Multi-Scale Terrain Texturing using Generative Adversarial Networks

Jonathan Klein
French-German Research Institute Saint-Louis / University of Bonn

Stefan Hartmann
University of Bonn

Michael Weinmann
University of Bonn

Dominik L. Michels
KAUST

***Abstract*—We propose a novel, automatic generation process for detail maps that allows the reduction of tiling artifacts in real-time terrain rendering. This is achieved by training a generative adversarial network (GAN) with a single input texture and subsequently using it to synthesize a huge texture spanning the whole terrain. The low-frequency components of the GAN output are extracted, down-scaled and combined with the high-frequency components of the input texture during rendering. This results in a terrain texture that is both highly detailed and non-repetitive, which eliminates the tiling artifacts without decreasing overall image quality. The rendering is efficient regarding both memory consumption and computational costs. Furthermore, it is orthogonal to other techniques for terrain texture improvements such as texture splatting and can directly be combined with them.**

I. Introduction

Terrain rendering is an important aspect of many real-time applications such as simulators or video games. Among the different terrain representations that have been proposed, combining a triangle mesh representing the coarse-scale geometry with texture maps representing the finer details is the most popular choice. While the creation of convincing textures is mainly performed by artists or designers in a time-consuming and costly process, automatic texture generation allows the reduction of the development and production costs significantly.

Texturing huge terrains imposes the challenge of covering a large area while still preserving a high detail density which normally demands a lot of texture memory. A common approach is therefore to repeat a single texture with high resolution but small spatial extend over the whole terrain. However, albeit the efficiency regarding memory consumption and rendering time, this strategy results in undesirable visible tiling artifacts such as seams or repetitions. As the human perception is well-trained to notice patterns, it is desirable to take measures that reduce those tiling artifacts.

In detail mapping, two textures are used: The first one spans the whole terrain but contains only low-frequent details, which allows high compression rates. The second one is much smaller but more detailed and contains only high frequencies. It is tiled over the terrain and added onto the first one.

The generation of those detail maps is however not straightforward and either requires manual work of an artist or the combination of available maps that might not fit well together. While there are visually superior techniques for terrain mapping, detail mapping is still popular, due to its simplicity and efficiency in implementation, especially in areas that are constrained by production budget or limitations of target hardware such as the mobile sector.

In this paper, we propose to overcome the above mentioned issues with a novel automatic detail mapping approach that is efficient and significantly reduces unpleasant tiling artifacts. From a single terrain texture, we train a generative adversarial network (GAN) [1], [2], [3] to capture its structure and color. Then the GAN is used to generate a huge texture map that covers the whole terrain and provides high details at every location. In order to overcome artifacts such as blurriness or color shifts, we only extract the low-frequency component of the generated huge texture map. High-frequency features from the exemplar are recombined with the low-frequency component. The low-frequency map can be substantially compressed (e.g. by a factor of 100) without visible loss, while the high-frequency map contains finer details than the one fabricated by the GAN. Thus, our approach strongly improves memory efficiency and is visually superior when compared to the synthesized texture that covers that whole terrain.

After the initial pre-processing, detail maps can be generated for arbitrary terrain sizes in a matter of seconds to minutes. The adapted rendering process is highly efficient, as it requires only a texture mixing step within the fragment shader. This allows easy integration into any other terrain texturing technique such as texture splatting, a well-known technique used to cover a terrain with multiple materials.

II. Related Work

Related work comprises the developments in terrain texturing as well as in image and texture synthesis.

Terrain texturing: Especially since the introduction of programmable shaders, there exists a wide variety of texturing approaches, many of which are tailored to specific projects and never have been documented publicly. However, there are some basic techniques which have been proven so useful, that they are applied in many projects. *Texture splatting* [4] uses a fixed set of materials, where each is represented by one or multiple textures, and assigns them in varying strengths to different parts of the terrain, using either vertex attributes or assignment maps. *Mega textures* use a high-resolution texture map that is specifically designed for a terrain representation and rely on advanced texture streaming and LOD algorithms to manage the enormous amount of data [5]. They also rely on manual artistic work for the creation and have high hardware demands, but offer a great level of control and extremely high visual quality.

