Received February 6, 2020, accepted March 2, 2020, date of publication April 1, 2020, date of current version April 9, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2981410

Texture Mixing by Interpolating Deep Statistics via Gaussian Models

ZHUCUN XUE AND ZIMING WANG
State Key Laboratory of Information Engineering in Surveying, Mapping and Remote Sensing (LIESMARS), Wuhan University, Wuhan 430079, China
Corresponding author: Ziming Wang (wangzm@whu.edu.cn)

ABSTRACT Recently, enthusiastic studies have devoted to texture synthesis using deep neural networks, because these networks excel at handling complex patterns in images. In these models, second-order statistics, such as Gram matrix, are used to describe textures. Although these models have achieved promising results, the structure of their parametric space is still unclear. Consequently, it is difficult to use them to mix textures. This paper addresses the texture mixing problem by using a Gaussian scheme to interpolate deep statistics computed from deep neural networks. More precisely, we first reveal that the statistics used in existing deep models can be unified using a stationary Gaussian scheme. We then present a novel algorithm to mix these statistics by interpolating between Gaussian models using optimal transport. We further apply our scheme to Neural Style Transfer, where we can create mixed styles. The experiments demonstrate that our method outperforms a number of baselines. Because all the computations are implemented in closed forms, our mixing algorithm adds only negligible time to the original texture synthesis procedure.

INDEX TERMS Texture modeling, texture mixing, Gaussian models, deep neural networks.

I. INTRODUCTION

Texture mixing is the process of generating new texture images that possess *averaged* visual characteristics of a given set of exemplars [1]–[5]. It can provide visually pleasing interpolations of difference textures, therefore, has numerous applications in computer vision and graphics [4], [6]. Besides, the ability to create smoothly morphing textures is regarded as a criterion for "good" texture synthesis algorithms [7] [8].

In the sense that a texture can be modeled by a set of statistics depicting the visual properties of its samples [9]–[11], texture mixing involves "averaging" the corresponding set of statistics. For copy-based texture synthesis methods [12], [13], textures can be mixed by combining pixels from multiple inputs using well-designed procedures such as in [4] or the *patch match* scheme [14]. These methods handle complex and geometric textures satisfactorily, but they tend to produce verbatim patterns and it is difficult to understand the mixing process. In contrast, statistical parametric texture methods [11], [15], [16] are more principled, and their parameters are better understood, although they are often not as good at handling structured textures. Moreover, with parametric texture models, the mixing of textures can be computed feasibly and more easily by "averaging" the corresponding set of parameters [1], [2], [5], [17].

A recent breakthrough in texture modeling involves the use of deep convolutional neural networks (CNNs) [18]–[22] for texture representation. This approach enables us, using parametric models, to synthesize comparable or better textures containing complex patterns than copy-based methods. Under this framework, researchers also cast the problem of style transfer into texture transfer [20], [23]. However, due to the complex structure of the parametric space of deep CNNs [18], [21], [23], it is still unclear how to mix textures or styles with these models.

In this paper, we address the problem of mixing textures using deep CNNs. More precisely, after studying existing deep texture models [18]–[22], we discover that the second-order statistics (e.g. Gram matrix, correlation matrix and their variations) used in these methods can be represented as continuous functions of a stationary Gaussian model, so the mixing of these statistics is reduced to the interpolation of Gaussian models, which is known to have a closed form solution. Therefore, we present a simple and efficient scheme illustrated in Fig 1, for mixing the statistics of deep CNNs via interpolation of Gaussian models. We further apply our scheme to neural-style morphing, where we can interpolate between different styles. We also demonstrate that our mixing algorithm is fully compatible with feed-forward CNNs

The associate editor coordinating the review of this manuscript and approving it for publication was Vicente Alarcon-Aquino.

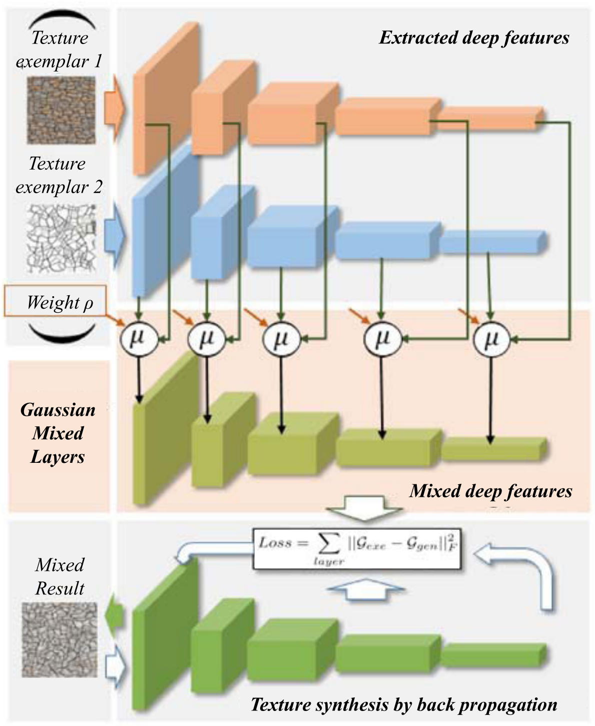


FIGURE 1. **The proposed texture mixing scheme. (Top) Two texture exemplars are passed through the CNN. (Middle) The outputs of selected layers are mixed using a Gaussian model. (Bottom) The Gram matrices of the mixed outputs are used as constraints to generate mixed textures.**

[20], [24], where mixed textures or stylish photos with mixed styles are generated in a fast forward pass. Experiments demonstrate that our method produces better or at least comparable results than the state-of-the-art methods. It is also worth noticing that our mixing algorithm adds only negligible time to the original texture synthesis procedure, because all mixing computations are in closed forms.

The rest of this paper is organized as follows: Section II briefly reviews the related work. Section III formulates the texture mixing problem. Section IV presents our scheme for mixing deep statistics with Gaussian models. Section V provides all the implementation details. Section VI compares the proposed methods with the state-of-the-arts and analyzes the experimental results. Section VII finally draws some conclusion remarks.

II. RELATED WORK

Exemplar-based texture synthesis is the basis of our work, of which the goal is to generate new texture samples from a given texture exemplar [25]. The works on texture synthesis can be roughly categorized into non-parametric models such as copy-based (also known as *patch-based*) methods [12] and statistic parametric models [11]. Patch-based models copy pixels or patches directly from the exemplar to the synthesized samples [12], [13]. These approaches can generate high fidelity textures but sometimes produce verbatim patterns, *i.e.* using the same parts of the exemplar repeatedly in the results. In contrast, statistical parametric methods aim to find parametric representations of textures, which allow more control over the synthesis processes. Portilla and Simoncelli [11] used wavelet and pyramid decomposition to build a parametric texture model, which can synthesize many nature textures, even those containing geometric patterns. Stationary Gaussian model [5], [16], [26]–[30] is an efficient texture model, as new textures can be synthesized fast in Fourier domain. Gatys *et al*. [18] used a CNN for texture synthesis. Their method achieved good performance over a large scope of nature textures, but it failed to synthesize textures with non-local structures and sometimes suffered from degraded quality [22]. In order to overcome this difficulty, later works added extra penalty terms such as Fourier spectrum [19] and correlation matrix [21] to Gatys' model [23]. Furthermore, Li *et al*. [22] proposed to use centred Gram matrix instead of Gram matrix to improve the quality of outputs.

Gatys' neural texture model [18] was later adapted to a neural style transfer algorithm [23], which sought to transfer the "style" of the input image while keeping its "content" fixed. Although Gatys' style transfer algorithm [23] can produce high quality stylish photos, its computational cost was prohibitively high. To accelerate this time consuming procedure, Johnson *et al*. [24] proposed a perceptual loss function and a transformation network, which can generate textures and stylish photos in a forward manner. Ulyanov *et al*. [20] further proposed to use instance normalization to improve the quality of outputs. Later, Li *et al*. [22] and Dumoulin *et al*. [31] proposed new network structures that can learn multiple styles in one network.

In the past decades, tremendous studies have devoted to texture mixing. It aims at generating an "averaged" texture from several texture exemplars. Some patch-based texture synthesis algorithms [4], [14] can be naturally extended to texture mixing by considering multiple inputs. In terms of statistic parametric models, texture mixing corresponds to averaging statistics from different exemplars. This "averaging" procedure has been investigated for different texture models. Bar-Joseph *et al*. [17] proposed to use wavelet and a tree structure to model and mix textures. Peyré [1] proposed to use "grouplet" for synthesizing and mixing locally parallel textures. Rabin *et al*. [2] used sliced optimal transport to mix textures. Mixing stationary Gaussian texture has also been studied in terms of optimal transport [5]. Although these algorithms can generate homogeneous mixed textures, they have difficulties in mixing structured textures. Recently, Yu *et al*. [32] proposed a deep model called MixNet for mixing nature textures. Although this model can handle nature textures relatively well, it can not be used for unseen images and generally takes several days to train.

It is worth noticing that not all texture models are able to mix textures. For example, even in the prominent work of Portilla and Simoncelli's [11], it is unclear how to mix textures satisfactorily. The deep texture models [18] also suffered from this problem, as linear interpolation of Gram matrices only results in low quality mixtures [22].

III. PROBLEM FORMULATIONS

Denote $I \in \mathbb{R}^{\Omega\times d}$ as an image with d channels defined on the grid $\Omega = \{0, \ldots, M-1\} \times \{0, \ldots N-1\}$. In particular, $d = 1$ for grey-scale images and $d = 3$ for color images. For each pixel $p \in \Omega$, the value $I(p)$ is a d-dimensional vector, and for each channel $c \in \{0, .., d-1\}$ at location $p \in \Omega$, the value $I(p, c)$ is a real scalar.

