Computational Visual Media
https://doi.org/10.1007/s41095-021-0243-7 Vol. 8, No. 2, June 2022, 289–302

Research Article

Co-occurrence based texture synthesis

Anna Darzi[1], Itai Lang[1], Ashutosh Taklikar[1], Hadar Averbuch-Elor[2] (✉), and Shai Avidan[1]

Abstract As image generation techniques mature, there is a growing interest in explainable representations that are easy to understand and intuitive to manipulate. In this work, we turn to co-occurrence statistics, which have long been used for texture analysis, to learn a controllable texture synthesis model. We propose a fully convolutional generative adversarial network, conditioned locally on co-occurrence statistics, to generate arbitrarily large images while having local, interpretable control over texture appearance. To encourage fidelity to the input condition, we introduce a novel differentiable co-occurrence loss that is integrated seamlessly into our framework in an end-to-end fashion. We demonstrate that our solution offers a stable, intuitive, and interpretable latent representation for texture synthesis, which can be used to generate smooth texture morphs between different textures. We further show an interactive texture tool that allows a user to adjust local characteristics of the synthesized texture by directly using the co-occurrence values.

Keywords co-occurrence; texture synthesis; deep learning; generative adversarial networks (GANs)

1 Introduction

Deep learning has revolutionized our ability to generate novel images. Most notably, generative adversarial networks (GANs) have shown impressive results in various domains, including textures. Nowadays, GANs can generate a distribution of texture images that is often indistinguishable from the distribution of real ones. The goal of the generator is conceptually simple and boils down to training the network weights to map a latent code, sampled from a random distribution, to realistic samples.

However, generative networks are typically non-explainable and hard to control. That is, given a generated sample, it is generally hard to explain its latent representation and to directly modify it to output a new sample with desired properties.

Recently, we are witnessing growing interest in interpreting the latent spaces of GANs, aspiring to better understand their behaviour [1–3]. These studies tap into vector arithmetics in latent space, or use more complicated entangled properties to achieve better control of the texture generation process. However, many questions remain open, and local control is still difficult to achieve.

In this work, we seek a generative model for textures that is intuitive to understand and easy to edit and manipulate. Our key insight is that we can bypass the need to understand a highly entangled latent space by using a structured latent space, with meaningful interpretable vectors.

We use a statistical tool, co-occurrence, to serve as an encoding of texture patches. Co-occurrence, first introduced by Julesz [4], captures the local joint probability of pairs of pixel values to co-occur. They have long been used to analyze textures [5, 6]. In our work, we take the opposite direction. Given a co-occurrence matrix, we can generate a variety of texture images that match it.

Technically, we train a fully convolutional conditional GAN (cGAN), conditioned on local co-occurrence statistics. The convolutional architecture allows for arbitrarily large generated textures, while local co-occurrence statistics gives us explainable control over *local* texture appearance. This allows

1 Tel Aviv University, Tel Aviv 6997801, Israel. E-mail: A. Darzi, annadarzi@mail.tau.ac.il; I. Lang, itailang@mail.tau.ac.il; A. Taklikar, ashutosht@mail.tau.ac.il; S. Avidan, avidan@eng.tau.ac.il.

2 Cornell-Tech, Cornell University, NYC, NY, 10044, USA. E-mail: hadarelor@cornell.edu (✉).

Manuscript received: 2021-04-05; accepted: 2021-05-28

the synthesis of a variety of inhomogeneous textures, such as those shown in Fig. 1, by conditioning locally on different co-occurrences collected from the input textures.

To enable end-to-end training using co-occurrence statistics, we introduce a new differentiable *co-occurrence loss*, which penalizes inconsistencies between the co-occurrence of the synthesized texture and the input condition. We demonstrate that our proposed training objective allows for a simple and interpretable latent representation, without compromising fidelity, on a large variety of inhomogeneous textures. We show that our approach can be used, for example, to interpolate between two texture regions by interpolating between their corresponding co-occurrence matrices, resulting in a *dynamic* texture morph. We also illustrate how to control the generated texture by editing the co-occurrence input.

Our contributions are twofold. Firstly, we introduce a stable, intuitive, and interpretable latent representation for texture synthesis, offering parametric control over the synthesized output using co-occurrence statistics. Secondly, we present a fully convolutional cGAN architecture with a differentiable co-occurrence loss, enabling end-to-end training. Our code is publicly available at `https://github.com/ashu7397/cooc_texture`.

2 Related work

There are different ways to model texture. Heeger and Bergen [7] represented textures as histograms of different levels of a steerable pyramid, while De Bonet [8] represented texture as the conditional probabilities of pixel values at multiple scales of the image pyramid. New texture is synthesized by matching the statistics of a noise image to that of the input texture. Portilla and Simoncelli [9] continued this line of research, using a set of statistical texture properties. None of these methods used co-occurrence statistics for texture synthesis.

Stitching-based methods [10–12] assume a non-parametric model of texture. In this case a new texture is generated by sampling patches from the input texture image. This sampling procedure can lead to poor results; one fix is to use an objective function that forces the distribution of patches in the output texture to match that of the input texture

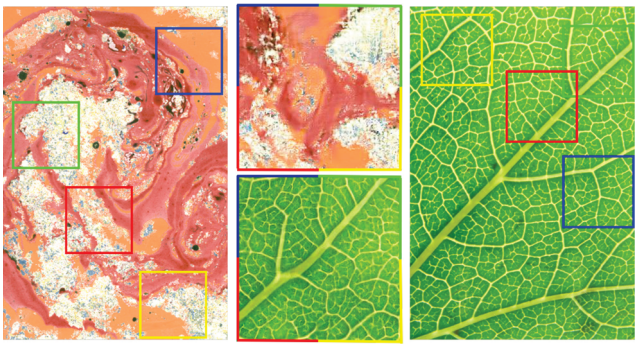


Fig. 1 Given a texture exemplar (left and right), co-occurrence statistics are collected for different texture crops. Using these co-occurrences, our method can synthesize a variety of novel textures with desired local properties (center), similar to those of the corresponding crops from the texture exemplar. Above we demonstrate textures synthesized from co-occurrence statistics collected at four random crops (marked in unique colors).

[13, 14]. These methods have proven to be very effective in synthesizing plausible textures. Please refer to the work by Barnes and Zhang [15] for a more comprehensive survey of patch-based synthesis techniques.

Some methods have been proposed to interpolate between textures rather than to synthesize new textures. For example, Matusik et al. [16] captured the structure of the induced space by a simplicial complex whose vertices represent input textures. Interpolating between vertices corresponds to interpolating between textures. Rabin et al. [17] interpolated between textures by averaging discrete probability distributions that are treated as a barycenter in the Wasserstein space. Darabi et al. [18] used a screened Poisson equation solver to meld images and, among other applications, showed how to interpolate between different textures.

Deep learning for texture synthesis by Gatys et al. [19] follows the approach of Heeger and Bergen [7]. Instead of matching histograms of the image pyramid, they matched the Gram matrix of different feature maps of the texture image, where the Gram matrix measures correlation between features in selected layers of a neural network. This approach was later improved [20, 21]. These methods look at pair-wise relationships between features, which is similar to what we do. However, the Gram matrix measures correlation of deep features, whereas we use co-occurrence statistics of pixel values.

