A Stochastic Hyperspectral Image Synthesizer for Use in Algorithm Comparison and Verification

Bjørn U. Dideriksen*, Jakob K. Thomsen†, James P. Harris‡, Kristoffer C. Derosche§ and Magnus B. B. Christensen¶
Signal and Information Processing, Department of Electronic Systems, Aalborg University, Aalborg, Denmark
Author Emails: {*bdider16, †jkth16, ‡jharri16, §kderos16, ¶mbbc16 }@student.aau.dk

***Abstract*—Within the field of hyperspectral image processing, the lack of publicly available and accurately labelled datasets may be impairing the comparison of algorithms within the field. This paper describes a stochastic model for hyperspectral image synthesis, which is able to generate labelled datasets. The model generates spatial features while introducing spectral variations in the form of noise and texture on provided material spectra. Attempts are made to calibrate the model's parameters to a set of output metrics. The calibration for the spatial parameter could not be completed as the chosen output metrics did not contain sufficient information about the input parameters. The method of generating textures using Haralick features presented in this paper is, to the best of the authors knowledge, novel.**

I. Introduction

A HyperSpectral Image (HSI) is similar to a conventional RGB image, however it contains additional frequency bands which range from ultraviolet to infrared. HSIs are used in a wide range of fields, such as agricultural monitoring (e.g. crop health) and the food industry (e.g. product quality) [1].

In all areas of science and engineering, testing and verification is crucial and serves the purpose of comparing the performance of newly designed methods against the state of the art. Within the field of HSI algorithm design, this is challenging due to a severe shortage of publicly available labelled datasets, rendering algorithm comparison inherently difficult. Additionally, this creates the risk of the designed algorithms becoming over-fitted to the datasets, thus it could become possible to pick an algorithm that has good performance on these datasets, but bad performance in reality. The authors, as well as others, have identified four to six sets commonly used throughout the field [2].

One way to alleviate the issues caused by the lack of labelled data is to synthesize HSIs with a known ground truth. Models for synthesizing HSIs can be divided into two distinct groups, realistic and simplified. Realistic models attempt to replicate real-world phenomena, often containing many input parameters and having comparatively long computation time, while simplified models replicate the required behaviour using as few parameters as possible, and generally with reduced computation time. Some examples of realistic models would be Digital Imaging and Remote Sensing Image Generation (DIRSIG) [3] and Monte Carlo Scene (MCScene) [4], whereas Automatic Multi-dimensional Image GeneratOr (AMIGO) [5] is an example of a simplified model.

In the AMIGO approach, a three-stage process is proposed. An image canvas is split into individual segments using a snake-based tracing algorithm, spectra of known materials are then applied to each region individually, with spectral variations being introduced last. This model requires six input parameters to create images. On the other hand, both MCScene and DIRSIG are very complex tools that have been under development for many years. DIRSIG is a non-public tool employed by actors such as NASA, the U.S. Department of Defence, and various U.S. intelligence agencies [3], whilst MCScene is a commercial tool.

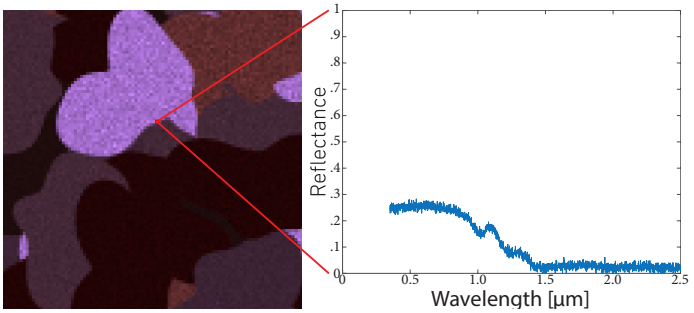


Figure 1. Example of false colour composite of an HSI generated using the proposed method. The spectrum of one the pixels is also shown.

This paper presents a hyperspectral image synthesizer to stochastically generate HSIs to aid researchers with evaluation and comparison of algorithms. A representation of the generated HSIs may be found in Figure 1.

