Solid Texture Synthesis using Generative Adversarial Networks

Xin Zhao1 Jifeng Guo2 Lin Wang1 Fanqi Li1 Junteng Zheng1 Bo Yang1
1Shandong Provincial Key Laboratory of Network Based Intelligent Computing, University of Jinan, Jinan 250022, China.
2School of Computer Science and Engineering, South China University of Technology, Guangzhou 510641, China.

Abstract

Solid texture synthesis (STS), as an effective way to extend 2D exemplar to a 3D solid volume, exhibits advantages in numerous application domains. However, existing methods generally synthesize solid texture with specific features, which may result in the failure of capturing diversified textural information. In this paper, we propose a novel generative adversarial nets-based approach (STS-GAN) to hierarchically learn solid texture with a feature-free nature. Our multi-scale discriminators evaluate the similarity between patch from exemplar and slice from the generated volume, promoting the generator to synthesize realistic solid textures. Experimental results demonstrate that the proposed method can generate high-quality solid textures with similar visual characteristics to the exemplar.

Introduction

Texture synthesis, a technique for extending textural information to images, has been applied widely in computer graphics and vision (Chen, Pan, and Tian 2019; Hörmann et al. 2021). Many studies focused on generating textures for two-dimensional images or three-dimensional object surfaces. However, in some fields, solid textures are favored because they can convey textural information not only on the surfaces, but also throughout the entire volume, i.e., the texture extends from the surface to the inside.

In its generating process, solid texture synthesis attempts to learn to generate a 3D solid texture from a given 2D exemplar, and the synthesized solid texture shares similar textural properties with the exemplar (see Figure. 1). It has gained numerous success stories in real-world applications, such as aiding scientists with 3D modeling (Xiao et al. 2021) or providing people with an exceptional visual experience (Hädrich et al. 2021). Solid texture synthesis, in general, extends the visual characters of a 2D exemplar into an object whose voxels belong to a volumetric domain $D \subset \mathbb{R}^3$, sharing a similar internal appearance with the exemplar. During the synthesization process, the color of each voxel in the generated solids is gradually modified by matching features, describing specific appearance properties. Eventually, the overall appearance of the solid is similar to the fundamental textural feature of a given instance.

Motivation The traditional non-neural networks STS methods, like statistical feature matching methods (Heeger and Bergen 1995; Ghazanfarbour and Dischler 1995) or markov random field-based methods (Wei 2002; Kopf et al. 2007), have a number of marvelous stories to their credit (Mariethoz and Lefebvre 2014; Turner and Kalidindi 2016). But due to the intricacy of artificial or natural textures, it is difficult to accurately map the appearance of a 2D exemplar into a 3D solid item in practice. Henzler et al. provided a neural networks solution to synthesize 3D textures using a point operation technique, exhibiting the potential of neural networks in the STS field (Henzler, Mitra, and Ritschel 2020). Nevertheless, the widespread existence of multi-scale appearance information in textures sets up hurdles for point operation. Thus, Gutierrez et al. investigated a convolutional neural networks (CNN)-based model to synthesize solid textures, taking full advantage of its hierarchical expressive capability and extracting features using the VGG-19 feature maps. Their visual effects of synthesized volumes are at least comparable to the state-of-the-art methods (Gutierrez et al. 2020).

However, the variety of textures makes it difficult to select appropriate features, as the chosen feature may not fit a specific exemplar very well. It is almost impossible to capture an infinite number of textural appearances with a limited number of features.

The Generative Adversarial Nets (GANs) (Goodfellow et al. 2014) have been proven to be a powerful model to
adapt to arbitrary distribution hidden behind data and synthesize diversified images (Bergmann, Jetchev, and Vollgraf 2017; Shaham, Dekel, and Michaeli 2019). Following point operation strategy from Henzler et al., the GramGAN (Portenier, Arjomand Bigdeli, and Goksel 2020) learns textural appearance with generative adversarial nets without feature extraction. However, it still suffers from the inaccuracy when describing the multi-scale spatial correlation because of simply providing spatial information as additional network inputs, failing in generating structured textures or complicated stochastic structures (Portenier, Arjomand Bigdeli, and Goksel 2020).