Alternatively, one can use a generative adversarial network (GAN) to synthesize textures that resemble

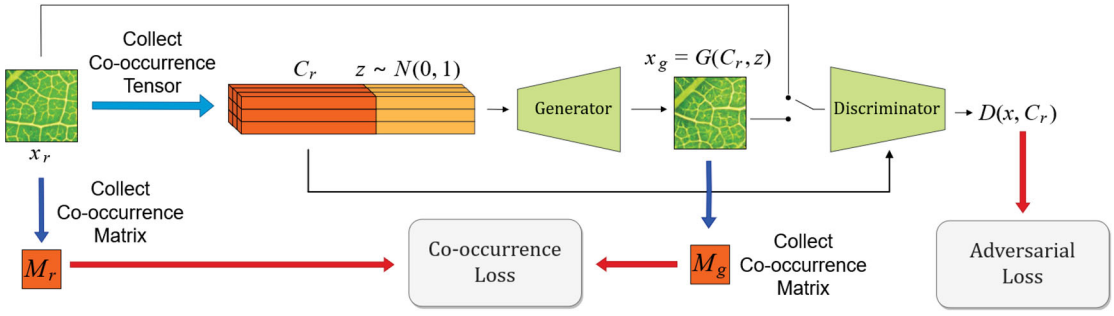


Fig. 2 Method overview. The generator receives as input a co-occurrence tensor C_r, corresponding to a real texture crop x_r. This tensor is concatenated to a random noise tensor z. The generator outputs a synthetic texture sample x_g. The discriminator's input alternates between real and synthetic texture samples. It also receives the co-occurrence tensor C_r. In addition, the co-occurrence of the synthesized texture is required to be consistent with the input statistics to the generator. By this architecture, the generator learns to synthesize samples with desired local texture properties, as captured by the co-occurrence tensor.

an input exemplar. Li and Wand [22] used a GAN combined with a Markov random field to synthesize texture images from neural patches. Gang et al. [23] improved the method of Gatys et al. [19] by adding constraints on the Fourier spectrum of the synthesized image, and Li et al. [24] used a feed-forward network to synthesize diversified texture images. Zhou et al. [25] used GANs to spatially expand texture exemplars, extending non-stationary structures. Frühstück et al. [26] synthesized large textures using a pre-trained generator, which can produce images at higher resolutions.

Texture synthesis using GANs is also used for texture interpolation. Jetchev et al. [27] suggested a spatial GAN (SGAN), where the input noise to the generator is a spatial tensor rather than a vector. This method was later extended to a periodic spatial GAN (PSGAN), proposed by Bergmann et al. [28]. Interpolating between latent vectors within the latent tensor results in spatial interpolation in the generated texture image. These works focus on spatial interpolation, and learn the input texture's structure as a whole. We focus on representing the appearance of different local regions of the texture in a controllable manner.

The most closely related work to ours is the recent texture mixer of Yu et al. [29], which performs texture interpolation in the latent space. However, unlike our work, their method requires encoding sample crops into the latent space and control is obtained in the form of interpolating between the representations of these sample crops. Our method, on the other hand, provides a parametric and interpretable latent representation which can be controlled directly.

Co-occurrences were introduced by Julesz [4] who conjectured that two distinct textures can be distinguished by their second order statistics (i.e., co-occurrences). This conjecture is not necessarily true because it is possible to construct distinct images that have the same first (histograms) and second order (co-occurrence) statistics. It was later shown by Yellott [30] that images with the same third-order statistics are indistinguishable. In practice, co-occurrences have long been used for texture analysis [5, 6], but not for texture synthesis, as we propose in this work.

3 Method

We use a conditional generative adversarial network (cGAN) to synthesize textures with spatially varying co-occurrence statistics. Before giving details, let us fix terminology first. A texture *patch* (i.e., a 64×64 pixel region) is represented by a local co-occurrence matrix. A texture *crop* (i.e., a 128×128 pixel region) is represented by a collection of local co-occurrence matrices, organized as a co-occurrence tensor. We train our cGAN on texture crops, because crops capture interaction between neighboring co-occurrences. This allows the generator to learn how to synthesize texture that fits spatially varying co-occurrence statistics. Once the cGAN is trained, we can feed the generator with a co-occurrence tensor and a random seed to synthesize images of arbitrary size. We can do that because both the discriminator and generator are fully convolutional.

Figure 2 provides an overview of our cGAN architecture. It is based on the one proposed by Bergmann et al. [28]. In what follows, we explain how the co-occurrence statistics are collected, followed by

a description of our cGAN and how we use it to generate new texture images.

We use co-occurrence matrices to represent the local appearance of patches in the texture image. For a given patch, we calculate the co-occurrence matrix M of the joint appearance statistics of pixel values [31, 32].

The size of the co-occurrence matrix scales quadratically with the number of pixel values and it is therefore infeasible to work directly with RGB values. To circumvent that we quantize the color space to a small number of clusters. The pixel values of the input texture image are first grouped into k clusters using standard k-means. This results in a $k \times k$ co-occurrence matrix.

Let (τ_a, τ_b) denote two cluster centers. Then $M(\tau_a, \tau_b)$ is given by

$$M(\tau_a,\tau_b)=\frac{1}{Z}\sum_{p,q}\exp\left(-\frac{d(p,q)^2}{2\sigma^2}\right)K(I_p,\tau_a)K(I_q,\tau_b) \tag{1}$$

where I_p is the pixel value at location p, $d(p,q)$ is the Euclidean distance between pixel locations p and q, σ is a user specified parameter, and Z is a normalizing factor designed to ensure that the elements of M sum to 1.

K is a soft assignment kernel function that decays exponentially with the distance of the pixel value from that of the cluster center:

$$K(I_p,\tau_l)=\exp\left(-\sum_i\frac{(I_p^i-\tau_l^i)^2}{(\sigma_l^i)^2}\right) \tag{2}$$

where i runs over the RGB color channels and σ_l^i is the standard deviation of color channel i of cluster l.

The contribution of a pixel value pair to the co-occurrence statistics decays with their Euclidean distance in the image plane. In practice, we do not sum over all pixel pairs (p,q) in the image, but rather consider only pixels q within a window around p. An illustrative image and its corresponding co-occurrence matrix are given in Fig. 3.

We collect the co-occurrence statistics of an image crop x of spatial dimensions $h \times w$, according to Eqs. (1) and (2). We row-stack each of these matrices to construct a co-occurrence volume of size $h \times w \times k^2$. This is then downsampled spatially by a factor of s to obtain a co-occurrence tensor C. This downsampling allows more variability for the texture synthesis process, implicitly making every spatial position in C a description of the co-occurrence of a certain receptive field in x.

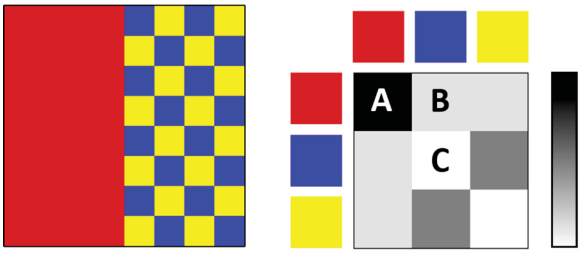


Fig. 3 Left: an image. Right: corresponding co-occurrence matrix, computed by Eqs. (1) and (2), using 4-neighborhood connectivity. Red pixels are frequently adjacent: their co-occurrence measure is high (A). Red pixels appear less frequently next to blue ones, giving a lower value in the co-occurrence matrix (B). Blue pixels are never adjacent, yielding a zero probability (C).

3.1 Texture synthesis

We denote a real texture crop by x_r, and its co-occurrence tensor C_r. The random noise seed tensor is denoted z; its entries are drawn from a normal distribution $\mathcal{N}(0,1)$. In order to balance the influence of the co-occurrence and the noise on the output of the generator, we also normalise C_r to have zero mean and unit variance.

The concatenated co-occurrence and noise tensor are given as input to the generator. The output of the generator G is a synthesized crop $x_g = G(C_r, z)$. The corresponding downsampled co-occurrence of x_g is denoted C_g.

The input to the discriminator D is a texture crop that is either a real crop from the texture image (x_r) or a synthetic one from the generator (x_g). In addition to the input texture crop, the original co-occurrence tensor C_r is also provided to the discriminator. For a real input texture, this is its corresponding co-occurrence tensor. For a synthetic input, the same co-occurrence condition given to the generator is used at the discriminator.

We emphasize that both the generator and discriminator are conditioned on the co-occurrence tensor. With this cGAN architecture, the discriminator teaches the generator to generate texture samples that preserve local texture properties, as captured by the co-occurrence tensor. In order to further guide the generator to respect the co-occurrence condition when synthesizing texture, we compute the co-occurrence statistics of the generated texture and demand consistency with the corresponding generator's input.

We train the network with the Wasserstein GAN with gradient penalty (WGAN-GP) objective function,

proposed by Gulrajani et al. [33]. In addition, we add a novel consistency loss on the co-occurrence of the generated texture and the input condition.

The generator is optimized according to the following loss function:

$$L(G) = -\mathbb{E}_{x_g \sim \mathbb{P}_g}[D(G(z, C_r), C_r)] + \lambda_M |M_g - M_r|_1 \tag{3}$$

where D is the discriminator network, and $|\cdot|_1$ is L_1 loss.