II. Methods and Models

The HSI generation method is a six-step process as shown in Figure 2 on the next page. First a set of shapes is generated. These shapes are converted to shape masks and scattered on canvases. These canvases are then merged into regions. Each region is then assigned a material spectrum, thus creating an HSI. In the last three steps the HSI is downsampled, texture is applied and noise is added. The steps of the generation are explained in the following sections:

- II-A Shape mask generation
- II-B Shape Merging and Partitioning
- II-C Material Selection
- II-D Downsampling
- II-E Texture generation
- II-F Noise model

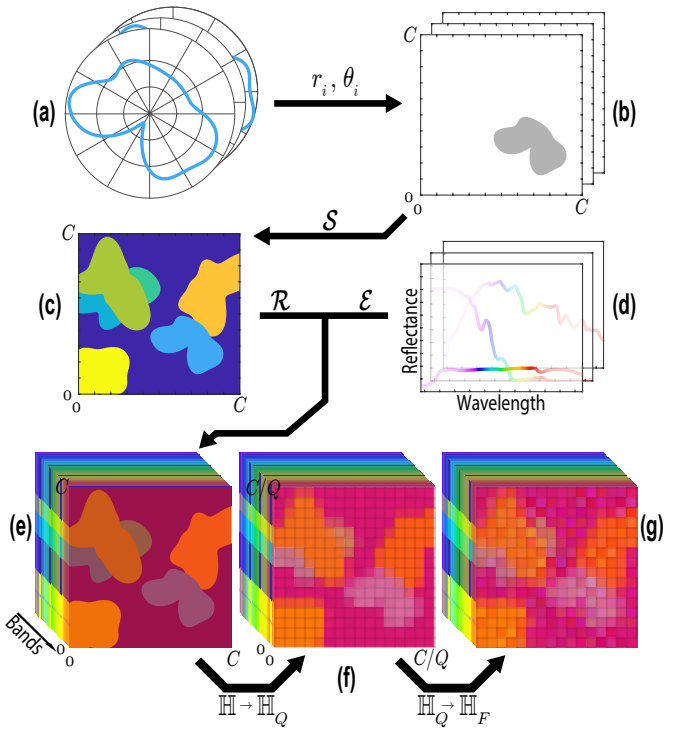


Figure 2. Shows generation steps. (a): Polar shape generated by Fourier method. (b): Shape masks from polar shapes. (c): Regions generated from merging shape masks. (d): Material spectra. (e): HSI generated from region and spectra. (f): Downsampled HSI. (g): Downsampled HSI with texture and noise.

Throughout these sections the model parameters will be explained. These parameters are:

TABLE I
USER-DEFINED PARAMETERS

Overview of user-defined parameters		
Variable	**Meaning**	**Limits**
ρ	Decay constant	$\mathbb{R}[\sim 0.481, \infty)$
r_{mean}	Mean shape radius	$\mathbb{R}[0, \infty)$
k	Number of shapes	$\mathbb{N}[0, \infty)$
C	Canvas size	$\mathbb{N}[Q, \infty)$
Q	Downsampling ratio	$\mathbb{N}[1, C]$, C modulo $Q = 0$
$\boldsymbol{\mathcal{N}}$	Set of number of grey levels	$\{n_i \in \mathbb{N} \mid 1 \leq n_i \leq \infty\}$
$\boldsymbol{\mathcal{P}}$	Set of probabilities	$\{p_i \in \mathbb{R} \mid 0 \leq p_i \leq 1\}$
σ^2	Variance of additive noise	$\mathbb{R}[0, \infty)$

A. Shape mask Generation

The shape mask generation will output a randomly placed shape, on a discrete $C \times C$ canvas, i.e. a Boolean shape mask. The method used to generate shapes is a Fourier-based approach, based on [6], selected due to intuitive relationship between input arguments and the output contour. Initially a radial shape is generated, this is then discretized and inserted into the aforementioned canvas. The user specified parameters relevant for this section are ρ, r_{mean} and C.

Let $\boldsymbol{\mathcal{O}}$ be a sequence of N_p vertices describing the shape. The coordinates of the vertices are denoted in polar form:

$$\boldsymbol{\mathcal{O}} = ((r_1, \theta_1), (r_2, \theta_2), \ldots, (r_{N_p}, \theta_{N_p})) \tag{1}$$

where

r_i specifies radius at angle θ_i
N_P defines the angular resolution
$\theta_i = \frac{2\pi i}{N_p}$

Using the following Fourier series, each shape may be considered a 2π-periodic discrete signal.

$$r_i(\theta_i) = r_{mean} + \sum_{n=1}^{N_H} [A_n \cos(n\theta_i) + B_n \sin(n\theta_i)] \tag{2}$$

where:

N_H number of harmonics
r_{mean} mean radius of shape
A_n,B_n Fourier coefficients of n^{th} harmonic

