MANIFOLD-BASED ANALYSIS OF NATURAL STOCHASTIC TEXTURES WITH APPLICATION IN TEXTURE SYNTHESIS

Ido Zachevsky and Yehoshua Y. Zeevi

ido@technion.ac.il, zeevi@ee.technion.ac.il
Technion - Israel Institute of Technology
Haifa 3200003, Israel

ABSTRACT

Embedding textured images in manifolds reveals latent information regarding texture structure and allows useful analysis of these high dimensional images in a low dimensional space. We present a framework for analysis and synthesis of natural stochastic textures (NST) which constitute an important subset of textures that are modelled as realizations of random processes. The randomness of NST differentiates them from other types of images and requires a dedicated method for analysis and synthesis. We demonstrate several applications of this framework. The first is synthesis of new types of NST. The second is NST analysis, reaffirming our previous findings regarding the fundamental properties of NST, and showing that they emerge naturally in the latent parameter space. Finally, we show the advantage of producing a manifold representation with intrinsic geometry.

***Index Terms*—** Natural stochastic textures, manifolds, texture synthesis

1. INTRODUCTION

Natural stochastic textures (NST) are a subset of textures, considered to be realizations of random processes, unlike the deterministic structure of regular textures or other contents of images [1]. In the latter case, an image can be semantically defined via a number of latent parameters. For instance, an image of a digit (e.g. MNIST) can be defined via its literal figure and other properties such as its roundness, boldness, skewness, and similar properties. Given these semantic characteristics, the image is deterministic.

NST are different in that their latent parameters, such as Gaussianity, fractal dimension and coherence [2], define only the distribution of the texture; sampled textures with the same latent parameters will span a range of pixel-wise realizations. As a simple example, consider the fractional Brownian motion (fBm) [3] with $H = 0.5$. H is the single deterministic latent parameter, while there are infinitely many realizations that satisfy this parameter.

This research was supported in part by the Minerva Center and the OMEK Consortium.

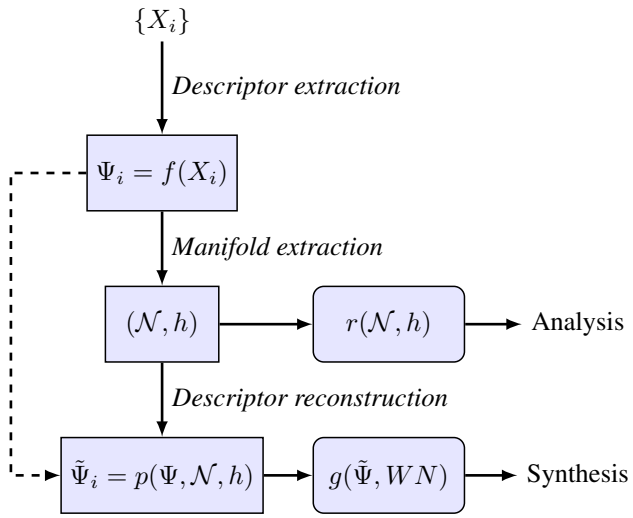


Fig. 1: Embedding framework – A high-level overview: A set of images $\{X_i\}$ is fed into a descriptor extraction function f. A manifold $\mathcal{N}$ is extracted with local geometry metric, h. Next, either data analysis (r) or synthesis (g) can be performed. Synthesis is performed by extracting the modified descriptor $\tilde{\Psi}_i$ (using information from Ψ_i and the manifold $(\mathcal{N}, h)$) for image i with white noise WN to obtain the synthesized image.

In this work we propose a framework for embedding NST in a manifold, based on a graph representation with intrinsic geometry (Fig. 1). Subsequent to presentation of the framework, we highlight its possible applications. These include a method for synthesis of new types of textures, and texture analysis based on the intrinsic geometry of the manifold. We also show that manually-introduced texture features arise naturally from the data when analyzed using the proposed framework.

1.1. Related work

Texture synthesis was successfully performed by accurately describing relations between wavelet coefficients [4]. In this

method, textures are decomposed using the steerable wavelet transform and dedicated, manually derived function are used to learn the joint constraints of adjacent wavelet scales and orientations. Then, white noise is fed to the inverse system to generate some type of new texture.

Inspired by this method, convolutional neural networks (CNNs) were recently used to automatically learn these features, by using only a small subset of hand-tailored functions [5]. A high dimensional set of Gram matrices is found to describe each texture. Similarly to such previous methods, white noise is fed in our method into the learned system to create textures of a realistic appearance.