A. EXEMPLAR-BASED TEXTURE SYNTHESIS WITH CNNs

Given a texture exemplar I_{exp}, the aim of exemplar-based texture synthesis is to produce new texture samples I_{syn} that are as similar as possible to I_{exp} regarding certain visual/perceptual measurements [11]. For instance, Zhu *et al.* [10] argued that I_{syn} and I_{ex} are equivalent on statistical feature sets,

$$\{\mathbf{F}^{(\ell_1)}_{syn}, \ldots, \mathbf{F}^{(\ell_k)}_{syn}\} \sim \{\mathbf{F}^{(\ell_1)}_{exp}, \ldots, \mathbf{F}^{(\ell_k)}_{exp}\},$$

where $\mathbf{F}_\times := \{\mathbf{F}^{(\ell_1)}_\times, \ldots, \mathbf{F}^{(\ell_k)}_\times\} = \mathscr{F} \circ I_\times$ are the sets of texture features extracted from $I_\times$ by a texture model $\mathscr{F}$. These models can be filter banks [10], wavelets [11] or Markovian models [12]. The image I_{syn} can thus be generated by feature projection [10], [33]. A survey of exemplar-based texture synthesis was recently provided in [25].

In this paper, we are interested in exemplar-based texture models using deep CNNs [18], [21], [23], because of their abilities to synthesize textures with complex structures. This type of methods utilize a pre-learned deep CNN $\mathscr{F}_{\text{CNN}}$ for texture description, and generate new textures I_{syn} by matching deep features such as Gram matrix. More precisely, one can initialize I_{syn} with a random noise and pursue an optimal output by minimizing the following objective:

$$\sum_{\ell=\ell_1}^{\ell_k} \|\mathcal{G}(\mathbf{F}^{(\ell)}_{syn}) - \mathcal{G}(\mathbf{F}^{(\ell)}_{exp})\|^2_F, \tag{1}$$

where $\mathcal{G}()$ is the Gram measure of matrix, and $\|\cdot\|_F$ denotes the Frobenius norm. The minimization problem in Eqn. (1) can be solved using back-propagation [18].

B. EXEMPLAR-BASED TEXTURE MIXING WITH CNNs

Given two input texture exemplars I_{exp_0} and I_{exp_1}, exemplar-based texture mixing aims to generate new textures whose perceptual properties are drawn from both the inputs. Denoting the deep features of the two inputs as $\mathbf{F}_{exp_0} = \mathscr{F}_{\text{CNN}} \circ I_{exp_0}$ and $\mathbf{F}_{exp_1} = \mathscr{F}_{\text{CNN}} \circ I_{exp_1}$ respectively, the mixing of I_{exp_0} and I_{exp_1} with ratio $\rho \in [0, 1]$ is to obtain I_{syn}, such that

$$\mathbf{F}_{syn} \sim \{\rho\mathbf{F}_{exp_0}, (1-\rho)\mathbf{F}_{exp_1}\},$$

where $\mathbf{F}_{syn} = \mathscr{F}_{\text{CNN}} \circ I_{syn}$. A straightforward solution is to pursue I_{syn} by minimizing

$$\sum_{\ell=\ell_1}^{\ell_k} \|\rho\,\mathcal{G}(\mathbf{F}^{(\ell)}_{exp_0}) + (1-\rho)\,\mathcal{G}(\mathbf{F}^{(\ell)}_{exp_1}) - \mathcal{G}(\mathbf{F}^{(\ell)}_{syn})\|^2_F, \tag{2}$$

which actually finds an I_{syn} with linear interpolation of the Gram matrices. As we shall discuss in Section VI, this mixing often produces results with conspicuous artifacts.

In what follows, we will develop a more effective algorithm to interpolate the deep CNN features for mixing textures.

IV. DEEP TEXTURE MIXING WITH GAUSSIAN MODELS

Pioneered by Gatys *et al.* [18], several studies have addressed texture synthesis with deep CNNs [19]–[22]. In this section, we first reveal that all the statistic measures used in these works can be unified into a stationary Gaussian scheme. We then show that this unified scheme enables us to mix textures by interpolating deep features through a simple and fast procedure.

A. GAUSSIAN SCHEME FOR DEEP TEXTURE SYNTHESIS

Given a deep feature $\mathbf{F} \in \mathbb{R}^{U\times k}$ with U pixels and k channels, deep texture models [18], [19], [21], [22] need to compute statistics of $\mathbf{F}$ as the textural signatures, and then synthesize new texture samples by matching the signatures. The main discover of this section is that all these statistics can be represented using an unified Gaussian model. Before approaching to the main result, we first recall the definitions of these statistics and the stationary Gaussian model as follows.

1) GRAM MATRIX $\mathcal{G}$

The Gram matrix $\mathcal{G} \in \mathbb{R}^{k\times k}$ was first used in Gatys' model [18]. It is defined as:

$$\mathcal{G}(i,j) = \frac{1}{|U|}\sum_{p\in U} \mathbf{F}(p,i)\mathbf{F}(p,j), \quad 1 \le i,j \le k. \tag{3}$$

2) CENTRED GRAM MATRIX $\bar{\mathcal{G}}$

Li *et al.* [22] suggested to use centred Gram matrix $\bar{\mathcal{G}}$ instead of Gram matrix $\mathcal{G}$ for better synthesis results:

$$\bar{\mathcal{G}}(i,j) = \frac{1}{|U|}\sum_{p\in U} \Big(\mathbf{F}(p,i) - m_i\Big)\Big(\mathbf{F}(p,j) - m_j\Big), \tag{4}$$

where $1 \le i, j \le k$. $m \in \mathbb{R}^k$ is the mean vector of $\mathbf{F}$:

$$m = \frac{1}{|U|}\sum_{p\in U} \mathbf{F}(p). \tag{5}$$

3) CORRELATION $\mathcal{S}'$

Sendik and Cohen-Or [21] proposed to use correlation $\mathcal{S}'$ to synthesize non-local textures, and reported the state-of-the-art results. Their deep correlation $\mathcal{S} \in \mathbb{R}^{U\times k}$ is defined as

$$\mathcal{S}'(p,n) = \sum_{p'\in U} w(p)\mathbf{F}(p',n)\mathbf{F}(p+p',n), \tag{6}$$

in which $p = (i, j)$ is the offset vector, and $i \in [-Q/2, Q/2]$ and $j \in [-M/2, M/2]$. w is the relative weight defined by

$$w(i,j) = ((Q - |i|)(M - |j|))^{-1}. \tag{7}$$

4) MODIFIED CORRELATION $\mathcal{S}$

By assuming periodic boundary, all the relative weights $w(p)$ in Eqn. (6) become the same, correlation $\mathcal{S}'$ therefore has a much simpler form:

$$\mathcal{S}(p,n) = \frac{1}{|U|}\sum_{p'\in U}\mathbf{F}(p',n)\mathbf{F}(p+p',n), \tag{8}$$

where $1 \leq n \leq k$, $p \in U$. We called $\mathcal{S}$ defined in Eqn. (8) modified correlations. For simplicity, in the rest of this paper, we only consider $\mathcal{S}$ instead of $\mathcal{S}'$. But as we will see in the experiment section, these two correlation matrices produce similar results.

5) SPECTRUM $\mathcal{F}$

The Fourier spectrum $\mathcal{F}$ has been considered in [19] for synthesizing non-local textures. Formally, $\mathcal{F}$ is defined as:

$$\mathcal{F} = |\hat{\mathbf{F}}|, \tag{9}$$

where $\hat{}$ denotes the Fourier transformation.

6) STATIONARY GAUSSIAN MODEL μ

The Gaussian models have been explored for modelling stationary textures [5], [16]. A stationary Gaussian model $\mu(m, \mathcal{C})$ consists of a mean vector $m \in \mathbb{R}^k$ and a covariance matrix $\mathcal{C} \in \mathbb{R}^{U\times k\times k}$.

$$\mathcal{C}(p,i,j) = \frac{1}{|U|}\sum_{p'\in U}\Big(\mathbf{F}(p',i) - m_i\Big)\Big(\mathbf{F}(p+p',j) - m_j\Big), \tag{10}$$

$$m = \frac{1}{|U|}\sum_{p\in U}\mathbf{F}(p). \tag{11}$$

where $p \in U,\ 1 \leq i,\ j \leq k$.

Although the four statistics considered above *i.e.*, $\mathcal{G}$, $\bar{\mathcal{G}}$, $\mathcal{S}$ and $\mathcal{F}$, seem irrelevant, the following proposition suggests that they can all be represented by the stationary Gaussian model μ.

Proposition 1: Given feature maps $\mathbf{F} \in \mathbb{R}^{U\times k}$, *its Gram matrix* $\mathcal{G}$, *centred Gram matrix* $\bar{\mathcal{G}}$, *correlation* $\mathcal{S}$ *and spectrum* $\mathcal{F}$ *can be derived from a stationary Gaussian model* $\mu(m, \mathcal{C})$*:*

$$\mathcal{G} = \mathcal{C}(0) + mm^T, \tag{12}$$

$$\bar{\mathcal{G}} = \mathcal{C}(0), \tag{13}$$

$$\forall p \in U,\ \mathcal{S}(p) = diag(\mathcal{C}(p)) + m \odot m, \tag{14}$$

$$\forall \omega \in U,\ \mathcal{F}(\omega) = (|U||\hat{\mathcal{S}}(\omega)|)^{\frac{1}{2}}. \tag{15}$$

where ^{T} *is the transpose operator and* $\odot$ *denotes the component-wise product.*

The derivations of Eqn. (12) (13) (14) are straightforward. Eqn. (15) holds because the correlation $\mathcal{S}$ is the auto-correlation of $\mathbf{F}$, which leads to

$$\hat{\mathcal{S}}(\omega) = \frac{1}{|U|}\hat{\mathbf{F}}(\omega) \odot \hat{\mathbf{F}}(\omega)^*,$$

where $*$ denotes the conjugate transpose.