The Fourier coefficients can be described by Fourier descriptors D_n, where each descriptor has a relation to the final shape. The magnitude of the descriptor is sufficient to describe the shape, allowing the use of a random phase technique [6]. Using this, the Fourier coefficients can be calculated:

$$\begin{aligned} A_n &= D_n \cos(\phi_n) \\ B_n &= D_n \sin(\phi_n) \end{aligned} \tag{3}$$

where:

$$\phi_n \overset{iid}{\sim} \mathcal{U}[-\pi, \pi] \tag{4}$$

To reduce input parameters for shape generation, the descriptors are generated using an exponential decay function:

$$D_n = e^{-\rho n} r_{mean} \tag{5}$$

where:

ρ user specified decay rate

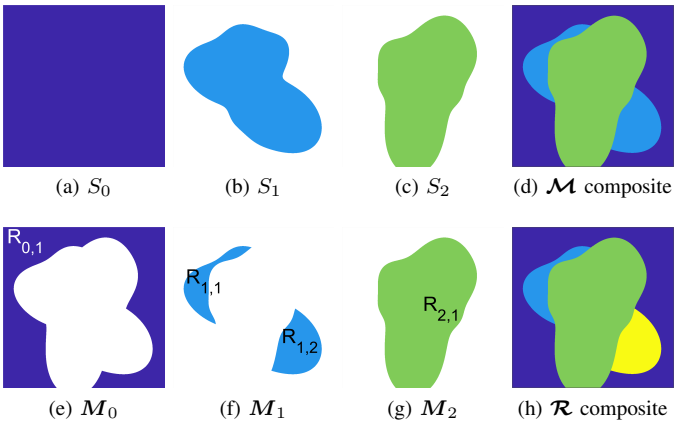
Note that $D_1 \neq 0$ results in a shift of the shape with respect to the origin, hence D_1 is always set to 0 [6].

To produce radially simple shapes (only one positive r at each θ) using this method, it should not be possible for higher harmonics to dominate the function, i.e. $D_1 + D_2 + \cdots + D_{N_H} \leq D_0$. This is ensured when:

$$\sum_{n=2}^{N_H} e^{-\rho n} \leq 1 \tag{6}$$

this constraint stems from the worst-case scenario, where the phases of D_2 to D_{N_H} align and oppose D_0. This sum converges to $\rho = \ln(\frac{\sqrt{5}+1}{2}) \approx 0.4812$ when $N_H \to \infty$.

A discrete contour can then be created using two parameters, ρ and r_{mean}. The contour is scaled by the canvas size C, such that r_{mean} acts as a normalized scaling factor. The contour is now randomly placed in a $C \times C$ Boolean matrix with all zero entries. This is achieved using a uniform binomial point process. All entries interior to the contour are now changed to ones, thus creating a shape mask.



(a) S_0 (b) S_1 (c) S_2 (d) $\boldsymbol{\mathcal{M}}$ composite

(e) $\boldsymbol{M}_0$ (f) $\boldsymbol{M}_1$ (g) $\boldsymbol{M}_2$ (h) $\boldsymbol{\mathcal{R}}$ composite

Figure 3. Figure (a), (b), and (c) illustrates three shape masks. Figure (e), (f), and (g) show the equivalent stacked masks. Their regions are also labelled. Finally, figure (d) and (h) illustrate composite images of $\boldsymbol{\mathcal{M}}$ and $\boldsymbol{\mathcal{R}}$ respectively.

B. Shape Merging and Partitioning

The first step of the synthesis, is to generate a set $\boldsymbol{\mathcal{S}}$ of k shape mask matrices of size $C \times C$, and a matrix S_0 representing the canvas:

$$\boldsymbol{\mathcal{S}} = \{\boldsymbol{S}_0, \boldsymbol{S}_1, \ldots, \boldsymbol{S}_k\} \tag{7}$$

Where $\boldsymbol{S}_0$ is a matrix of logical ones of size $C \times C$. The remaining masks, $\boldsymbol{S}_1, \boldsymbol{S}_2, ..., \boldsymbol{S}_k$, are constructed using the shape generation method described in section II-A.