Variational autoencoders (VAEs) are also useful in manifold extraction and image synthesis [6]. However, they are less suitable for random textures, such as NST, inasmuch as VAE-based synthesis assumes the image is normally distributed around a mean given by a deterministic function (usually expressed via a neural network) of its descriptor.

2. A FRAMEWORK FOR ANALYSIS AND SYNTHESIS OF NST

The framework consists of several stages (Fig. 1). First, a descriptor is extracted for each image. Then, the descriptor undergoes dimensionality reduction (based on a collection of images) by means of PCA and metric assignment. The resulting manifold can then be used for analysis, by inspecting its properties, or for synthesis. In synthesis, the processed descriptor is projected back into the original descriptor space, using inverse PCA, and is implemented with a white noise source to generate new texture using a texture synthesis algorithm.

2.1. Dimensionality reduction

The first step in the proposed framework, *descriptor extraction* (Fig. 1), is based on [5], which uses a pre-trained VGG-19 CNN network to extract features. This algorithm provides the descriptor extraction function, f, and the synthesis function, g.

The CNN-based synthesis framework used for descriptor extraction yields $\sim 852,000$ parameters for each $224 \times 224 \times 3$ dimensional texture. In a low-dimensional representation, this number should be limited to a very low number of parameters. The purpose of this dimensionality reduction is twofold: first, we would like to extract the most meaningful set of latent parameters of each texture. Second, we wish to use the reduced dimensional space as a manifold, by means of which new textures can be interpolated from existing points on the manifold.

In the CNN-based texture synthesis, the texture descriptors (per image) are based on filter response matrices F_i, in which every column is a vectorized response, matching a certain filter in the ith layer. We observe that much of the texture information is encapsulated in three components: the response mean, variance and singular values. Let $G^i = F_i^T F_i$ denote a Gram matrix for layer i. Let $\mu_i(k)$ and $\sigma_i^2(k)$, defined as

$$\mu_i(k) = \frac{1}{s_i} \sum_l F_i(l,k)$$
$$\sigma_i^2(k) = \frac{1}{s_i} \sum_l \left[F_i(l,k) - \mu_i(k)\right]^2,$$

where s_i is the length of each filter, denote the mean and variance of each filter's response, respectively. The singular values are obtained by means of the singular value decomposition (SVD) of G^i. We have the following connection between the SVD of G^i and the SVD of F^i:

$$F_i = U_i S_i V_i^T,\ F_i^T F_i = V_i S_i U_i^T U_i S_i V_i^T$$
$$G^i = V_i S_i^2 V_i^T,$$

where U_i and V_i are orthogonal matrices, and S_i is a matrix with nonzero singular values being clustered along the main diagonal. The Gram matrix G^i is a function of a right orthogonal matrix, V_i, and the singular values of F_i. We find that V_i has a limited role in texture representation, especially when similar textures are considered. We, therefore, extract λ_i, the set of singular values on the diagonal of S_i, as the third component for texture representation. In the sequel we show that textures can indeed be generated using this set of limited parameters: $\theta_i \triangleq \left(\mu_i, \sigma_i^2, \lambda_i\right)$, for each Gram matrix.

The dimensions of these parameters are of the order of the number of filters for each layer, whereas the original descriptor dimension was of the order of the number of filters squared.

The final representation, Ψ_i (Fig. 1), is given by performing PCA on each of the parameters in θ_i, based on a dataset of analyzed textures.

2.2. Synthesis of new textures by interpolation in the PCA space

We synthesize new textures using the PCA space of compact texture representation, expressed by the response means and variances. These parameters correspond to the deterministic quantities of the texture, whereas the white noise used in texture generation corresponds to random content. This method allows us to synthesize various random textures with the same set of deterministic parameters (expressed by θ_i), which is an advantage over VAE-based synthesis.

Using the PCA as a dimensionality reduction method, we can use existing data points, but we can also interpolate new points on the manifold by using a few existing analyzed images. Intuitively, we treat the texture space as a smooth and continuous manifold, on which we have access to only a small subset of images that are samples on this manifold. Interpolation of new points corresponds to sampling of new points on the manifold.

