \in I_a} \|N_s(r) - N_a(t)\| \quad (1)$$

The ||.|| norm is the Euclidean norm.

3. Choose the minimum distance neighborhood and copy its corresponding pixel (λ-dimensional vector*)* to the output image.
4. Repeat for each of the pixels of the output texture image until finished.

A multiresolution approach will help us better capture the structure of the original image. A Gaussian pyramid [32] is built from *Ia* and *Is* with *L* levels. The synthesis process is started using the lowest resolution with the same steps explained previously. Then to

synthesize the next level, say level *L*, an interpolation of *Is* at level *L+1* is done, and the synthesis process is repeated until we have synthesized at the highest resolution level.

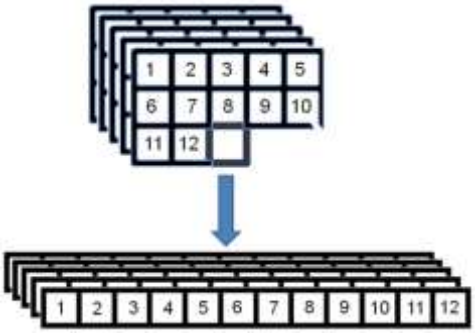


Figure 5: *Neighborhood shape*

3.2 Image Quilting

This texture synthesis method is presented in [33], but instead of taking just the 3 RGB bands, the whole pixel spectral vector is used. A MATLAB implementation of this algorithm can be found in [34].

From the pixel based algorithm we can appreciate that neighbors are highly correlated. So in this case the idea is very similar, instead of obtaining the P(r | N(r)), we want P(B | N(B)), where B is the block. This makes the algorithm much faster because we're synthesizing a whole block at once. This technique will fail for highly structured patterns due to boundary inconsistencies, but for many stochastic textures it works very well. For this algorithm we have two variables: block size and overlap size. The block must be big enough to capture the relevant structures in the texture. In our case the width of the edge overlap (on

one side) was 1/6 of the size of the block. The error was computed using the L_2 norm on pixel values. The error tolerance was set to be within 0.1 times the error of the best matching block. The algorithm is explained next:

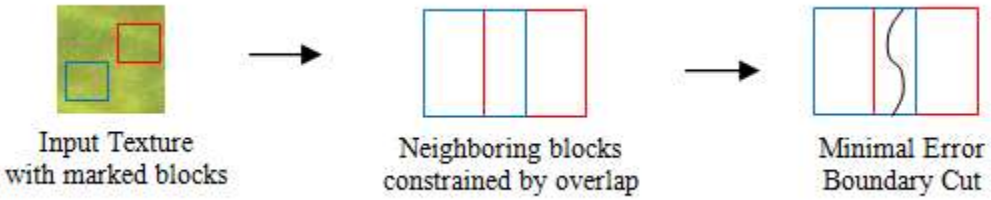


Figure 6: *Quilting Texture*

1. Pick size of block and size of overlap.
2. Square blocks from the input texture are patched together to create a new texture sample. This is done in raster order. To start pick a block randomly.

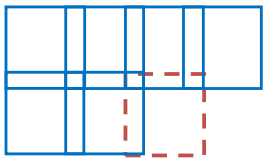


Figure 7: *Overlap for the dotted block.*

3. Search input texture for a block that satisfies overlap constraints (above and left), within some error tolerance, as it can be appreciated in Figure 3. The blocks overlap and each new block is chosen so that it "agrees" with its neighbors in the region of overlap.
4. Paste new block into resulting texture.

5. Use dynamic programming to compute the error surface between the newly chosen block and the old blocks at the overlap region, and then find the minimum cost path through the error surface at the overlap and make that the boundary of the new block. This is done to reduce blocking effect at the boundary between blocks.
6. Repeat.

3.3 Wavelet Based

A general idea of the work presented in [35] was implemented. Their concept was extended from color images to hyperspectral images by using a 3D wavelet transform and a different texture synthesis algorithm to obtain the unknown wavelet coefficients.

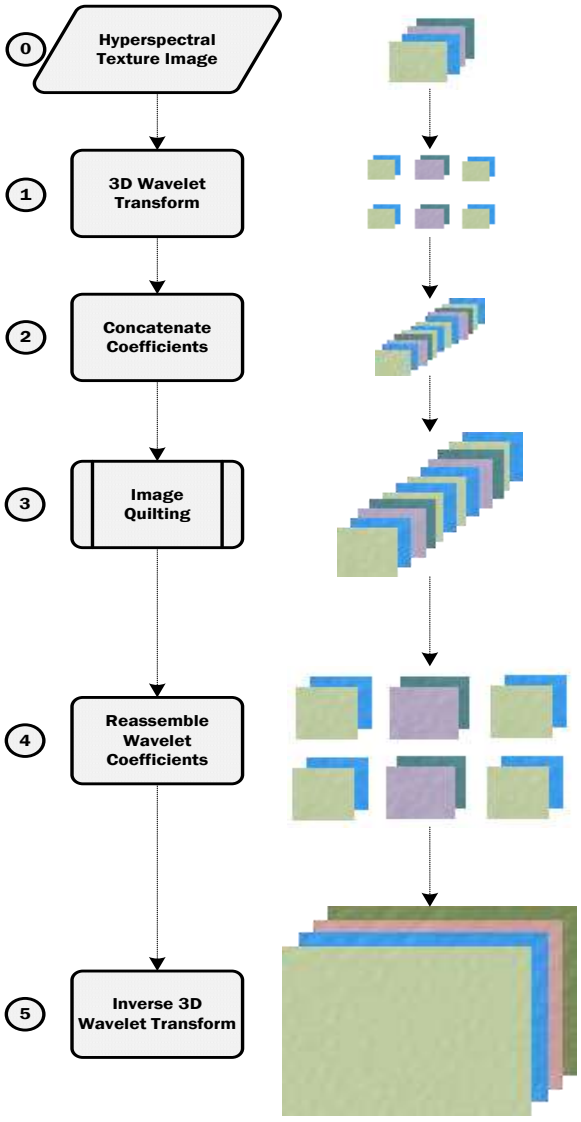


Figure 8: 3D Wavelet Algorithm

Figure 7 shows a flow chart of this algorithm. Numbers at the left match the description of the steps presented next. Given the initial sample image *Ia* of size $x \times y$ pixels with λ spectral bands and the required output size $X \times Y$ of the image to be synthesized *Is*, the algorithm proceeds as follows:

1. Apply the 3D wavelet transform to the input texture *Ia*. A description of this transform can be found in [36] and its MATLAB code can be found in [37].
2. Concatenate the sub images obtained (coefficients) from the wavelet transform in the 3rd dimension, this will result in a 3D matrix, with size $x/2 \times y/2 \times ((\lambda * k)/2)$ where k is the number of wavelet coefficient matrixes.
3. Synthesize this new matrix using the algorithm presented in section 2.2, specifying the scale factor so that the resulting matrix is of size $X/2 \times Y/2 \times ((\lambda * k)/2)$.
4. Reassemble the wavelet coefficients.
5. Do the wavelet reconstruction. This will result in $N \times M \times l$ matrix, which was our goal.

The resources used to run this experiment were:

- SOC-700 VIS/NIR hyperspectral camera, to capture some texture images.
- Desktop Computer from LARSIP
- MATLAB R200b, to run all the algorithms mentioned in this work.

4 DATA ANALYSIS AND VALIDATION

In this Section we show the synthesis results of three hyperspectral scenarios using the three algorithms presented in Section 2. The RGB composite of the hyperspectral images is shown in the following order: first, the pixel based algorithm (Section 2.1), second, the image quilting algorithm (Section 2.2) and finally the 3D wavelet transform based algorithm (Section 2.3).