Following this, the set $\boldsymbol{\mathcal{M}}$ of stacked masks is constructed:

$$\boldsymbol{\mathcal{M}} = \{\boldsymbol{M}_0, \boldsymbol{M}_1, \ldots, \boldsymbol{M}_k\} \tag{8}$$

Each element of $\boldsymbol{\mathcal{M}}$ is derived from the elements in $\boldsymbol{\mathcal{S}}$:

$$\boldsymbol{M}_n = \begin{cases} \boldsymbol{S}_n \oplus \left(\boldsymbol{S}_n \wedge \left(\bigvee_{i=n+1}^{k} \boldsymbol{S}_i \right) \right), & n \neq k \\ \boldsymbol{S}_n, & n = k \end{cases} \tag{9}$$

where:

$\oplus$ element-wise XOR operator
$\wedge$ element-wise AND operator
$\bigvee$ element-wise OR operator of multiple elements

Alternatively, this can be thought of as stacking the shapes on top of each other, with $\boldsymbol{S}_0$ being at the bottom and $\boldsymbol{S}_k$ being at the top. $\boldsymbol{M}_n$ then corresponds to the logical ones of $\boldsymbol{S}_n$ not occluded by the ones of the shapes stacked on top of it. Figure 3 shows a graphical illustration of this. Additionally, the elements in $\boldsymbol{\mathcal{M}}$ have the property that if all elements are XOR'd together the result is an all-ones matrix. This implies that out of all elements of $\boldsymbol{\mathcal{M}}$, there exists one, and only one logical one entry at a given index. Furthermore, in each element of $\boldsymbol{\mathcal{S}}$, it is guaranteed that there only exists one connected region of logical ones. This is not necessarily the case for the elements of $\boldsymbol{\mathcal{M}}$, as the operations applied by (9) can result in the region of $\boldsymbol{S}_n$ being partitioned into multiple disjoint regions. The set of matrices $\boldsymbol{\mathcal{R}}$ containing the disjoint regions in the set $\boldsymbol{\mathcal{M}}$ are then found using binary 8-neighbors connected-component labelling [7]. To expand on this, each matrix $\boldsymbol{M}_n$ contains r_n disjoint regions. Each of these regions are extracted as a Boolean mask $\mathcal{R}_{n,i}$, to form the set of regions:

$$\boldsymbol{\mathcal{R}}_n = \{\boldsymbol{R}_{n,1}, \boldsymbol{R}_{n,2}, \ldots, \boldsymbol{R}_{n,r_n}\} \tag{10}$$

such that:

$$\boldsymbol{M}_n = \bigvee_{i=1}^{r_n} \boldsymbol{R}_{n,i} \tag{11}$$

Thus the set of all regions is:

$$\boldsymbol{\mathcal{R}} = \{\boldsymbol{R}_{0,1}, \ldots, \boldsymbol{R}_{0,r_0}, \ldots, \boldsymbol{R}_{k,1}, \ldots, \boldsymbol{R}_{k,r_k}\} \tag{12}$$

$$= \{\boldsymbol{\mathcal{R}}_0, \boldsymbol{\mathcal{R}}_1, \ldots, \boldsymbol{\mathcal{R}}_k\} \tag{13}$$

The cardinality of $\boldsymbol{\mathcal{R}}$, i.e. the total number of regions can then be expressed as:

$$R = \sum_{n=0}^{k} r_n \tag{14}$$

To improve the readability of future sections, we define a sequence of regions:

$$\boldsymbol{V}_n \coloneqq v(n), \quad n \in \{1, 2, \ldots, R\} \tag{15}$$

where v is bijective function that maps the natural numbers in the range 1 to R to each element of $\boldsymbol{\mathcal{R}}$:

$$v : \{1, 2, \ldots, R\} \to \boldsymbol{\mathcal{R}} \tag{16}$$

C. Material Selection

Following the creation of the region set $\boldsymbol{\mathcal{R}}$, each region needs to be mapped to a material. The set of l materials with b spectral bands, $\boldsymbol{\mathcal{E}}$ is defined as:

$$\boldsymbol{\mathcal{E}} = \{\boldsymbol{e}_1, \boldsymbol{e}_2, ..., \boldsymbol{e}_l\}, \quad \boldsymbol{e}_i \in \mathbb{R}^b \mid 0 \leq_e \boldsymbol{e}_i \leq_e 1 \tag{17}$$

where:

