

# Exemplar-based Pattern Synthesis with Implicit Periodic Field Network

Haiwei Chen<sup>1,2</sup>, Jiayi Liu<sup>1,2</sup>, Weikai Chen<sup>3</sup>, Shichen Liu<sup>1,2</sup>, Yajie Zhao<sup>1,2</sup>

<sup>1</sup>University of Southern California

<sup>2</sup>USC Institute for Creative Technologies

<sup>3</sup>Tencent Game AI Research Center

{haiweich, jliu8328}@usc.edu chenwk891@gmail.com {lshichen, zhao}@ict.usc.edu

## Abstract

*Synthesis of ergodic, stationary visual patterns is widely applicable in texturing, shape modeling, and digital content creation. The wide applicability of this technique thus requires the pattern synthesis approaches to be scalable, diverse, and authentic. In this paper, we propose an exemplar-based visual pattern synthesis framework that aims to model the inner statistics of visual patterns and generate new, versatile patterns that meet the aforementioned requirements. To this end, we propose an implicit network based on generative adversarial network (GAN) and periodic encoding, thus calling our network the Implicit Periodic Field Network (IPFN). The design of IPFN ensures scalability: the implicit formulation directly maps the input coordinates to features, which enables synthesis of arbitrary size and is computationally efficient for 3D shape synthesis. Learning with a periodic encoding scheme encourages diversity: the network is constrained to model the inner statistics of the exemplar based on spatial latent codes in a periodic field. Coupled with continuously designed GAN training procedures, IPFN is shown to synthesize tileable patterns with smooth transitions and local variations. Last but not least, thanks to both the adversarial training technique and the encoded Fourier features, IPFN learns high-frequency functions that produce authentic, high-quality results. To validate our approach, we present novel experimental results on various applications in 2D texture synthesis and 3D shape synthesis.*

## 1. Introduction

The synthesis of visual patterns, may that be a wooden texture for painting, or a simulation of natural cave systems, is a technique that is applied ubiquitously in computer-aided design and digital content creation. Visual patterns can be understood as arts, shapes, or natural textures following certain geometric structures. In an application context, let us start by defining several characteristics that are desirable for

an algorithm that generates visual patterns:

- *Authenticity.* Probably the most prioritized quality of synthesized visual patterns is its visual quality. When patterns are synthesized from an exemplar, the quality is determined by whether they faithfully recreate the source pattern.
- *Diversity.* It would be undesirable for a synthesizer to only copy patterns from the source. Diversity is thus an equally important measurement that evaluates whether the synthesized patterns vary from the source and each other. We strive to achieve two different levels of diversity: the patterns should be diversified both *within* a generated sample and *across* samples.
- *Scalability.* As patterns are usually demanded at different and potentially large scales for many practical applications, we want a scalable synthesizer to be able to efficiently generate patterns of arbitrary size. Scalability is particularly valuable when it comes to the synthesis of 3D models, as the extra dimension translates to a much larger amount of computations.

A *scalable* design choice leads us to formulate the synthesis problem as generating patterns from a continuous, real coordinate space. This is generally known as the implicit formulation, where a nonlinear function maps points defined in  $\mathbb{R}^2$  or  $\mathbb{R}^3$  to features that represent the synthesized subjects. In particular, the implicit function has been shown to be an efficient representation for synthesizing 3D volumes [12, 22, 24].

Patterns that scale well to an infinitely large space, in general, possess a stationary property - a shift-invariant structure that can be expanded by tiling or stacking blocks of elements. We therefore develop our method by pivoting on the fact that many types of natural and artistic patterns can be analyzed and recreated in a stationary framework. The goal of synthesizing an *authentic* and *diverse*

stationary pattern from an exemplar, however, requires careful modeling that is compatible with the underlying structure of the pattern.

Generative adversarial networks (GAN) [9] is one of the most promising techniques so far to model data distribution in an unsupervised manner and has been frequently adapted to convolutional models that synthesize visually authentic images [2, 14, 31, 32, 37, 42]. How would a GAN generator be leveraged to modeling stationary pattern? As all stationary patterns contain a repeating structure with local variations that “vivifies” its appearance, in an ideal situation, a stationary pattern can be modeled by a discrete random field, where each random variable is associated with a patch of the basic element. Thus a natural GAN formulation models image patches with a spatially defined latent field [2, 14]. In a convolutional framework, however, problems arise when fake samples generated from a discrete noise image are discriminated from randomly sampled patches from a real image. The first problem is that the sampled patch does not necessarily agree with the scale of the repeating structure. The second problem is that the sampled patch can be arbitrarily shifted from the center of a stationary element. A typical deconvolutional network [6, 39] that upsamples from an evenly-spaced noise image may not sufficiently address the previously mentioned problems. To study their effects we designed a convolutional network following the DCGAN architecture [29] to synthesize a honeycomb pattern from a  $2 \times 2$  noise map, which is trained against random patches sampled from a source image. The comparison between its result and that synthesized by our generator network that is trained with an identical discriminator is shown in Figure. 1. We found that the noise map does not capture well the honeycomb structure as seams and intercepting elements are visible from the synthesized image.