Therefore, a question that arises here is: can we synthesize arbitrary solid texture by GANs to avoid feature extraction while capturing the multi-scale textural information hierarchically?

**Contribution** In this paper, we propose a generative adversarial nets-based solid texture synthesis framework, named STS-GAN. As a feature-free hierarchical framework, STS-GAN could capture multi-scale textural information in a 2D exemplar and extend it to 3D solids.

**Related Works**

Solid texture synthesis has attracted considerable research interest in the field of computer graphics and vision since it was proposed by Perlin and Peachey in 1985. Among them, procedural methods (Perlin 1985; Peachey 1985) were the earliest families with the advantage of low computational cost. They synthesized textures using a function of pixel coordinates and a set of manually tuning parameters. Perlin Noise, perhaps the most widely used one, is a smooth gradient noise function that is used to create pseudo-random patterns by perturbing mathematical equations.

Nevertheless, determining a suitable set of parameters for a given image necessitates tedious trial-and-error. Furthermore, the semantic gap inhibits individuals from linking notions such as marble or gravel with precise parameters. By contrast, exemplar-based STS methods can generate a new texture of arbitrary size from a given exemplar. These methods, in general, are automated and do not rely on the accurate texture description. The following is a brief review of various families of these methods.

**Statistical Feature-Matching Methods** These methods use a set of statistical features extracted from a given texture and apply it to solid textures. The pyramid histogram matching (Heeger and Bergen 1995) method pioneered the work on solid texture synthesis from 2D exemplars, using an image pyramid to capture the characteristics of textures at various resolutions. It is useful to create stochastic textures. Ghazanfarpour and Dischler (Ghazanfarpour and Dischler 1995) presented a solid texture generation method based on the spectral analysis of a 2D texture in various types. Jagnow et al. proposed a solid texture synthesis method using stereoscopic techniques (Jagnow, Dorsey, and Rushmeier 2004), which effectively preserve the structure of the texture.

Textures, in general, are diverse and complicated. Statistical feature-based methods tend to synthesize specific texture based on the certain image feature but fail to work on a broad set of textures.

**Markov Random Field-based Methods** These methods model texture as a Markov Random Field (MRF). That is, each pixel in a texture image is only dependent on the pixels around it, and all pixels follow this dependency. Based on non-parametric MRF model, Wei first applied a nearest neighborhood matching strategy coupled with an image pyramid technique to synthesize solid textures (Wei 2002).

Kopf et al. synthesized 3D solid textures by adopting MRF as a similarity metric (Kopf et al. 2007). In this method, the color histogram matching forces the global statistics of the synthesized solid to match those of exemplars. Chen and Wang integrated position and index histogram matching into the MRF optimization framework using the k-coherence search, which effectively improves the quality of synthetic solids (Chen and Wang 2010). In general, while these MRF-based techniques may capture hierarchical texture features and generate outstanding results, the conflict between texture diversity and the difficulty of learning a non-parametric MRF model precludes them from producing high-quality solid textures.

**Neural Nets-based Methods** Recently, neural networks have been used to synthesize solid textures because of their capability to approximate any nonlinear functions.

One branch of neural networks STS is based on the point operation, started from Henzler et al. (Henzler, Mitra, and Ritschel 2020), which is a generative model of natural textures by feeding multiple transformed random 2D or 3D fields into a multi-layer perceptron that can be sampled over infinite domains. Based on this work, GramGAN combined ideas from both style transfer and generative adversarial nets to generate 3D textures (Portenier, Arjomand Bigdeli, and Goksel 2020). Despite the fact that it avoids feature extraction, simply providing spatial information as additional network input makes it hard to capture hierarchical details at multi-scales. Consequently, the method fails to generate structured or complex stochastic textures.