It is interesting to notice the similarity between $\mathcal{G}$ and $\bar{\mathcal{G}}$. As shown in Eqn. (12) and Eqn. (13), these two statistics both contain the orderless part of $\mathcal{C}$, *i.e.*, $\mathcal{C}(0)$. As a result, they can not encode ordered elements and can not be used for synthesizing non-local textures. On the contrary, $\mathcal{S}(p)$ contain $diag(\mathcal{C}(p))$, which is in the the ordered part of $\mathcal{C}$. Therefore, $\mathcal{S}(p)$ is sensitive to ordered elements in textures and can be used for synthesizing non-local textures. It is also worth noticing that Eqn. (15) can explain the similar effects of $\mathcal{S}$ and $\mathcal{F}$ in synthesizing non-local structures, because these two statistics can be derived from each other.

B. INTERPOLATING DEEP STATISTICS VIA GAUSSIAN MODEL

In this section, we discuss the interpolation of Gram matrix $\mathcal{G}$ for simplicity, but all discussions apply for other deep statistics.

In order to synthesize mixed textures in Gatys' model [18], we need to find an intermediate Gram matrix to represent the average of the two textures. Formally, given two feature maps F_0 and F_1 corresponding to two exemplar textures, we seek to find a continuous function $\mathcal{G}(\rho)$, $\rho \in [0, 1]$, such that $\mathcal{G}(\rho) = \mathcal{G}(F_\rho)$ when $\rho = 0, 1$. Even though the solution is not unique, it is generally difficult to find a natural and effective interpolation method. For instance, the simple linear interpolation $\rho\mathcal{G}_0 + (1-\rho)\mathcal{G}_1$ satisfies this requirement. However, it performs poorly in texture mixing (see experiments in Fig. 4 and 6), because linear interpolation of Gram matrices does not necessarily result in Gram matrices.

The significance of Proposition. 1 is that it provides a method to interpolate Gram matrix. In other words, if we are able to interpolate stationary Gaussian models, Proposition. 1 directly enables us to calculate interpolated Gram matrices. Specifically, given Gaussian models μ_0 and μ_1 corresponding to F_0 and F_1 respectively, if we can find a continuous function $\mu(\rho) = (m_\rho, \mathcal{C}_\rho)$, $\rho \in [0, 1]$ such that $\mu(\rho) = \mu_\rho$ when $\rho = 0, 1$. Eqn. (12) in Proposition. 1 asserts that an interpolated Gram matrix can be generated as follows:

$$\mathcal{G}(\rho) = \mathcal{C}_\rho(0) + m_\rho m_\rho^T. \tag{16}$$

Namely, the interpolation of Gram matrices is reduced to the interpolation of Gaussian models.

Several ways to interpolate Gaussian models have been investigated, such as linear interpolation, Fisher-Rao interpolation [34] and optimal transport interpolation [5]. However, linear interpolations of Gaussian models are no longer Gaussian, and no explicit formula is known for high dimensional Fisher-Rao interpolation. Alternatively, optimal transport interpolation provides a closed-form solution to the problem and the interpolated $\mu(\rho)$ remains Gaussian [5].

According to [5], the interpolated $\mu(\rho)$ can be calculated in two steps. First, we calculate the interpolated feature map $\hat{\mathbf{F}}_\rho$ in in Fourier domain:

$$\forall \rho \in [0, 1],\ \hat{\mathbf{F}}_\rho = (1-\rho)\hat{\mathbf{F}}_0 + \rho\hat{\mathbf{G}}, \tag{17}$$

$$\forall w,\ \hat{\mathbf{G}}(w) = \hat{\mathbf{F}}_1(w)\frac{\hat{\mathbf{F}}_1(w)^*\hat{\mathbf{F}}_0(w)}{|\hat{\mathbf{F}}_1(w)^*\hat{\mathbf{F}}_0(w)|}, \tag{18}$$

where * represents the conjugate transpose. Then, the interpolated $\mu(\rho)$ is the corresponding stationary Gaussian model of $\hat{\mathbf{F}}_\rho$, *i.e.*, $\mu(\rho)$ can be calculated using Eqn. (10) and Eqn. (11).

Now, we are able to calculate interpolated Gram matrix by combining Proposition. 1, Eqn. (17) and Eqn.(18). Note that the practical algorithm for interpolating these deep statistics does not require computing $\mu(\rho)$, because $\mathcal{G}_\rho$ can be derived directly from $\mathbf{F}_\rho$. In summary, we have the following proposition for interpolating deep statistics. A conceptual illustration is given in Fig. 1.

Proposition 2: Given feature maps $\mathbf{F}_0, \mathbf{F}_1 \in \mathbb{R}^{U\times k}$, *and a relative weight* $\rho \in [0,1]$, *the interpolated Gram matrix* $\mathcal{G}_\rho$, *centred Gram matrix* $\bar{\mathcal{G}}_\rho$, *correlation* $\mathcal{S}_\rho$ *and spectrum* $\mathcal{F}_\rho$ *can be written as follows*:

$$\forall \rho \in [0,1],\ \hat{\mathbf{F}}_\rho = (1-\rho)\hat{\mathbf{F}}_0 + \rho\hat{\mathbf{G}}, \tag{19}$$

$$\forall w,\ \hat{\mathbf{G}}(w) = \hat{\mathbf{F}}_1(w)\frac{\hat{\mathbf{F}}_1(w)^*\hat{\mathbf{F}}_0(w)}{|\hat{\mathbf{F}}_1(w)^*\hat{\mathbf{F}}_0(w)|}, \tag{20}$$

$$\mathcal{G}_\rho(i,j) = \frac{1}{|U|}\sum_{p\in U}\mathbf{F}_\rho(p,i)\mathbf{F}_\rho(p,j), \tag{21}$$

$$\bar{\mathcal{G}}_\rho(i,j) = \frac{1}{|U|}\sum_{p\in U}\Big(\mathbf{F}_\rho(p,i)-m_{\rho i}\Big)\Big(\mathbf{F}_\rho(p,j)-m_{\rho j}\Big), \tag{22}$$

$$\mathcal{S}_\rho(p,g) = \frac{1}{|U|}\sum_{p'\in U}\mathbf{F}_\rho(p',g)\mathbf{F}_\rho(p+p',g), \tag{23}$$

$$\mathcal{F}_\rho(\omega) = (|U||\hat{\mathcal{S}}_\rho(\omega)|)^{\frac{1}{2}}. \tag{24}$$

where $p, \omega \in U,\ 1 \le i,\ j,\ g \le k$.

V. IMPLEMENTATION DETAILS

This section presents the implementation details of our Gaussian scheme for texture mixing. We follow the pipeline proposed by Gatys *et al.* [18]. However, our algorithm can be combined with a forward generator as in TextureNet [20] without difficulty.

By considering "styles" as textures, our algorithm can be further used to synthesize stylish photos with mixed styles. For simplicity, we only consider mixing two styles/textures in our algorithm, but our algorithm can be extended to mixing more styles/textures easily.

A. TEXTURE MIXING

Thanks to Proposition. 2, we can calculate interpolated Gram matrices efficiently. In this section, we present an algorithm to synthesize mixed textures using interpolated Gram matrix.

Specifically, given two input exemplars I_{exp_0} and I_{exp_1}, we seek to synthesize a mixed texture I_{syn} with relative weight $\rho \in [0,1]$, where ρ control the similarity between I_{syn} and I_{exp_0} or I_{exp_1}. To be precise, I_{syn} should be similar to I_{exp_0} when $\rho = 0$, and be similar to I_{exp_1} when $\rho = 1$. When $\rho \in (0,1)$, I_{syn} should be the "average" of I_{exp_0} and I_{exp_1}.

First, we feed I_{exp_0} and I_{exp_1} to a pre-trained deep CNN $\mathscr{F}_{\text{CNN}}$, and record their feature maps $\{\mathbf{F}_i^{(\ell_1)}, \mathbf{F}_i^{(\ell_2)}\ldots, \mathbf{F}_i^{(\ell_k)}\}$ at selected layers $\{\ell_1, \ell_2, \ldots, \ell_k\}$, where $i = 0, 1$. Then, we calculate interpolated feature maps $\mathbf{F}_\rho^{(\ell)}$ in each layer using Eqn. (17) (18). Mixed deep statistics including $\mathcal{G}_\rho^{(\ell)}$, $\bar{\mathcal{G}}_\rho^{(\ell)}$, $\mathcal{F}_\rho^{(\ell)}$ and $\mathcal{S}_\rho^{(\ell)}$ can be calculated using Proposition. 2. Finally, to generate new textures I_{syn}, I_{syn} is initialized as Gaussian noise and fed into $\mathscr{F}_{\text{CNN}}$. Deep statistics such as Gram matrices $\{\mathcal{G}_{syn}^{(\ell_1)}, \ldots, \mathcal{G}_{syn}^{(\ell_k)}\}$ at selected layers are calculated. Back-propagation is used to match the deep statistics of I_{syn} with the mixed statistics. Specifically, for stationary textures, we only consider Gram matrix:

$$\sum_{\ell=\ell_1}^{\ell_k}\|\mathcal{G}_\rho^{(\ell)} - \mathcal{G}_{syn}^{(\ell)}\|_F^2, \tag{25}$$

For non-local textures, we need to add $\mathcal{S}_\rho^{(\ell)}$ (or $\mathcal{F}_\rho^{(\ell)}$) as the extra penalty term:

$$\sum_{\ell=\ell_1}^{\ell_k}\|\mathcal{G}_\rho^{(\ell)} - \mathcal{G}_{syn}^{(\ell)}\|_F^2 + \lambda\sum_{\ell=\ell_1}^{\ell_k}\|\mathcal{S}_\rho^{(\ell)} - \mathcal{S}_{syn}^{(\ell)}\|_F^2, \tag{26}$$

where λ is the regularization weights for non-local structures. In addition, it is straightforward to use a generator net $\mathscr{T}$ for fast synthesis as in TextureNet [20], *i.e.*, minimize

$$\sum_{\ell=\ell_1}^{\ell_k}\|\mathcal{G}_\rho^{(\ell)} - \mathcal{G}(\mathscr{T}(z))^{(\ell)}\|_F^2, \tag{27}$$

where z is a random noise, and $\mathscr{T}(z)$ is a texture generated by $\mathscr{T}$.