Though various techniques have been proposed in the past to address the aforementioned issues (e.g. [2, 14]), in this paper, we consider a more natural way to model stationary patterns with an implicit periodic generator. The core of the formulation is to match the repeatable structure of a stationary pattern to the period of a learnable continuous periodic function. Instead of modeling the pattern with a discrete noise tensor, we define latent variables in a continuous space, where the extent of each latent factor is learned to match with the extent of the repeating elements. The benefits of this design align well with the desirable characteristics for visual pattern synthesis: 1) learned periodicity of the implicit field encourages latent factors to model the stationary variations observed from the exemplar pattern; 2) a continuous representation provides flexibility during training to learn a distribution from randomly shifted patches cropped from the exemplar; 3) a Fourier encoding scheme learns high-frequency details from the exemplar. This al-

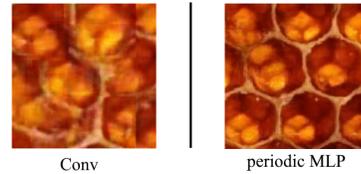


Figure 1. Comparison between synthesized honeycomb from a DCGAN convolution generator and the periodic MLP generator. Seams and intercepting patterns are visible in the former result due to difficulty for the convolution generator to capture the repeating structure.

lows our model to synthesize visually authentic results, and 4) multilayer perceptron (MLP) that implicitly takes coordinates as input scales well to the generation of 3D shapes when compared to 3D convolution. Based on these design choices, we term our network the Implicit Periodic Field Network (IPFN).

We validate our proposed design by showing various applications in texture image synthesis and 3D volume synthesis in Section. 4. Specifically, besides synthesizing stationary patterns, we design a conditional formulation of our model to tackle the synthesis of directional patterns and to provide application in controllable shape generation. An ablation study is also conducted to verify the effectiveness of our design choices.

## 2. Related Work

**Pattern Synthesis** 2D visual patterns are generally referred to as “texture” due to their prevalent applications in computer-aided design. Well-known early attempts to synthesize texture derive patterns from smoothly interpolated noise [25, 26, 36] and create aesthetically pleasing materials that display a high level of randomness. [12, 27] are two recent works related to us that utilize randomness from a continuously-defined Perlin noise [25] to synthesize exemplar-based textures in an implicit field. Their works demonstrate the advantage of smooth noise field and implicit formulation in efficiently and diversely generating a 3D texture field. However, just like their procedural precedence, these methods have been shown to be limited in synthesizing patterns that are more complicated in structures (e.g. bricks, rocky surface) [27].

Later works in traditional image synthesis are generally divided into pixel-based method (e.g. [8, 35]), patch-based method (e.g. [7, 16, 18]) and optimization-based method (e.g. [16, 17]) and have shared with us important considerations for recreating new patterns from an exemplar. For instance, synthesis on a patch level encourages preservation of fine local details, and the efforts are focused on “quilting” the discontinuity between patches [7, 18] and encouraging global similarity [16]. Early statistical model [8, 28] utilizes a random field representation that cap-

tures variations in stationary patterns and synthesizes a variety of new patterns under the Julesz’ conjecture. Our work is inspired by key ideas from the traditional modeling of stationary textures, while we strive to unify authenticity, diversity, and scalability with a neural representation to overcome limitations that existed in the traditional approaches.

Compared to earlier parametric models, the artificial neural network is powerful in its generalizability in pattern recognition [11]. Thus neural synthesis of texture images has marked the advances in recent endeavors. How does a neural network learn the stylish representation of a texture without simply reconstructing it? A milestone that unifies both synthesized quality and sampling power adversarially trains a generator network and a discriminator network to learn the mapping from a latent space to the texture distribution [9]. We are particularly interested in the generative adversarial networks (GANs) that models the inner statistics of the exemplars. This is marked by a patch-based approach that represents an image as a collection of smaller images: [14, 19] formalizes patch-based synthesis in the GAN setting with the concept of Markovian and spatial GAN. [2] motivated us with its periodic design in latent codes, which effectively captures stationary patterns from an input exemplar. [42] can be seen as a complement to our design by focusing on addressing non-stationary texture synthesis by learning to expand an image patch. In addition, [31, 32] present multi-scale, global-focused approaches to effectively recreate natural images. While the aforementioned approaches all utilize convolutional designs, our work extends texture synthesis to the continuous domain with an implicit representation of textures, as we argue that such representation provides a more natural and efficient way to synthesize stationary patterns.

The synthesis of 3D shapes is of particular interest in computer graphics and thus has a rich history. To name a few, this includes volumetric field design [4, 23, 40], procedural generation [13, 21] and 3D texturing [3, 41]. However, very few works have considered the synthesis of 3D patterns with neural networks, with the exception of [12, 27], which explores the generation of 3D solid textures.

**Implicit Network** Implicit network refers to multilayer perceptron that learns a continuous function in real coordinate space. In particular, the implicit network is mainly utilized in the reconstruction of 3D shapes [20, 24, 30, 33], where shapes are represented by a distance function, or a radiance field [22, 38]. We are motivated by the signed distance representation of shapes and the Fourier encoding explored in [22, 34] in our design of the implicit networks, and our work adopts these features to a generative setting where novel 3D patterns are synthesized.

### 3. Method

Our method is best introduced by expanding from the Wasserstein GAN value function [10] constructed as the *Earth-Mover* distance between the real data distribution  $\mathbb{P}_{data}$  and the generator distribution  $\mathbb{P}_g$ :

$$\min_G \max_D \mathbb{E}_{\mathbf{x} \sim \mathbb{P}_{data}} [\log(D(\mathbf{x}))] - \mathbb{E}_{\mathbf{z} \sim \mathbb{P}_Z} [\log(D(G(\mathbf{z})))] \tag{1}$$