$\leq_e$ element-wise inequality applied on vector elements

Each element $\boldsymbol{e}_i$ is a vector representing the materials reflectance at b wavelengths. The material mapping is done using function $f : \boldsymbol{\mathcal{R}} \to \boldsymbol{\mathcal{E}}$, where f is a deterministic mapping, that maps a region of $\boldsymbol{\mathcal{R}}$ to an arbitrary element of $\boldsymbol{\mathcal{E}}$. Using these definitions, the hyperspectral cube $\mathbb{H}$ is defined as a $C \times C$ array where the elements are material vectors from $\boldsymbol{\mathcal{E}}$:

$$\mathbb{H} = \begin{bmatrix} \boldsymbol{e}(1,1) & \cdots & \boldsymbol{e}(1,C) \\ \vdots & \ddots & \vdots \\ \boldsymbol{e}(C,1) & \cdots & \boldsymbol{e}(C,C) \end{bmatrix}, \boldsymbol{e}(i,j) \in \boldsymbol{\mathcal{E}} \tag{18}$$

where each element $\boldsymbol{e}(i,j)$ is found as:

$$\boldsymbol{e}(i,j) = \sum_{n=1}^{R} I(\boldsymbol{V}_n(i,j)) \cdot f(\boldsymbol{V}_n) \tag{19}$$

$I : \mathbb{B} \to \{0, 1\}$ where $\mathbb{B}$ is the Boolean set, such that I maps the Boolean values to integers. I.e. a logical zero maps to zero, and a logical one maps to one.

These elements represent the pixels of the hyperspectral image.

D. Downsampling

In the generated hyperspectral cube all regions are separated leading to pure pixels containing only one material. This is not representative of real-world hyperspectral images, which are likely to contain multiple materials within the area captured by a pixel. Thus, the spectrum of a hyperspectral pixel may be the result of multiple material spectra being mixed. The pure pixels are therefore mixed to mimic this phenomenon. After considering other mixing methods, such as bilinear- and intimate-mixed models, a linear mixed model was selected, based on its popularity, simplicity and physical interpretability [8]. This is done by downsampling $\mathbb{H}$ by a factor of Q, i.e. $\mathbb{H}$ is split into non overlapping $Q \times Q$ arrays where the elements of the vectors in each array are averaged to form the linearly mixed spectrum vector in $\mathbb{H}_Q$, this is expressed as:

$$\mathbb{H}_Q(i,j) = \sum_{k=d(i)}^{Q\cdot i} \sum_{w=d(j)}^{Q\cdot j} \mathbb{H}(k,w) \cdot \frac{1}{Q^2} \tag{20}$$

where:

$$i,j \in \mathbb{N} \mid 1 \leq i,j \leq \frac{C}{Q}$$
$$d(a) = Q(a-1)+1$$

Note that the limits of i and j requires that Q is a factor of the canvas size C.

E. Texture Generation

The Grey-Level Co-occurrence Matrix (GLCM) is a $N \times N$ symmetric matrix, where N is the number of distinct grey levels in a greyscale image [9]. The GLCM provides insight to texture through the 14 Haralick features [9]. Of the 14 Haralick features, we extract three typical features: *Energy* f_1, *Contrast* f_2 and *Homogeneity* f_5:

$$f_1 = \sum_{i}^{N_G} \sum_{j}^{N} \{p(i,j)\}^2 \tag{21}$$

$$f_2 = \sum_{n=1}^{N-1} n^2 \{ \sum_{\substack{i=1 \\ |i-j|=n}}^{N_G} \sum_{j=1}^{N} p(i,j) \} \tag{22}$$

$$f_5 = \sum_{i}^{N} \sum_{j}^{N} \frac{1}{1+(i-j)^2} p(i,j) \tag{23}$$

where:

$p(i,j)$ is the $(i,j)^{th}$ entry in the normalized GLCM

In the following, the idea of using the GLCM and Haralick features as a synthesis tool for textures will be motivated which leads to a novel texture generation algorithm.

In the normalized GLCM the $(i,j)^{th}$ entry describes the joint probability that Random Variable (r.v.) $\mathcal{I}$ and $\mathcal{J}$ are equal to grey level i and j respectively, i.e. $p(i,j) = \Pr(\mathcal{I}=i, \mathcal{J}=j)$.

Thus, any symmetric bivariate discrete distribution describes a texture with Haralick features that can be computed directly from the distributions Probability Density Function (PDF). An important aspect of this approach is that the bivariate distribution can be calibrated to give desired textures since it fully describes $p(i,j)$. Given desired Haralick features (21) through (23) poses a system of equations with the distribution parameters as unknowns.

