Deep Tiling: Texture Tile Synthesis Using a Constant Space Deep Learning Approach

Vasileios Toulatzis and Ioannis Fudos

October 6, 2021

Deep Tiling: Texture Tile Synthesis Using a Constant Space Deep Learning Approach*

Vasilis Toulatzis[1] and Ioannis Fudos[1]

University of Ioannina, Ioannina, Greece
{vtoulatz,fudos}@cse.uoi.gr

Abstract. Texturing is a fundamental process in computer graphics. Texture is leveraged to enhance the visualization outcome for a 3D scene. In many cases a texture image cannot cover a large 3D model surface because of its small resolution. Conventional techniques like repeating, mirroring or clamping to edge do not yield visually acceptable results. Deep learning based texture synthesis has proven to be very effective in such cases. All deep texture synthesis methods that attempt to create larger resolution textures are limited in terms of GPU memory resources. In this paper, we propose a novel approach to example-based texture synthesis by using a robust deep learning process for creating tiles of arbitrary resolutions that resemble the structural components of an input texture. In this manner, our method is firstly much less memory limited owing to the fact that a new texture tile of small size is synthesized and merged with the existing texture and secondly can easily produce missing parts of a large texture.

Keywords: Texture Synthesis · Deep Laarning.

1 Introduction

Texture synthesis aims at generating a new texture such that its resolution and structure are appropriate for using it on wrapping a 3D model. Texture expansion plays a cardinal role in many applications where a large texture is required. Games along side with Geographic Information System (GIS) apps are such cases in which large unbounded resolution textures are needed. In addition, the same applies not only for diffuse textures but also for specular, normal, bump and height maps.

Structure similarity with the original input texture is one of the most investigated topics on texturing. Many texture synthesis methods aim at expanding a texture and usually on doubling its width and height. However, they simultaneously introduce an increased consumption of memory resources which severely restricts its scalability. To this end, many such methods end up on running on

* This work has been co-financed by the European Union (European Regional Development Fund- ERDF) and Greek national funds through the Interreg Greece Albania 2014-2020 Program (project VirtuaLand).

CPU without leveraging the power and speed of GPU. Consequently, memory efficiency is a key factor for texture synthesis and expansion.

Example-based texture synthesis techniques employ deep learning based optimization processes that seek larger resolution textures that resemble an input texture. Such methods by targeting on producing synthesized images of larger resolution textures [17], [19] do not have the capacity to create smaller or arbitrary resolution textures. Tiling is the only alternative for building arbitrary texture images. Therefore, tiling texture synthesis [2], [6], [7], is not only capable of synthesizing larger textures but also a novel way of constructing step by step a brand-new texture or for completing missing parts of a larger texture.

In this work, we propose a new texture synthesis approach that follows the aforementioned procedure. Thus, our method is capable of generating new tiles that match structurally and have the same morphology with the original input texture. We utilize a space invariant deep neural network to produce a new tile that can be used to expand the original texture. Subsequently our system builds a new texture of arbitrary shape and size by artificially synthesizing tiles in any direction by using constant memory.

2 Related Work

2.1 Texture Synthesis

Texture synthesis is a field of research that has drawn the attention of researchers for many years. Starting from simple ideas of tiling patterns and stochastic models to state of the art techniques based on exemplars all of them aim to produce new synthetic visually acceptable textures.

The most effective approaches have proven to be example-based methods that employ deep learning approaches [8], optimization-based techniques [12], pixel-based [5], [18] and patch-based methods [13], [4].

Expanding texture synthesis is the most challenging among the texture synthesis goals. Therefore, several techniques that aim at expanding texture synthesis have been developed [10], [19]. The most recent ones rely on deep learning by producing remarkable results on expanding and even for super-resolution texture synthesis [14]. By using Convolutional Neural Networks (CNNs) of many layers [17] or Generative Adversarial Networks (GANs) [6], [19] these methods correlate image features to produce a new synthesized high resolution texture map and constitute the state-of-the-art methods on texture expansion. Nevertheless, except for their efficacy on visually acceptable results they do have some limitations like memory consumption and a very restricted either on the way of new texture shapes are generated [19] (fixed new texture dimensions) or on adding new patterns not included in the input image [6] (huge data-set of images that contain different patterns) which in some cases is not the intention on texture expansion.

Another work utilizing a Generative Adversarial Network (GAN) trained to expand a texture in a uniform manner is [9]. On the other hand, there are optimization techniques like Self Tuning [13,3], which is a method that extends texture optimization, that accomplish visually acceptable results.