To hierarchically synthesize volumetric textures, a convolutional neural networks-based method was introduced (Gutierrez et al. 2020), taking advantage of CNN’s powerful expressive capability for spatial autocorrelation data. It takes part of VGG-19 (Simonyan and Zisserman 2014) as an image descriptor to conceptualize features extracted from an exemplar (Gutierrez et al. 2020). The results proved that it can generate a solid texture of arbitrary size while reconstructing the conceptualized visual features of an exemplar along with some directions. But the diversity of textures makes it hard to use a limited number of VGG features to fit an infinite number of appearances. Thus, this method may not accurately capture and extend arbitrary exemplars’ texture properties.

**STS-GAN**

The proposed STS-GAN can extend a given 2D exemplar to volumetric texture by utilizing adversarial learning between the solid texture generator (STG) and a set of slice texture
discriminators (STD). The STG is in charge of generating "realistic" solid textures that have the similar textural appearance to a given exemplar at cross-sections. In contrast, slice texture discriminators, like STG's competitors, attempt to distinguish slices of created solid (the fake one) from the supplied instance (the real one) at various scales. During optimization, adversarial learning forces STG to gradually capture the global structure as well as fine texture details of the given exemplar.

Framework

A solid textural appearance can be described by a joint distribution in the volumetric domain $D$. We first assume the information of this joint distribution can be captured approximately by enough marginal distributions of different cross-sections along distinct directions.

Given that we are interested in the texture along a certain direction, the distribution of cross-sections belonging to this direction should be same as each other, describing the common textural property in them. As a result, if we can learn the distribution of texture in a given exemplar along this direction and fit those distributions to it, the textural appearance in the 2D exemplar can be extended into 3D domain $D$ along this direction. Particularly, for an isotropic solid texture, distributions of all cross-sections along all directions should be the same. In addition, the textural appearance usually exhibits different properties at different scales, implying the difference of distributions between scales.

Thus, as an universal distribution learner, the GANs are used in this work to design a framework for synthesizing arbitrary 3D textural appearances by learning texture distribution in a 2D exemplar.

The framework of STS-GAN is described in Figure 2, which is a hierarchical architecture containing $N$ discriminant scales, with the intention of learning texture information at various scales. At each level $n$, the solid texture generator $G$ reconstructs a solid texture $v_n$, by processing a group of input noises $z_n$:

$$v_n = G(z_n).$$

At each level of the framework, the STG’s objective is to synthesize a fake solid whose slice $u_n$ is similar with a real patch $x_n$, which is randomly cropped from the given exemplar. Meanwhile, as STG’s competitor, the STD consists of a collection of multi-scale discriminators $\{D_1, \ldots, D_N\}$. $D_n$ is responsible for learning to differentiate fake slice $u_n$ from real patch $x_n$, promoting the STG to synthesize more realistic solid textures to fool the STD. When the Nash equilibrium is achieved (Goodfellow 2016), the STG can synthesize such realistic solid textures that STD cannot distinguish it from the given exemplar.

Solid Texture Generator In order to improve the stability of the adversarial training, we need to provide adequate fake slices at multiple scales. Inspired by (Ulyanov et al. 2016), the STG employs the trick of multi-scale inputs, influencing on the textural characteristic of the generated solid at each scale to increase the diversity of texture.

The input noise $z_n$ for $G$ contains $K$ volumetric noises $\{z_{n,1}, \ldots, z_{n,K}\}$ with incremental size. During the generation, the multi-scale noises are preprocessed by convolution blocks, and are fed into generator at the corresponding intermediate layer. To merge the preprocessed noises in different scales, the small one are upsampled to match the size of the larger one for concatenation on channels. The concatenated tensors are finally mapped by another convolution block to three channels, each of which corresponds to a primary color. As the STG is a fully convolutional network, it is able to customize the size of generated solid texture easily.
Orthogonal Slicing 45-degree-added Slicing

Exemplar
Orthogonal Slicing
45-degree-added Slicing

Figure 3: Different slicing strategies have diverse impact on the model performance. The left part shows the given exemplar. The middle section presents the outcome of the orthogonal slicing strategy, and the results with the 45 degree-added slicing strategy are shown on the right. In particular, we offer the carved solid and the 2D slice at 45 degree selected randomly from the volume respectively.