Our first change of the above objective is to draw real-valued coordinates  $\mathbf{c} \in \mathbb{R}^k$  ( $k = 2$  for image and  $k = 3$  for volume) from a distribution  $\{s\mathbf{c} \mid \mathbf{c} \sim \mathbb{P}_c\}$  as input to the generator, where  $s$  is a constant scalar. This underlies the implicit nature of the generative network, as  $G$  learns a mapping from the real coordinate space in a stochastic process. Instead of being sampled from a prior distribution  $\mathbb{P}_Z$ , latent variables are drawn from a random field  $f_z(\mathbf{c})$  defined on the real coordinate space.  $G$  is therefore a function of both the coordinates  $\mathbf{c} \in \mathbb{R}^k$  and the latent variables  $f_z(\mathbf{c}) \in \mathbb{R}^d$ . This updates Eq.1 to

$$\min_G \max_D \mathbb{E}_{\mathbf{x} \sim \mathbb{P}_{data}} [\log(D(\mathbf{x}))] - \mathbb{E}_{\mathbf{c} \sim \mathbb{P}_c, \mathbf{z} \sim \mathbb{P}_{f_z(\mathbf{c})}} [\log(D(G(\mathbf{z}, \mathbf{c})))] \tag{2}$$

In the implementation, our goal is to synthesize color, distance function, normal vectors, etc. that are defined in a grid-like structure  $X : \mathbb{R}^{H \times W \times C}$  or  $X : \mathbb{R}^{H \times W \times D \times C}$  and thus we also sample grid coordinate input  $C : \mathbb{R}^{H \times W \times 2}$  or  $C : \mathbb{R}^{H \times W \times D \times 3}$  whose center is drawn from a uniform distribution  $U(-s, s)$ . The randomness to the grid center position is critical in encouraging smooth and seamless transition between blocks of patterns as it models the distribution of randomly sampled patches from the input pattern.

In the following sections, key modules are discussed in detail: 1) a deformable periodic encoding of the coordinates to model stationary patterns; 2) implementation of the latent random field; 3) a conditional variance of our objective for the synthesis of directional exemplar and controllability of the synthesized patterns, and 4) the overall network structure that completes our pattern synthesis framework.

#### 3.1. Periodic Encoding

As discussed in the introduction, it is critical to represent stationary patterns from the input by a repeated structure that avoids simply reconstructing the original exemplar. Simply mapping spatial coordinates to the visual pattern does not satisfy this requirement: since each coordinate is unique in the real-value space, the network would learn to overfit the coordinates to their associated positions in the exemplar and therefore fail to capture a repeatable structure.

The benefits of a periodic encoding to the coordinates are two-fold: Firstly, it disentangles patch-level appearance from their specific position in the exemplar, which

allows the pattern synthesized by the generator to be shift-invariant. Secondly, recent advances in implicit network [22, 33, 34] have found a Fourier feature mapping with a set of sinusoids effective in learning high-frequency signals by addressing the ‘‘spectral bias’’ [1] inherent to MLPs. In our work, we use the following periodic mapping for the input coordinates:

$$\gamma(\mathbf{c}) = [\cos(2^0\pi a\mathbf{c}), \sin(2^0\pi a\mathbf{c}), \dots, \cos(2^i\pi a\mathbf{c}), \sin(2^i\pi a\mathbf{c})], \quad (3)$$

where the learnable parameter  $a \in \mathbb{R}^k$  determines the period of the encoding. This design allows the network to learn to match the period of the encoding to the repeatable structure of the exemplar. It thus provides robustness to the scale of the patches sampled in training as such scale no longer dictates the period of the synthesized pattern.

### 3.2. Latent Random Field

In a generative framework, latent noise sampled from a prior distribution model the variation in observations. A noise function that is smoothly defined in the real coordinate space encourages smooth transition between the synthesized patches. In a 2D example, we start with a discrete random field that maps a uniform grid of coordinates to random variables  $\{f_z(\mathbf{c}) \mid \mathbf{c} : \mathbb{R}^{H \times W \times 2}\}$ . Then the discrete random field is smoothly interpolated to form a smooth latent field (see the visualization in Figure 2). In our implementation, we used an exponential interpolation:

$$f(\mathbf{x}) = \sum_{i=1}^K w_i(\mathbf{x}) f_z(\mathbf{c}_i), \quad w_i(\mathbf{x}) = \frac{e^{-\frac{\|\mathbf{x}-\mathbf{c}_i\|_2}{\sigma}}}{\sum_{j=1}^K e^{-\frac{\|\mathbf{x}-\mathbf{c}_j\|_2}{\sigma}}}, \quad (4)$$

where the latent code at spatial position  $\mathbf{x}$  is interpolated from  $K = 4$  latent vectors defined at the grid corners. In implementation, the discrete grid used to define the random field has a spacing of 1. To match the extent of a latent factor with the learned period  $a$ , we simply scale the uniform grid of the discrete random field accordingly.

### 3.3. Conditional IPFN

Extending our GAN objective to be conditional enables many practical applications. Assume each input patch is paired with a guidance factor  $\mathbf{g}$ , the conditional objective is simply an extension:

$$\min_G \max_D \mathbb{E}_{\mathbf{x} \sim \mathbb{P}_{data}} [\log(D(\mathbf{x}|\mathbf{g}))] - \mathbb{E}_{\mathbf{c} \sim \mathbb{P}_c, \mathbf{z} \sim \mathbb{P}_{f_z(\mathbf{c})}} [\log(D(G(\mathbf{z}, \mathbf{c}|\mathbf{g})))]. \quad (5)$$

Here we outline two applications in pattern synthesis using the conditional formulation:

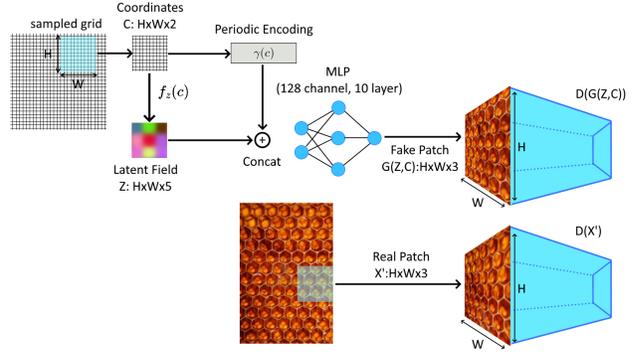


Figure 2. Overview of our network architecture discussed in Section 3.4.