The interpolation method is as follows: the texture descriptor contains a set of 16 vectors θ_i per image (this number is dictated by the descriptor extraction method [5]); let I_a and I_b denote two images with corresponding parameters $\{\theta_i^a\}$ and $\{\theta_i^b\}$, respectively. Let $\alpha \in [0,1]^{16}$ denote a weight vector. The interpolated parameters are given by

$$\begin{aligned} \mu_i' &= \alpha \circ \mu_i^a + (1-\alpha) \circ \mu_i^b \\ v_i' &= \alpha \circ v_i^a + (1-\alpha) \circ v_i^b, \end{aligned} \tag{1}$$

where μ_i^j is the mean of some filter response of image j in the PCA space, μ_i' is the new texture and $v \equiv \sigma^2$ denotes variance. The rest of the parameters are extracted from the first image. This is shown to be reasonable due to the semantic closeness of images considered for interpolation. We note that this interpolation method is linear and does not take into account the intrinsic geometry of the manifold; it can be further improved as by our method for obtaining the manifold's intrinsic geometry, presented in the sequel.

2.3. Intrinsic geometry of the manifold

The proposed framework (Fig. 1) implements a PCA-based manifold extraction. By using PCA it is assumed that the data structure resides in some linear subspace. It is useful due to the stability and invertibility properties associated with it, but it does not necessarily reflect the intrinsic geometry of the data.

To introduce local geometry to this manifold, we use the so-called pushforward metric ([7]§2), h. To calculate it we represent the data points as a weighted graph where each weight is determined via a Gaussian kernel:

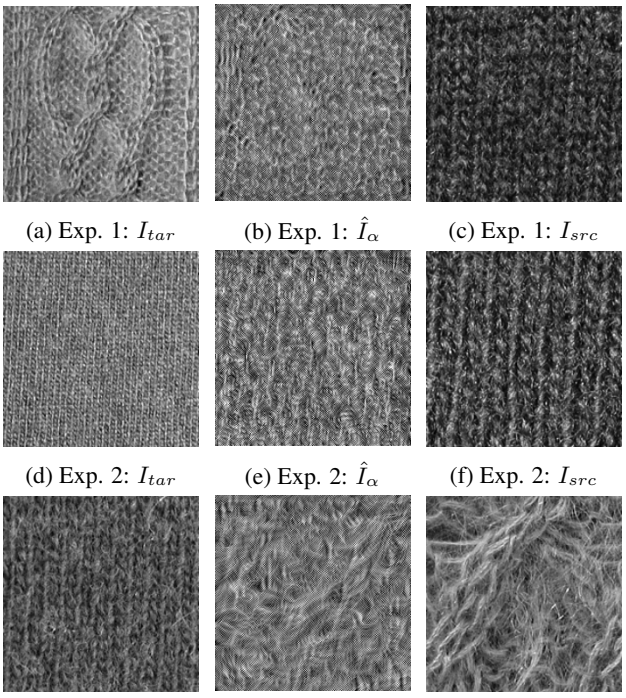
$$W_{i,j} = \exp(-\left\|p_i - p_j\right\|^2 / \epsilon), \tag{2}$$

where p_i is a coordinate in the embedding space. This graph is then used to construct the graph Laplacian. The metric, h, is calculated for each point, thereby obtaining the local geometry. [For more details see [7]§2, Algorithm 2.]

Equipped with h, the manifold $(\mathcal{N}, h)$ is isometric with the Riemannian manifold representing the data. We can then measure distances using intrinsic geometry which represents better the distortions between data points, compared with Euclidean distances. We exemplify this phenomenon in the sequel.

3. EXPERIMENTS

In all the experiments we use the KTH-TIPS2 dataset [8]. This dataset contains 11 texture classes, each containing 4 samples of the same type of material. We implement the CNN-based texture synthesis algorithm described in [5] using the Keras framework on top of TensorFlow [9]. We use the same network by importing the trained weights provided online by the authors. Code as well as further demonstrations is available online [10].



(a) Exp. 1: I_{tar} (b) Exp. 1: $\hat{I}_\alpha$ (c) Exp. 1: I_{src}

(d) Exp. 2: I_{tar} (e) Exp. 2: $\hat{I}_\alpha$ (f) Exp. 2: I_{src}

(g) Exp. 3: I_{tar} (h) Exp. 3: $\hat{I}_\alpha$ (i) Exp. 3: I_{src}

Fig. 2: Synthesis of a new texture. First row: (a, b, c) depict I_{tar}, $\hat{I}_\alpha$ and I_{src} , respectively. Rows 2–3 depict similar images for different experiments. Observe the new textures, $\hat{I}_\alpha$, which combine visual properties from I_{tar} and I_{src} to generate new type of naturally-appearing texture.