Our Gaussian scheme for texture mixing is summarized in Algorithm. 1.

Algorithm 1 Deep Texture Mixing With Gaussian Models

```
Input: exemplar textures $I_{exp_0}, I_{exp_1}$, $\rho \in [0,1]$, a pre-trained
    CNN $\mathscr{F}_{\text{CNN}}$.
Output: a sample of mixed texture $I_{syn}$.
  for $i = 0, 1$ do
    $\{\mathbf{F}_i^{(\ell_1)}, \mathbf{F}_i^{(\ell_2)}\ldots, \mathbf{F}_i^{(\ell_k)}\} \leftarrow \mathscr{F}_{\text{CNN}} \circ I_{exp_i}$.
  end for
  for $\ell = \{\ell_1, \ell_2, \ldots, \ell_k\}$ do
    $\mathbf{F}_\rho^{(\ell)} \leftarrow Mixing(\mathbf{F}_0^{(\ell)}, \mathbf{F}_1^{(\ell)}, \rho)$ using Eqn. (17) (18);
    Compute $\mathcal{G}_\rho^{(\ell)}$, $\bar{\mathcal{G}}_\rho^{(\ell)}$, $\mathcal{F}_\rho^{(\ell)}$ and $\mathcal{S}_\rho^{(\ell)}$ as needed using
  Proposition. 2.
  end for
  Generate $I_{syn}$ by minimizing Eqn. (25), Eqn. (26) or
  Eqn. (27).
```

B. STYLES MORPHING

Our scheme can also be applied to morphing the styles of two images. Given a content image I_{ori} and two style images I_{sty_0} and I_{sty_1}. The goal of *styles morphing* is to transfer the style

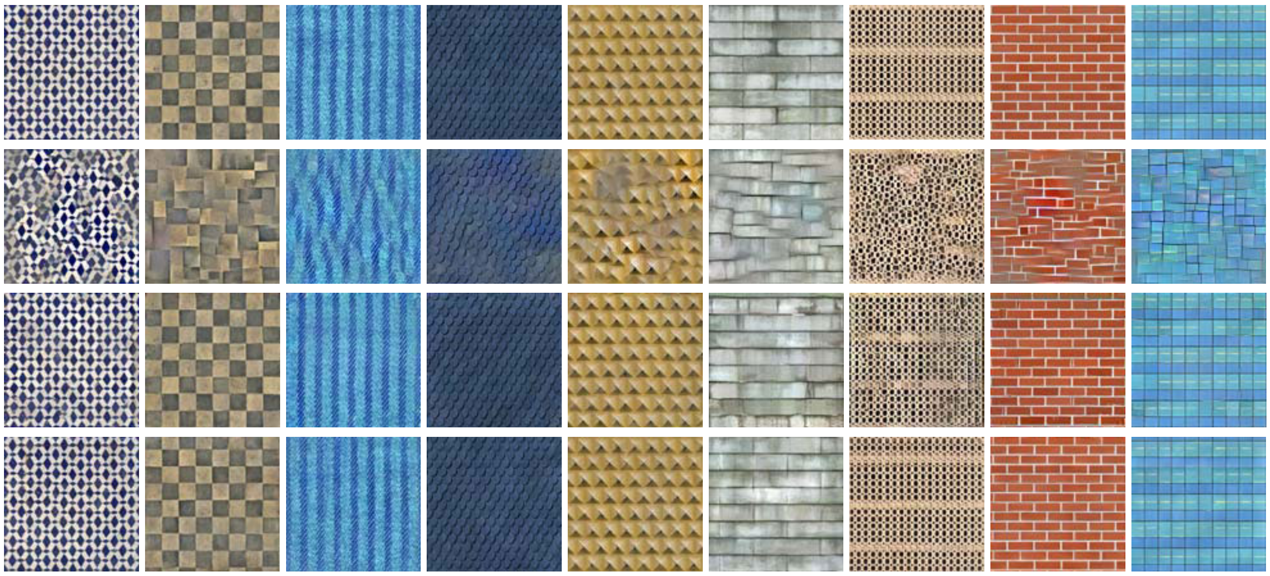


FIGURE 2. Comparison of using different correlation matrices. From top to bottom, 1*st* row: input images; 2*nd* row: results achieved using Gram matrix $\mathcal{G}$ [18]; 3*rd* row: results achieved using correlation matrix $\mathcal{S}'$ [21]; 4*th* row: results achieved using our modified correlation matrix $\mathcal{S}$ described in Eqn. (14). Observe that both $\mathcal{S}$ and $\mathcal{S}'$ can synthesize non-local textures faithfully, but Gram matrix can not.

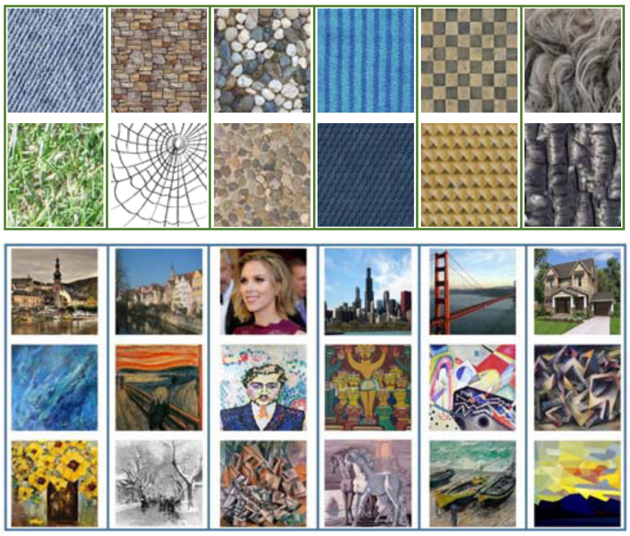


FIGURE 3. Top: exemplar textures used in texture mixing experiments. Bottom: photos and style images used in style morphing experiments.

of I_{ori} to the interpolation of I_{sty_0} and I_{sty_1}, while keeping the content of I_{ori} fixed.

Similar to the texture model, Gram matrices at selected layers are used to parametrized the style. Denote $\mathbf{F}_{ori}$, $\mathbf{F}_{syn}$ as the feature maps of I_{ori} and synthesized image. Let $\mathcal{G}_{sty_0}$, $\mathcal{G}_{sty_1}$ and $\mathcal{G}_{syn}$ be the Gram matrices of the style images and the synthesized image. Similar to Algorithm. 1, after interpolating $\mathbf{F}_{sty_0}$ and $\mathbf{F}_{sty_1}$ using Eqn. (17) and Eqn. (18), mixed Gram matrix $\mathcal{G}_\rho^{(\ell)}$ can be computed using Eqns. (21). The stylish image I_{syn} can be generated by minimizing

$$\sum_{\ell=\ell_1}^{\ell_k} \|\mathcal{G}_\rho^{(\ell)} - \mathcal{G}_{syn}^{(\ell)}\|_F^2 + \alpha \|\mathbf{F}_{ori}^{(\ell)} - \mathbf{F}_{syn}^{(\ell)}\|_F^2. \tag{28}$$

where $\mathcal{G}_\rho^{(\ell)}$ is an interpolated Gram matrix of $\mathcal{G}_{sty_0}$ and $\mathcal{G}_{sty_1}$. α is a parameter to control the degree of style bending. It is also possible to generate stylish images using a forward generator $\mathscr{T}$, which seeks to transform the content image I_{ori} to a stylish photo $\mathscr{T}(I_{ori})$. $\mathscr{T}$ can be trained by minimizing

$$\sum_{\ell=\ell_1}^{\ell_k} \|\mathcal{G}_\rho^{(\ell)} - \mathcal{G}(\mathscr{T}(I_{ori}))^{(\ell)}\|_F^2 + \alpha \|\mathbf{F}_{ori}^{(\ell)} - \mathbf{F}(\mathscr{T}(I_{ori}))^{(\ell)}\|_F^2. \tag{29}$$

VI. EXPERIMENTAL RESULTS AND ANALYSIS

In this section, we first evaluate the performance of our modified correlation matrix $\mathcal{S}$ on non-local textures by comparing with Sendik's results [21]. Then we present our texture mixing results for both non-local textures and stationary textures. Our results are compared with other sate-of-the-art algorithms. Finally, we apply our algorithm to style transfer and compare our results with other style morphing algorithms.

For all experiments, we use VGG-19 [35] network pre-trained on ImageNet dataset [36]. 10 values of relative weight ρ are used, *i.e.*, we let $\rho = \frac{0}{9}, \frac{1}{9}, \frac{2}{9}, \frac{3}{9}, \frac{4}{9}, \frac{5}{9}, \frac{6}{9}, \frac{7}{9}, \frac{8}{9}, \frac{9}{9}$. In texture mixing experiments, input images are down-sampled to (256, 256) or (128, 128) depending on their original sizes. Pairs of texture and style exemplars are shown in Fig. 3, which are from the DTD dataset [37] or collected from Internet. All input images are initialized as white Gaussian noise. In style morphing experiments, input images are down-sampled to (256, 256). We use the L-BFGS algorithm [38] for optimization. All experimental results are available at http://captain.whu.edu.cn/TexMixDeepG.