Slice Texture Discriminators  Given a 2D exemplar and a specific direction, we could simply assume the exemplar comes from a “real” solid, and its cross-sections at this direction share same distribution with the exemplar. Thus, recognizing the appearance of 2D exemplar(s) accurately is quite important to guide STG.

However, considering the complexity of solid textures, it is hard to train a discriminator to assemble multi-scale knowledge into a single model for differentiating slices at different scales. In other words, a discriminator focusing on small-scale texture during learning may fail to guide the STG to model the global volumetric appearance and vice versa.

Therefore, to distill the information on the distribution of “real” solids from 2D exemplar as much as possible, we adopt a group of slice discriminators to differentiate textures hierarchically at different scales. The primary purpose is to assist the STG in learning cross-scale appearances to produce realistic solid textures. The STD consists of \( N \) discriminators sharing the same structure but operating at different image scales.

Training  The STG and STD were trained alternatively during adversarial learning. It is worth noting that all discriminators of different scales are trained once in a single iteration, whereas the generator is trained only once at a random chosen scale, ensuring that the training is balanced.

The STD takes the slices of generated solid as fake images. One would wonder how we slice a generated solid texture, especially along which directions. As the voxels in a solid texture should be spatially autocorrelated, non-orthogonal slicing may result in the redundancy of information between different slices for training. Thus, we orthogonally slice the generated solid from STG as the fake samples which will be further fed into the discriminators.

This framework adopts the Wasserstein GAN with gradient penalty (Gulrajani et al. 2017) loss during optimization to improve the stability of training. When the generator is optimized, the loss is minimized using equation:

\[
L_G = -\mathbb{E}[D(u_n)],
\]

\[L_{D_n} = \mathbb{E}[D(u_n)] - \mathbb{E}[D(x_n)] + \lambda \mathbb{E}[(\|\nabla r_n D(r_n)\|_2 - 1)^2],
\]

where \( u_n \) is a 2D slice taken at random from the 3D solid generated by STG. Meanwhile, each discriminator is optimized by minimizing its loss function, denoted by equation:

\[
\text{Figure 4: The model’s performance with different number of discriminant scales. The generated solids of each model are shown on top, with randomly picked slices from them on the bottom.}
\]

Experiment

In this section, we examine the efficacy of our hierarchical architecture’s multi-scale components through ablation and performance experiments. Furthermore, we compare our method to the state-of-the-art STS methods, particularly those based on neural networks.

Experimental Specifications

In the experiment, for the STD, we adopt five discriminant scales in STS-GAN, i.e., \( N = 5 \). The network of discriminators are fully convolutional, and there are two sizes of convolution kernels, \( 3 \times 3 \) and \( 1 \times 1 \). The STD takes an image as the input and gets a two-dimensional field, whose mean will be regarded as the discriminator score. In addition, the first three convolutional blocks of VGG-19 (Simonyan and Zisserman 2014) are used as pre-training module during the training process.

For the STG, noises with three different sizes are fed into the generator, i.e., \( K = 3 \). The noises follow the normal distribution. Each 3D convolution block consists of three 3D convolution layers, with kernel sizes of \( 3 \times 3 \times 3 \) for the first two layers and \( 1 \times 1 \times 1 \) for the last. The final convolution block is made up of two convolution layers, with kernel size of \( 3 \times 3 \times 3 \). In addition, we employ Batch Normalization (Ioffe and Szegedy 2015) and Leaky ReLU (Xu et al. 2015) to stabilize and accelerate the training process. To prevent the checkerboard pattern of artifacts, we use upsampling with nearest-neighbor interpolation followed by convolution rather than transposed convolution (Odena, Dumoulin, and Olah 2016).
Figure 5: The performance of STS-GAN on three isotropic exemplars, illustrated in seven different ways. (a) the given texture exemplars, (b) - (d) the slices of the generated solid across the three orthogonal directions, (e) the 45 degree slices, (f) the synthetic solid textures, (g) the eroded solid textures, (h) the texture mapping on 3D mesh models. The eroded visual effect is obtained by removing a range of colors and adding shadow. Source: The 3D mesh model (bunny) is from Stanford 3D Scanning Repository.