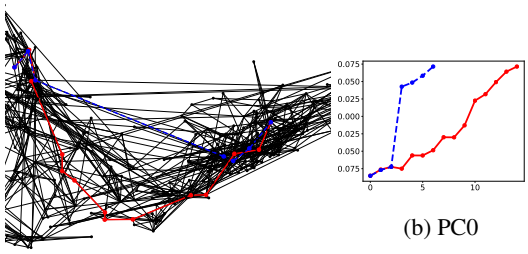
3.1. Synthesis of new textures

We synthesize new textures by mixing parameters as discussed earlier (1), by using two images as $I_a = I_{src}$ and $I_b = I_{tar}$ with some predetermined mixing vector α. The resulting images, $\hat{I}_\alpha$, shown in Fig. 2, indicate that using this method one can create new textures that do not appear identical to the originating textures, but nevertheless appear natural. A value of $\alpha = 0.8 \cdot \underline{1}$ is used in all experiments.

3.2. Intrinsic geometry and image paths

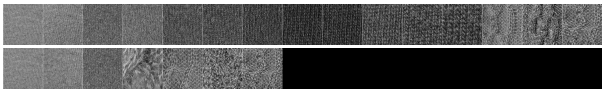
In this set of experiments, we produce the manifold using one class of the KTH-TIPS2 dataset (*wool*) to obtain a single cluster. Each class contains 4 sub-classes of different samples, which introduce intra-class variability. We build the graph using $\sqrt{\epsilon} = 0.01$ (2). No significant dependency w.r.t this parameter is observed. The intrinsic and embedding dimensions are 3 and 6, respectively [7]. The data used for manifold extraction is an aggregated mean vector for all filter responses.

To demonstrate the use of geometry, we seek the path between two arbitrary images on the manifold. In the graph-based representation, each node has at most 3 neighbors, obtained by disconnecting edges with sufficiently small weights.



(a) Manifold graph

(b) PC0



(c)

Fig. 3: Comparison of intrinsic vs. Euclidean geometry. (a) A graph depicting the manifold of one class of KTH-TIPS2. A Euclidean path (dashed blue) and a geodesic path (red) are shown between two points. (b) The first principal component (with 63% explained variance) progression for the Euclidean path (dashed blue) and the intrinsic path (red). (c) The sets of images along the intrinsic path (top row) and Euclidean path (bottom row, padded with blank images).

The graph distances are calculated using the Dijkstra algorithm.

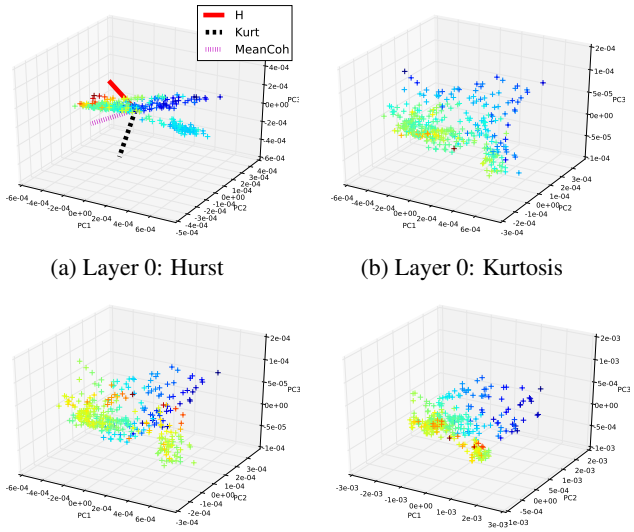
Comparing the paths (Fig. 3 depicts an excerpt of the manifold graph), we observe that using the intrinsic geometry, the path proceeds within the manifold structure (Fig. 3a), whereas in the Euclidean case the path extends itself beyond the manifold. Inspecting the first principal component, which accounts for 63% of the explained variance, we observe that the manifold-intrinsic path progresses more gradually compared with the Euclidean path (Fig. 3b). Finally, we observe that the images in each path (Fig. 3c; top row depicts the intrinsic path and the bottom row depicts the Euclidean path) vary more gradually in terms of texture content in the intrinsic case.

3.3. Interpretation of learned parameters

We briefly recite main NST properties relevant to our application. These are discussed in more detail elsewhere [3, 11, 12].

The 3 main properties (or features) of NST are Gaussianity, self-similarity and coherence. These properties, expressed by means of the kurtosis, the slope of the variance of the signal increments, and the mean of the log-coherence [12], are simple to calculate, and have been used in various applications for analysis and/or processing (e.g. [3, 13, 12]). We have previously used these features in various applications [14, 2, 15, 11].