This implementation uses a bivariate distribution where the marginal distributions are Independent and Identically Distributed (i.i.d.) binomial r.v.s. This results in a computationally efficient texture generation algorithm, as drawing a sample from the bivariate distribution is the same as drawing two samples from the binomial distribution.

The binomial distribution is parameterised by number of grey levels N and probability p. Each material has a unique texture, thus we form the sets $\boldsymbol{\mathcal{N}} = \{N_1, \ldots, N_l\}$ and $\boldsymbol{\mathcal{P}} = \{p_1, \ldots, p_l\}$, each containing l elements. The set of materials is then mapped to $\boldsymbol{\mathcal{N}}$ and $\boldsymbol{\mathcal{P}}$ using two bijective functions, $g : \boldsymbol{\mathcal{E}} \to \boldsymbol{\mathcal{N}}$ $h : \boldsymbol{\mathcal{E}} \to \boldsymbol{\mathcal{P}}$: The texture is applied on the downsampled hypercube pixelwise:

$$\mathbb{H}_T(i,j) = \mathbb{H}_Q(i,j) \frac{X_{i,j}}{N_{i,j}} \tag{24}$$

where:

$$X_{i,j} \sim \mathcal{B}(N_{i,j}, p_{i,j})$$

The binomial parameters $N_{i,j}$ and $p_{i,j}$ of $X_{i,j}$ are found as:

$$N_{i,j} = \left\lfloor \sum_{k=d(i)}^{Q\cdot i} \sum_{w=d(j)}^{Q\cdot j} g(\mathbb{H}(k,w)) \cdot \frac{1}{Q^2} \right\rceil \tag{25}$$

$$p_{i,j} = \sum_{k=d(i)}^{Q\cdot i} \sum_{w=d(j)}^{Q\cdot j} h(\mathbb{H}(k,w)) \cdot \frac{1}{Q^2} \tag{26}$$

where

$\lfloor a \rceil$ rounds a to the nearest integer

F. Noise model

Noise is modelled as additive white Gaussian noise applied on $\mathbb{H}_T$ to get the final synthesized HSI $\mathbb{H}_F$:

$$\mathbb{H}_F(i,j) = \mathbb{H}_T(i,j) + \boldsymbol{W}, \quad \boldsymbol{W} \in \mathbb{R}^b \tag{27}$$

where:

$$W_n \sim \mathcal{N}(0, \sigma^2), \quad \sigma^2 \in \mathbb{R}_+, \quad n \in \mathbb{N} \mid 1 \leq n \leq b \tag{28}$$

G. Model Calibration

To better allow the user to select appropriate input values, the model can be calibrated on a set of features. These features can be selected or calculated from user-defined reference images whose features are to be imitated. These output features, or metrics, can be decided based on the individual application, however for this paper five metrics are chosen. The five metrics are calculated for each region. They are defined in TABLE II on the following page. If multiple regions are present within

TABLE II
OUTPUT METRICS, GIVEN BY [10] EXCEPT # OF REGIONS

Title	
Metric	**Description**
Area	Number of pixels in region
Perimeter	Distance around border of region (8-directional)
Extent	Proportion of bounding box filled by region
Eccentricity	Ratio between minor and major axes of region
# of Regions	Number of regions generated, found in (14) on page 3

a single pixel, as may be caused by downsampling, the pixel is mapped to the region with the highest presence within the pixel.

To gain full knowledge of the relationship between the output metrics and the input parameters the entire parameter space must be analysed. However, as this is an infeasible task only a subset of the parameter space defined by ρ, r_{mean} and k is considered. In this analysis ρ was varied linearly in 11 discrete steps between 0.481 and 1.481. k was varied logarithmically (base 1.5) in 14 steps between 4 and 657. r_{mean} was varied logarithmically (base 1.5) in 13 steps between 0.01 and 1.3. For each parameter combination, 192 images were synthesized. Finally, the five output metrics for each image were found.

In figure 4 a subset of the calculated output metrics is shown. Each blue dot represents the test metric value for a given region. As hundreds of regions were generated for each parameter combination, only a small subset of the calculated output metrics are displayed to maintain visual clarity. The line in red represents the average metric value over all regions generated using the same input parameters.

From Figure 4, a clear relationship is present between all of the output metrics and r_{mean}. The same cannot be said about the other input parameters. For k a clear relationship is only present for the number of regions metric, and perhaps the area metric. No apparent correlation is seen between ρ and the output metrics, as it appears that none of them contain any information about ρ.