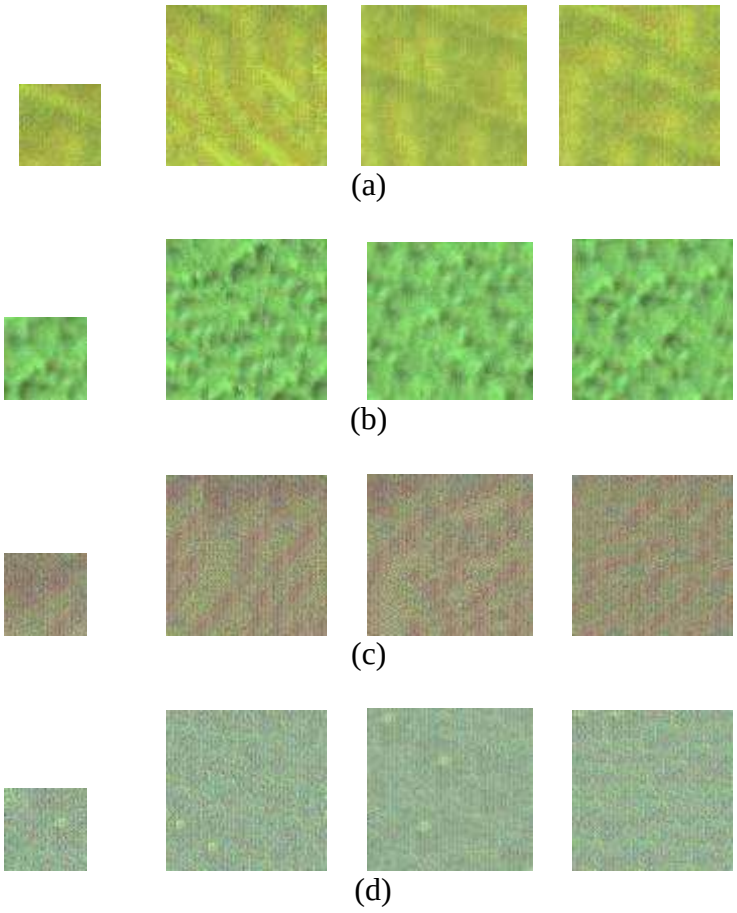
To verify the correspondence between the original and synthetic textures, not only a visual check using different bands was made. After obtaining the texture features from both, the original and the synthesized hyperspectral images, we proceeded to train a Quadratic classifier (for SOC camera images and AVIRIS Cuprite Image) and a SVM classifier (for Enrique Reef) using the original images as the training set, and the synthetic ones as the testing set. Features f1 to f6 where calculated for each of the images, as described in chapter 2. For the classification of the SOC camera textures, the following five classes were chosen: Dry Leaves, Brick, Wood, Bottled Water and Wet Sand. The resulting classification accuracy and kappa coefficient are shown in Table 10.

4.1 SOC-700 VIS/NIR Hyperspectral Camera Textures

4.1.1 Experiment 1

Hyperspectral images of different textures are collected using the SOC-700 hyperspectral camera. This camera has a spectral resolution of 4 nm with 120 bands and a

spectral range from 400 to 1000 nm. The texture synthesis results are presented in Figure 9. The images shown are the RGB color-composite from the original hyperspectral images using bands 90, 68 and 29. For the pixel based algorithm each texture is generated using a 3-level Gaussian pyramid, with neighborhood sizes 3x3, 5x5 and 7x7, respectively, from lower to higher resolutions. For the image quilting algorithm, textures a,b,c,d and e were generated using patch sizes of 7x7 pixels, the rest of them using a 12x12 patch. For the wavelet based algorithm, all textures were generated using 7x7 patches for the image quilting algorithm and a single decomposition level for the wavelet transform.



(a)

(b)

(c)

(d)

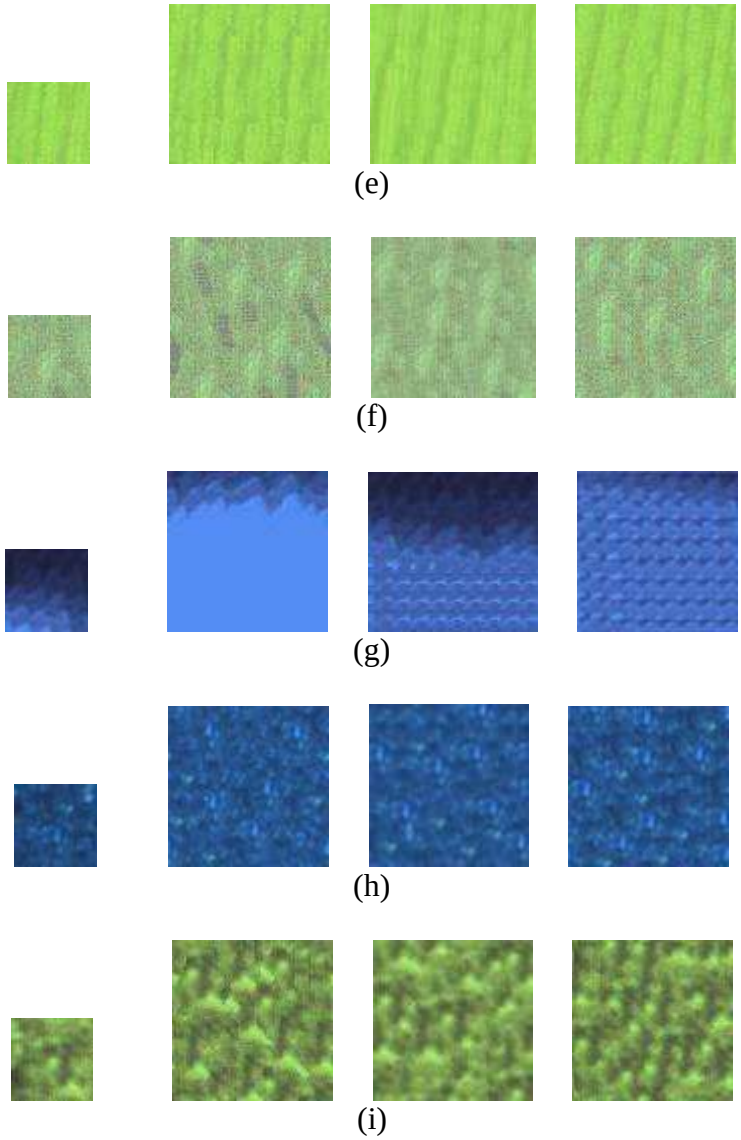


Figure 9: *Texture synthesis results. The smaller patches (size 41x41) are the input textures, and to their right are synthesized results using the pixel based algorithm, the image quilting algorithm and the wavelet based algorithm, respectively, which approximately double the spatial dimensions of the input size. SOC camera textures: (a) Dry Leaves (b) Brick (c) Can (d) Tire (e) Wood (f) Coral (g) Bottled water (h) Asphalt (i) Wet Sand.*

Table 1: Confusion Matrix using Pixel Based Algorithm and Quadratic Classifier for SOC Camera Textures

	Dry Leaves	Brick	Wood	Bottled Water	Wet Sand	Total
Dry Leaves	5348	0	0	0	1052	6400
Brick	0	6340	58	2	0	6400
Wood	0	529	5867	4	0	6400
Bottled Water	0	0	0	6400	0	6400
Wet Sand	48	0	0	0	6352	6400
Total	5396	6869	5925	6406	7404	32000

Table 2: Confusion Matrix using Image Quilting Algorithm and Quadratic Classifier for SOC Camera Textures

	Dry Leaves	Brick	Wood	Bottled Water	Wet Sand	Total
Dry Leaves	5864	0	0	0	536	6400
Brick	0	6309	91	0	0	6400
Wood	0	107	6286	7	0	6400
Bottled Water	0	0	0	6400	0	6400
Wet Sand	173	0	0	0	6227	6400
Total	6037	6416	6377	6407	6763	32000

Table 3: Confusion Matrix using Wavelet based Algorithm and Quadratic Classifier for SOC Camera Textures

	Dry Leaves	Brick	Wood	Bottled Water	Wet Sand	Total
Dry Leaves	3857	0	0	0	2543	6400
Brick	0	5960	435	5	0	6400
Wood	0	2558	3810	32	0	6400
Bottled Water	0	0	0	6400	0	6400
Wet Sand	121	0	0	0	6279	6400
Total	3978	8518	4245	6437	8822	32000

4.1.2 Experiment 2

In order to consider more varied textures, a new set of experiment was run using different materials like cardboard and concrete, as well as natural materials such as coconut shell and a pod and also a group of "Big Ass" Ants stacked together. A similar procedure was followed as in Experiment 1. The results are shown below:

Table 4: Confusion Matrix using Pixel Based Algorithm and Quadratic Classifier for SOC Camera 2nd Textures

	Ants	Cardboard	Concrete	Coconut	Pod	Total
Ants	9981	0	0	19	0	10000
Cardboard	0	8149	0	0	1851	10000
Concrete	0	0	8873	2	1125	10000
Coconut	170	0	62	7673	273	8178
Pod	33	7	289	944	4927	6200
Total	10184	8156	9224	8638	8176	44378

Table 5: Confusion Matrix using Image Quilting Algorithm and Quadratic Classifier for SOC Camera 2nd Textures