A. COMPARISON OF CORRELATIONS MATRICES $\mathcal{S}$ AND $\mathcal{S}'$

Because our mixing algorithm involves the use of modified correlation $\mathcal{S}$, which is slightly different from the original definition of Sendik's correlation $\mathcal{S}'$, we need to verify that these two statistics actually produce similar results. We follow the same experimental settings as in [21]: we use layers `pool1`, `pool2`, `pool3`, `pool4` for Gram loss, and use layer `pool2` for correlation loss. Additional total variation loss is added on layer `conv1`.

Fig. 2 presents the comparison of $\mathcal{S}$ and $\mathcal{S}'$. One can see that Gram matrix fails to capture non-local structures, and sometimes causes blurry effects. On the contrary, both $\mathcal{S}$ and $\mathcal{S}'$ can re-produce non-local structures faithfully. The results of $\mathcal{S}$ and $\mathcal{S}'$ are comparable for most of exemplars. For some exemplars (the brick texture), $\mathcal{S}$ produces even better results as the results are less noisy or preserve the structures better.

B. COMPARISONS WITH STATE-OF-THE-ART TEXTURE MIXING METHODS

This section evaluates our texture mixing method by comparing it with several other texture mixing algorithms. We show that our method is able to mix textures effectively in all scenarios. The baselines are listed as follows:

- **GaussTexton** [5]: A simple and fast texture mixing algorithm based on stationary Gaussian models.
- **Image Melding** [4]: An efficient texture mixing method based on patch match algorithm.
- **Diversified Feed-forward Networks (DFN)** [22]: Texture mixing using linear combinations of different ''selectors''. In all experiments, we used the pretrained model provided by the authors.
- **Linear Interpolation Algorithm (LIA)**: Linear interpolation of Gram matrix as given in Eqn. (2).
- **Our scheme + TextureNet**: The combination of TextureNet [20] and our method. This method can synthesis mixed textures in a fast forward pass.
- **LIA + TextureNet**: The combination of TextureNet [20] and linear interpolation algorithm, *i.e.* the Gram matrices used in TextureNet is linear interpolated as in Eqn. (2).
- **TexMixer** [32]: Texture mixing by interpolating the latent code of an generator. It should be noticed that this model requires a large number of training data, which is infeasible in our settings. For fair comparison, we use the pretrained model provided in the papers.

Following the settings of Gatys [18], layers `conv1_1`, `pool1`, `pool2`, `pool3` and `pool4` are selected for Gram loss.

Fig. 4 displays the results of mixing two stationary textures. This type of textures have relatively simple structures, as they can be feasibly modeled by Gaussian texture models [5]. In this experiment, we compare our algorithm with the GaussTexton [5], Image Melding [4] and TexMixer [32]. As GaussTexton is specifically designed for Gaussian texture mixing, and the detailed shape of the grass is completely missed. Image melding and TexMixer can indeed generate comparable textures, but they both produce new structures or artifacts, *i.e.* vertical strips or wrinkle-like noise. Linear algorithms (LIA and LIA + TextureNet) lead to poor quality results, where different textures are joint together in patch-wise. Our algorithms (both with or without TextureNet) produce the best results, as the details are preserved and no extra structures are produced.

Fig. 5 presents the results of mixing a pair of nature textures belong to the same category. This type of mixing is of particular interesting in real applications, because the mixed textures create more inner variance in a category. As the considered textures can not be modeled by Gaussian models, we only compare our method with Image Melding [4] and TexMixer [32]. Observe that our mixing algorithm can mix the edges and the shapes of pebbles simultaneously, and create smooth transitions from one exemplar texture to the other without ''ghosting''. Image Melding can also create such transitions, but it generates obviously repeated patterns, *i.e.* some pebbles in mixed textures are completely the same. The results of TexMixer are also inferior to ours, as there are several visible blurring areas in the mixing results.

Fig. 6 shows the results of mixing a pair of nature textures belong to different categories. This type of synthesis may have great theoretical importance, as it not only enables us to create textures belong to new intermediate categories, but also uncovers the connections between different categories. This is the most difficult scenario in texture mixing, as the exemplars contain regular textural elements and their visual properties, such as color and patterns, are significantly different. We compare our method with Image Melding, TexMixer [32] and DFN [22]. Notice all baseline methods fail to generate intermediate textures in this experiments. In contrast, our methods can produce considerably better results, creating smooth transitions both in color and texture patterns from one to the other input.

Fig. 7 compares our algorithm with baseline methods on non-local textures. This task is different from others as it requires to preserve the regular structures in the mixed textures. In this experiments, we use our algorithm with additional correlation penalty $\mathcal{S}$, *i.e.*, we optimize Eqn. (26) to enforce the non-local structures. Our results are compared with Image Melding and TexMixer. Notice Image Melding indeed creates smooth transition between textures, but the regular structures are no longer preserved. TexMixer fails to handle this task, probably because it can not model non-local structures. On the contrary, our correlation-based algorithm successfully produces mixed textures, as it creates smooth transition between input exemplars and preserves regular structures in every intermediate textures.

C. STYLE MORPHING

In this experiment, we extend our texture mixing algorithm to style morphing. Our goal is to create ''intermediate'' styles between different styles, in other words, to create

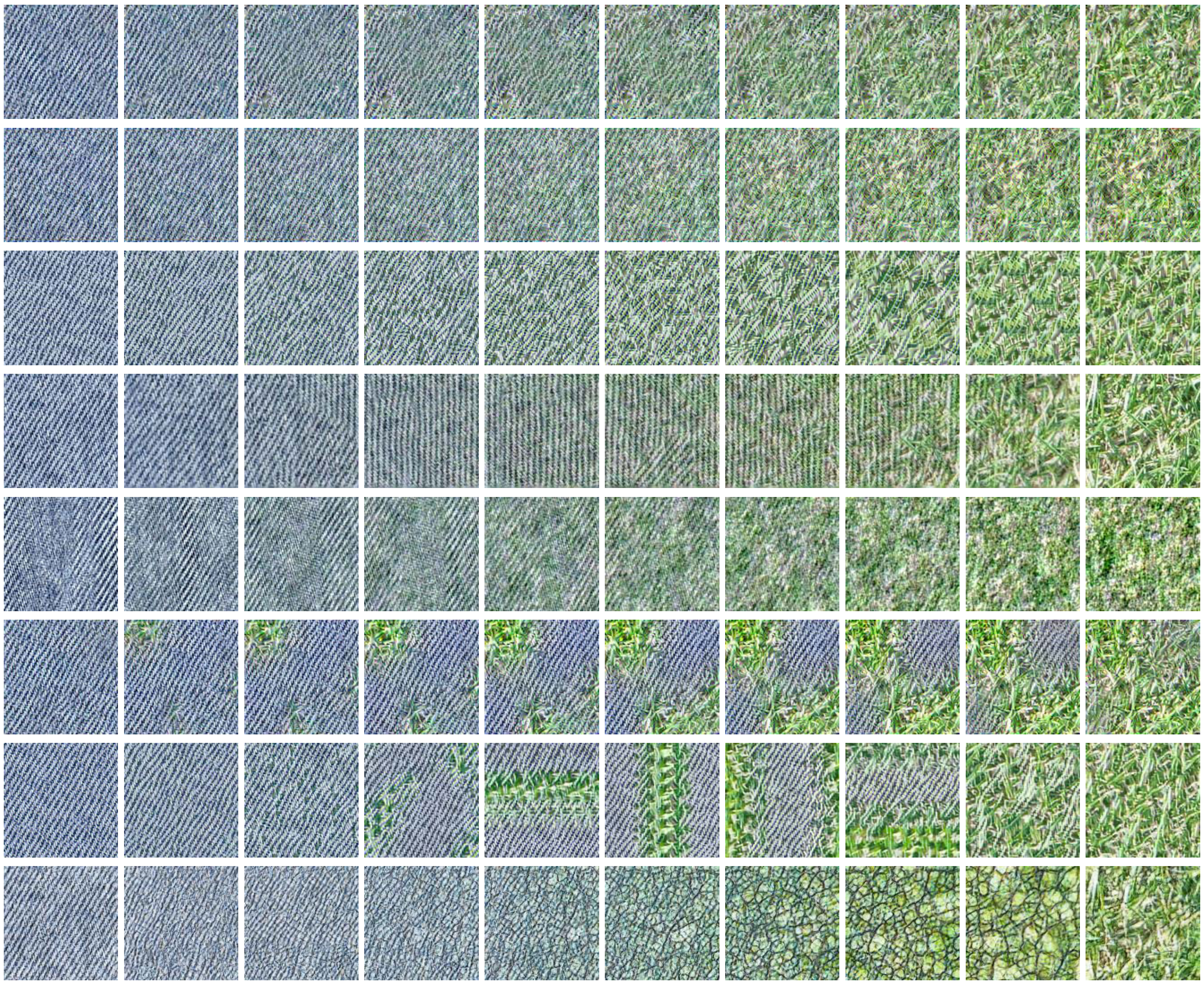


FIGURE 4. Mixing micro textures using our method with Gram matrix (1-st row), our method with correlation (2-nd row), our method + TextureNet (3-rd row), Image Melding (4-th row) [4], GaussTexton (5-th row), LIA (6-th row), LIA + TextureNet (7-th row), and TexMixer (8-th row). Notice that our method, both with or without TextureNet, can smoothly interpolate between two exemplars. See text for more details.