Figure 6: The continuous slices of solid textures in one direction. Left: the synthesized solid textures. Right: the continuous slices.

We adopt Chen et al.’s approach (Chen and Wang 2010), Gutierrez et al.’s approach (Gutierrez et al. 2020) and GramGAN (Portenier, Arjomand Bigdeli, and Goksel 2020) as competitors. In our comparison experiments, the experimental settings of the compared methods are the recommended ones in their paper.

This framework is implemented by PyTorch and run on GPU Nvidia GeForce TITAN RTX. The optimization is carried out using the Adam optimizer (Kingma and Ba 2014), with STG and STD learning rates of 0.0005 and 0.0003, respectively. The batch sizes for STG and STD are set to 1 and 72, respectively. All parameters are fine-tuned through trial and error.

Ablation Experiments
Slicing Direction To reduce computational overhead, we slice cross-sections in three orthogonal directions in STS-GAN. For comparison, we also slice additional 45-degree-angle cross-sections into the fake patch set to evaluate the sufficiency of orthogonal slicing strategy.

The solid textures obtained by the orthogonal slicing method and the 45 degree-added slicing strategy are shown in Figure 3. There is no significant difference between these two slicing strategies, demonstrating that slicing along orthogonal plane is sufficient to create high-quality solid textures.

Multi-scale Discriminant As previously stated, STS-GAN learns textural information on multiple scales. To demonstrate the efficacy of the multi-scale discriminative strategy, we investigate its effects by varying the number of discriminant scales (i.e., $N$). In this experiment, models with different discriminant scales are trained to learn the same texture exemplar.

Figure 4 shows that as the number of discriminant scales increases, the model provides clearer solids. Furthermore, the overall structure, as well as the local features, gradually mimic the texture of the given exemplar. This experiment proves that a model with a sufficient number of discriminant scales can learn hierarchical textural information and generate realistic solid textures with visual qualities similar to the
Dependency of Direction

Isotropic Exemplar  In this experiment, the STS-GAN is evaluated with various isotropic exemplars. Figure 5 depicts the generated solid textures from given textures as well as several visualizations of solids. It is apparent that the created 3D solids closely resemble the specified 2D exemplars. Furthermore, the textural features of the slices from the 3D solid at different angles are the same as the exemplar. Furthermore, the generated isotropic solid is used in surface mapping. Based on the spatial coordinate information, the synthesized solid distributes color to the surface pixels of the 3D mesh model. The outcome exhibits similar visual qualities to the exemplar, as shown in Figure 5(h). In addition, in Figure 6 we show continuous slices in the generated solid. The interior texture has been observed to have a high level of structure consistency.

These experiments show that our model is capable of extending a given 2D texture to 3D solids.

Anisotropic Exemplar  Since the STS-GAN learns the textural distribution from a 2D exemplar, it can also generate 3D solid textures with anisotropic properties when given textures in different orientations. In this experiment, the STS-GAN was trained using anisotropic exemplars in multiple orthogonal orientations based on different learning configurations.

As shown in Figure 7, the synthesized texture solids are consistent with anisotropic exemplars in different directions. Furthermore, the internal texture in these solids exhibits high coherence and consistency, presented in the last two columns. These findings suggest that STS-GAN is able to learn anisotropic texture and extend it to 3D solids.

Performance Comparison

The visual quality performance of the STS-GAN is compared against three state-of-the-art methods including a non-neural method (Chen and Wang 2010) and two neural networks-based methods (Gutierrez et al. 2020; Portenier, Arjomand Bigdeli, and Goksel 2020). Figure 8 compares our method with the method of Chen and Wang. Despite the fact that Chen et al.’s approach produces solid textures that are similar to exemplars, they still have failed attempts with significant variances between the generated solid and the exemplar (especially for the first exemplar). It can be observed that the solid texture produced by our method is substantially closer to the exemplar. Furthermore, the STS-GAN provides clearer borders and textures that are consistent in color, shape, and distribution with the exemplars. Our model can approximate arbitrary non-linear functions and has a more remarkable ability to learn texture properties compared to non-neural method. Thus, our method can generate more realistic solid textures.