	Ants	Cardboard	Concrete	Coconut	Pod	Total
Ants	9964	0	0	36	0	10000
Cardboard	0	9868	0	0	132	10000
Concrete	0	0	8590	1	1409	10000
Coconut	801	0	11	7355	11	8178
Pod	0	10	123	0	6067	6200
Total	10765	9878	8724	7392	7619	44378

Table 6: Confusion Matrix using Wavelet based Algorithm and Quadratic Classifier for SOC Camera 2nd Textures

	Ants	Cardboard	Concrete	Coconut	Pod	Total
Ants	9956	0	0	44	0	10000
Cardboard	0	9015	0	0	985	10000
Concrete	0	0	9584	0	416	10000
Coconut	10	0	0	8168	0	8178
Pod	0	11	606	0	5583	6200
Total	9966	9026	10190	8212	6984	44378

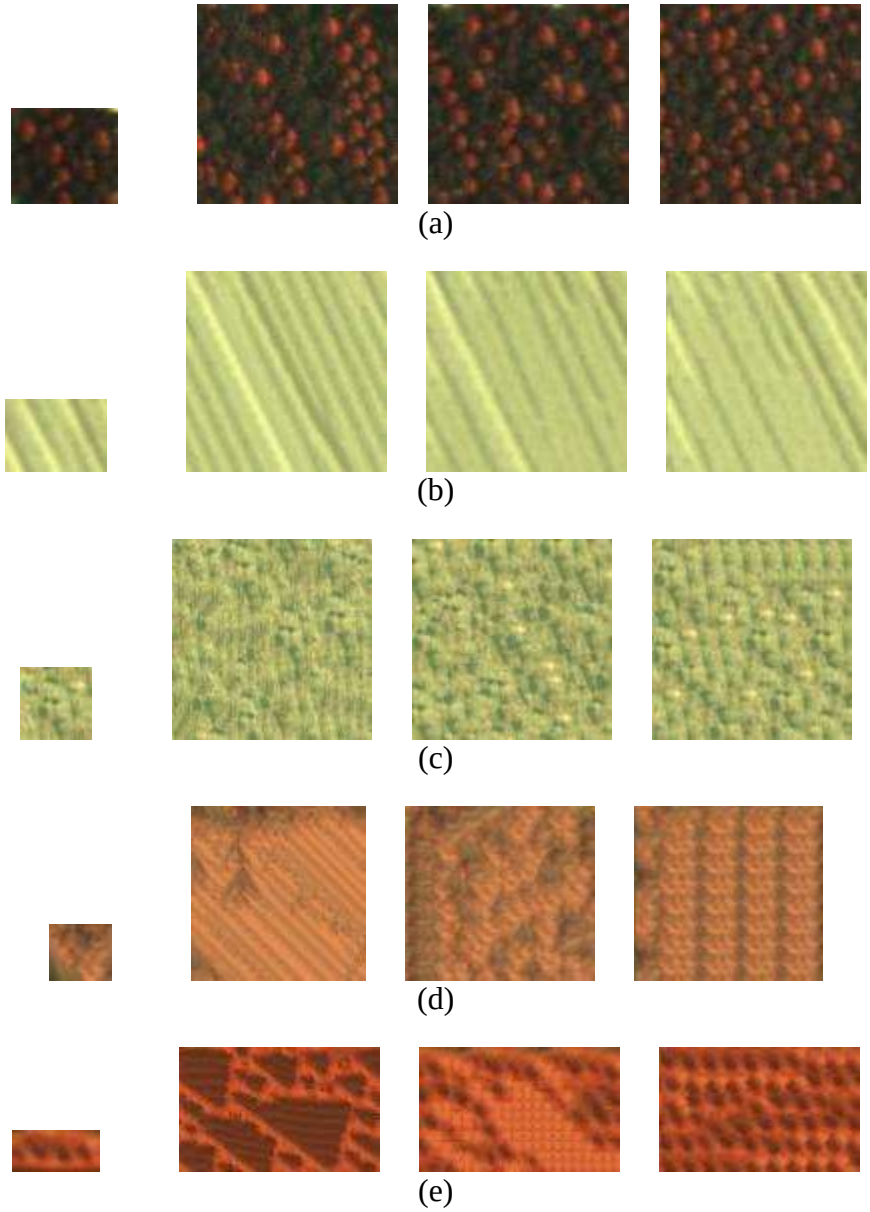


Figure 10: *Texture synthesis results. The smaller patches are the input textures, and to their right are synthesized results using the pixel based algorithm, the image quilting algorithm and the wavelet based algorithm, respectively, which approximately double the spatial dimensions of the input size. SOC camera 2nd textures: (a) Ants (b) Cardboard (c) Concrete (d) Coconut Shell (e) Pod.*

In Figure 10 we can notice that the spatial attributes are preserved between input and output images. To show that the spectrum of the input texture is very similar to the output

texture, the average spectrum of all the pixels in the input texture was calculated. This was also done for the output textures as shown in Figures 11-13, one figure for each algorithm. The similarity between input and output textures suggests that the spectral attributes are conserved.