As texture tiling often results in visible artifacts, several approaches have been proposed to reduce these artifacts. Perturbing the UV coordinates offsets texture features and greatly reduces repetitions [6]. *Wang tiles* [7] use an arbitrary number of different texture patches or tiles. These can be combined with a subset of other tiles sharing the same tile edge colors. This technique allows the synthesis of complex textures with strongly reduced visibility of repetitions. However, the creation of these tiles is a hard problem and it is more difficult to implement than other techniques described inhere. In order to avoid the complex tiling problems, procedural techniques might be used to generate textures of arbitrary sizes. Such textures might be used to automatically compute mega textures and thus share their properties. However, they require specialized tools for their initial creation, which might be less intuitive than traditional texture painting or the processing of photographed textures [8]. In addition, the reproduction of photo-realistic textures is difficult.

Image and texture synthesis: While early work on texture synthesis (e.g. [9]) already indicated the potential towards automatically synthesizing textures with characteristics similar to the reference image, recent neural network based approaches based on generative models for image [10] and texture synthesis [11], [12] have lead to significant improvements.

Especially GANs have shown to produce a near infinite variety of realistic imagery from a low-dimensional seed sampled from a simple distribution. Apart from images and textures, this technique has been investigated for authoring the terrain of virtual worlds. Beckham and Pal. [13] synthesize a heightmap and a corresponding texture using unconditional [10] and conditional [14] GANs. The heightmap is generated from a low-dimensional seed producing a gray-scale image. This serves as input to an encoder-decoder network in order to produce a colored terrain texture conditioned on the relief. Guerin [15] propose an interactive terrain authoring system where GANs are trained dedicated to different types of terrain features that are easy to sketch.

A different line of research focuses on high-quality image generation as the output of GANs still contains unpleasant artifacts such a blurriness, color shifts, etc.. One way to overcome such artifacts are a combination of GANs with classic gradient-based algorithms. Wu et al. [16] propose Gaussian-Poisson GANs and formulate image compositing as a joint optimization problem that is constrained by both color generated by a GAN and gradient information from high-resolution images. Apart from that, image quality can also be increased by incorporating techniques from image quality analysis (IQA) as done by Vertolli et al. [17]. They show that adapted model loss functions using and combining recent distance measures from (IQA) improve the overall image quality when used with encoder-decoder networks.

Our work is built on top of the approaches based on GANs. However, in contrast to using the unchanged output fabricated by the GAN, we only use the low-frequency characteristics of the generated huge texture map. This map is then enhanced by a subsequent transfer of the high-frequency characteristics from the exemplar image. As a consequence, our approach avoids blurriness, wash-out colors and checkerboard artifacts that are strongly visible in previous neural network based approaches.

III. Methodology

The essence of detail mapping is the splitting of texture details into low- and high-frequency maps. The low-frequency components are used to uniquely cover every part of the terrain, which is important as several of these parts become visible when viewing the terrain from far away. Then, the high spatial resolution of the texture cannot be perceived due to the limited resolution of the viewer and becomes irrelevant. In contrast, a high spatial resolution is mandatory for the parts of the terrain that are viewed from a short distance. In this case tiling artifacts are not noticeable because only a small part of the terrain is visible to the observer. Thus, detail mapping produces visually pleasant results from either perspective.

Overview: As depicted in Figure 1, the first step of our approach consists of training a GAN that later allows the synthesis of a huge texture that spans the whole terrain while preserving the main characteristics of the input texture. To support arbitrary large output sizes, we split the generation process into multiple unique tiles that later can be assembled seamlessly. As the synthesized textures tend to have a shifted color tone, an additional post-processing can be performed to closely match the color distribution of the input texture.

While the result preserves the coarse characteristics of the input texture adequately, it is lacking in visual quality for fine details due to blurriness, color shifts and other artifacts produced by the network. For this reason, a subsequent optional color correction step is applied and the texture is split into its low-frequency and high-frequency components. The low-frequency image can be down-sampled without a significant loss of information allowing a memory reduction by a factor of 100 with usual parameters. The same splitting strategy is applied to the input texture, while only the high frequency part is maintained. During the rendering, the high-frequency component of the reference image and the low-frequency part of the synthesized texture can be efficiently combined within the fragment shader. This yields a high visual quality at a low memory consumption.