The discriminator is subject to the following objective:

$$L(D) = \mathbb{E}_{x_r \sim \mathbb{P}_r}[D(x_r, C_r)] - \mathbb{E}_{x_g \sim \mathbb{P}_g}[D(x_g, C_r)] + \lambda_p \mathbb{E}_{\hat{x} \sim \mathbb{P}_{\hat{x}}}[(||\nabla_{\hat{x}} D(\hat{x}, C_r)||_2 - 1)^2] \tag{4}$$

where $\hat{x}$ is randomly sampled as a linear interpolation between real and synthetic textures $\hat{x} = ux_r + (1-u)x_g$, with a uniform interpolation coefficient $u \in U[0,1]$.

Training is done with batches of texture samples and their corresponding co-occurrence tensors. When training the discriminator, co-occurrences of the real texture samples are used to generate the synthetic samples. For generator training, only co-occurrences of the batch are used.

4 Results and evaluation

We evaluate our technique on a wide variety of textures, including texture images from the Describable Textures Dataset [34], from the work of Bellini et al. [35], textures with perspective variations as studied by Wu et al. [36], and also several texture images we found online. We demonstrate the performance of our method in terms of: (i) fidelity and diversity, (ii) novelty, and (iii) stability. Additionally, we compare our approach to previous work (Section 4.3). We also perform an ablation study, demonstrating the importance of the different components of our work (Section 4.4). Finally we conclude by discussing limitations of our approach (Section 4.5).

4.1 Experimental details

For all texture images we set k equal to 2, 4, or 8 clusters. We collect the co-occurrence statistics for each pixel. We take a patch of 65×65 around that pixel, and calculate the co-occurrence statistics using a window of size 51×51 around each pixel in that patch and set σ^2 in Eq. (1) to 51. Thus, for each pixel we have a $k \times k$ co-occurrence matrix, which is reshaped to k^2, and for an image of $w \times h \times 3$, we have a co-occurrence volume of $w \times h \times k^2$.

The dataset for a texture image contains $N = 2000$ crops of $n \times n$ pixels. As our work aims to extract and analyze local properties, we use $n = 128$. The down sampling factor of the co-occurrence volume is $s = 32$. Thus, the spatial dimensions of the condition co-occurrence tensor are 4×4.

The architecture of the generator and discriminator is fully convolutional, so large textures can be synthesized at inference time. The architecture is based on that of PSGAN [28]. The generator in our case is a stack of 5 convolutional layers, each having an upsampling factor of 2 and filter of size 5×5, stride of 1, and a padding of 2, with ReLU activation and batch normalization.

The discriminator too is a stack of 5 convolutional layers, with filter size of 5×5, stride of 2, and padding of 2, with sigmoid activation. The stride here is 2 to bring down the upscaled spatial dimensions from the generator back to the original input size. After activation of the third layer, we concatenate the co-occurrence volume in the channel dimension, to help the discriminator also condition on co-occurrence (see Fig. 2).

For each texture image, we train our method for 120 epochs using the Adam optimizer with momentum 0.5. The learning rate is set to 0.0002, and the gradient penalty weight is $\lambda = 1$. The training time is about 3 hours using an nVidia 1080Ti GPU.

4.2 Evaluation

4.2.1 Fidelity and diversity

Texture synthesis algorithms must balance the contradicting requirements of fidelity and diversity. Fidelity refers to how similar the synthesized texture is to the input texture. Diversity expresses the difference in the resulting texture, with respect to the input texture. We maintain fidelity by keeping the co-occurrences fixed and achieve diversity by varying the noise seed.

Figure 1 shows that the generated samples resemble their corresponding texture crop. Utilizing the adversarial loss during training can be viewed as an implicit demand of this property, while the co-occurrence loss requires it more explicitly. Additional fidelity and diversity results are presented in the Electronic Supplementary Material (ESM).

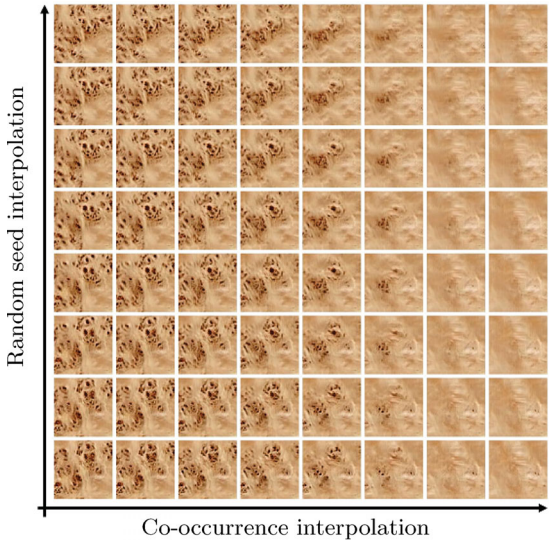


Fig. 4 Our technique synthesizes textures given a co-occurrence matrix and a random seed vector. Above we illustrate textures generated from interpolating between two co-occurrence vectors (across the columns) and two random seed vectors (across the rows).

We demonstrate the smoothness of the latent space by interpolating along two axes: between co-occurrence tensors and between noise tensors (see Fig. 4). Interpolation between two different co-occurrences results in a smooth traversal between different texture characteristics. On the other hand, for a given co-occurrence tensor, interpolation between different noise tensors smoothly transitions between texture samples with similar properties. Any intermediate result of the interpolation yields a plausible texture result, suggesting that the generator has learned a smooth texture manifold, with respect to both co-occurrence and noise.

4.2.2 Novelty of texture samples

To verify that our network is truly generating novel, unseen texture samples and not simply memorizing seen examples, we examine the nearest neighbors in the training set. To do so, we generate textures using unseen co-occurrences from the test set and search for their nearest neighbors in the training set, in terms of L_1 of RGB values. For comparison, we also compute the co-occurrence of the generated samples and look for the nearest neighbor, in an L_1 sense, in the co-occurrence space.

We demonstrate the results of this experiment in Fig. 5. The spatial arrangement of nearest neighbors in RGB space resembles that of the synthesized texture, yet they are not identical. Nearest neighbors in the co-occurrence space may have different spatial arrangement. In any case, while the training samples bear some resemblance to our generated texture samples, they are not the same, and visual differences are noticeable using both co-occurrence and RGB measures.

4.2.3 Stability of synthesized textures

We use a texture degradation test [21, 37] to measure the stability of our algorithm. Specifically, we do the following: given a co-occurrence tensor from the training set, we generate a texture sample, compute its co-occurrence tensor, and feed it back to the generator (with the same noise vector) to generate another sample. This process is repeated several times.

Figure 6 shows representative results of our stability test. The appearance of the synthesized textures remains roughly constant. We attribute this stable behaviour to the use of the co-occurrence as a condition for the texture generation process.

To quantitatively evaluate the stability of our method, we repeated this test for the entire test set of different texture images. Following each looping iteration, we measured the L_1 difference between the input co-occurrence and that of the generated sample, and average over all the examples in the set. For 10 looping iterations the average L_1 measure remains within 2% of the average L_1 measure of the first iteration, for all examined textures, indicating that the co-occurrence of the generated samples is indeed stable.

4.3 Comparison to previous work

We compare our method to two previous texture generation algorithms related to our work: PSGAN [28] and TextureMixer [29]. First, we examine the

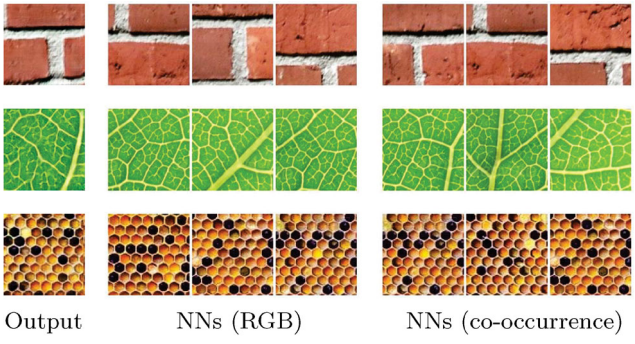


Fig. 5 Novelty of generated samples. Left: generated textures using co-occurrences from the test set. For each generated sample, we find its nearest neighbors (NNs) in the training set, measured in terms of RGB values (middle) and co-occurrences (right). As illustrated above, the generated textures are different from their nearest neighbors in the training set.