- **Synthesis of directional pattern:** Many natural patterns have a directional distribution that is oftentimes considered non-stationary. A typical example is a leaf texture - a midrib defines the major direction that separates the blade regions by half (see Figure 3). A conditional extension of our model is able to model patch distribution along a specified direction. For simplicity, we present a 2D formulation that can be easily extended to 3D. With a user-defined implicit 2D line equation  $ax+by+c=0$ , the guidance factor is defined as  $\mathbf{g}(x, y) = ax + by + c$ . Pixel coordinates  $(p_x, p_y)$  from an input texture image with width  $w$  and height  $h$  are transformed as  $(x, y) = (\frac{2p_x}{w} - 1, \frac{2p_y}{h} - 1)$  to be normalized to the value range of  $[-1, 1]$ . In our experiments we have found it sufficient to condition our model on the horizontal ( $y = 0$ ) and vertical direction ( $x = 0$ ) for the evaluated exemplars.
- **Controlling synthesis of 3D shapes:** In the modeling of geometric patterns, it is often desirable for the synthesis algorithm to provide explicit control of certain geometric properties such as density and orientation. These geometric properties can be calculated from the exemplar shape. Let  $g(x)$  be a shape operator that defines the geometric property of interest, our conditional model trained with the sample pair  $(x, g(x))$  then learns a probabilistic mapping from a guidance vector field to the target 3D shape. An intuitive example of this application can be found in Section 4.5.

### 3.4. Network Structure

The overall structure of IPFN is visualized in Figure 2. The Generator Network  $G$  is a 10-layer MLP with ReLU activations between layers and a sigmoid function in the end. A grid of coordinates is sampled based on a randomly shifted center. The coordinates are then passed to two separate branches: 1) the periodic encoder, and 2) a projection on the latent field to obtain a 5-dim latent vector for each coordinate. The latent codes and the periodically encoded

coordinates are then concatenated as input to the generator mlp, which outputs the fake sample. The discriminator  $D$  discriminates between the generated samples and randomly cropped patches from the real input. We implement  $D$  by following the DCGAN architecture [29] with a stride of 2. For discriminating 3D volumes, 2D convolution layers are replaced by 3D convolution.

## 4. Experiments

We hypothesize that our approach is most suitable for synthesizing texture patterns and 3D shapes with repeating structures and local variations in appearance. To this end, we demonstrate our main results by applying IPFN to the synthesis of 2D texture images and 3D structured shapes. In addition, IPFN is adapted to two applications in 3D texturing and shape manipulation. To evaluate the effectiveness of the proposed techniques, we have also conducted an ablation study where several key designs are altered.

**Evaluation metric** While the quality for pattern synthesis is not easily quantifiable, human eyes usually provide a reasonable qualitative assessment for whether the synthesized patterns capture the aesthetics and structure of the exemplar. In our evaluation, we present comparisons of visual results that are self-manifesting, since the synthesized patterns bear obvious characteristics of the underlying designs of the synthesizer. In addition, we have provided quantifiable metrics in terms of Single Image Fréchet Inception Distance and inference time and memory.

**Implementation details** For all of our experiments, the network is optimized under WGAN loss with gradient penalty [10]. Adam optimizer [15] is used with a learning rate of  $1e-4$  for both the discriminator  $D$  and the generator  $G$ . In each iteration, both  $D$  and  $G$  are updated for 5 steps sequentially. Input images and volumes are randomly cropped to a smaller-scale patch. For positional encoding, we choose a bandwidth  $i = 5$  as a wider kernel tends to produce sinusoidal artifacts, whereas a narrower kernel produces blurry results. The input coordinates are randomly shifted by an offset in the range  $[-4, 4]$  to accommodate for the chance that the network may learn an increased period for the periodic encoding. Accordingly, noises are interpolated from a  $5 \times 5$  grid ( $5^3$  for 3D volume) discrete random field, where the point locations are ranged between  $[-5, 5]$ . A single-exemplar experiment is typically trained for 12,500 iterations with a batch size of 8 and runs on a single Nvidia GTX 1080 GPU, which takes about 6-9 hours to complete. **Inference Time:** IPFN only requires 24 milliseconds to generate a  $1024 \times 1024$  image. 3D volumes are generated iteratively and a large-scale  $512^3$  volume takes only 22.9 seconds to be generated. Source code will be made publicly available upon acceptance.

### 4.1. Texture Pattern Synthesis

Image sources selected from the Oxford Describable Textures Dataset (DTD) [5] are shown in Figure 3. Specifically, we selected two exemplars with stationary patterns (top 4 rows in Figure 3) and two exemplars with directional patterns (bottom 4 rows in Figure 3) to demonstrate that IPFN synthesizes visually similar patterns in both cases. During training, images were randomly cropped into patches of size  $128 \times 128$ . During inference, the synthesized images were scaled up four times to a size of  $512 \times 512$ . Our results are compared to the three most relevant baseline generative methods that synthesize texture patterns from a single image:

- Henzler et al. [12]: A method that similarly utilizes implicit network and smooth noise field for texture synthesis. The synthesized results were obtained from running the officially released code.
- Bergmann et al. [2]: A convolutional method that combines noise with periodic signals to synthesize stationary pattern. The synthesized results were obtained from running the officially released code.
- Zhou et al. [42]: A convolutional image expansion approach targeted for non-stationary texture synthesis. Since [42] expands from an image input deterministically and does not utilize latent code, only one synthesized result is shown per row. The synthesized results were obtained from the authors.