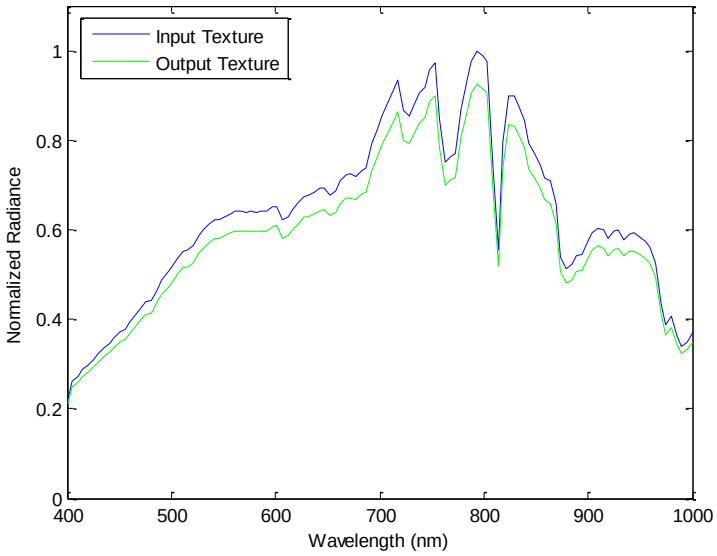


Figure 11: Spectral Signature of Ants class for Pixel Based Algorithm

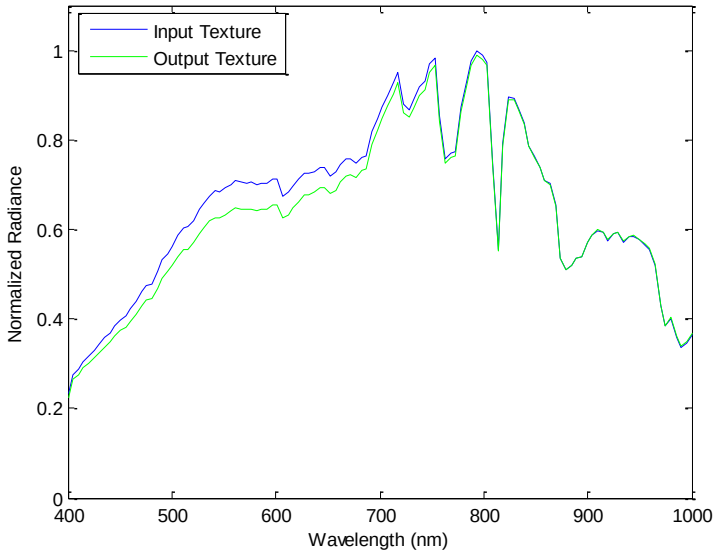


Figure 12: Spectral Signature of Ants class for Patch Based Algorithm

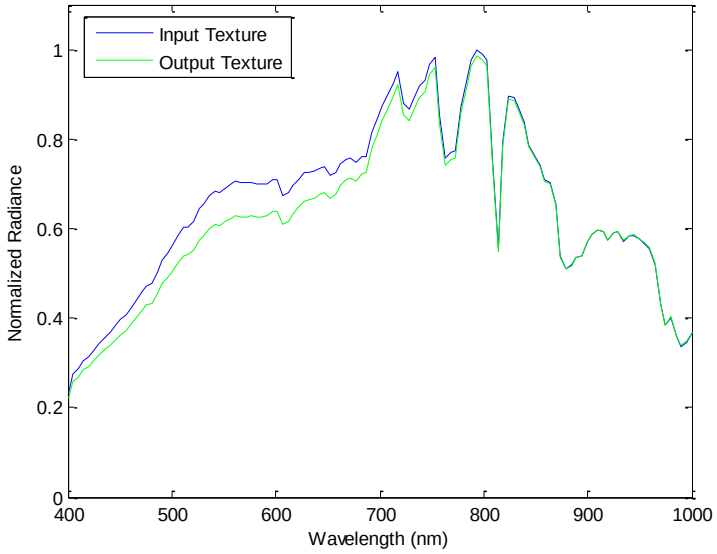


Figure 13: Spectral Signature of Ants class for Wavelet Based Algorithm

4.2 Enrique Reef

The hyperspectral texture images are patches taken from an image collected over Enrique Reef in La Parguera, Puerto Rico using the AISA sensor (Figure 10). Each image has 128 bands. The texture synthesis results are presented in Figure 11. The smaller patches are the original textures and to the right are synthesized results, which are three times the size of the original textures. The texture images shown in Figure 11 are the RGB color-composite from the original hyperspectral images using bands 60, 38 and 19. For the algorithm pixel based each texture is generated using a 3-level Gaussian pyramid, with neighborhood sizes 3x3, 5x5 and 5x5, respectively, from lower to higher resolutions. For the image quilting algorithm, textures were generated using a 7x7 patch. For the wavelet based algorithm, all textures were generated using 7x7 patches for the image quilting algorithm and a single decomposition level for the wavelet transform.

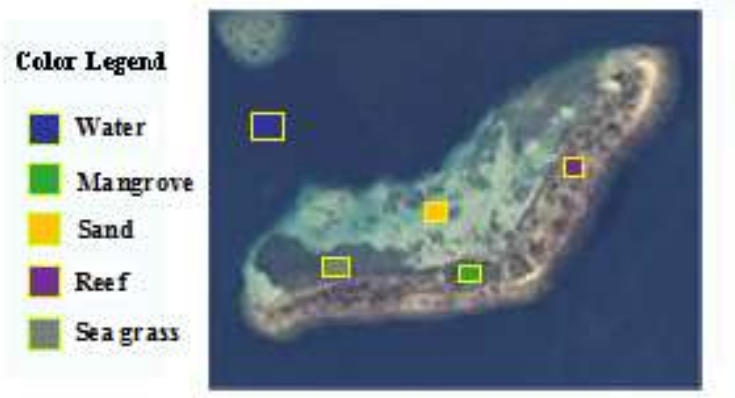


Figure 14: *Selected Texture primitives from Enrique Reef Image*

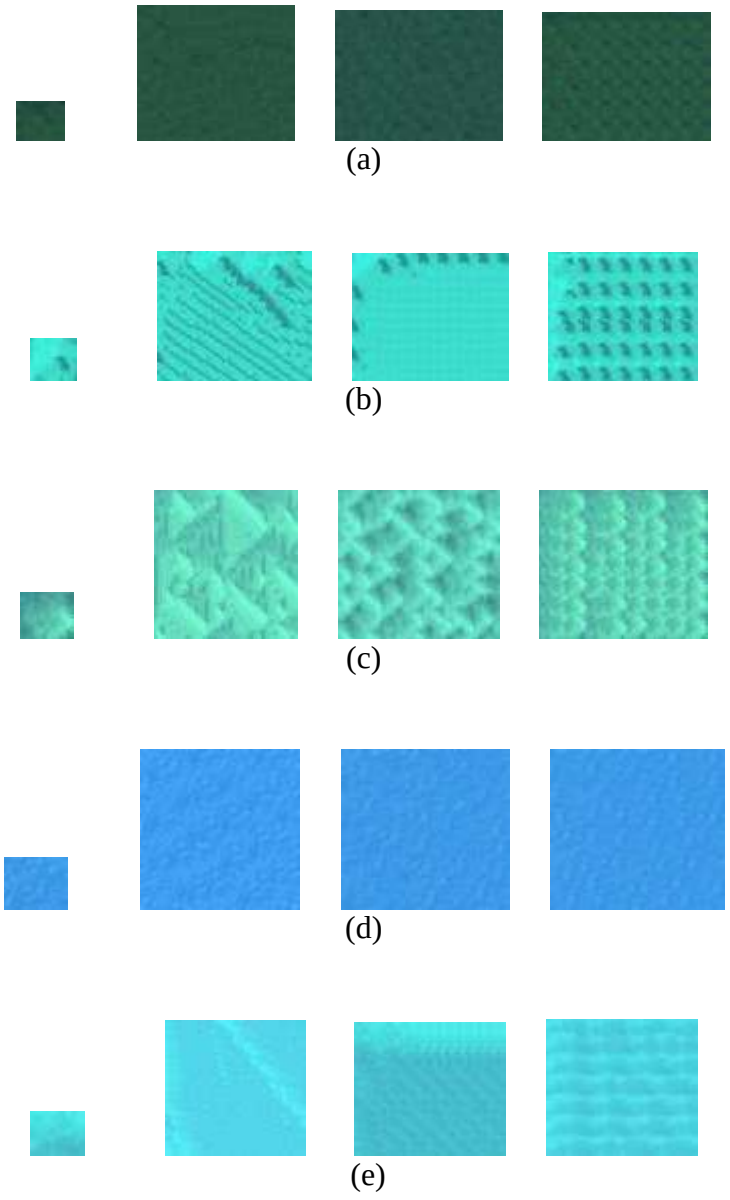


Figure 15: *Texture synthesis results. The smaller patches are the input textures and to their right are synthesized results the pixel based algorithm, the image quilting algorithm and the wavelet based algorithm, respectively, which approximately double the spatial dimensions of the input size. Enrique Reef textures: (a) Mangrove (b) Sand (c) Coral Reef (d) Water (e) Sea grass.*

Table 7: Confusion Matrix using Pixel Based Algorithm and SVM Classifier for Enrique Reef Textures

	Mangrove	Sand	Coral Reef	Water	Sea Grass	Total
Mangrove	3007	4	36	1438	289	4774
Sand	0	6106	294	0	0	6400
Coral Reef	409	2409	2883	0	59	5760
Water	438	0	3	3510	657	4608
Sea Grass	2723	0	113	2101	823	5760
Total	6577	8519	3329	7049	1828	27302

Table 8: Confusion Matrix using Image Quilting Algorithm and SVM Classifier for Enrique Reef Textures

	Mangrove	Sand	Coral Reef	Water	Sea Grass	Total
Mangrove	1430	107	1207	90	1940	4774
Sand	0	6329	71	0	0	6400
Coral Reef	417	0	5278	7	58	5760
Water	0	0	106	0	4502	4608
Sea Grass	2122	0	52	2177	1409	5760
Total	3969	6436	6714	2274	7909	27302

Table 9: Confusion Matrix using Wavelet Based Algorithm and SVM Classifier for Enrique Reef Textures

	Mangrove	Sand	Coral Reef	Water	Sea Grass	Total
Mangrove	2059	30	1671	26	988	4774
Sand	0	5593	807	0	0	6400
Coral Reef	0	791	4969	0	0	5760
Water	44	0	30	4419	115	4608
Sea Grass	668	99	491	0	4502	5760
Total	2771	6513	7968	4445	5605	27302

4.3 AVIRIS Cuprite

The AVIRIS Cuprite image was taken over the mining district, 2 km north of Cuprite, Nevada, by NASA/Ames on June 19, 1997. This image has 640 x 2378 pixels and 224 bands in the 370-2500 nm range. A 400 x 350 pixel portion of the fourth scene is used. 50 bands in 172-221 that corresponds to the 2000-2480nm wavelength absorption region were selected by the USGS for mapping minerals in the Cuprite image. Figure 12 shows the RGB version for the bands 208, 206 and 215. The classes selected are: Goethite, nano Hematite, Amorphous Iron oxides and Fe^{2+} -bearing minerals + Hematite. For the pixel based algorithm each texture is generated using a 3-level Gaussian pyramid, with neighborhood sizes 3x3, 5x5 and 5x5, respectively, from lower to higher resolutions. For the image quilting algorithm, textures were generated using a 7x7 patch. For the wavelet based algorithm, all textures were generated using 7x7 patches for the image quilting algorithm and a single decomposition level for the wavelet transform. Synthesis results are presented in Figure 13.