Figure 9 exhibits the comparison between STS-GAN and two other neural networks-based methods. It can be observed that the solid textures and slices generated by ours are more visually similar to the exemplar. In most cases, the approach of Gutierrez et al. focuses solely on the major parts of colors and structures in the texture while ignoring a number of minor but crucial colors or structures. GramGAN’s solid textures are blurred. Besides, the color and shape of textures differ from the exemplar. Our method can adapt to arbitrary data distributions using GANs, and hierarchical modeling enables our method to learn hierarchical information about textures. Thus, our method produces realistic and
Figure 9: The comparison of STS-GAN’s outcomes with those of neural networks-based approaches. The generated solids are shown on the right. The solid slices are shown in the middle. In particular, the slices come from the diagonal direction of the solid.

| Method            | Rank From Study |
|-------------------|-----------------|
| Non-neural Networks |                 |
| Ours              | 1.14 ± 0.35     |
| Chen and Wang     | 1.86 ± 0.35     |
| Neural Networks   |                 |
| Ours              | 1.47 ± 0.74     |
| GramGAN           | 2.00 ± 0.68     |
| Gutierrez et al.  | 2.53 ± 0.66     |

Table 1: User ranking for the similarity of the textures generated by the different methods to the exemplar. We report $\mu \pm \sigma$ for user study ranking (lower is better).

high-quality solid textures compared to the other two methods.

We also conducted a single-blind formal user study to compare the visual effect between approaches. A group of 26 volunteers were given texture exemplars and their corresponding slices from synthesized solid texture by different competitors. They were asked to rank the slices based on their similarity to the corresponding reference exemplar. Apart from their corresponding exemplars, the questionnaire contains 20 groups of slices from non-neural networks synthesizers and 20 groups from neural networks ones. Table II exhibits the average ranking for each approach. It can be observed that the users are more accepting of our results.

1All volunteers have signed an informed consent form, guaranteeing to make independent and objective choices.

Conclusion

In this research, a novel approach to synthesize solid texture, STS-GAN, is proposed, which adversarially and hierarchically learn the appearance with a feature-free nature. It can successfully capture the multi-scale textural information hid in a 2D exemplar and further extend it into volumetric domain. In experiments, our method generates more realistic solid textures than the other state-of-the-art methods.

There are, however, still some limitations in this method. The high computational cost of STS-GAN learning process is a significant burden. In the future, the simplification method should be further studied for the acceleration of learning. Furthermore, the generating process must be hastened to meet the requirements of efficiency in real-world applications.

References

Bergmann, U.; Jetchev, N.; and Vollgraf, R. 2017. Learning Texture Manifolds with the Periodic Spatial GAN. In Precup, D.; and Teh, Y. W., eds., Proceedings of the 34th International Conference on Machine Learning, volume 70 of Proceedings of Machine Learning Research, 469–477. International Convention Centre, Sydney, Australia: PMLR.

Chen, G.; Pan, G.; and Tian, Y. 2019. Lung Surface Texture Synthesis Based on Convolutional Neural Network. *Journal of Jiamusi University (Natural Science Edition)*, 01.

Chen, J.; and Wang, B. 2010. High quality solid texture synthesis using position and index histogram matching. *The Visual Computer*, 26(4): 253–262.
Ghazanfarpour, D.; and Dischler, J. 1995. Spectral analysis for automatic 3-d texture generation. *Computers & Graphics*, 19(3): 413–422.

Goodfellow, I. 2016. Nips 2016 tutorial: Generative adversarial networks. *arXiv preprint arXiv:1701.00160*.

Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y. 2014. Generative Adversarial Nets. In Ghahramani, Z.; Welling, M.; Cortes, C.; Lawrence, N.; and Weinberger, K. Q., eds., *Advances in Neural Information Processing Systems*, volume 27, 2672–2680. Curran Associates, Inc.