Visual inspection is sufficient to show that IPFN provides promising results. When compared to [12], IPFN synthesized results with obvious structures as noise is not directly mapped to the output. While [2] synthesizes periodic samples that display diversity across samples and similarity to the stationary exemplars, their synthesized patterns lack variation within the image. In comparison, our synthesized patterns show a higher level of local variations and adapt well to the directional cases. [42] has provided the most visually authentic results among the baselines. However, in the stationary cases, radial distortion is noticeable near the boundaries of its synthesized images. Moreover, without requiring image input, IPFN provides a more direct approach to synthesizing diversified samples from random noise.

	honey	crosshatch	rock	leaf
Henzler [12]	332.66	310.49	351.23	225.11
Bergmann [2]	62.75	177.88	120.64	164.37
Zhou [42]	14.54	154.63	118.29	<b>38.13</b>
Ours	<b>10.15</b>	<b>130.83</b>	<b>113.81</b>	103.6

Table 1. SIFID scores between the exemplars and the generated patterns from ours and different baselines.

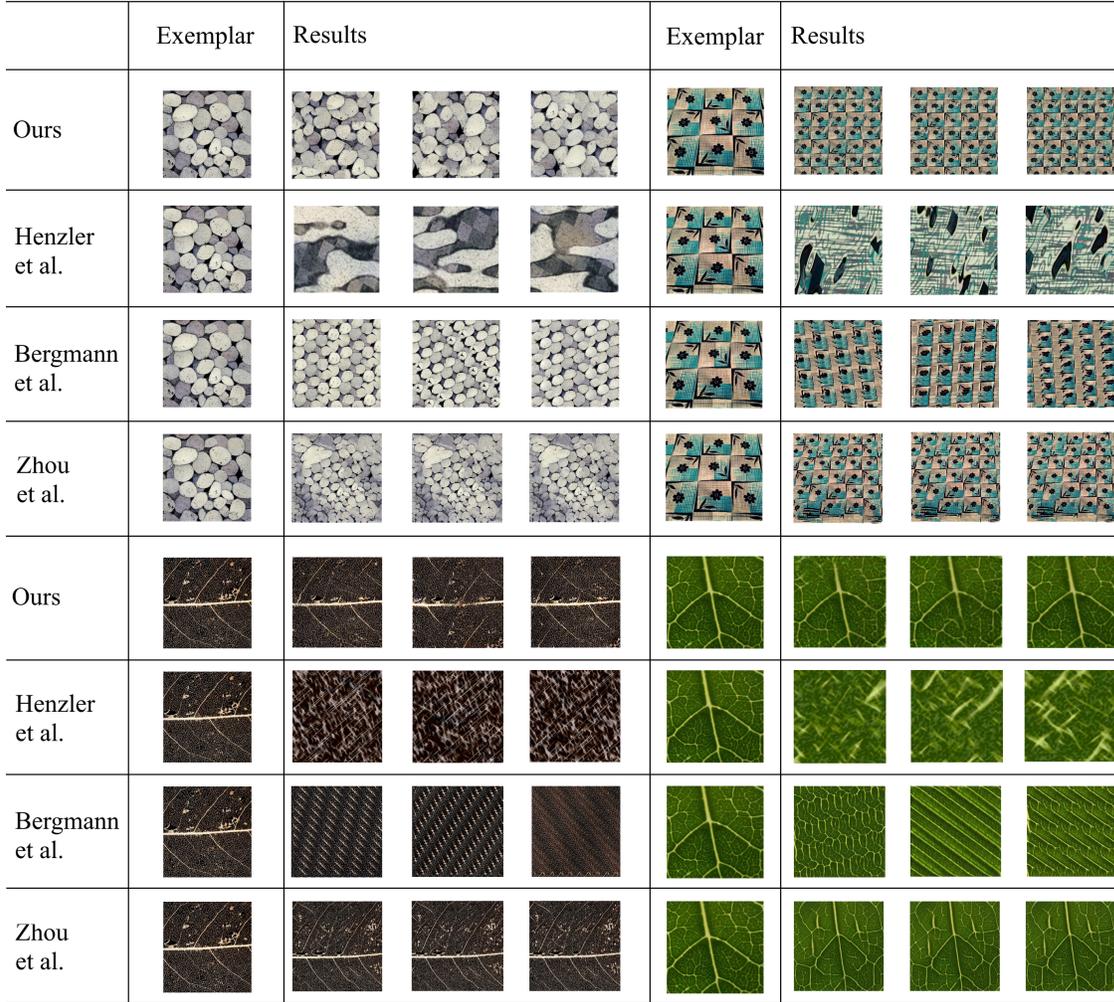


Figure 3. Main results for 2D texture synthesis with comparisons to Henzler et al. [12], Bergmann et al. [2], and Zhou et al. [42] on synthesizing two stationary patterns (top four rows) and two directional patterns (bottom four rows).

**Single Image Fréchet Inception Distance (SIFID)**  
SIFID introduced in [31] is a metric commonly used to assess the realism of generated images. For the computation of SIFID, we have used a patch size of  $128 \times 128$  in all experiments, where the synthesized patterns have the same resolution as the original exemplars. Table 1 shows the SIFID comparisons between ours and the baselines in various categories of exemplars. For Zhou et al. [42], only the generated (expanded) portion of the images were used. The results show that our method can generate results that better resemble the distribution of the real texture in the stationary categories (honey, crosshatch, rock) as our generated patterns receive lower SIFID scores. For the leaf category, a typical directional pattern, Zhou et al. [42] achieves the best performance as its method specifically targets non-stationary expansions, while our method still performs bet-

Time (ms) / memory (GB)	$128^2$	$256^2$	$512^2$	$1024^2$
Henzler [12]	218/1.38	278/1.62	328/2.72	458/6.45
Bergmann [2]	7/2.37	13/5.79	42/19.68	115/31.88
Zhou [42]	356/1.20	349/1.34	510/2.00	612/4.66
Ours	<b>8/0.76</b>	<b>11/0.85</b>	<b>15/1.23</b>	<b>24/2.81</b>

Table 2. Comparisons of inference time and inference memory consumption, measured in milliseconds (ms) / gigabytes (GB), when patterns of increasing size (top row) are generated.

ter than other baselines that have not taken into consideration the synthesis of non-stationary patterns.