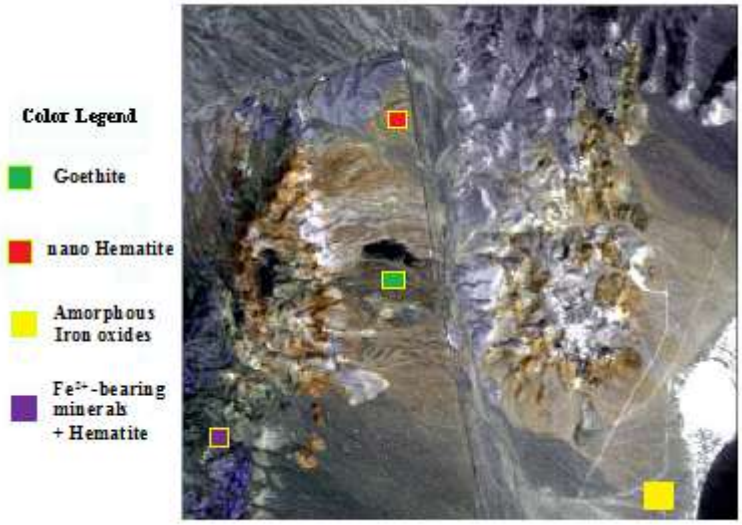


Figure 16: Selected Texture primitives from Cuprite Image.

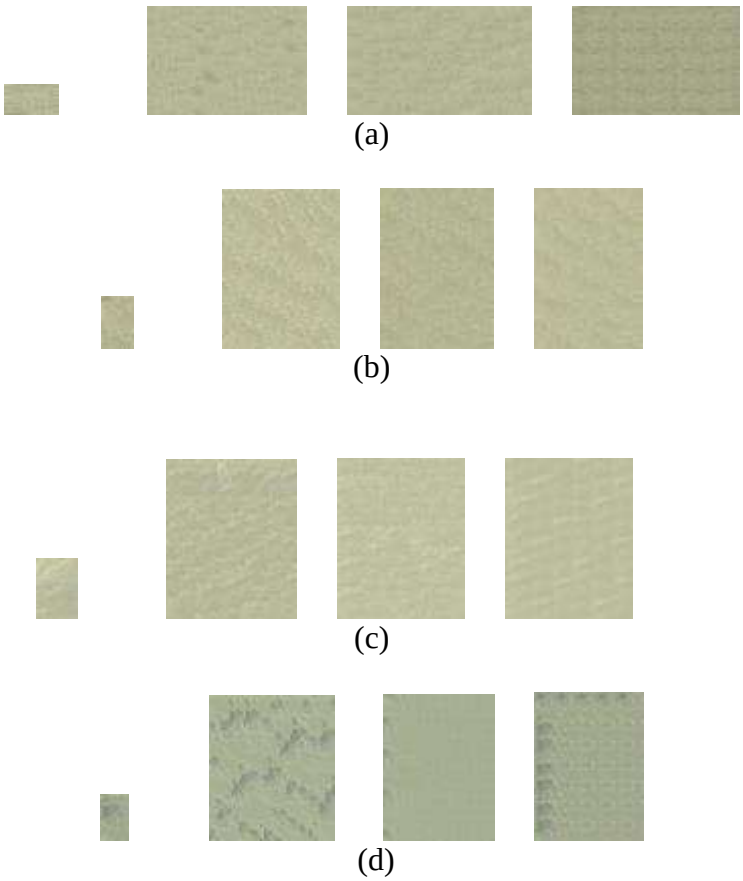


Figure 17: Texture synthesis results. The smaller patches are the input textures and to their right are synthesized results using the pixel based algorithm, the image quilting algorithm and the wavelet based algorithm, respectively, which approximately double the spatial dimensions of the input image. Cuprite textures: (a) Goethite (b) nano Hematite (c) Amorphous Iron oxides (d) Fe^{2+} -bearing minerals + Hematite.

Table 10: Confusion Matrix using Pixel Based Algorithm and Quadratic Classifier for Cuprite Image Textures

	Goethite	**nano Hematite**	**Amorphous Iron oxides**	**Fe^{2+} -bearing minerals + H.**	**Total**
Goethite	3736	0	0	584	4320
nano Hematite	0	4052	195	73	4320
Amorphous Iron oxides	0	770	3528	22	4320
Fe^{2+} -bearing minerals + H.	76	153	651	3440	4320
Total	3812	4975	4374	4119	17280

Table 11: Confusion Matrix using Image Quilting Algorithm and Quadratic Classifier for Cuprite Image Textures

	Goethite	nano Hematite	Amorphous Iron oxides	Fe^{2+} -bearing minerals + H.	Total
Goethite	3650	0	0	670	4320
nano Hematite	0	4074	225	21	4320
Amorphous Iron oxides	0	444	3846	30	4320
Fe^{2+} -bearing minerals + H.	0	81	722	3517	4320
Total	3650	4599	4793	4238	17280

Table 12: Confusion Matrix using Wavelet Based Algorithm and Quadratic Classifier for Cuprite Image Textures

	Goethite	nano Hematite	Amorphous Iron oxides	Fe^{2+} -bearing minerals + H.	Total
Goethite	4236	0	0	84	4320
nano Hematite	0	3670	523	127	4320
Amorphous Iron oxides	0	767	3551	2	4320
Fe^{2+} -bearing minerals + H.	93	228	665	3334	4320
Total	4329	4665	4739	3547	17280

Table 13: Overall Accuracy and kappa Statistics (κ)

Dataset	Pixel Based		Image Quilting		Wavelet Based	
	OA	κ	OA	κ	OA	κ
SOC Camera	.9471	.9339	.9714	.9643	.8221	.7776
SOC Camera 2nd	.8024	.8652	.9429	.9283	.9533	.9413
Enrique Reef	.5981	.4984	.5291	.4057	.7890	.7343
AVIRIS Cuprite	.8539	.8052	.8731	.8308	.8560	.8079

4.4 Histograms

The following histograms were calculated after computing the SAD between each pixel and the spectral mean of the input textures. These histograms were calculated to verify if there is any relationship between the SAD differences of the generating textures that were classified with better overall accuracy than those that were not.