2.2 Texture Tiling

The drawback of the aforementioned approaches is the large requirement of memory resources that makes them unsuitable for GPUs. To this end, tiling seems to be the only viable approach for synthesizing large textures.

One of the simplest texture design techniques is repeating tile patterns such that the produced texture does not include seams. Moreover, methods generating stochastic tiles have been developed [2], [7] for the same purpose. However, texture tiling is still an open field of research in terms of increasing texture resolution without considerable performance downgrade.

Thus, tiling forms a new challenge for texture synthesis and deep learning methods have already started being used to this end. Their main advantage is the capability of synthesizing new textures that are not repeated. Instead, they use one or more original input textures and produce random texture tiles in any direction matching the original structure. One recent work that focuses on creating high resolution texture tiles is [6]. This work introduces an approach to homogenizing texture tiles outputs of GANs trained on lower resolution textures to produce a high resolution texture with no seam artifacts by using Markov Random Fields (MRF). However, this method has very high GPU memory space requirements making it inappropriate for medium GPU configurations.

3 Deep Texture Tiling

The main issue of most of state-of-the-art methods is memory consumption that is a key factor both on computation but also hardware cost which is more obvious on using GPU for texture synthesis. To this end, we are focusing on creating smaller textures that are merged together to form a new synthesized texture of greater size but with respect on keeping the seamless manner of expanding in every direction with new tiles.

We propose a novel algorithm for synthesizing tiles by extending the fundamental work by [8] on neural texture synthesis. In general, by leveraging the power of a CNN of multiple layers we extract and correlate feature maps across layers of two instances of a VGG19 [15] network given two different resolution textures in each network. Our algorithm has the ability to synthesize texture tiles in a seamless manner by optimizing the distance of feature maps across the layers of our model by using as input textures the original and a new white noise tile merged with the original texture towards a specific orientation (up, down, right, left tiling).

Consequently, we embrace the main idea of deep texture synthesis but we abandon the specific size expansion and replace it with tiling. More specifically, we correlate feature spaces targeting to produce similar representations across network layers. On the first network we forward the original image while on the

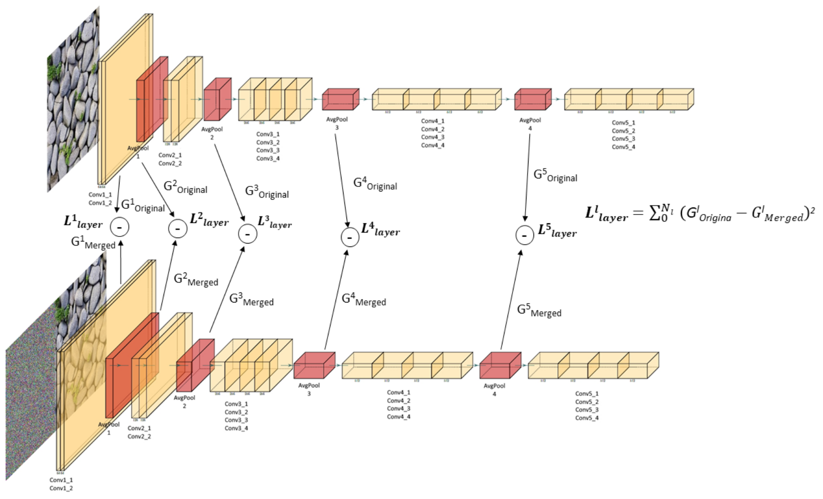


Fig. 1: Deep texture tiling: $G^l{}_{layer}$ is the Gram Matrice of feature maps in layer l which depends on the number of filters N_l. The network structure adopted is VGG19 [15] by changing $MaxPooling$ layers to $AvgPooling$ layers. This figure is generated by PlotNeuralNet (https://github.com/HarisIqbal88/PlotNeuralNet) and then modified.

second network we utilize two user input tiling factors for width and height for a new white noise tile creation that is forwarded along with the original as a merged input texture.

To capture correlations among network layers we extract their feature space representation F^l_{li} of a general feature map $F^l \in R^{n_f \times vs_f}$, where l is a layer having n_f filters of size vs_f reshaped into one dimensional vectors. This is achieved by the use of Gram Matrices:

$$G^l_{rc} = \sum_i F^l_{ki} F^l_{li} \tag{1}$$

The total layer loss is the sum of all layer losses that are computed as the mean squared displacement of the Gram Matrices of the two VGG19 instances. As a consequence, the total loss function is defined as follows:

$$L_{total}(I_{original}, I_{merged}) = \sum_{l=1}^{N^L} \frac{w^l}{4{n^l_f}^2 {vs^l_f}^2} \sum (G^l_{original} - G^l_{merged})^2 \tag{2}$$

where $I_{original}$ is the original texture and I_{merged} is a white noise texture merged with the original one having been forwarded to our system as described above and N^L is the number of contributory layers. The whole process and the layers contributing to the total loss function are shown in Figure 1. We correlate feature representations along $layer1$, $pool1$, $pool2$, $pool3$, $pool4$ and the corresponding weight setting for every layer contributory factor is $\frac{1}{5}$.