**Inference Time and Memory** Table 2 measures the inference time and memory consumption of our network compared to the baselines when generating image at different sizes. Our implicit formulation is shown to be significantly

more efficient in both time and space without the needs to rely on computation of pseudo-random noise ([12]) or convolutional operations ([2, 42]). This validates our claim on the scalability of our design.

### 4.2. Ablation Study

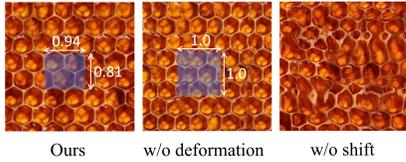


Figure 4. Synthesized honeycomb textures for the ablation study. The blue boxes represent the learned scale of periodic encoding in ours, where in *w/o deformation*, the period is default to 1, which does not match with the repeating structure of the honeycomb pattern and results in visual artifact.

To validate our design choices, we have conducted an ablation study by removing two designs that we consider critical for our network to be effective. The comparison results are shown in Figure 4. *w/o deformation* is a network model that encodes input coordinates without the learnable parameters  $a$  described in Section 3.1. *w/o shift* is a model that is trained without randomly shifting the input coordinates. The resulted patterns are indicative of the effects of these designs: when coordinates are encoded at a fixed scale, the *w/o deformation* model generates hexagons that are seemingly glued together as the presumed scale does not match with the actual period of the repeating structure. The *w/o shift* model synthesized distorted patterns as we speculate that, without the random sampling of the input coordinates, the network faces difficulty in matching the patch-based priors of the image patches.

### 4.3. Volumetric Shape Synthesis

For the evaluation on volumetric shape synthesis, We have obtained two 3D models from *turbosquid.com*: a porous structure (Figure 5.a) and a foam structure ((Figure 5.d). The 3D meshes are preprocessed into signed distance fields by uniformly sampling points in a volumetric grid. For the porous structure, we have sampled  $256 \times 256 \times 256$  points and extracted  $64 \times 64 \times 64$  patches during training. For the foam structure, we have sampled  $200 \times 200 \times 128$  points and extracted  $32 \times 32 \times 32$  patches during training. During inference, porous structures are synthesized at their original resolution, while we scale the synthesized foam structures to be twice as large as the original shape in the XY direction. Figure 5.c and Figure 5.f show the synthesized shapes. For the porous structures, both outer structures and interior structures are learned (see Figure 5.b for zoom-in interior views) and the structures are diversified both across and within samples. For the foam structures, we have shown different results by varying the

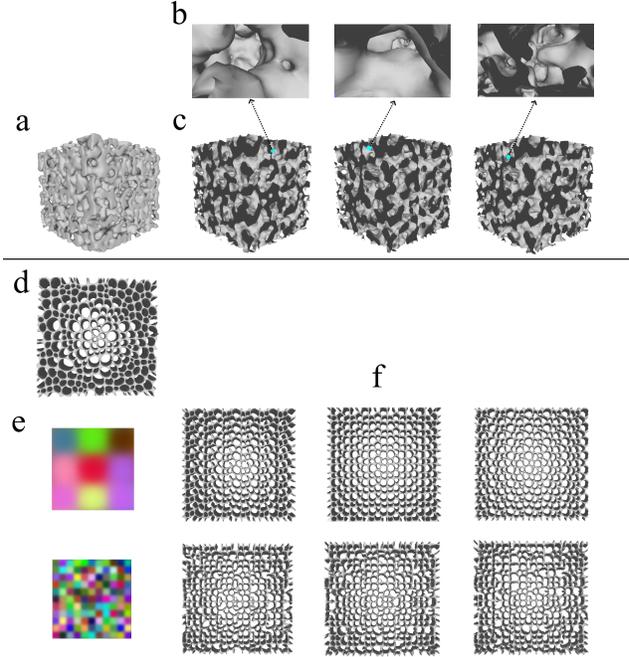


Figure 5. Main results for 3D volume synthesis. **a.** Exemplar porous structure. **b.** Synthesized structure models interior tunnels. **c.** Global views of synthesized porous structures. **d.** Exemplar foam structure. **e.** Two scales of noise fields for the foam structure synthesis. **f.** Synthesized foam structures. Larger scale of the noise field leads to more isotropic foam structures.

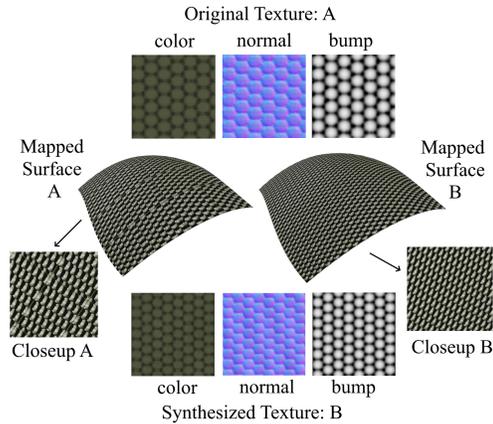


Figure 6. IPFN learns multi-channel textures that are applicable to seamless 3D texturing. The original 3D texture in this example is not symmetric and therefore visible seams can be found on the texture-mapped surface and in the closeup view (A in figure). As synthesized patterns learnt from this exemplar can be tiled in any direction, the mapped surface (B in surface) is seamless.

extent of the latent random field (see Figure 5.e). A larger-scale random field encourages the synthesizer to generate globally varied structures, whereas a smaller scale produces locally anisotropic structures.