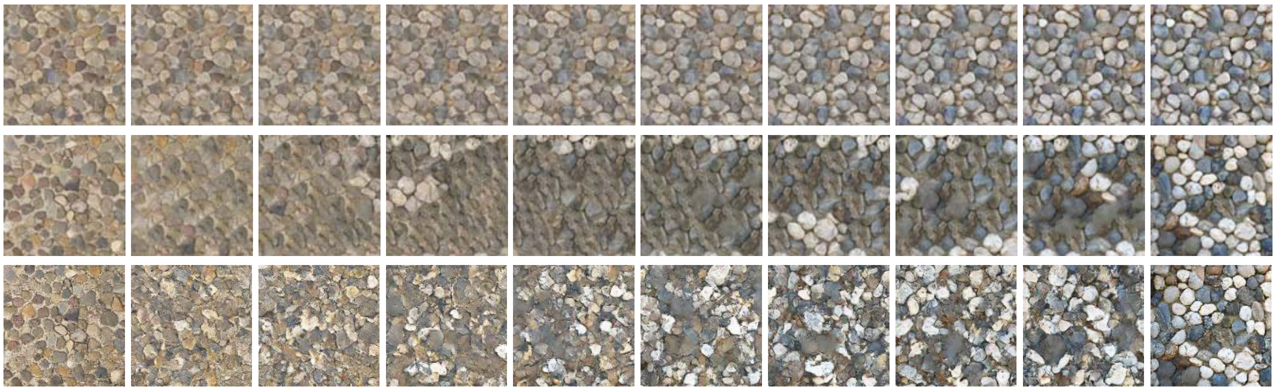


FIGURE 5. Mixing a pair of nature textures in the category "pebbles" using our method (1-st row), Image Melding (2-nd row) and TexMixer (3-nd row).

smooth transitions between stylish photos. We use Jonson's feed-forward structure [24] together with instance normalization [20]. We set style layers as `relu1_1`, `relu2_1`, `relu3_1` and `relu4_1`, content layer as `relu4_2`. Style weight is set to 5. All other parameters are left as default. We compare our result with Dumoulin's algorithm [31].

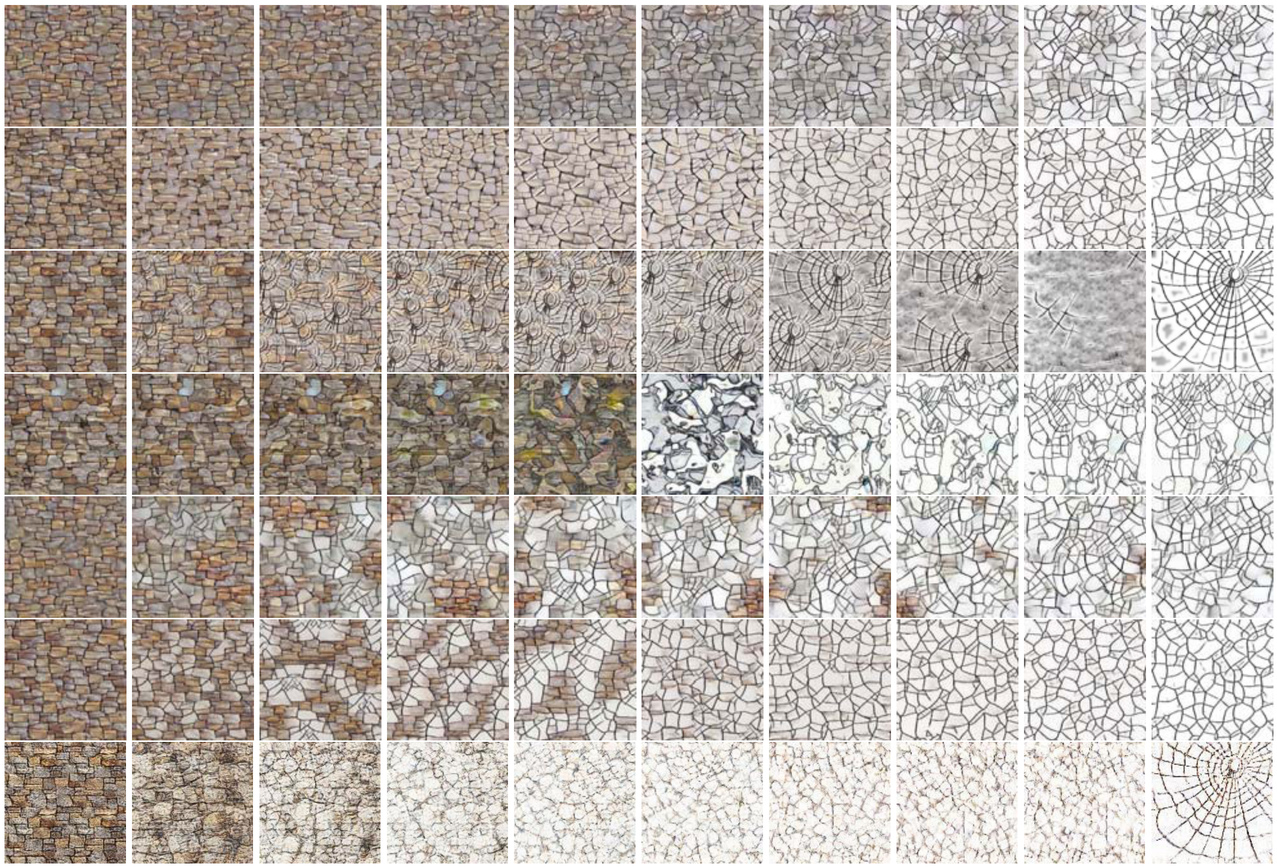


FIGURE 6. Mixing a pair of nature textures belong to different categories using our method (1-st row), our method + TextureNet (2-nd row), Image Melding (3-rd row), DFN (4-th row), LIA (5-th row), LIA + TextureNet (6-th row), and TexMixer (7-th row). Note that our scheme, either combined with TextureNet or not, creates more smooth transitions, both in color and texture patterns, between exemplars.

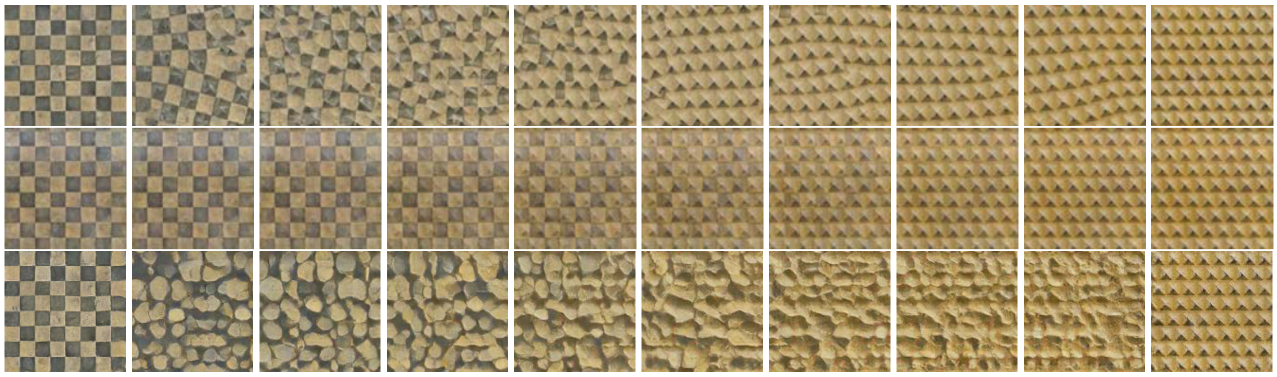


FIGURE 7. Mixing non-local textures using Image Melding (1-st row), our method + correlation matrix (2-nd row) and TexMixer (3-rd row). Note that our algorithm can preserve structures in every mixed textures.

To produce better results, we use a technique called *lag constraint*. Specifically, instead of calculating all mixed features directly, we calculate mixed feature maps at `pool1`, `pool2` and `pool3` layer, and propagate the mixed feature maps to style layers `relu2_1`, `relu3_1` and `relu4_1` respectively. Results with or without lag constraint are showed in Fig. 8.

Fig 9 compares results between our algorithm and Dumoulin's algorithm [31]. As we can see in this experiment, although Dumoulin's algorithm can morph different styles continuously, it failed to represent most of detailed structures. In contrast, our method can preserve more detailed structures in the images and create smooth transitions between styles.