In some texture input cases the output of our method produces some noise in the boundaries of the original and deep generated tile. Therefore, we have developed an additional preprocessing phase we call Seam Removal in which we attempt to vanish the seam effect of deep texture tiling method. In the noise part of the second network instance ($Merged$ in Figure 1) instead of using a simple noise we utilize a mirrored version of the input original texture and then we apply noise that increases exponentially based on the distance of each column from the seam. Specifically, every pixel for the $Merged$ part of our model is computed as:

$$Noise(i,j) = w_1 Original(i, width - j - i) + w_2 RandomColor \tag{3}$$

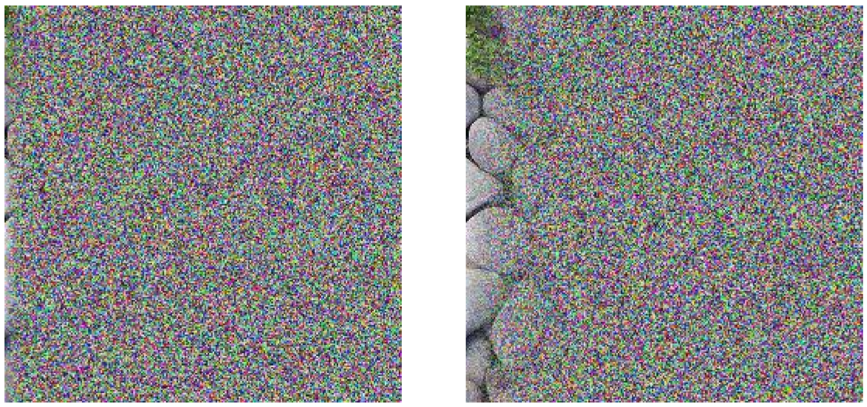

(a) $alpha = 0.25$ (b) $alpha = 0.05$

Fig. 2: Seam Removal: exponentially fading column mirroring

where $w_1 = e^{-\alpha j}$ with $\alpha \in (0,1)$, $w_2 = 1 - w_1$, i and j rows and columns accordingly.The produced noise outcome with $\alpha = 0.25$ and 0.05 is illustrated in Figures 2a and 2b respectively. An optimal α can been determined by

$$\alpha = -\frac{50\ ln(0.5)}{c}$$

where $c \times r$ is the resolution of the input texture. To obtain this we have determined experimentally running our whole methodology in the same input texture that the optimal visual result is derived by setting as target an attenuation of 50% (i.e. $w_1 = 0.5$) of the original mirrored image when we reach the 2% of the total number of columns (i.e. $j = c/50$). By doing so we achieve a seamless join of the two images without the mirroring effect being noticeable.

Texture expansion can be accomplished utilizing the aforementioned method in all directions by comparing input image with noise that is merged with a

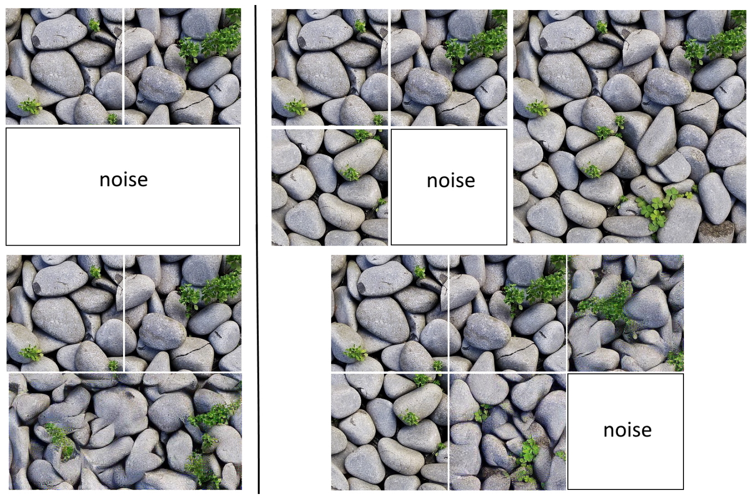


Fig. 3: The two Deep Texture Tiling Expansion methods. (left) Simple Right & then Down Tiling, (right) Create tiles of equal size - Right & Down Tiling are performed and then the Bottom Right Tile is produced by training our model lowering the mean square displacement of all 4 merged tiles to an original texture of double size. The second method is capable of keeping constant the amount of memory needed to expand a texture on any direction by following the exact same steps. Both methods are able to generate tiled textures of arbitrary size.

non noise part, using Gram matrices of arbitrary size. We have developed two different ways of creating large non-homogeneous textures.