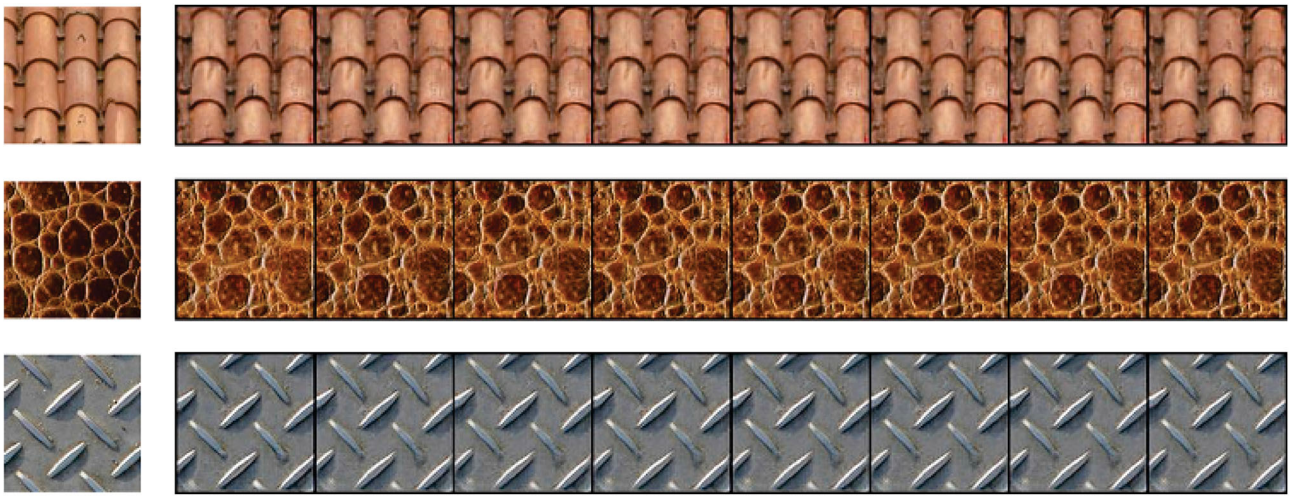


Fig. 6 Stability test. Textures were generated in a loop. In the first iteration, co-occurrences are taken from the test set. In the next, co-occurrence of the generated texture is used. The original crop from the texture exemplar is on the left and the generated sequence for eight iterations is on the right. It can be noted that the textures remain stable over the iterations.

diversity of our synthesized texture samples against those from PSGAN, illustrated in Fig. 7. PSGAN is an unconditional texture synthesis algorithm. Thus, there is no control on the appearance of the generated samples, resulting in limited diversity. Our method, on the contrary, is conditioned on co-occurrence statistics that represent the texture's local properties. Different input co-occurrences result in texture samples with different appearance, which allows us to generate diverse samples.

Next, we compare our generated textures to the ones obtained with Texture Mixer [29]. Our comparison focuses on three different aspects: fidelity and diversity (Fig. 8), stability (Fig. 9), and interpolation between texture regions with different properties (Fig. 10). We use their publicly available model, which was trained on earth textures.

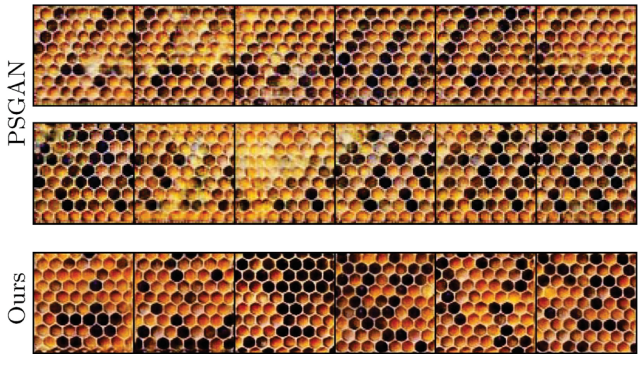


Fig. 7 Diversity comparison. Texture samples generated by PSGAN [28] are demonstrated on the top rows. Several of these results seem to have a repetitive pattern and their visual diversity is limited. Our technique is conditioned on co-occurrence statistics and thus generates significantly more diverse results.

Figure 8 shows that, while both methods generate diverse results, our method has somewhat higher fidelity to the input texture. In terms of stability, as demonstrated in Fig. 9, their method gradually breaks, while ours remains stable. This experiment illustrates that their encoder is sensitive to the distribution on which it was trained, and as the generated samples deviate from this distribution, it cannot accurately encode it into the latent space. Lastly, Fig. 10 shows that the local properties change more steadily and smoothly using our method.

We have also compared our results to the classical image quilting method for texture synthesis [10], which shows that the quality of classical techniques can be plausible, but, unlike our method, they do not offer control and interpretability. We refer the reader to the ESM for further details of this comparison.

4.4 Ablation study

In order to validate the importance of the different components in our framework, we performed several experiments, omitting key components one at a time. Specifically, we trained our cGAN without the co-occurrence condition at the generator or at the discriminator, or without the co-occurrence loss. Results are shown in Fig. 11.

When the generator is non-conditional, the control over the generation process is completely lost. Omitting the co-occurrence condition from the

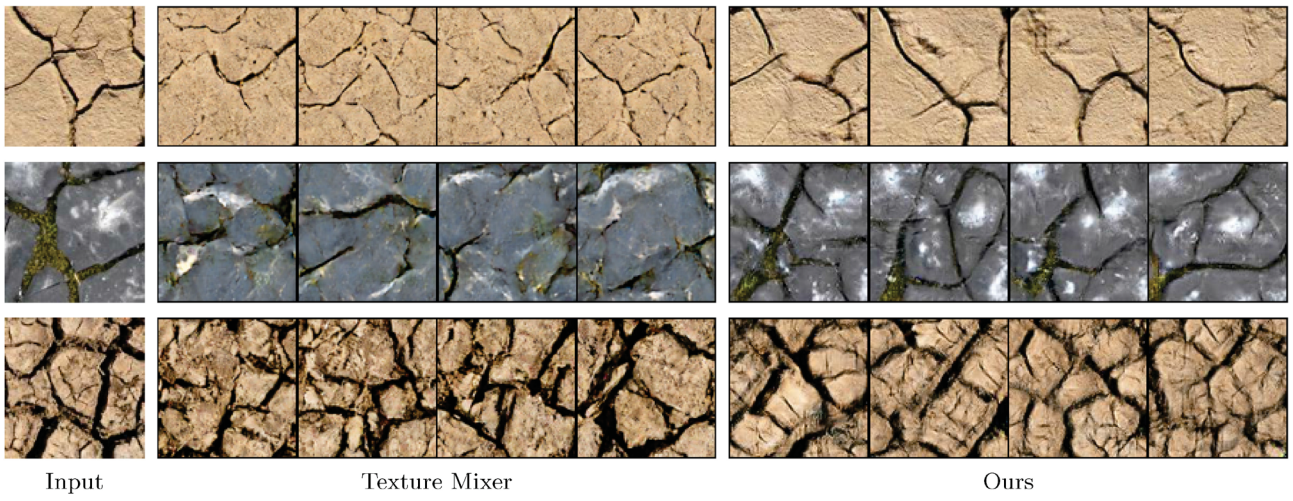


Fig. 8 Fidelity and diversity comparison. Given an input crop (left), we demonstrate samples synthesized using Texture Mixer [29] (center) and our technique (right). Note that unlike Texture Mixer, we do not encode the input crop directly—we generate samples conditioned on its co-occurrence matrix. While both methods generate diverse results, our method seems to better respect the properties of the input texture crop.