4.4.1 SOC Camera Textures

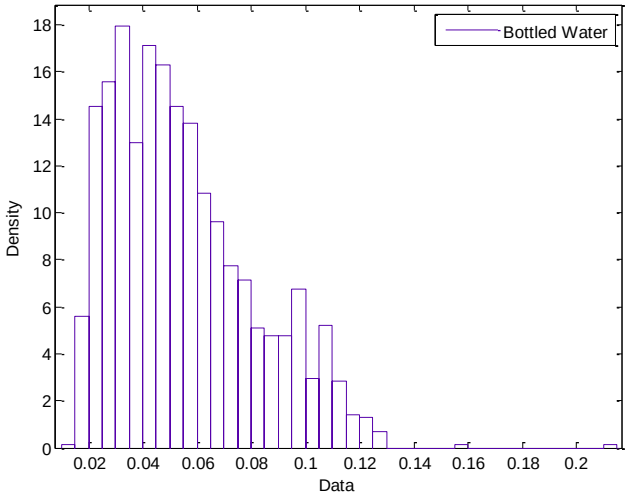


Figure 18: Histogram of SAD feature space for Bottled Water

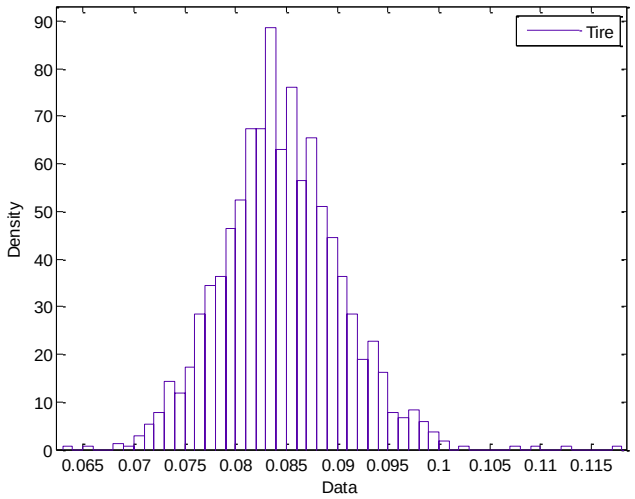


Figure 19: Histogram of SAD feature space for Tire

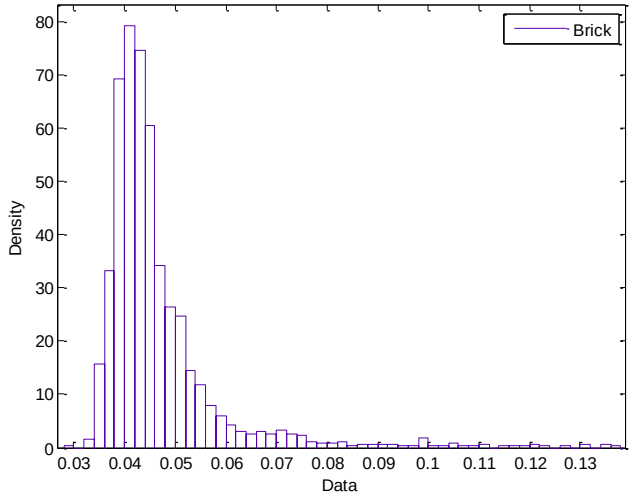


Figure 20: Histogram of SAD feature space for Brick

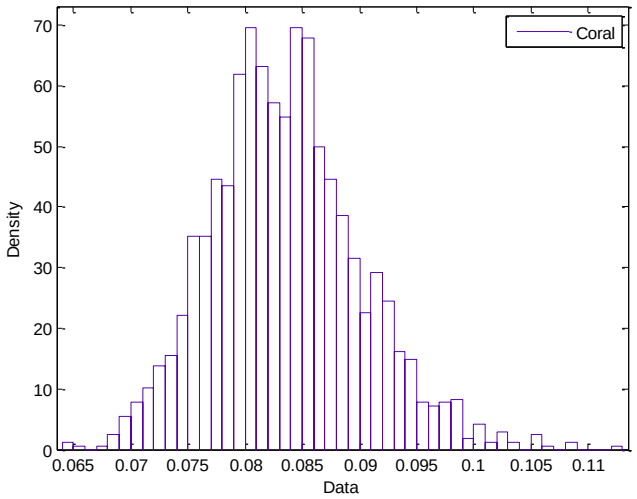


Figure 21: Histogram of SAD feature space for Coral

4.4.2 SOC Camera 2nd Textures

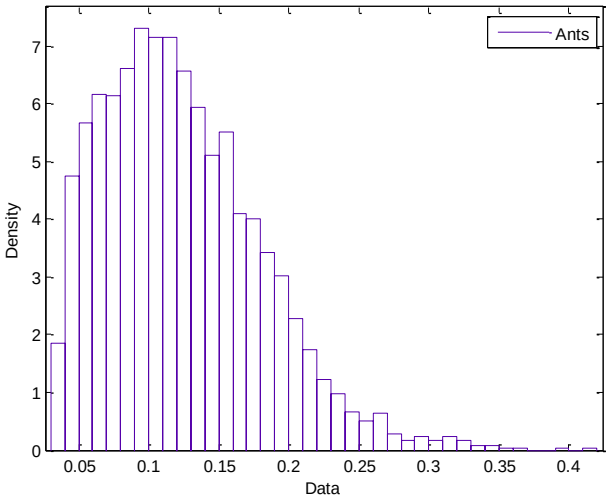


Figure 22: Histogram of SAD feature space for Ants

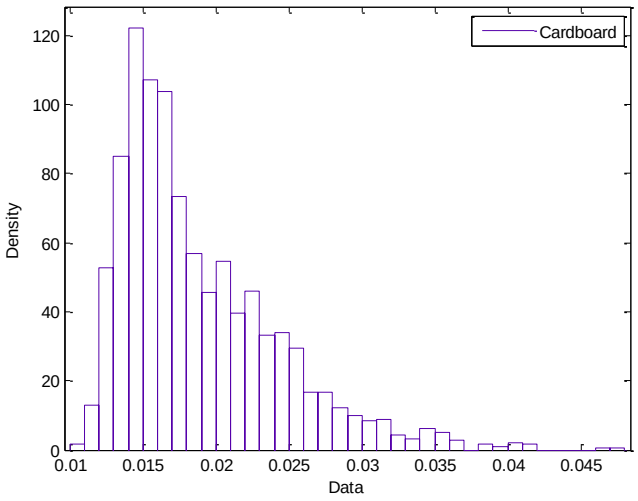


Figure 23: Histogram of SAD feature space for Cardboard

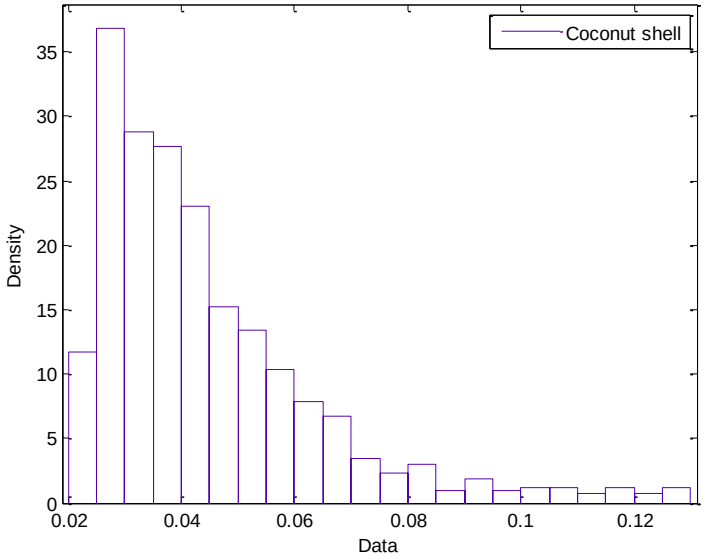


Figure 24: Histogram of SAD feature space for Coconut shell

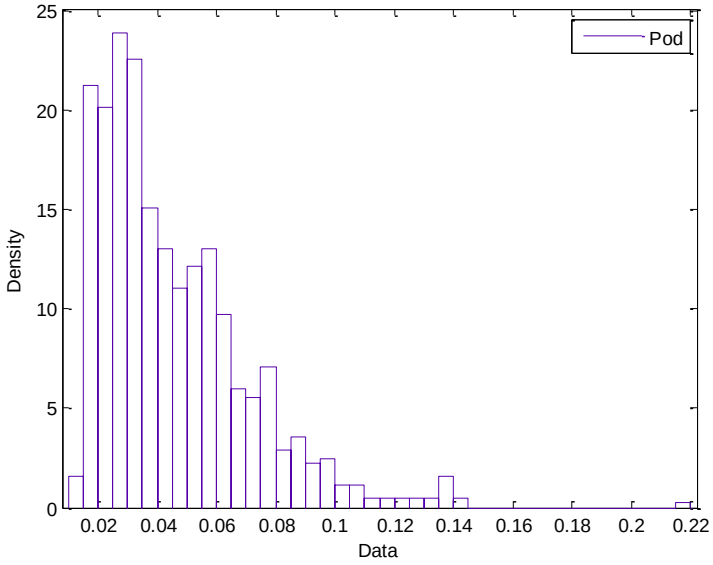


Figure 25: Histogram of SAD feature space for Pod

4.4.3 AVIRIS Cuprite Images

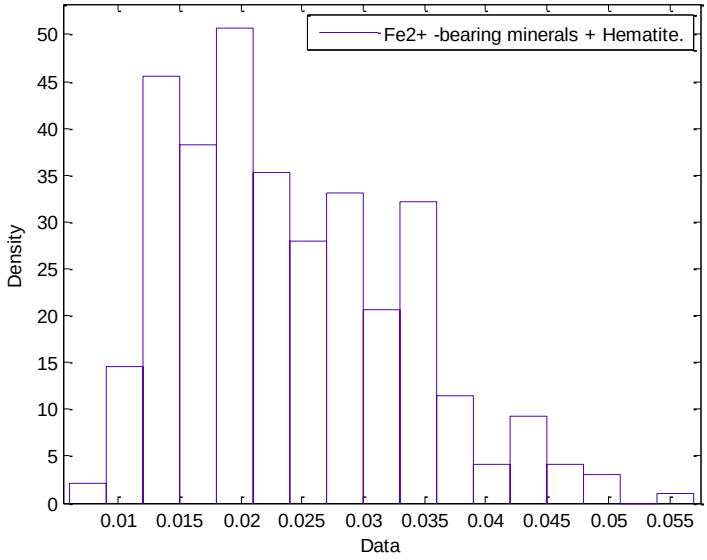


Figure 26: Histogram of SAD feature space for Fe^{2+} -bearing minerals

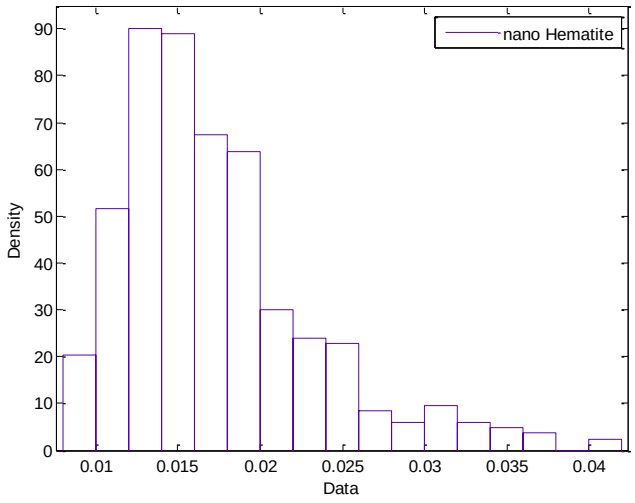


Figure 27: Histogram of SAD feature space for nano Hematite

4.4.4 Enrique Reef Image

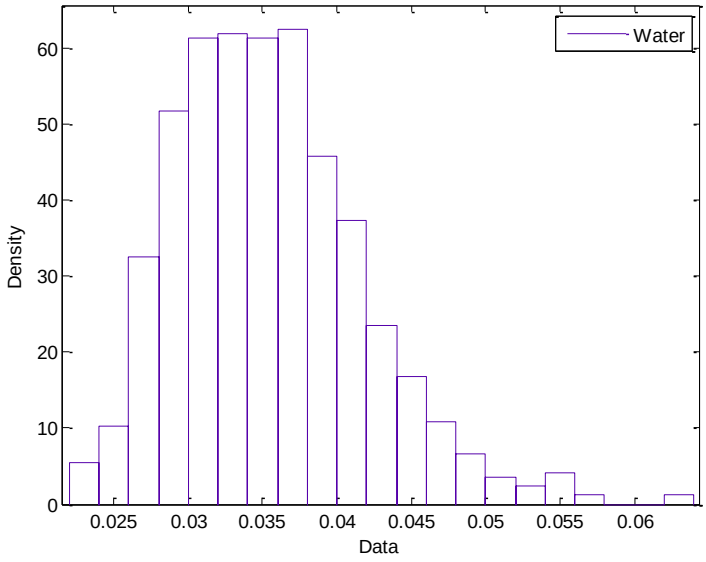


Figure 28: Histogram of SAD feature space for Water

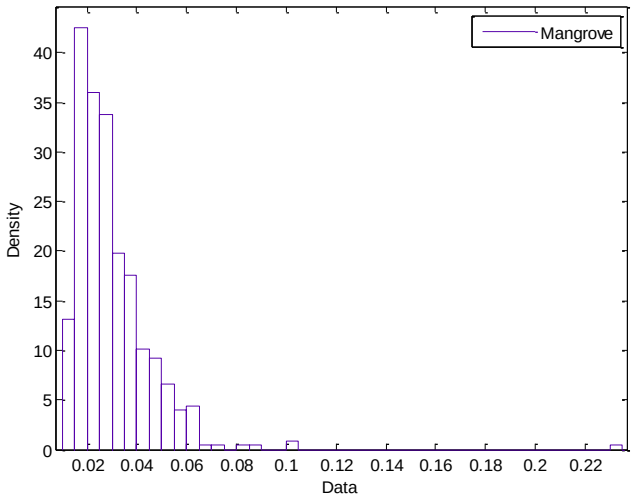


Figure 29: Histogram of SAD feature space for Mangrove

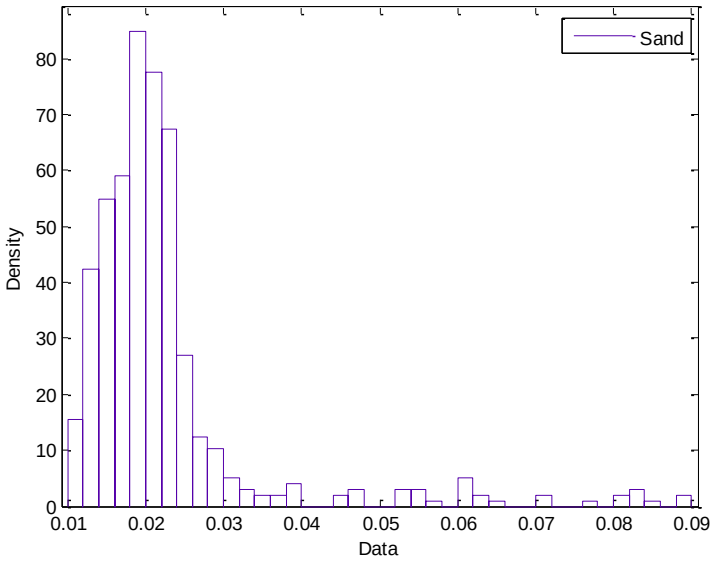


Figure 30: Histogram of SAD feature space for Sand

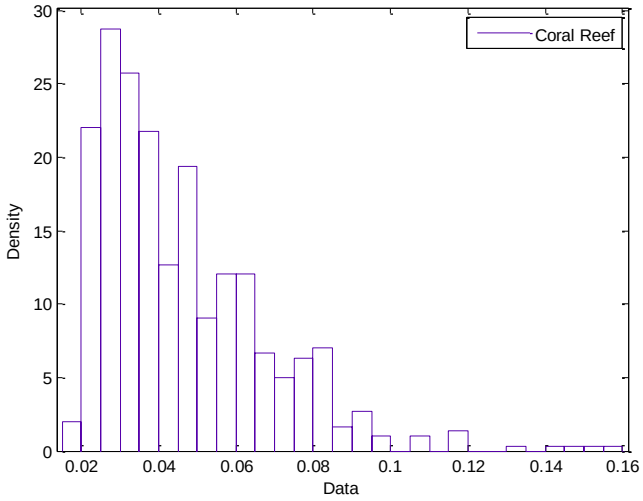


Figure 31: Histogram of SAD feature space for Coral Reef

5 CONCLUSIONS AND FUTURE WORK

5.1 Conclusions

Three texture synthesis algorithms for hyperspectral images were successfully implemented. For most of the experiments it was verified that there is a substantial agreement between the synthetic textures and their generating input textures, as the OA and kappa coefficient show in Table 10. For the SOC camera textures and Cuprite textures there was very good agreement for the chosen classes. For most of the stochastic textures, good results were obtained, as well as for the more structured texures, as in Figure 10, specifically *Ants* (class a) and Cardboard (class b). It is important to mention that for *Wood* (class e) the directionality of the texture was preserved by the algorithms. For the *Bottled water* class (*g*) in Figure 9, it can be appreciated that the input patch does not exhibit