The first one is a two step method. Firstly, a tile expansion to one of the four directions is performed with scaling factor that makes height or width of the starting image being doubled. For example, by doing a right tiling the original texture is doubled in width. Secondly, a tiling step is conducted, so that the other dimension is doubled. In the aforementioned case a down tiling follows to double the height, as well.

For limiting GPU memory requirements, we have developed an additional tiling method that it is slower but needs constant memory on the GPU. The method comprises three tiling steps. First we perform a side tiling, then a down tiling and eventually we end up merging the three tiles with a noise and our method goal is to converge on resembling to a fixed size original texture that we additionally provide as input. This approach is capable of serving as a missing tile filling method and it is presented along with the aforementioned Simple Tiling method in Figure 3. The drawback of this method that needs to be improved in the future is the forwarding of noise that is incrementally passed on to the new synthesized tiles.

4 Results & Evaluation

Our method has been developed on Python, using Tensorflow [1] and it has been tested on an NVIDIA GeForce RTX 2080 Ti with 11GB GDDR6 RAM and 1350MHz base clock. Input texture resolution was 256×256 and we used a tiling factor of 1 for both width and height of the generated input noise in both right and up tiling with which the original textures were merged (second VGG19 input). All outputs have been produced by 100000 *iterations* by utilizing the Adam optimizer [11] with $learning\ rate = 0.0005$ on our system learning process and the running average time was ≈ 2400 *secs*. However, using this learning rate to which we concluded experimentally we observe after 50000 epochs our method has already converged (Figure 4) in all input cases and it is just searching for a better minima and visual outcome with minor changes on loss valuese.

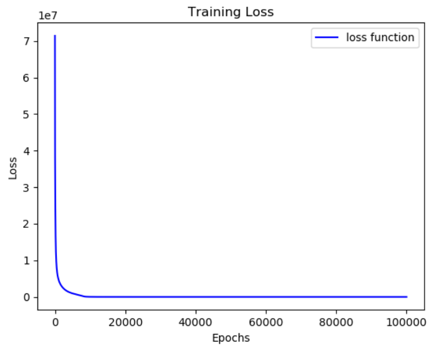


Fig. 4: Loss function per epoch during training for producing a new right tile with Adam optimizer [11] and $learning\ rate = 0.0005$ on a range of 100000 *iterations*.

In Figures 5 and 6 results of our algorithm are presented showing that synthesis of texture tiles which highly match an original texture is plausible by using our deep learning system with acceptable visual quality. In addition, Figure 7 depicts how effective is our method producing synthesized textures (consisted of new synthetic tiles) of larger size comparing to the original input tile. In this specific case, our method produces textures of double size (256×256 to 512×512) and in all cases Seam Removal with $a = 0.15$ is used. Our method is able to be used in left and down tiling and in cases in which a part of texture is missing, as well. In the latter case, the merged texture is the union of all other tiles surrounding the missing part (noise) which should then resemble to a fixed size image, as presented in Section 3.

Finally, we compare our method with state-of-the-art methods as shown in Figures 8a and 8b with [19] and [6] being the corresponding methods accordingly. Both methods are designed for different applications. However, this is a valid