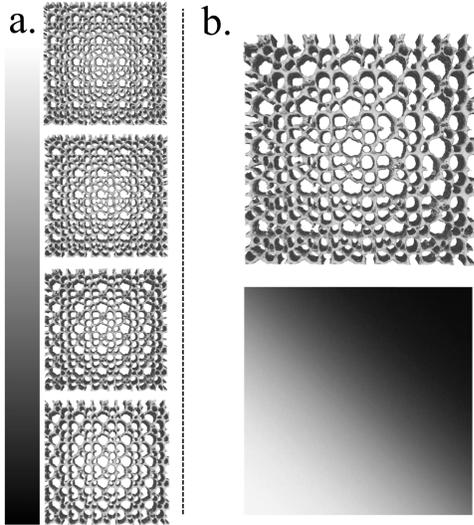


Figure 7. Synthesized foam structures with controllable density. **a.** The grey scale bar controls the synthesized structure from the highest density (white) to the lowest (black). **b.** Smooth interpolation of the guidance factor allows us to synthesize a foam structure with smoothly changing densities.

#### 4.4. Application: Seamless 3D Texturing

Due to the periodic nature of the synthesized patterns, noise manipulation allows IPFN to create textures that are mirror-symmetric. This property provides an immediate application to seamless 3D texture mapping: in Figure 6, the original 9-channel texture, composed of color, normal, and bump maps, is tiled and directly mapped to a planar surface. Due to discrepancies on the edges, as the textures are wrapped to create the tiled patterns, the mapped surfaces show visible seams. We recreate this texture through our network under a constant latent vector (Figure 6 B). When repeatedly mapped to the surface, the symmetric texture is seamless while faithfully reflecting the appearance and structure of the original texture.

#### 4.5. Application: 3D Foam with controllable density

The original foam shape used in our experiment contains holes of various sizes, which corresponds to the density of the foam structure. This geometric property  $g$  can be easily approximated with an unsigned distance field representation. For a patch  $X'$  of size  $H' \times W' \times D'$ , we estimate its density by:

$$g(X') = \frac{1}{m} \sum_{i=1}^{H'} \sum_{j=1}^{W'} \sum_{k=1}^{D'} |sdf(X_{ijk})| \quad (6)$$

, where  $m$  is a normalization factor. Figure 7 shows the synthesized foam structures by gradually increasing the density factor (Figure 7.a) and by a linearly interpolated density map ((Figure 7.b).

## 5. Limitations and Discussions



Figure 8. The examples demonstrating limitations of our network

The main limitation of our method is its emphasis on modeling stationary patterns. While this is based on our observation that a broad range of natural patterns is stationary or directional, our method does not provide a natural way to address a class of patterns that are radial, which are exemplified by web structures and spiral patterns (see the third column in Figure 8).

While our conditional formulation is in theory compatible with the synthesis of landscape images, experiments found that the quality of the synthesized landscapes are subpar - while the synthesized landscape appears globally similar to the exemplar, some local regions contain "fading-out" elements that are blended with the background (see the first two columns in Figure 8). We speculate that this phenomenon is due to an under-representation of these elements in the exemplar.

The above limitations have inspired us to consider many potential improvements of our methods in future works. A multi-scale synthesis approach marked in [31, 32] strikes a good balance between learning the distribution of global structure and local, high-frequency details of an image. Different geometric encoding schemes may also extend our framework to synthesize beyond stationary patterns. We believe there are still ample opportunities for the extension of our methods to a broader range of 3D applications.

## Acknowledgements

This research was sponsored by the Army Research Office and was accomplished under Cooperative Agreement Number W911NF-20-2-0053, and by the U.S. Army Research Laboratory (ARL) under contract number W911NF-14-D-0005. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Office or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation.