Similar trends are found over the remaining parameter space. Thus, before the spatial properties of the model can effectively be calibrated, more informative metrics need to be found. This should be the primary aim of future research of the proposed method.

The texture calibration relies on the system of equations that the Haralick features form when the PDF of the bivariate distribution used for texture generation is known. In general, to avoid overdetermined systems of equations, the number of Haralick features should equal the amount of model parameters. In this implementation the bivariate distribution is a binomial bivariate distribution resulting from two i.i.d. binomial marginal distributions. Analytic solutions to the system of equations were sought, however none were found and instead a numerical solution that approximates the intersection between the equations was obtained. As it is only an approximation the solution may not be exact, though it may be improved at the cost of computation time. However, even then there is

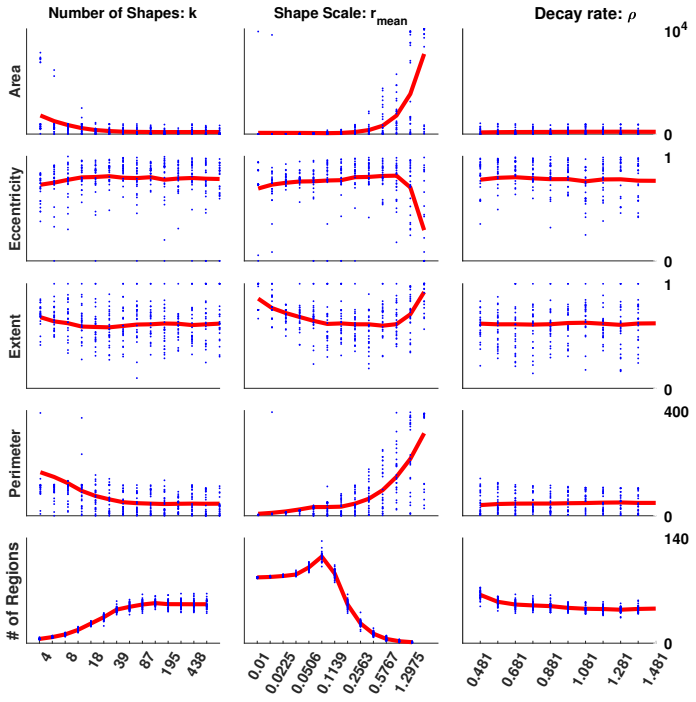


Figure 4. Scatter plot between metrics (rows) against input parameters (columns), with mean (red). In the first column k is varied while $r_{mean} = 0.1709$, and $\rho = 0.681$. In the second column $k = 87$ and $\rho = 0.681$. In the third column $k = 87$ and $r_{mean} = 0.1709$

no guarantee that a solution exists. To calibrate the texture generation, the desired Haralick features must be specified. These can be stated explicitly, or a texture may be supplied from which the Haralick features can be derived.

III. TEXTURE CALIBRATION TESTING AND RESULTS

A. Testing

The texture generation may be tested separately as it does not affect any aforementioned shape metric. In particular, the texture generation with calibration will be tested. The calibration is done on two Haralick features at a time because this implementation relies on a binomial distribution, which has two unknown parameters.

B. Results

In Figure 5 on the next page, 9 generated textures can be seen and compared to the texture they were calibrated with respect to. Three textures are generated for each input texture because three Haralick features are derived but only two Haralick features can be calibrated at a time. This gives three possible combinations of Haralick features to calibrate with. As a result of this the feature which is not considered in the calibration is not directly controlled.

C. Results Discussion

As it can be seen from Figure 5 on the following page the texture generation and calibration work well as none of the generated textures differ much in the features they were

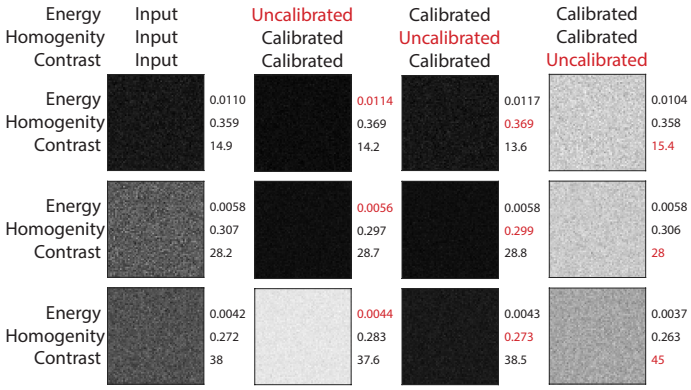


Figure 5. Calibrated texture generation. Each row shows three generated textures calibrated to reproduce two of the Haralick features present in the input texture of the leftmost column. Red numbers indicate the Haralick feature that was ignored in the calibration.

calibrated with respect to. An interesting result is that even though the calibration works well, the resulting textures are quite easily distinguished from the input and even each other. This shows that the calibration is good at extracting exactly the features that are specified. Thus, if visually identical textures are wanted, Haralick features which capture visual information should be used. If all three parameters are of importance, any discrete distribution with three or more parameters could be used to likely obtain even better solutions.