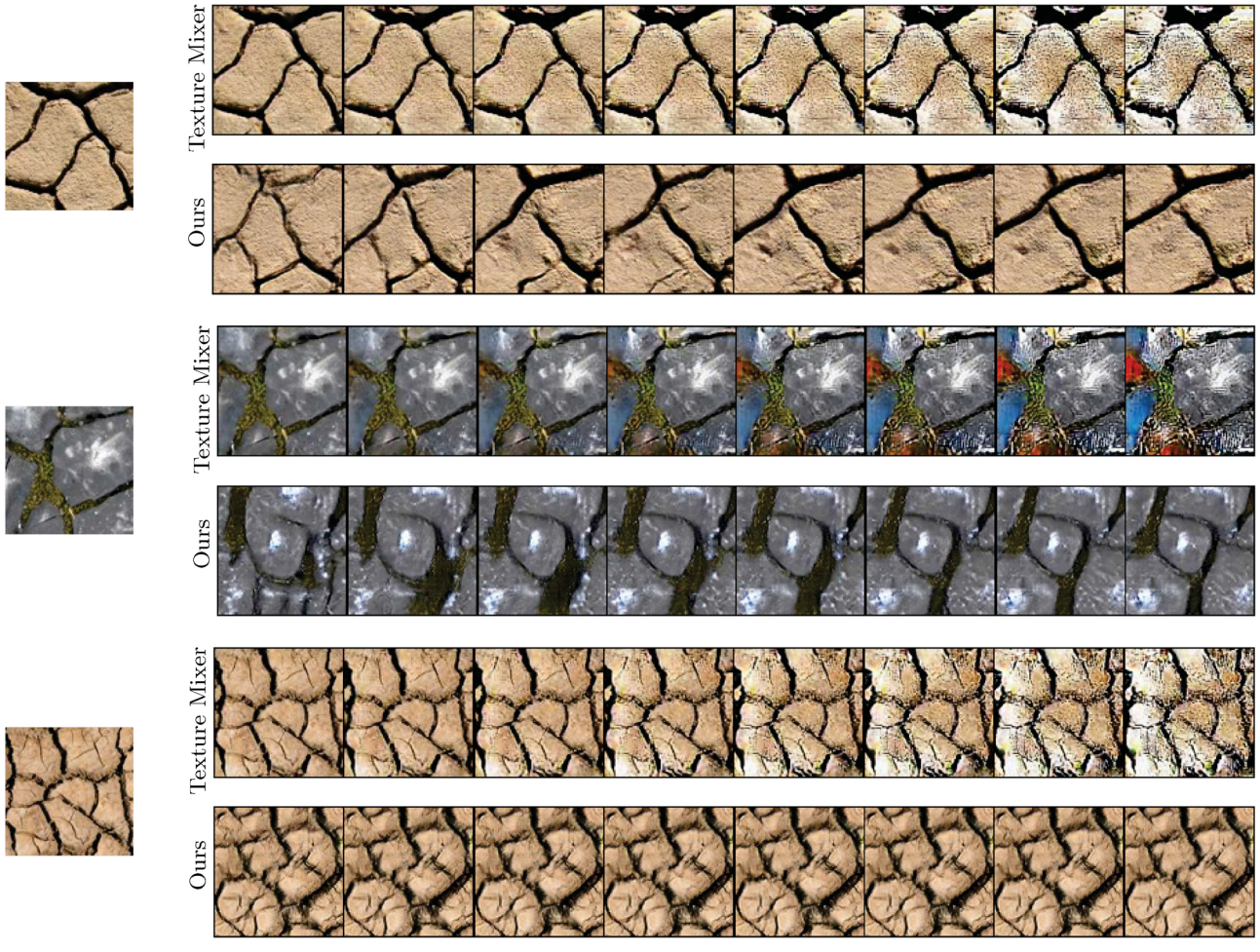


Fig. 9 Stability comparison. Given an input crop (left), we iteratively generate a sample with Texture Mixer [29] (top rows) and with our method (bottom rows). While Texture Mixer suffers from severe artifacts along the iterations, our texture synthesis method remains stable.

discriminator or not using the co-occurrence loss degrades the fidelity of generated texture samples: their properties differ from those of the reference crop from which the co-occurrence was collected.

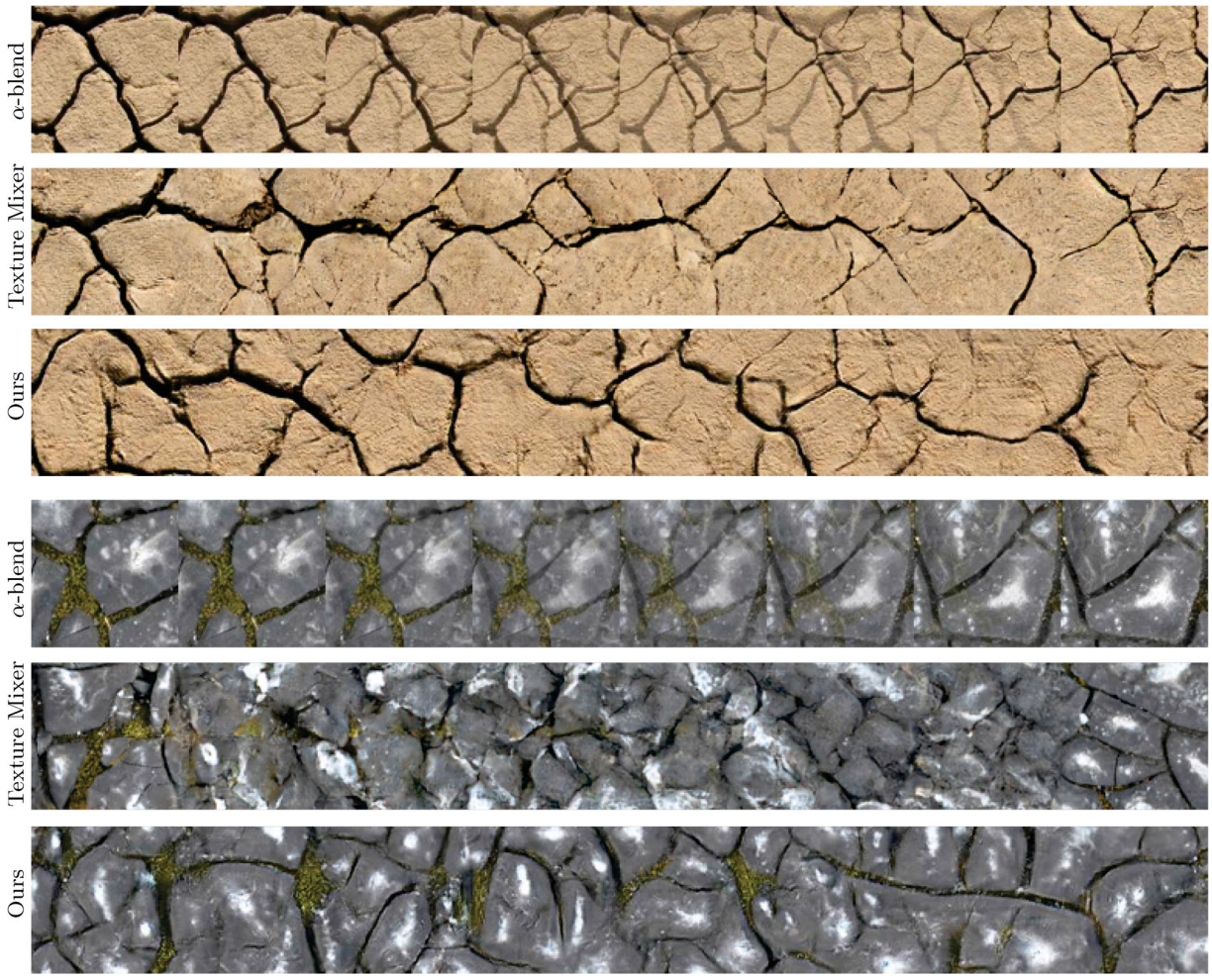


Fig. 10 Interpolation comparison. We compare interpolation results obtained with α-blend (top rows), Texture Mixer [29] (middle rows), and our method (bottom rows). The interpolation stripes are of size 1024×128 pixels. As illustrated above, our interpolations tend to be smoother and less repetitive than those of Texture Mixer.