Gulrajani, I.; Ahmed, F.; Arjovsky, M.; Dumoulin, V.; and Courville, A. 2017. Improved training of wasserstein gans. *arXiv preprint arXiv:1704.00028*.

Gutierrez, J.; Rabin, J.; Galerne, B.; and Hurtut, T. 2020. On Demand Solid Texture Synthesis Using Deep 3D Networks. In *Computer Graphics Forum*.

Hädrich, T.; Banuti, D. T.; Pałubicki, W.; Pirk, S.; and Michels, D. L. 2021. Fire in paradise: mesoscale simulation of wildfires. *ACM Transactions on Graphics (TOG)*, 40(4): 1–15.

Heeger, D. J.; and Bergen, J. R. 1995. Pyramid-based texture analysis/synthesis. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, 229–238.

Henzler, P.; Mitra, N. J.; and Ritschel, T. 2020. Learning a neural 3d texture space from 2d exemplars. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 8356–8364.

Hörmann, S.; Bhowmick, A.; Weiher, M.; Leiss, K.; and Rigoll, G. 2021. Face Texture Generation And Identity-Preserving Rectification. In *2021 IEEE International Conference on Image Processing (ICIP)*, 2448–2452. IEEE.

Ioffe, S.; and Szegedy, C. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, 448–456. PMLR.

Jagnow, R.; Dorsey, J.; and Rushmeier, H. 2004. Stereological techniques for solid textures. *ACM Transactions on Graphics (TOG)*, 23(3): 329–335.

Kingma, D. P.; and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kopf, J.; Fu, C.-W.; Cohen-Or, D.; Deussen, O.; Lischinski, D.; and Wong, T.-T. 2007. Solid texture synthesis from 2d exemplars. In *ACM SIGGRAPH 2007 papers*, 2-es.

Mariethoz, G.; and Lefebvre, S. 2014. Bridges between multiple-point geostatistics and texture synthesis: Review and guidelines for future research. *Computers & Geosciences*, 66: 66–80.

Odena, A.; Dumoulin, V.; and Olah, C. 2016. Deconvolution and checkerboard artifacts. *Distill*, 1(10): e3.

Peachey, D. R. 1985. Solid texturing of complex surfaces. In *Proceedings of the 12th annual conference on Computer graphics and interactive techniques*, 279–286.

Perlin, K. 1985. An image synthesizer. *ACM Siggraph Computer Graphics*, 19(3): 287–296.

Portenier, T.; Arjomand Bigdeli, S.; and Goksel, O. 2020. GramGAN: Deep 3D Texture Synthesis From 2D Exemplars. In Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M. F.; and Lin, H., eds., *Advances in Neural Information Processing Systems*, volume 33, 6994–7004. Curran Associates, Inc.

Shaham, T. R.; Dekel, T.; and Michaeli, T. 2019. SinGAN: Learning a Generative Model From a Single Natural Image. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*.

Simonyan, K.; and Zisserman, A. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Turner, D. M.; and Kalidindi, S. R. 2016. Statistical construction of 3-D microstructures from 2-D exemplars collected on oblique sections. *Acta Materialia*, 102: 136–148.

Ulyanov, D.; Lebedev, V.; Vedaldi, A.; and Lempitsky, V. 2016. Texture networks: Feed-forward synthesis of textures and stylized images. In *33rd International Conference on Machine Learning, ICML 2016*, 2027–2041.

Wei, L. 2002. *Texture Synthesis by Fixed Neighborhood Searching*. Ph.D. thesis, Stanford, CA, USA. AAI3038169.

Xia, H.; He, L.; Li, J.; Zou, C.; and Shao, C. 2021. Permeability prediction for porous sandstone using digital twin modeling technology and lattice Boltzmann method. *International Journal of Rock Mechanics and Mining Sciences*, 142: 104695.

Xu, B.; Wang, N.; Chen, T.; and Li, M. 2015. Empirical Evaluation of Rectified Activations in Convolutional Network. *arXiv e-prints*, arXiv–1505.