homogeneity, which means that the statistical properties of any one region of the image were not the same as any other region. This differs from our initial assumptions and due to this fact the synthetic images differ a lot from one another. This also happened for classes *Sand* (*b*) and *Coral Reef* (*c*) in Figure 11 and for class Fe^{2+} -bearing minerals + Hematite (*d*).

Since some remote sensing images do not have good spatial resolution, sometimes it will happen that a big enough patch cannot be selected as the input texture, or those that are big enough do not comply with the initial assumptions. This happened with *Sand* and *Coral Reef classes* for Enrique Reef Textures and the resulting images were not as desired. For this

case the wavelet based algorithm gave the best results, since it better captures the underlying frequencies of the images. This resulted in a better OA for Enrique Reef.

The histograms were calculated trying to find a relationship between them and the synthesis results. No relationship was found but histograms are a good tool to see the spectral differences between pixels. For example the Cardboard histogram in Figure 23 shows that the spectral differences are more similar than those of the Ants, shown in Figure 24. This could be because the Cardboard is a uniform material, and the Ants have different spectral response of the different parts of their bodies.

In general, the image quilting algorithm gave better results, as the classification accuracy in Table 13 shows. This could be because for most of the experiments, a small patch is able to capture better the distribution of the input texture than a pixel based algorithm. The pixel based and the wavelet based performed similarly.

5.2 Future Work

- It is desired to have automatic window size selection to reduce the input parameters to the algorithm.

- A different validation framework for algorithms of hyperspectral texture synthesis is desired, in which we could have a measure of the probability of success given a single input texture.
- Implement other texture synthesis methods for hyperspectral images using tensor algebraic approach, which would take advantage of the 3D nature of hyperspectral images.
- Continue synthesizing textures using other hyperspectral data. More tests in other scenarios, like medical tissue textures should be done.