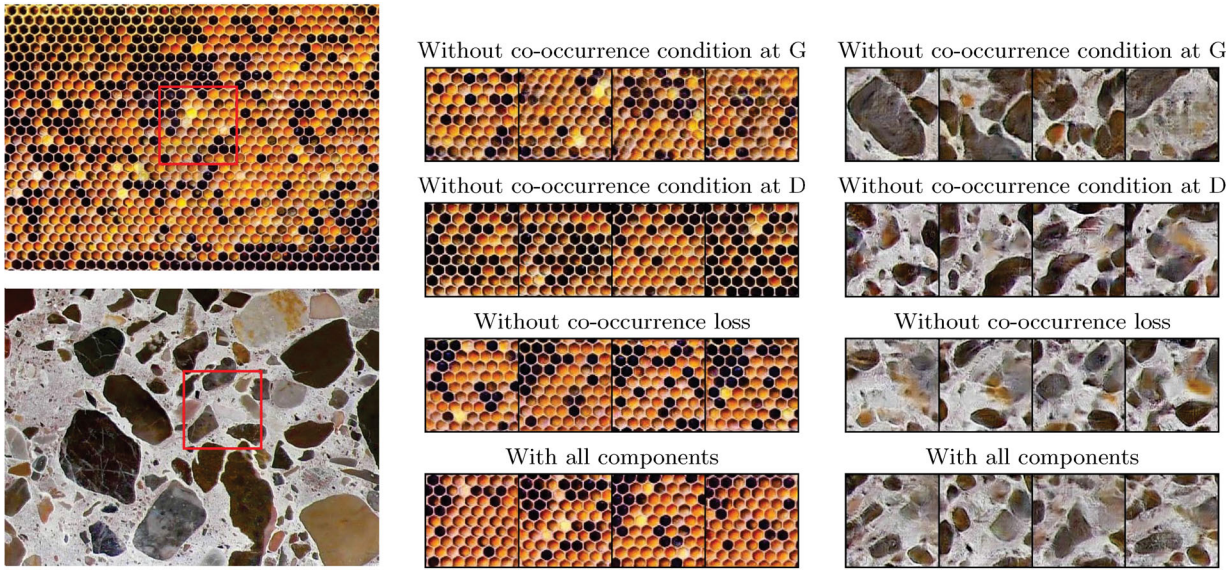


Fig. 11 Ablation study. Each row on the center and right shows several synthesized texture samples, conditioned on the co-occurrence of the reference crop, marked on the texture exemplars on the left. Each time one component of our method is turned off. G and D stand for Generator and Discriminator, respectively. When one of the components is missing, the fidelity to the input co-occurrence is compromised. When all the components are used, the generated samples better respect the properties of the reference crop.

When utilizing all components, our method can better preserve local texture properties, represented by the co-occurrence statistics.

4.5 Limitations

The quality of our results degrades when texture elements are larger than the crop size we use (128×128 pixels). The method also fails to capture texture characterized by global structure. We show examples of such textures in Fig. 12. Additionally, for jointly modeling multiple textures, we believe that larger models and additional optimization are needed. This is further discussed in the ESM.

Fig. 12 Limitation example. When the input texture includes elements of scale larger than our crop size of 128 × 128, they are not preserved in the generated samples.

5 Applications

We next demonstrate our co-occurrence based texture synthesis in the context of several applications: an interactive texture editing framework, dynamic texture morph generation, and controllable textures.

5.1 Interactive texture editing

Our technique lets users edit locally generated texture by modifying the corresponding co-occurrence matrix. A user can modify, for instance, a single bin $M(\tau_i, \tau_j)$ in the co-occurrence matrix. We can then generate a texture image with the modified co-occurrence (after normalizing the matrix to sum to 1). In Fig. 13, we demonstrate texture sequences obtained by increasing a single bin in the co-occurrence matrix.

We conducted a user study to quantitatively evaluate how easy it is to use our interactive texture editing approach for users who are unfamiliar with co-occurrence statistics. Participants were first provided with a brief introduction to co-occurrences using the toy example illustrated in Fig. 3. Then, they were presented with image triplets containing an original texture image and two edited images. The participants were asked to match the edited images with their co-occurrence matrices. The co-occurrence matrix corresponding to the original image was provided. Participants were also asked to rank how confident they were in their answer using a 5-point Likert scale.

Seventy-two users participated in the study. Each participant was shown image triplets generated from 5 different textures and the number of clusters k was set to either $k = 2$ or $k = 4$. In Fig. 14, we provide a few sample questions from our user study.

On average, 83% of the time the participants successfully matched the image to its corresponding co-occurrence matrix. The participants were confident in their selection 72% of the time. For these confident selections, the success rate was 86% on average. As our study illustrates, in most cases users understand how an edit in the image is reflected in the co-occurrence matrix. Therefore, directly editing the co-occurrence matrix can provide meaningful local control of the texture generation process.

5.2 Texture morphing

We generate a dynamic texture morph by interpolating and extrapolating between randomly

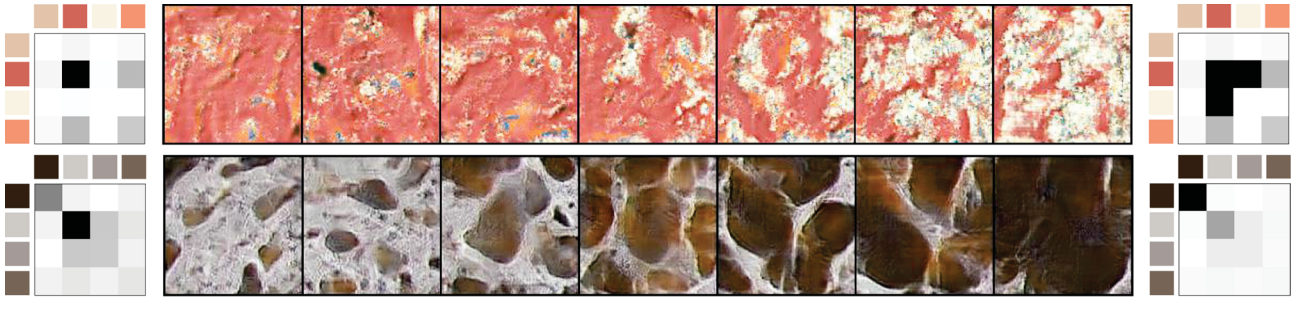


Fig. 13 Interactive texture editing. With our representation, a user can locally adjust the generated image by editing its corresponding co-occurrence matrix. On the left, we illustrate the initial result and its co-occurrence matrix. We demonstrate the sequence of results obtained by multiplying a selected bin with an increasing factor (and normalizing accordingly, as described in the text). The co-occurrence matrix on the right corresponds to the rightmost image. Note that darker colors correspond to pixel values which co-occur with high probability.

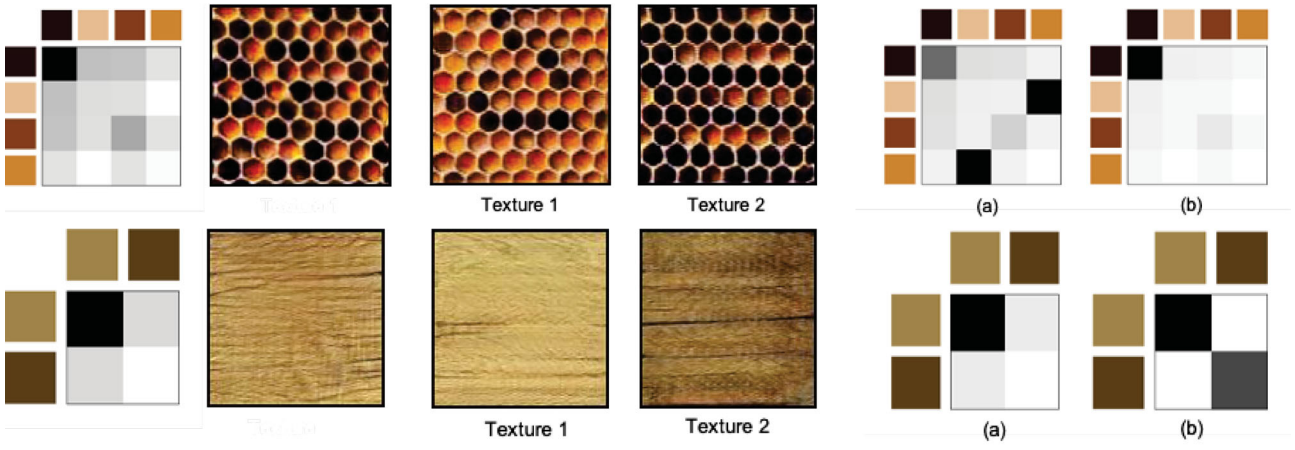


Fig. 14 User study. Given an original texture image and two different co-occurrence based edits, participants were asked to match between the edited textures (in the center) and their edited co-occurrence matrix (on the right). The correct matches are provided[①]. Note that darker colors correspond to pixel values which co-occur with high probability.

selected co-occurrence tensors of various local texture regions. The generated image sequence, obtained by conditioning on the smooth co-occurrence sequence and a fixed random noise seed, exhibits a unique temporal behavior. In Fig. 15, we illustrate a few representative frames along the sequence. We provide full dynamic sequences in the accompanying video in the ESM.