## References

- [1] Ronen Basri, Meirav Galun, Amnon Geifman, David Jacobs, Yoni Kasten, and Shira Kritchman. Frequency bias in neural networks for input of non-uniform density. In *International Conference on Machine Learning*, pages 685–694. PMLR, 2020. 4
- [2] Urs Bergmann, Nikolay Jetchev, and Roland Vollgraf. Learning texture manifolds with the periodic spatial gan. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 469–477, 2017. 2, 3, 5, 6, 7
- [3] Pravin Bhat, Stephen Ingram, and Greg Turk. Geometric texture synthesis by example. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pages 41–44, 2004. 3
- [4] Weikai Chen, Xiaolong Zhang, Shiqing Xin, Yang Xia, Sylvain Lefebvre, and Wenping Wang. Synthesis of filigrees for digital fabrication. *ACM Transactions on Graphics (TOG)*, 35(4):1–13, 2016. 3
- [5] Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed, and Andrea Vedaldi. Describing textures in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3606–3613, 2014. 5
- [6] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Image super-resolution using deep convolutional networks. *IEEE transactions on pattern analysis and machine intelligence*, 38(2):295–307, 2015. 2
- [7] Alexei A Efros and William T Freeman. Image quilting for texture synthesis and transfer. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 341–346, 2001. 2
- [8] Alexei A Efros and Thomas K Leung. Texture synthesis by non-parametric sampling. In *Proceedings of the seventh IEEE international conference on computer vision*, volume 2, pages 1033–1038. IEEE, 1999. 2
- [9] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014. 2, 3
- [10] Ishaan Gulrajani, Faruk Ahmed, Martín Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In *NIPS*, 2017. 3, 5
- [11] Lars Kai Hansen and Peter Salamon. Neural network ensembles. *IEEE transactions on pattern analysis and machine intelligence*, 12(10):993–1001, 1990. 3
- [12] Philipp Henzler, Niloy J Mitra, and Tobias Ritschel. Learning a neural 3d texture space from 2d exemplars. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8356–8364, 2020. 1, 2, 3, 5, 6, 7
- [13] Takashi Ijiri, Radomír Mech, Takeo Igarashi, and Gavin Miller. An example-based procedural system for element arrangement. In *Computer Graphics Forum*, volume 27, pages 429–436. Wiley Online Library, 2008. 3
- [14] Nikolay Jetchev, Urs Bergmann, and Roland Vollgraf. Texture synthesis with spatial generative adversarial networks. *arXiv preprint arXiv:1611.08207*, 2016. 2, 3
- [15] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 5
- [16] Johannes Kopf, Chi-Wing Fu, Daniel Cohen-Or, Oliver Deussen, Dani Lischinski, and Tien-Tsin Wong. Solid texture synthesis from 2d exemplars. pages 2–es, 2007. 2
- [17] Vivek Kwatra, Irfan Essa, Aaron Bobick, and Nipun Kwatra. Texture optimization for example-based synthesis. In *ACM SIGGRAPH 2005 Papers*, pages 795–802. 2005. 2
- [18] Vivek Kwatra, Arno Schödl, Irfan Essa, Greg Turk, and Aaron Bobick. Graphcut textures: Image and video synthesis using graph cuts. *Acm transactions on graphics (tog)*, 22(3):277–286, 2003. 2
- [19] Chuan Li and Michael Wand. Precomputed real-time texture synthesis with markovian generative adversarial networks. In *European conference on computer vision*, pages 702–716. Springer, 2016. 3
- [20] Shichen Liu, Shunsuke Saito, Weikai Chen, and Hao Li. Learning to infer implicit surfaces without 3d supervision. *Advances in Neural Information Processing Systems*, 32:8295–8306, 2019. 3
- [21] Paul Merrell. Example-based model synthesis. In *Proceedings of the 2007 symposium on Interactive 3D graphics and games*, pages 105–112, 2007. 3
- [22] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *European conference on computer vision*, pages 405–421. Springer, 2020. 1, 3, 4
- [23] Jonathan Palacios, Chongyang Ma, Weikai Chen, Li-Yi Wei, and Eugene Zhang. Tensor field design in volumes. In *SIGGRAPH ASIA 2016 Technical Briefs*, pages 1–4, 2016. 3
- [24] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. DeepSDF: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 165–174, 2019. 1, 3
- [25] Ken Perlin. An image synthesizer. *ACM Siggraph Computer Graphics*, 19(3):287–296, 1985. 2
- [26] Ken Perlin. Improving noise. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 681–682, 2002. 2
- [27] Tiziano Portenier, Siavash Arjomand Bigdeli, and Orcun Goksel. Gramgan: Deep 3d texture synthesis from 2d exemplars. *Advances in Neural Information Processing Systems*, 33, 2020. 2, 3
- [28] Javier Portilla and Eero P Simoncelli. A parametric texture model based on joint statistics of complex wavelet coefficients. *International journal of computer vision*, 40(1):49–70, 2000. 2
- [29] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015. 2, 5
- [30] Shunsuke Saito, Zeng Huang, Ryota Natsume, Shigeo Morishima, Angjoo Kanazawa, and Hao Li. Pifu: Pixel-aligned

- implicit function for high-resolution clothed human digitization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2304–2314, 2019. 3
- [31] Tamar Rott Shaham, Tali Dekel, and Tomer Michaeli. Singan: Learning a generative model from a single natural image. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4570–4580, 2019. 2, 3, 6, 8
- [32] Assaf Shocher, Shai Bagon, Phillip Isola, and Michal Irani. Ingan: Capturing and retargeting the” dna” of a natural image. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4492–4501, 2019. 2, 3, 8
- [33] Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. *Advances in Neural Information Processing Systems*, 33, 2020. 3, 4
- [34] Matthew Tancik, Pratul P Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *arXiv preprint arXiv:2006.10739*, 2020. 3, 4
- [35] Li-Yi Wei. Texture synthesis from multiple sources. In *ACM Siggraph 2003 Sketches & Applications*, pages 1–1. 2003. 2
- [36] Steven Worley. A cellular texture basis function. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 291–294, 1996. 2
- [37] Wenqi Xian, Patsorn Sangkloy, Varun Agrawal, Amit Raj, Jingwan Lu, Chen Fang, Fisher Yu, and James Hays. Texturegan: Controlling deep image synthesis with texture patches. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8456–8465, 2018. 2
- [38] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelnerf: Neural radiance fields from one or few images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4578–4587, 2021. 3
- [39] Matthew D Zeiler, Dilip Krishnan, Graham W Taylor, and Rob Fergus. Deconvolutional networks. In *2010 IEEE Computer Society Conference on computer vision and pattern recognition*, pages 2528–2535. IEEE, 2010. 2
- [40] Eugene Zhang, Konstantin Mischaikow, and Greg Turk. Vector field design on surfaces. *ACM Transactions on Graphics (TOG)*, 25(4):1294–1326, 2006. 3
- [41] Kun Zhou, Xin Huang, Xi Wang, Yiying Tong, Mathieu Desbrun, Baining Guo, and Heung-Yeung Shum. Mesh quilting for geometric texture synthesis. In *ACM SIGGRAPH 2006 Papers*, pages 690–697. 2006. 3
- [42] Yang Zhou, Zhen Zhu, Xiang Bai, Dani Lischinski, Daniel Cohen-Or, and Hui Huang. Non-stationary texture synthesis by adversarial expansion. *ACM Transactions on Graphics (TOG)*, 37(4):1–13, 2018. 2, 3, 5, 6, 7