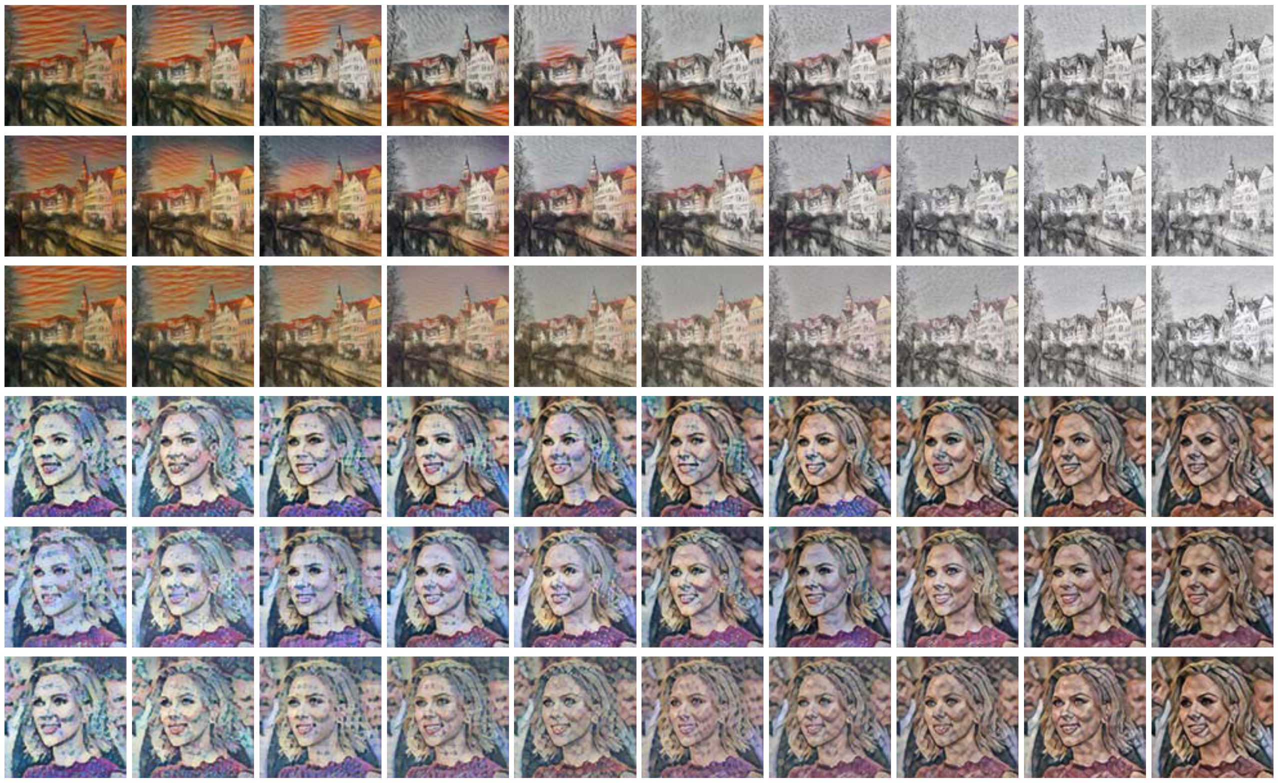


FIGURE 8. Comparisons of style morphing results using linear algorithm (1-st and 4-th row), and our scheme with (3-rd and 6-rd row) or without (2-nd and 5th row) lag constraint. Note that lag constraint technique (see text for details) produces better results.

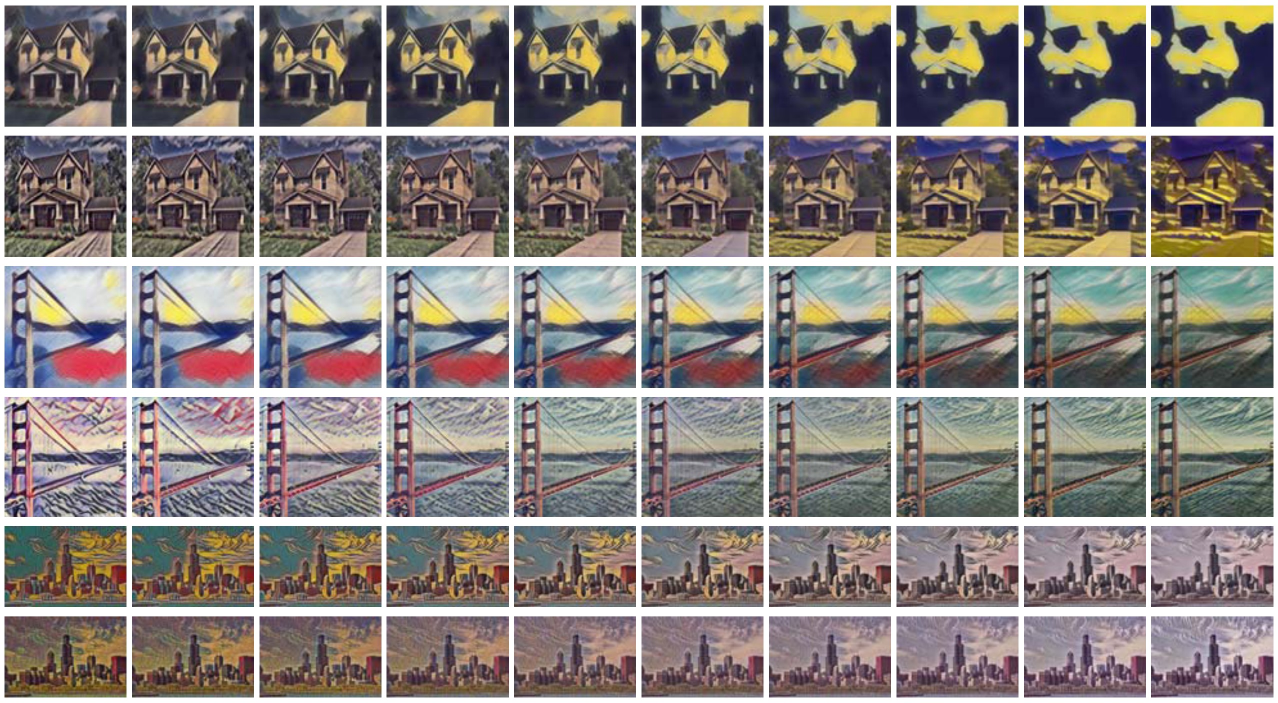


FIGURE 9. Comparison on style morphing between Dumoulin's algorithm [31] (1-st, 3rd and 5-th row) and our method (2-nd, 4-th and 6-th row). Dumoulin's algorithm can indeed create smooth transitions between different styles, but it fails to represerve detailed structures. On the contrary, our algorithm can preserve more detailed structures and create smooth transitions simultaneously.

D. INCREMENTAL TRAINING

In the scenarios where one needs to mix textures/styles with a large number of different relative weights, it can be time-consuming to initialize each optimization process with random noises. We can use incremental training to reduce time consumption and create more smooth transitions.

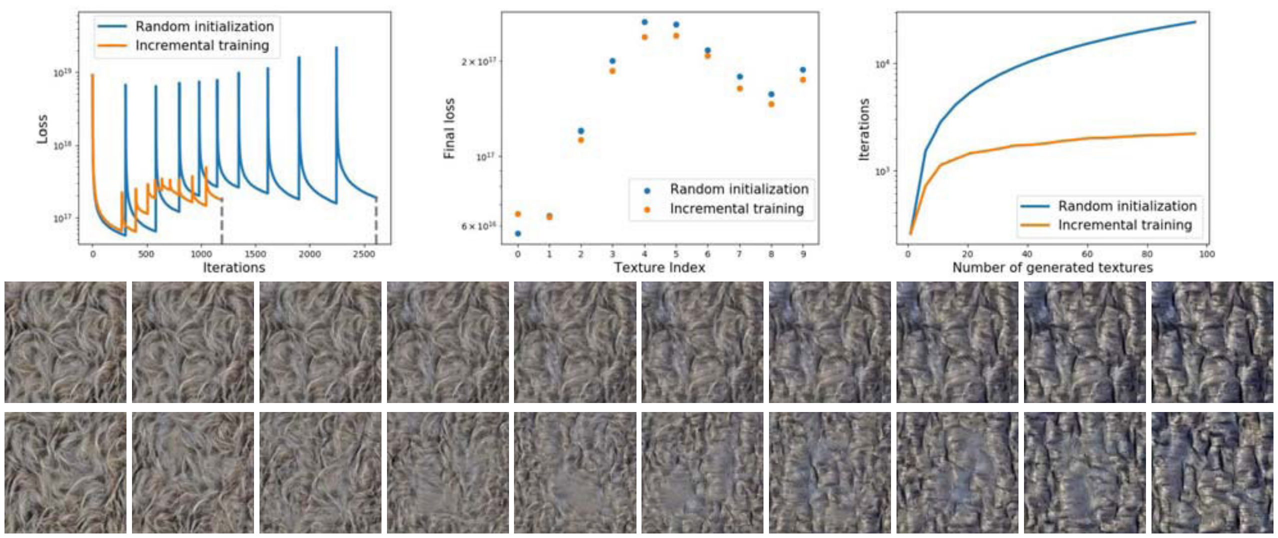


FIGURE 10. Comparison on incremental and random training for mixing wood and wool textures. Incremental training converges much faster than random training (Top left). It also achieves slightly lower loss (Top middle). The difference of converge speed is more obvious when synthesizing more images (Top right). Compared with the results of random training (Bottom, 2-nd row), transitions created by incremental training (Bottom, 1-st row) are more smooth and visually pleasant.

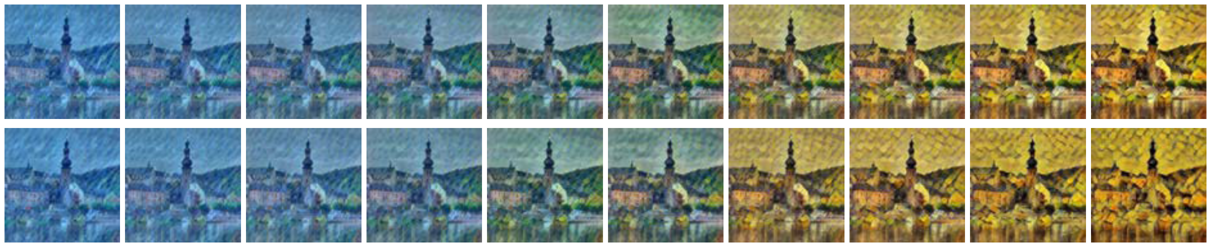


FIGURE 11. Comparison on style morphing using incremental training (Bottom, 1-st row) and random training (Bottom, 2-nd row). Notice that the transitions created by incremental training are slightly more smooth.

Specifically, our goal is to synthesis N images whose relative weights ρ are equally space in interval $[0, 1]$: $\frac{0}{N-1}$, $\frac{1}{N-1}, \dots, \frac{N-2}{N-1}, \frac{N-1}{N-1}$. In incremental training, instead of initializing each image as random noise, we generate images in sequence from small weight to the larger weight: the first image is initialized with random noise, while the rest of images are initialized with the image synthesized before. The convergence error is set to 0.001 for texture mixing, and the maximum number of iterations is fixed to be 10000.

Fig 10 illustrates the differences between these two training procedures. For texture mixing, incremental training can speed up the optimization process by offering a better initial point, and also lead to a lower final loss. It is also worth noticing that incremental training creates more smooth and visually pleasing transitions than random training. Similar results can be observed in style morphing, which is presented in Fig. 11.