5.3 Large controllable textures

Our texture synthesis algorithm is fully convolutional and thus can generate arbitrarily large textures. In addition, since it is conditioned on co-occurrence, we have control over the local appearance of the synthesized texture.

In order to demonstrate this ability, we take two crops of 128×128 pixels with different properties from a texture exemplar. Then, we tile their corresponding co-occurrence tensors in a desired spatial arrangement, and run it through the generator. In this way we can obtain a texture with desired appearance.

Figure 16 shows large synthesized textures made in that manner. The fidelity to the co-occurrence

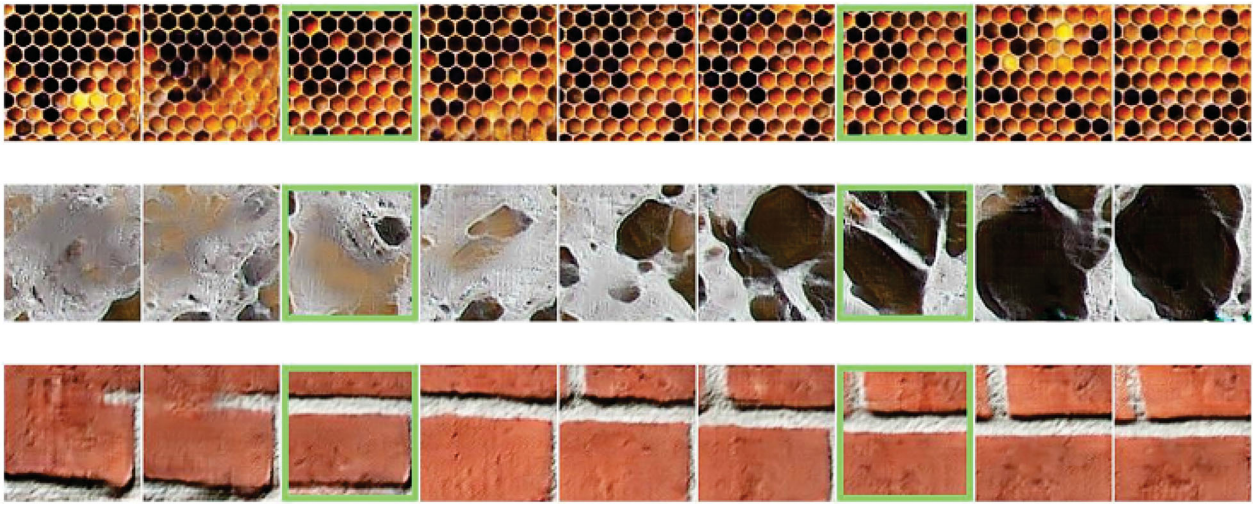


Fig. 15 Texture interpolation and extrapolation. Examples of interpolated and extrapolated textures, generated between two sample co-occurrence vectors (corresponding to the generated images marked in green). As the figure illustrates, the extrapolated examples extend the smooth interpolation sequence, magnifying the differences in the local texture properties. Please refer to the accompanying video in the ESM for the full dynamic sequences.

① The co-occurrence matrix corresponding to Texture 1 is (a) in both cases.

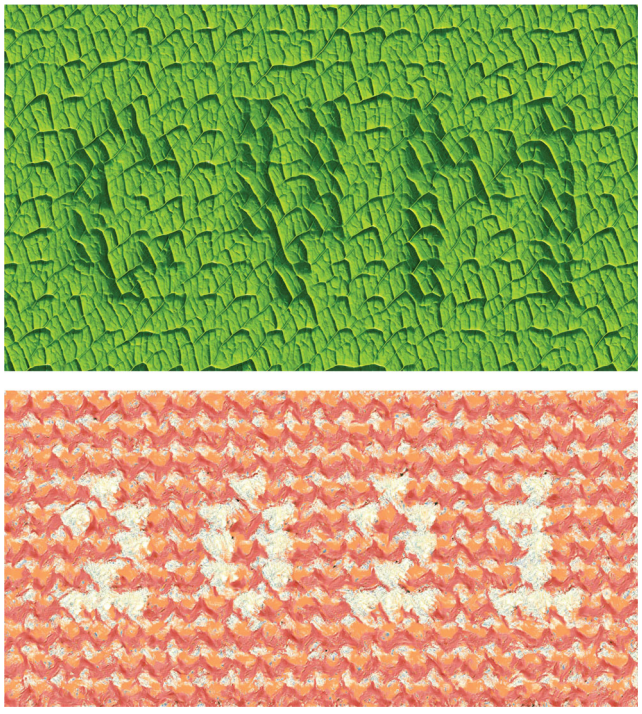


Fig. 16 Generating large textures with control. We can control the local appearance of the texture, by constructing the co-occurrence condition to the generator network. For these examples, we used co-occurrences from *only* two 128×128 crops of the original textures!

condition enables us to depict the word "CVM" and the number "2021". The diversity property of our algorithm enables the production of such large textures with a plausible appearance.

6 Conclusions

We have proposed a co-occurrence based texture synthesis method, where local texture properties are captured by a co-occurrence tensor. While the computation of the co-occurrence for a given texture crop is deterministic, the same is not true in the opposite direction: different texture crops can have similar co-occurrence statistics. As we have shown, our fully convolutional cGAN can learn an accurate and stable non-deterministic mapping from the co-occurrence space back to the image space.

We believe that co-occurrences strike the right balance between non-parametric methods that are difficult to control and manipulate and parametric methods that may have limited expressive power. Generally speaking, we hope to explore and learn about other deep frameworks where powerful traditional tools are integrated into neural networks to advance their respective fields.

Electronic Supplementary Material Supplementary material is available in the online version of this article at `https://doi.org/10.1007/s41095-021-0243-7`.

References

[1] Radford, A.; Metz, L.; Chintala, S. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint* arXiv:1511.06434, 2015.

[2] Bau, D.; Zhu, J. Y.; Strobelt, H.; Zhou, B. L.; Tenenbaum, J. B.; Freeman, W. T.; Torralba, A. Visualizing and understanding generative adversarial networks. *arXiv preprint* arXiv:1901.09887, 2019.

[3] Shen, Y. J.; Gu, J. J.; Tang, X. O.; Zhou, B. L. Interpreting the latent space of GANs for semantic face editing. *arXiv preprint* arXiv:1907.10786, 2019.

[4] Julesz, B. Visual pattern discrimination. *IEEE Transactions on Information Theory* Vol. 8, No. 2, 84–92, 1962.

[5] Haralick, R. M.; Shanmugam, K.; Dinstein, I. Textural features for image classification. *IEEE Transactions on Systems, Man, and Cybernetics* Vol. SMC-3, No. 6, 610–621, 1973.

[6] Isola, P.; Zoran, D.; Krishnan, D.; Adelson, E. H. Crisp boundary detection using pointwise mutual information. In: *Computer Vision–ECCV 2014. Lecture Notes in Computer Science, Vol. 8691*. Fleet, D.; Pajdla, T.; Schiele, B.; Tuytelaars, T. Eds. Springer Cham, 799–814, 2014.

[7] Heeger, D. J.; Bergen, J. R. Pyramid-based texture analysis/synthesis. In: Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques, 229–238, 1995.

[8] De Bonet, J. S. Multiresolution sampling procedure for analysis and synthesis of texture images. In: Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques, 361–368, 1997.

[9] Portilla, J.; Simoncelli, E. P. A parametric texture model based on joint statistics of complex wavelet coefficients. *International Journal of Computer Vision* Vol. 40, No. 1, 49–70, 2000.

[10] Efros, A. A.; Freeman, W. T. Image quilting for texture synthesis and transfer. In: Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques, 341–346, 2001.