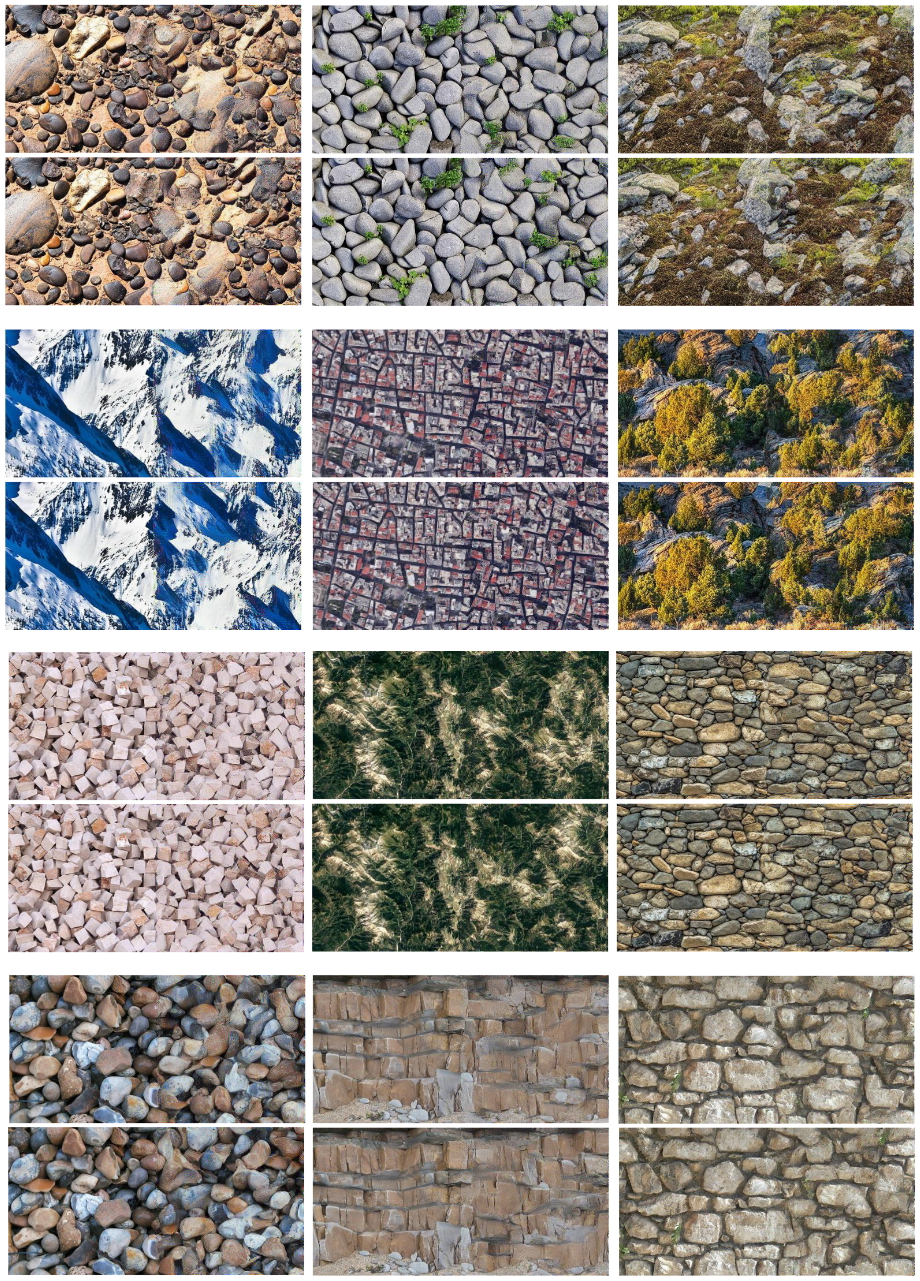


Fig. 5: Results of deep right texture tiling (half texture tile on the right is synthesized): Odd rows have been generated by deep texture tiling for right tile construction without seam removal applied, even rows illustrate the same but with seam removal by exponential column mirroring with $\alpha = 0.15$.