In our experiment (Fig. 4), we plot the transformed data using the first 3 principal components, with color gradients



(a) Layer 0: Hurst

(b) Layer 0: Kurtosis

(c) Layer 0: Mean coh.

(d) Layer 1: Mean coh.

Fig. 4: Analysis of texture latent feature space. (a), (b), (c) and (d) depict the data points colored according to H, Kurtosis, and mean coherence for layers 0 and 1, respectively. (a) depicts also the linear trend of the first 3 features.

matching one of the NST features. We use the mean vector, μ, for this analysis. If these features are indeed highlighted already in this low dimensional space, we expect to find some structure. Otherwise, the distribution should appear to be random.

Inspecting the smooth gradients, we observe (Fig. 4) that the features correlate with a linear combination of the first 3 principal components. The regression plots (e.g. Fig. 4a) show that the three features span the 3D space. This result shows that these 3 features are indeed fundamental to texture analysis, as they are extracted directly from the data.

4. DISCUSSION

Synthesis of new texture is challenging. Our preliminary results indicate that one can generate new textures by incorporating latent properties that are extracted from existing textures represented in a learned parameter space. Unlike other texture synthesis methods, we perform this task based on a small number of meaningful parameters. While we use simple mixtures, better results may be obtained by interpolating new points that are intrinsic to the structure of the learned parameter manifold.

The knowledge of the intrinsic geometry of a system allows the definition of better distance measures. These, in turn, can improve analysis algorithms such as clustering. In an ongoing work, we proposed an intrinsic-metric-based clustering algorithm. Augmenting PCA with local metric information allows the harnessing of such geometries.

5. REFERENCES

[1] Wen-Chieh Lin, James Hays, Chenyu Wu, Vivek Kwatra, and Yanxi Liu, "A comparison study of four texture synthesis algorithms on regular and near-regular textures," Tech. Rep., Carnegie Mellon University, 2004.

[2] Ido Zachevsky and Yehoshua Y Zeevi, "Statistics of Natural Stochastic Textures and Their Application in Image Denoising," *IEEE Trans. Image Process.*, vol. 25, no. 5, pp. 2130–2145, 2016.

[3] Beatrice Pesquet-Popescu and Jacques L Vehel, "Stochastic fractal models for image processing," *IEEE Signal Process. Mag.*, vol. 19, no. 5, pp. 48– 62, sep 2002.

[4] Javier Portilla and Eero P Simoncelli, "A Parametric Texture Model Based on Joint Statistics of Complex Wavelet Coefficients," *Int. J. Comput. Vis.*, vol. 40, no. 1, pp. 49–71, 2000.

[5] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge, "Texture Synthesis Using Convolutional Neural Networks," *Neural Image Process. Syst.*, pp. 1–10, 2015.

[6] Diederik P Kingma and Max Welling, "Auto-Encoding Variational Bayes," *arXiv Prepr. arXiv1312.6114*, pp. 1–14, 2013.

[7] Dc Perrault-Joncas, "Learning and Manifolds: Leveraging the Intrinsic Geometry," 2013.

[8] Barbara Caputo, Eric Hayman, and P. Mallikarjuna, "Class-specific material categorisation," *Proc. IEEE Int. Conf. Comput. Vis.*, vol. II, pp. 1597–1604, 2005.

[9] Francois Chollet, "Keras," https://github.com/fchollet/keras, 2015.

[10] Ido Zachevsky, "NST Manifold project website: https://idozach.github.io/manifolds/," .

[11] Ido Zachevsky and Yehoshua Y. Zeevi, "Model-based Color Natural Stochastic Textures Processing and Classification," in *IEEE Glob.*, Orlando, FL, dec 2015, pp. 1357—-1361.

[12] Joachim Weickert, *Anisotropic Diffusion in Image Processing*, Teubner Stuttgart, 1998.

[13] L M Kaplan, "Extended fractal analysis for texture classification and segmentation.," *IEEE Trans. Image Process.*, vol. 8, no. 11, pp. 1572–85, jan 1999.

[14] Ido Zachevsky and Yehoshua Y. Zeevi, "On the Statistics of Natural Stochastic Textures and their Application in Image Processing," in *IEEE Int. Conf. Acoust. Speech Signal Process.*, Florence, Italy, may 2014, pp. 5829–5833.

[15] Ido Zachevsky and Yehoshua Y Zeevi, "Superresolution of self-similar textures," *CCIT Report. EE Pub, Tech. Isr. Inst. Technol.*, vol. 838, no. 1795, 2013.