[11] Kwatra, V.; Schödl, A.; Essa, I.; Turk, G.; Bobick, A. Graphcut textures. *ACM Transactions on Graphics* Vol. 22, No. 3, 277–286, 2003.

[12] Kwatra, V.; Essa, I.; Bobick, A.; Kwatra, N. Texture optimization for example-based synthesis. *ACM Transactions on Graphics* Vol. 24, No. 3, 795–802, 2005.

[13] Wexler, Y.; Shechtman, E.; Irani, M. Space-time completion of video. *IEEE Transactions on Pattern Analysis and Machine Intelligence* Vol. 29, No. 3, 463–476, 2007.

[14] Simakov, D.; Caspi, Y.; Shechtman, E.; Irani, M. Summarizing visual data using bidirectional similarity. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 1–8, 2008.

[15] Barnes, C.; Zhang, F. L. A survey of the state-of-the-art in patch-based synthesis. *Computational Visual Media* Vol. 3, No. 1, 3–20, 2017.

[16] Matusik, W.; Zwicker, M.; Durand, F. Texture design using a simplicial complex of morphable textures. *ACM Transactions on Graphics* Vol. 24, No. 3, 787–794, 2005.

[17] Rabin, J.; Peyré, G.; Delon, J.; Bernot, M. Wasserstein barycenter and its application to texture mixing. In: *Scale Space and Variational Methods in Computer Vision. Lecture Notes in Computer Science, Vol. 6667*. Bruckstein, A. M.; ter Haar Romeny, B. M.; Bronstein, A. M.; Bronstein, M. M. Eds. Springer Berlin Heidelberg, 435–446, 2012.

[18] Darabi, S.; Shechtman, E.; Barnes, C.; Goldman, D. B.; Sen, P. Image melding: Combining inconsistent images using patch-based synthesis. *ACM Transactions on Graphics* Vol. 31, No. 4, Article No. 82, 2012.

[19] Gatys, L. A.; Ecker, A. S.; Bethge, M. Texture synthesis using convolutional neural networks. In: Proceedings of the 28th International Conference on Neural Information Processing Systems, 262–270, 2015.

[20] Ulyanov, D.; Lebedev, V.; Vedaldi, A.; Lempitsky, V. S. Texture networks: Feed-forward synthesis of textures and stylized images. In: Proceedings of the International Conference on Machine Learning, 1349–1357, 2016.

[21] Sendik, O.; Cohen-Or, D. Deep correlations for texture synthesis. *ACM Transactions on Graphics* Vol. 36, No. 5, Article No. 161, 2017.

[22] Li, C.; Wand, M. Precomputed real-time texture synthesis with Markovian generative adversarial networks. In: *Computer Vision–ECCV 2016. Lecture Notes in Computer Science, Vol. 9907*. Leibe, B.; Matas, J.; Sebe, N.; Welling, M. Eds. Springer Cham, 702–716, 2016.

[23] Gang, L.; Gousseau, Y.; Xia, G. S. Texture synthesis through convolutional neural networks and spectrum constraints. In: Proceedings of the 23rd International Conference on Pattern Recognition, 3234–3239, 2016.

[24] Li, Y. J.; Fang, C.; Yang, J. M.; Wang, Z. W.; Lu, X.; Yang, M. H. Diversified texture synthesis with feed-forward networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 266–274, 2017.

[25] Zhou, Y.; Zhu, Z.; Bai, X.; Lischinski, D.; Cohen-Or, D.; Huang, H. Non-stationary texture synthesis by adversarial expansion. *ACM Transactions on Graphics* Vol. 37, No. 4, Article No. 49, 2018.

[26] Frühstück, A.; Alhashim, I.; Wonka, P. TileGAN: Synthesis of large-scale non-homogeneous textures. *ACM Transactions on Graphics* Vol. 38, No. 4, Article No. 58, 2019.

[27] Jetchev, N.; Bergmann, U.; Vollgraf, R. Texture synthesis with spatial generative adversarial networks. In: Proceedings of the Workshop on Adversarial Training, 2016.

[28] Bergmann, U.; Jetchev, N.; Vollgraf, R. Learning texture manifolds with the periodic spatial GAN. *arXiv preprint* arXiv:1705.06566, 2017.

[29] Yu, N.; Barnes, C.; Shechtman, E.; Amirghodsi, S.; Lukác, M. Texture mixer: A network for controllable synthesis and interpolation of texture. *arXiv preprint* arXiv:1901.03447, 2019.

[30] Yellott, J. I. Implications of triple correlation uniqueness for texture statistics and the Julesz conjecture. *Journal of the Optical Society of America A* Vol. 10, No. 5, 777–793, 1993.

[31] Jevnisek, R. J.; Avidan, S. Co-occurrence filter. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 3184–3192, 2017.

[32] Kat, R.; Jevnisek, R.; Avidan, S. Matching pixels using co-occurrence statistics. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 1751–1759, 2018.

[33] Gulrajani, I.; Ahmed, F.; Arjovsky, M.; Dumoulin, V.; Courville, A. C. Improved training of Wasserstein GANs. In: Proceedings of the Annual Conference on Neural Information Processing Systems, 2017.

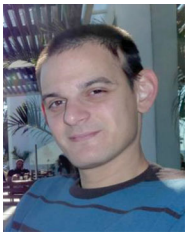
[34] Cimpoi, M.; Maji, S.; Kokkinos, I.; Mohamed, S.; Vedaldi, A. Describing textures in the wild. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 3606–3613, 2014.

[35] Bellini, R.; Kleiman, Y.; Cohen-Or, D. Time-varying weathering in texture space. *ACM Transactions on Graphics* Vol. 35, No. 4, Article No. 141, 2016.

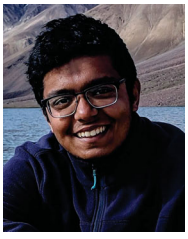
[36] Wu, F. Z.; Dong, W. M.; Kong, Y.; Mei, X.; Yan, D. M.; Zhang, X. P.; Paul, J.-C. Feature-aware natural texture synthesis. *The Visual Computer* Vol. 32, No. 1, 43–55, 2016.

[37] Kaspar, A.; Neubert, B.; Lischinski, D.; Pauly, M.; Kopf, J. Self tuning texture optimization. *Computer Graphics Forum* Vol. 34, No. 2, 349–359, 2015.

Anna Darzi received her M.Sc. degree in electrical engineering from Tel Aviv University in 2019, as a researcher in Avidan's lab. She completed her B.Sc. degree in electrical engineering at the Technion in 2016. She is currently working as an algorithm developer in computer vision, deep learning, and object detection.



Itai Lang is a Ph.D. researcher at Tel Aviv University, working with Avidan. In addition, he is a senior algorithm engineer at Samsung Israeli Research Center. He received his B.Sc. degree in electrical engineering and his B.A. degree in physics from the Technion, Israel, in 2005. In 2013, he completed his M.Sc. degree in electrical engineering at Tel Aviv University. His research interests include computer vision and learning methods for 3D geometry and images.



Ashutosh Taklikar is a graduate researcher at Tel Aviv University working with Avidan in the field of computer vision. He completed his B.Sc. degree in electrical engineering at Tel Aviv University in 2019, where he is currently pursuing his M.Sc. degree. His research mainly focuses on image generation and super-resolution tasks.

Hadar Averbuch-Elor is a postdoctoral researcher at Cornell-Tech working with Snavely as part of the Cornell Graphics and Vision Lab. She completed her Ph.D. degree in electrical engineering at Tel Aviv University in 2017. She received her B.Sc. degree in electrical engineering from the Technion in 2012. Her research focuses on modeling and manipulating visual concepts by combining pixels with more structured modalities, including natural language and 3D geometry.

Shai Avidan received his Ph.D. degree from the School of Computer Science, Hebrew University, Jerusalem, Israel, in 1999. He is currently a professor in the Faculty of Engineering, Tel Aviv University. In between, he worked for Adobe, Mitsubishi Electric Research Labs, MobilEye, and Microsoft Research. He has published extensively in the fields of object tracking in video and 3D object modeling from images. He is also interested in Internet vision applications such as privacy-preserving image analysis, distributed algorithms for image analysis, and image retargeting.