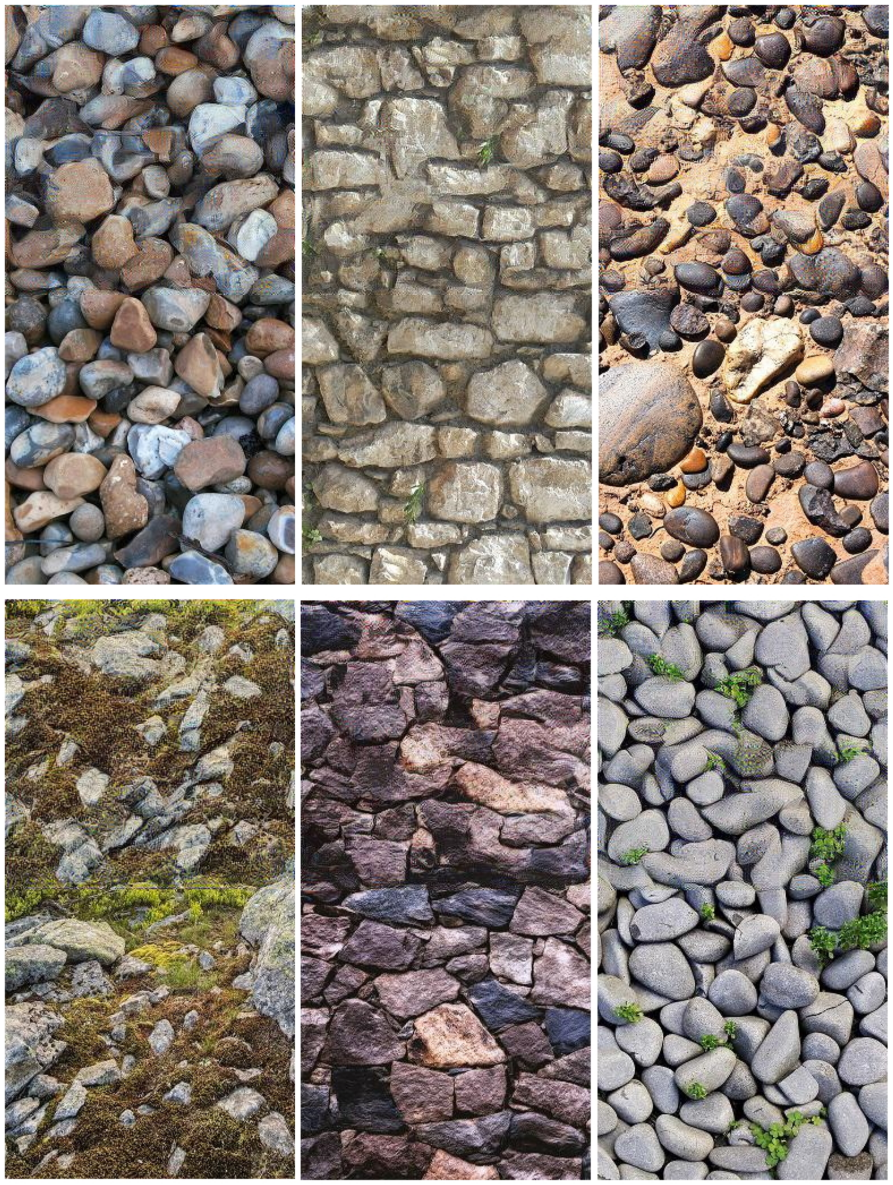


Fig. 6: Results of deep up texture tiling (upper half texture tile is synthesized): tiles synthesized on up direction with no seam removal.

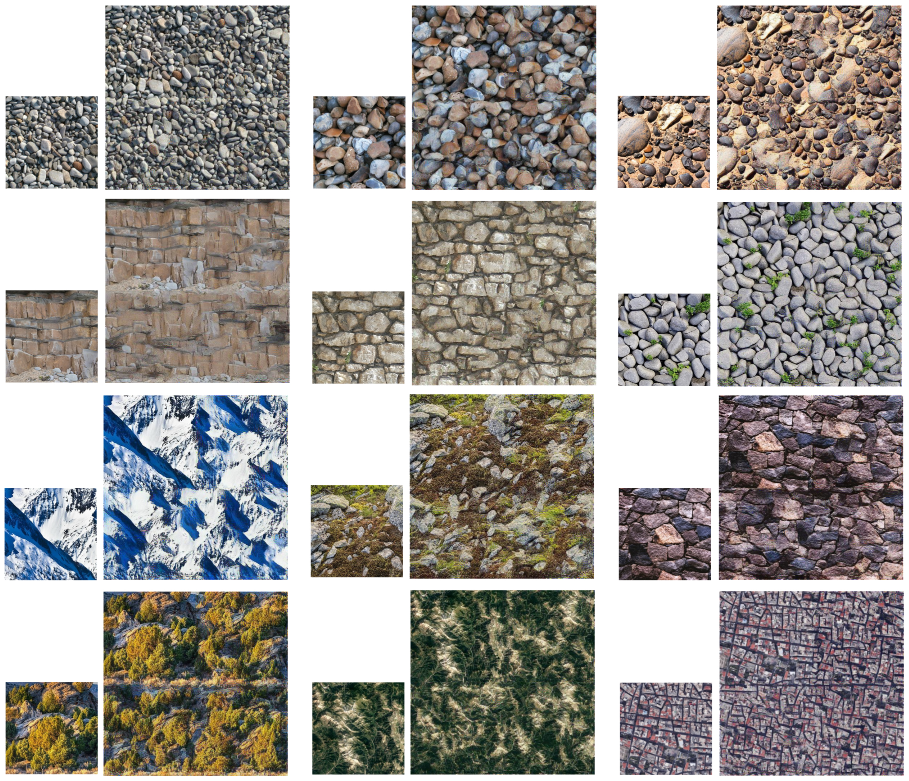
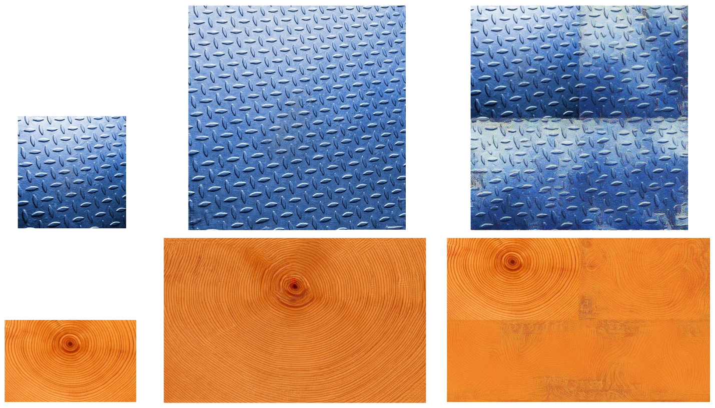
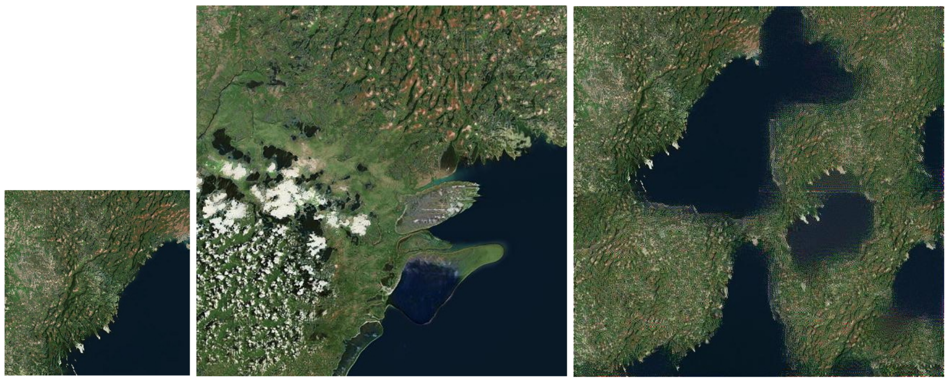