IV. Discussion

The proposed synthesizer can generate a range of hyperspectral scenery using only eight input parameters. Upon inspection, the generated regions appear curvilinear implying that the method may be better suited for replicating 'round' regions, such as those found in rural landscapes, than for imitating highly angular regions. This is largely due to the chosen shape generation method. By modifying this it is theorised that the general method can be used to model a wider range of scenery. In addition, the general methodology allows for rapid synthesis of large amount of HSIs. This was demonstrated when sweeping over the parameter space as described in Section II-G on page 4. In this sweep a 48-core machine was used to synthesize over 384 thousand images in only six hours. This implementation was not optimized for speed and could likely be improved.

The proposed synthesizer also includes a method to generate textures, which to the best of the authors knowledge is novel. It has been shown that a number of Haralick features equal to the amount of model parameters can be reliably calibrated using proper model assumptions. This implementation has assumed independence between the identical marginal distributions. This results in the GLCM becoming directionally invariant, which makes the Haralick feature analysis very straightforward, however it also results in "grainy" textures. Having dependant marginal distributions allows for more complex textures but requires more extensive analysis when interpreting the Haralick features.

Like with the texture parameters, efforts have been made to create a calibration model for the spatial parameters. This was not completed, and as such it would be an ideal topic for future research. Among others, this should include investigations into metrics containing information about ρ. Another area of interest is the evaluation of the effectiveness of the proposed method over a range of hyperspectral algorithms. This research is critical if the proposed synthesizer is to be used and trusted by the hyper spectral research community.

V. Conclusion

A stochastic model has been developed, which synthesizes HSIs based on eight input parameters. The proposed synthesizer generates spatial features, while introducing spectral variations in the form of noise and texture on provided material spectra. In addition to the synthesizer, it was desired to relate a set measurable output metrics with the defined input parameters.

To the best of the author's knowledge, a novel method for texture generation was developed, allowing selection of desired Haralick features.

Similarly, five output metrics describing the spatial generation were defined. However, clear relationships were only found between the output metrics and r_{mean}. If the other spatial parameters are to be inferred, more informative output metrics are needed.

The proposed synthesizer has not been used in conjunction with any hyperspectral image algorithms, thus its merits as a tool cannot be established.

Acknowledgements

The authors would like to thank Troels Pedersen and Ramoni Ojekunle Adeogun for their insight and supervision of this project.

References

[1] P. Ghamisi, N. Yokoya, J. Li, W. Liao, S. Liu, J. Plaza, B. Rasti, and A. Plaza, "Advances in hyperspectral image and signal processing: A comprehensive overview of the state of the art," 2017.

[2] (Apr. 2014). Hyperspectral remote sensing scenes. ehu.eus/ccwintco/index.php?title=Hyperspectral_Remote_Sensing_Scenes.

[3] Rochester Institute of Technology. (Sep. 2019). Digital imaging and remote sensing image generation. dirsig.cis.rit.edu/.

[4] R. Sundberg, S. Richtsmeier, and R. Haren, "Monte carlo based hyperspectral scene simulation," 2010.

[5] B. Priego and R. Duroy, "Amigo: A tool for the generation of synthetic hyperspectral images," 2018.

[6] G. Mollon and J. Zhao, "Fourier–voronoi-based generation of realistic samples for discrete modelling of granular materials," 2012.

[7] L. He, X. Ren, Q. Gao, X. Zhao, B. Yao, and Y. Chao, "The connected-component labeling problem: A review of state-of-the-art algorithms," 2017.

[8] R. Heylen, M. Parente, and P. Gader, "A review of nonlinear hyperspectral unmixing methods," 2014.

[9] K. S. Robert M. Haralick and I. Dinstein, "Textural features for image classification," 1973.

[10] MathWorks Inc. (Dec. 2019). Regionprops documentation. mathworks.com/help/images/ref/regionprops.html.