VII. CONCLUSION

This paper proposed a novel algorithm to mix textures with CNN. To this end, we revealed the statistics used in CNN based texture models can be represented by a Gaussian model, thus, interpolating this statistics is reduced to interpolating Gaussian models, which has a closed form solution. Experimental results show that our algorithm excels in mixing high quality textures, and creating mixed styles different from exemplar styles.

There are still some issues need to be further investigated. For example, we notice that the optimization based CNN methods [18] [23] produce some low level noise. Although in most cases one can polish the results with total variation de-noise techniques as in [24]. This problem might be completely overcome by carefully padding the feature maps [31], or by using upsampling and convolution instead of deconvolution as suggest in [39]. Another important aspect is the choice of the training set in training feed forward networks. Current researches use the whole ImageNet dataset as the training set, and it is time consuming to iterate through the whole data set. It is still unclear wether it's possible to use a smaller training set.

Finally, note that we only described mixing of two given textures/styles, but our algorithm can be extended to mixing more textures/styles without difficulty.

ACKNOWLEDGMENT
(Zhucun Xue and Ziming Wang contributed equally to this work.)

REFERENCES

[1] G. Peyre, "Texture synthesis with grouplets," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 4, pp. 733–746, Apr. 2010.

[2] J. Rabin, G. Peyré, J. Delon, and M. Bernot, "Wasserstein barycenter and its application to texture mixing," in *Proc. Int. Conf. Scale Space Variational Methods Comput. Vis.* Springer, 2011, pp. 435–446.

[3] S. Ferradans, G.-S. Xia, G. Peyré, and J.-F. Aujol, "Static and dynamic texture mixing using optimal transport," in *Proc. 4th Int. Conf. Scale Space Variational Methods Comput. Vis.*, 2013, pp. 137–148.

[4] S. Darabi, E. Shechtman, C. Barnes, D. B. Goldman, and P. Sen, "Image melding: Combining inconsistent images using patch-based synthesis," *ACM Trans. Graph.*, vol. 31, no. 4, pp. 1–10, Jul. 2012.

[5] G.-S. Xia, S. Ferradans, G. Peyré, and J.-F. Aujol, "Synthesizing and mixing stationary Gaussian texture models," *SIAM J. Imag. Sci.*, vol. 7, no. 1, pp. 476–508, Jan. 2014.

[6] E. Risser, C. Han, R. Dahyot, and E. Grinspun, "Synthesizing structured image hybrids," *ACM Trans. Graph.*, vol. 29, no. 4, p. 85, Jul. 2010.

[7] N. Jetchev, U. Bergmann, and R. Vollgraf, "Texture synthesis with spatial generative adversarial networks," 2016, *arXiv:1611.08207*. [Online]. Available: http://arxiv.org/abs/1611.08207

[8] L.-Y. Wei, S. Lefebvre, V. Kwatra, and G. Turk, "State of the art in example-based texture synthesis," in *Eurographics*. Aire-la-Ville, Switzerland: State of the Art Report, EG-STAR. Eurographics Association, 2009, pp. 93–117.

[9] B. Julesz, "Textons, the elements of texture perception, and their interactions," *Nature*, vol. 290, no. 5802, pp. 91–97, Mar. 1981.

[10] S. C. Zhu, Y. Wu, and D. Mumford, "Filters, random fields and maximum entropy (frame): Towards a unified theory for texture modeling," *Int. J. Comput. Vis.*, vol. 27, no. 2, pp. 107–126, 1998.

[11] J. Portilla and E. P. Simoncelli, "A parametric texture model based on joint statistics of complex wavelet coefficients," *Int. J. Comput. Vis.*, vol. 40, no. 1, pp. 49–70, Oct. 2000.

[12] A. A. Efros and T. K. Leung, "Texture synthesis by non-parametric sampling," in *Proc. 7th IEEE Int. Conf. Comput. Vis.*, vol. 2, Sep. 1999, pp. 1033–1038.

[13] L.-Y. Wei and M. Levoy, "Fast texture synthesis using tree-structured vector quantization," in *Proc. 27th Annu. Conf. Comput. Graph. Interact. Techn. (SIGGRAPH)*. New York, NY, USA: ACM, 2000, pp. 479–488.

[14] R. Ruiters, R. Schnabel, and R. Klein, "Patch-based texture interpolation," *Comput. Graph. Forum*, vol. 29, no. 4, pp. 1421–1429, 2010.

[15] D. J. Heeger and J. R. Bergen, "Pyramid-based texture analysis/synthesis," in *Proc. 22nd Annu. Conf. Comput. Graph. Interact. Techn.* New York, NY, USA: ACM, 1995, pp. 229–238.

[16] B. Galerne, Y. Gousseau, and J.-M. Morel, "Random phase textures: Theory and synthesis," *IEEE Trans. Image Process.*, vol. 20, no. 1, pp. 257–267, Jan. 2011.

[17] Z. Bar-Joseph, R. El-Yaniv, D. Lischinski, and M. Werman, "Texture mixing and texture movie synthesis using statistical learning," *IEEE Trans. Vis. Comput. Graphics*, vol. 7, no. 2, pp. 120–135, 2001.

[18] L. Gatys, A. S. Ecker, and M. Bethge, "Texture synthesis using convolutional neural networks," in *Proc. NIPS*, 2015, pp. 262–270.

[19] G. Liu, Y. Gousseau, and G.-S. Xia, "Texture synthesis through convolutional neural networks and spectrum constraints," in *Proc. 23rd Int. Conf. Pattern Recognit. (ICPR)*, Dec. 2016, pp. 3234–3239.

[20] D. Ulyanov, A. Vedaldi, and V. Lempitsky, "Improved texture networks: Maximizing quality and diversity in feed-forward stylization and texture synthesis," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 6924–6932.

[21] O. Sendik and D. Cohen-Or, "Deep correlations for texture synthesis," *ACM Trans. Graph.*, vol. 36, no. 4, p. 161, Jul. 2017.

[22] Y. Li, C. Fang, J. Yang, Z. Wang, X. Lu, and M.-H. Yang, "Diversified texture synthesis with feed-forward networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 3920–3928.

[23] L. A. Gatys, A. S. Ecker, and M. Bethge, "Image style transfer using convolutional neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 2414–2423.

[24] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual losses for real-time style transfer and super-resolution," in *Proc. ECCV*. Springer, 2016, pp. 694–711.

[25] L. Raad, A. Davy, A. Desolneux, and J.-M. Morel, "A survey of exemplar-based texture synthesis," *CoRR*, vol. abs/1707.07184, 2017.

[26] B. Galerne, Y. Gousseau, and J.-M. Morel, "Micro-texture synthesis by phase randomization," *Image Process. Line*, vol. 1, pp. 213–237, Sep. 2011.

[27] G.-S. Xia, S. Ferradans, G. Peyre, and J.-F. Aujol, "Compact representations of stationary dynamic textures," in *Proc. 19th IEEE Int. Conf. Image Process.*, Sep. 2012, pp. 2993–2996.

[28] L. Raad, A. Desolneux, and J.-M. Morel, "A conditional multiscale locally Gaussian texture synthesis algorithm," *J. Math. Imag. Vis.*, vol. 56, no. 2, pp. 260–279, Oct. 2016.

[29] B. Galerne and A. Leclaire, "Texture inpainting using efficient Gaussian conditional simulation," *SIAM J. Imag. Sci.*, vol. 10, no. 3, pp. 1446–1474, Jan. 2017.

[30] B. Galerne, A. Leclaire, and L. Moisan, "Texton noise," *Comput. Graph. Forum*, vol. 36, no. 8, pp. 205–218, Dec. 2017.

[31] V. Dumoulin, J. Shlens, and M. Kudlur, "A learned representation for artistic style," *CoRR*, vol. 2, no. 4, p. 5, 2016. [Online]. Available: https://arxiv.org/abs/1610.07629

[32] N. Yu, C. Barnes, E. Shechtman, S. Amirghodsi, and M. Lukac, "Texture mixer: A network for controllable synthesis and interpolation of texture," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 12164–12173.

[33] Y. Lu, S.-C. Zhu, and Y. N. Wu, "Learning frame models using CNN filters," in *Proc. 13th AAAI Conf. Artif. Intell.* Palo Alto, CA, USA: AAAI Press, 2016, pp. 1902–1910.

[34] C. Atkinson and A. F. S. Mitchell, "Rao's distance measure," *Sankhyā, Indian J. Statist., A*, vol. 43, no. 3, pp. 345–365, 1981.

[35] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*. [Online]. Available: http://arxiv.org/abs/1409.1556

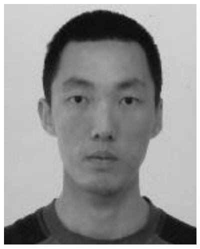
[36] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 248–255.

[37] M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, and A. Vedaldi, "Describing textures in the wild," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 3606–3613.

[38] C. Zhu, R. H. Byrd, and P. Lu, "L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization," *Acm Trans. Math. Softw.*, vol. 23, no. 4, pp. 550–560, 1994.

[39] A. Odena, V. Dumoulin, and C. Olah, "Deconvolution and checkerboard artifacts," *Distill*, vol. 1, no. 10, p. e3, Oct. 2016. [Online]. Available: http://distill.pub/2016/deconv-checkerboard

ZHUCUN XUE is currently pursuing the M.S. degree with the State Key Laboratory of Information Engineering in Surveying, Mapping and Remote Sensing (LIESMARS), Wuhan University, China. Her research interests include low-level vision and structure from motion.



ZIMING WANG received the M.S. degree in bio-informatics from the University of Chinese Academy of Sciences, Shanghai, China, in 2017. He is currently pursuing the Ph.D. degree in computer vision with the State Key Laboratory of Information Engineering in Surveying, Mapping and Remote Sensing (LIESMARS), Wuhan University, China. His research interests include texture modeling and image processing.