Fig. 7: Results for doubling the texture size with Deep Tiling: First, a right tiling is performed to produce a 512×256 tile and then a down tiling to produce the final 512×512 outcome. Seam removal by exponential column mirroring with $\alpha = 0.15$ is used. The last two input images are part of Pindos mountains and Athens city respectively and they have been extracted from Google Maps.

informal comparison of our method with two of the most competent methods in texture synthesis.



(a) Visual comparison with [19]. Generating from 256×256 texture a synthesized of 512×512 size.



(b) Visual comparison with TileGAN [6]. Guidance images resolutions for TileGAN used on this experiment is 32×32 so that the output of this method being a 512×512 image. In this manner, we are able to make our method comparable to the aforementioned work.

Fig. 8: Non-stationary expansion sub-figure 8a and TileGAN sub-figure 8b outputs are presented in second column and ours in last column.

Our method based on the the outcome of running it in non-stationary textures is not capable of synthesizing new seamless tiles. Thus, this constitutes one of the limitations of our method. This is mainly because our method is not designed to capture such patterns as [19] does and due to the tiling nature. Therefore, we see our method failing and not capturing well such type of textures on a seamless manner. However, [19]'s goal is generating such textures mainly.

As a consequence, it makes it not compatible for comparing our method with, in a formal way.

On the other hand, although TileGAN [6] is closer to our rational and way of producing new textures, it is not targeting to expand a texture towards a direction as we do, but their outcome is a synthesized texture of greater size trying to keep the structure as it is shown in sub-figure 8b. To this end, this method cannot be formally compared with ours owing to the fact that the two approaches are targeting into tackle different problems. Nevertheless, both technique's results are visually and perceptually acceptable as synthesized textures.

5 Conclusions & Future Work

We presented an innovative tiling synthesis method that is capable of producing new texture tiles in any direction. Moreover, we have introduced a new method for reducing the seam effect in texture synthesis. Based on the results, our method has proven to be very effective on tile texture synthesis bearing an essential advantage due to the fact that tiles can be generated in small resolutions step by step, making even low memory GPUs capable of synthesizing high resolution textures. A limitation of our approach is that noise is passed on from one tile to another and this is an issue that could be reduce using Gaussian pyramid as in [16]. As future work, we are also targeting at expanding our method with style transfer by creating new tiles of mixed styles. This is required for generating high resolutions textures or terrains that differ from area to area.