5.3 Contributions

5.3.1 Publications

- N. Diaz and V. Manian, "Hyperspectral texture synthesis by multiresolution pyramid decomposition." In Proceedings of SPIE: Algorithms and Technologies for Multispectral, Hyperspectral, and Ultraspectral Imagery XV, Vol. 7334, April 2009.

5.3.2 Poster Sessions

- Gordon-CenSSIS NSF Site Visit at Boston University, Boston, MA. April 14th – 15th 2010.
- Computing Alliance of Hispanic-Serving Institutions (CAHSI) Annual Meeting. Redmond, Washington. April 5th – 7th 2010.
- Gordon-CenSSIS NSF Site Visit at Boston University, Boston, MA. April 22nd – 23rd 2009.

ACKNOWLEGMENT

This work was supported by the Department of Defense under grant HM1582-08-1-0047 and by the Center for Subsurface Sensing and Imaging Systems, University of Puerto Rico at Mayaguez.

REFERENCES

1. A. Witkin and M. Kass. "Reaction-diffusion textures". In T. W. Sederberg, editor, *Computer Graphics (SIGGRAPH '91 Proceedings*), volume 25, pages 299–308, July 1991.
2. S. P. Worley. "A cellular texture basis function". In H. Rushmeier, editor, *SIGGRAPH 96 Conference Proceedings, Annual Conference Series,* pages 291–294. ACM SIGGRAPH, Addison Wesley, Aug. 1996.
3. J. Dorsey, A. Edelman, J. Legakis, H. W. Jensen, and H. K. Pedersen. "Modeling and rendering of weathered stone". *Proceedings of SIGGRAPH 99,* pages 225–234, August 1999.
4. A. Efros and T. Leung. Texture synthesis by non-parametric sampling. *In International Conference on Computer Vision,* volume 2, pages 1033–8, Sep 1999.
5. L. Y. Wei and M. Levoy, "Fast texture synthesis using tree structured vector quantization" *Proc. Intl. Conf. Computer graphics and interactive techniques*, pp. 479-488, 2000.
6. D. J. Heeger and J. R. Bergen, "Pyramid based texture analysis/synthesis," *Proc. Intl. Conf. Computer graphics and interactive techniques,* pp. 229-238, 1995.
7. J. S. Bonet, "Multiresolution sampling procedure for analysis and synthesis of texture images," *Proc. Intl. Conf. Computer graphics and interactive techniques,* pp.361-368, 1997.
8. Subhadip Sarkar and Glenn Healey , "Models for hyperspectral image synthesis and implications for algorithm evaluation", *Proc. SPIE 7334, 73340R* (2009).
9. Dimitris Manolakis, David Marden, and Gary A. Shaw, "Hyperspectral Image Processing for Automatic Target Detection Applications", *Lincoln Laboratory Journal*, Volume 14, Number 1, 2003.
10. Multivariate Texture Measured by Local Binary Pattern for Multispectral Image Classification Cuiyu Song; Peijun Li; Fengjie Yang; Geoscience and Remote Sensing Symposium, 2006. IGARSS 2006. IEEE International Conference on Digital Object Identifier: 10.1109/IGARSS.2006.555 Publication Year: 2006 , Page(s): 2145 – 2148
11. D. J. Marceau, P. J. Howarth and J. M. Dubois, et al, "Evaluation of the Grey-Level Co-Occurrence Matrix method for land-cover classification using SPOT imagery" J. IEEE Transactions on Geoscience and Remote Sensing. 28(4), 1990, pp. 513-519
12. P. Gong, D. J. Marceau, and P. J. Howarth, "A comparison of spatial feature extraction algorithms for land-use classification with SPOT HRV data" J. Remote Sensing of Environment. 40, 1992, pp. 137-151
13. Canada Centre for Remote Sensing - Glossary of remote sensing http://www.ccrs.nrcan.gc.ca/glossary/index_e.php (Accessed: 2010, April 12th).
14. Wikipedia. http://en.wikipedia.org (Accessed: 2010, April 12th).
15. R. O. Duda, P. E. Hart, D. G. Stork (2001). Linear Discriminant Functions. *In Pattern classification (2nd edition)* (pp. 215-270), Wiley, New York.

16. Xiangying Wang; Yixin Zhong; , "Statistical learning theory and state of the art in SVM," In *Proceedings of The Second IEEE International Conference on Cognitive Informatics*, vol., no., pp. 55- 59, 18-20 Aug. 2003.
17. A. Plaza., "Parallel processing of remotely sensed hyperspectral imagery: Full-pixel versus mixed-pixel classification," *Concurrency and Computation: Practice and Experience*, vol. 20, no. 13, pp. 1539–1572, 2008.
18. G. Camps-Valls, T. V. Bandos, and D. Zhou, "Semi-supervised graph based hyperspectral image classification," *IEEE Transaction on Geoscience and Remote Sensing*, vol. 45, no. 10, pp. 3044–3054, 2007.
19. G. Camps-Valls, L. Gomez-Chova, J. Munoz-Mari, J. Vila-Frances, and J. Calpe-Maravilla, "Composite kernel for hyperspectral image classification," *IEEE Geoscience and Remote Sensing Letters*, vol. 3, no. 1, pp. 93–97, January 2006.
20. D. L. Donoho, "Unconditional bases are optimal for data compression and for statistical estimation," *Applied and Computational Harmonic Analysis* 1, pp. 100–115, Dec. 1993.
21. E. J. Candes and D. L. Donoho, "Curvelets, multiresolution representation, and scaling laws," in *Proceedings of SPIE*, 4119, (San Diego), July 2000. Wavelet Applications in Signal and Image Processing VIII.
22. M. N. Do and M. Vetterli, "Pyramidal directional filter banks and curvelets," in *Proc. IEEE Int. Conf. Image Processing*, Oct. 2001.
23. R. H. Bamberger and M. J. T. Smith, "A filter bank for the directional decomposition of images: Theory and design," *IEEE Trans. on Signal Processing* 40, pp. 882–892, Apr. 1992.
24. M. N. Do and M. Vetterli, "Contourlets: A directional multiresolution image representation," in *Proc. IEEE Int. Conf. Image Processing*, 2002.
25. F. Fernandes, R. van Spaendonck, M. Coates, and S. Burrus, "Directional complex-wavelet processing," in *Proceedings of SPIE*, 4119, (San Diego), July 2000. Wavelet Applications in Signal and Image Processing VIII.
26. W. T. Freeman and E. H. Adelson, "The design and use of steerable filters," *IEEE Trans. Patt. Anal. Mach. Intell.* 13, pp. 891–906, Sept. 1991.
27. E. P. Simoncelli, W. T. Freeman, E. H. Adelson, and D. J. Heeger, "Shiftable multi-scale transforms," *IEEE Trans. Inform. Theory* 38, pp. 587–607, Mar. 1992.
28. N. G. Kingsbury, "Image processing with complex wavelets," *Phil. Trans. Royal Society London A*, Sept. 1999.
29. N. G. Kingsbury, "Complex wavelets for shift invariant analysis and filtering of signals," *Applied and Computational Harmonic Analysis* 10, pp. 234–253, May 2002.
30. L. Y. Wei and M. Levoy, "Fast texture synthesis using tree structured vector quantization" *Proc. Intl. Conf. Computer graphics and interactive techniques*, pp. 479-488, 2000.
31. N. Diaz and V. Manian. "Hyperspectral Texture Synthesis by Multiresolution Pyramid Decomposition". *In Proceedings of SPIE: Algorithms and Technologies for Multispectral, Hyperspectral, and Ultraspectral Imagery XV*, Vol. 7334, April 2009.
32. Burt, P. J. and Adelson, E. H., "The Laplacian pyramid as a compact image code", *IEEE Trans. Commun., vol. COM-31*, 532. 1983.

33. A. A. Efros and W. T. Freeman, "Image Quilting for Texture Synthesis and Transfer", SIGGRAPH 01.
34. http://mesh.brown.edu/dlanman/courses/en256/TextureSynthesis.zip. (Accessed: 2010, March 9th).
35. C. Gallaguer and A. Kokaram, Wavelet Based Texture Synthesis, In Proceedings of IEEE International Conference on Image Processing, volume II, 2004, 462-465.
36. Nick Kingsbury, "Shift invariant properties of the dual-tree complex wavelet transform," *in Proceedings of IEEE Conference on Acoustics, Speech and Signal Processing*, March 1999.
37. Wavelet Software at Brooklyn Poly. http://taco.poly.edu/WaveletSoftware/dt3D.html. (Accessed: 2010, March 9th).