References

1. Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Kudlur, M., Levenberg, J., Monga, R., Moore, S., Murray, D.G., Steiner, B., Tucker, P., Vasudevan, V., Warden, P., Wicke, M., Yu, Y., Zheng, X.: Tensorflow: A system for large-scale machine learning. In: Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation. p. 265–283. OSDI'16, USENIX Association, USA (2016)
2. Cohen, M., Shade, J., Hiller, S., Deussen, O.: Wang tiles for image and texture generation. ACM Transactions on Graphics **22** (05 2003). https://doi.org/10.1145/1201775.882265
3. Darabi, S., Shechtman, E., Barnes, C., Goldman, D.B., Sen, P.: Image melding: Combining inconsistent images using patch-based synthesis. ACM Trans. Graph. **31**(4), 82:1–82:10 (Jul 2012). https://doi.org/10.1145/2185520.2185578, `http://doi.acm.org/10.1145/2185520.2185578`
4. 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. pp. 341–346. SIGGRAPH '01, ACM, New York, NY, USA (2001). https://doi.org/10.1145/383259.383296, `http://doi.acm.org/10.1145/383259.383296`
5. Efros, A.A., Leung, T.K.: Texture synthesis by non-parametric sampling. In: Proceedings of the International Conference on Computer Vision-Volume 2 - Volume

2. pp. 1033–. ICCV '99, IEEE Computer Society, Washington, DC, USA (1999), http://dl.acm.org/citation.cfm?id=850924.851569

6. Frühstück, A., Alhashim, I., Wonka, P.: Tilegan: synthesis of large-scale non-homogeneous textures. ACM Transactions on Graphics **38**(4), 1–11 (Jul 2019). https://doi.org/10.1145/3306346.3322993, http://dx.doi.org/10.1145/3306346.3322993
7. Fu, C.W., Leung, M.K.: Texture tiling on arbitrary topological surfaces using wang tiles. pp. 99–104 (01 2005). https://doi.org/10.2312/EGWR/EGSR05/099-104
8. 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 - Volume 1. pp. 262–270. NIPS'15, MIT Press, Cambridge, MA, USA (2015), http://dl.acm.org/citation.cfm?id=2969239.2969269
9. Goodfellow, I.J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2. pp. 2672–2680. NIPS'14, MIT Press, Cambridge, MA, USA (2014), http://dl.acm.org/citation.cfm?id=2969033.2969125
10. Kaspar, A., Neubert, B., Lischinski, D., Pauly, M., Kopf, J.: Self tuning texture optimization. Comput. Graph. Forum **34**(2), 349–359 (May 2015). https://doi.org/10.1111/cgf.12565, http://dx.doi.org/10.1111/cgf.12565
11. Kingma, D., Ba, J.: Adam: A method for stochastic optimization. International Conference on Learning Representations (12 2014)
12. Kwatra, V., Essa, I., Bobick, A., Kwatra, N.: Texture optimization for example-based synthesis. ACM Trans. Graph. **24**(3), 795–802 (Jul 2005). https://doi.org/10.1145/1073204.1073263, http://doi.acm.org/10.1145/1073204.1073263
13. Kwatra, V., Schödl, A., Essa, I., Turk, G., Bobick, A.: Graphcut textures: Image and video synthesis using graph cuts. In: ACM SIGGRAPH 2003 Papers. pp. 277–286. SIGGRAPH '03, ACM, New York, NY, USA (2003). https://doi.org/10.1145/1201775.882264, http://doi.acm.org/10.1145/1201775.882264
14. Sajjadi, M.S.M., Schölkopf, B., Hirsch, M.: Enhancenet: Single image super-resolution through automated texture synthesis (2017)
15. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv 1409.1556 (09 2014)
16. Snelgrove, X.: High-resolution multi-scale neural texture synthesis. pp. 1–4 (11 2017). https://doi.org/10.1145/3145749.3149449
17. Toulatzis, V., Fudos, I.: Deep Terrain Expansion: Terrain Texture Synthesis with Deep Learning. In: Vidal, F.P., Tam, G.K.L., Roberts, J.C. (eds.) Computer Graphics and Visual Computing (CGVC). The Eurographics Association (2019). https://doi.org/10.2312/cgvc.20191262
18. Wei, L.Y., Levoy, M.: Fast texture synthesis using tree-structured vector quantization. In: Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques. pp. 479–488. SIGGRAPH '00, ACM Press/Addison-Wesley Publishing Co., New York, NY, USA (2000). https://doi.org/10.1145/344779.345009, http://dx.doi.org/10.1145/344779.345009
19. Zhou, Y., Zhu, Z., Bai, X., Lischinski, D., Cohen-Or, D., Huang, H.: Non-stationary texture synthesis by adversarial expansion. ACM Trans. Graph. **37**(4), 49:1–49:13 (Jul 2018). https://doi.org/10.1145/3197517.3201285, http://doi.acm.org/10.1145/3197517.3201285