Generative Adversarial Networks: Given an input texture, the first step of our pipeline is given by the generation of a larger texture that preserves the characteristics of the original image as much as possible. This is achieved by exploiting Generative Adversarial Networks (GANs) [1], [2], [3], a recently introduced deep-learning technique.

Inspired by game theory, GANs are based on the concept of a competition between two players, the generator G and the discriminator D. The generator G takes an input sample $z \in \mathbb{R}^d$ drawn from a d-dimensional prior distribution $p_z(z)$

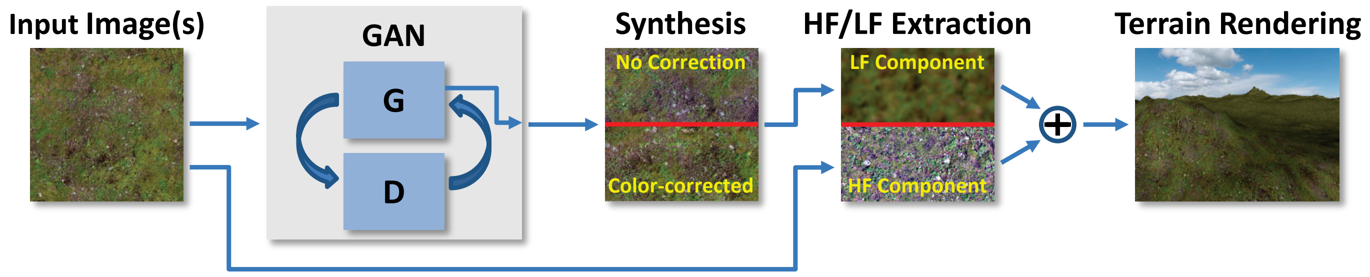


Fig. 1. Our pipeline for automatic detail maps generation: In a first step, a GAN is trained that allows the creation of a huge texture preserving the main characteristics of the exemplar texture. As the visual quality of this synthesized texture is not sufficient for its direct usage, we split the texture into its low-frequency and high-frequency components. Finally, we add the high-frequency components of the original texture onto the low-frequency texture obtained from the GAN to generate a plausible texture.

and, based on the parameters $\boldsymbol{\theta}^{(G)}$, tries to transform it into a plausible image x'. The discriminator D takes an image or an image patch x' and, based on the parameters $\boldsymbol{\theta}^{(D)}$, computes a scalar that judges whether x' has been sampled from the training data or whether it has been fabricated by the generator G. During the training procedure the discriminator tries to optimize its ability to distinguish synthetic images fabricated by the generator G from real images x that have been sampled from the training set. In contrast, the generator G tries to optimize its ability to fabricate images that cannot be distinguished from real ones by the discriminator D. In other words, the goal of the generator is the maximization of the misclassification rate in the decisions by the discriminator D, which can be stated as a cost function according to

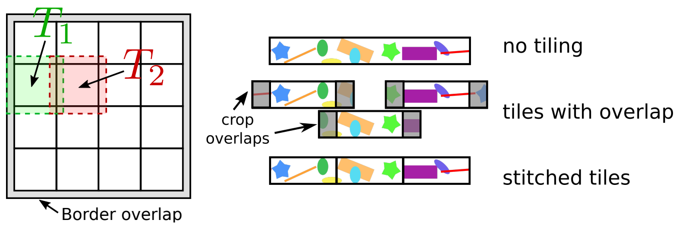
$$J^{(G)}(\boldsymbol{\theta}^{(D)}, \boldsymbol{\theta}^{(G)}) = -J^{(D)}(\boldsymbol{\theta}^{(D)}, \boldsymbol{\theta}^{(G)}). \tag{1}$$

A solution to these contradicting objectives can be obtained in terms of the minimax game with the objective

$$\arg \min_{\boldsymbol{\theta}^{(G)}} \max_{\boldsymbol{\theta}^{(D)}} -J^{(D)}(\boldsymbol{\theta}^{(D)}, \boldsymbol{\theta}^{(G)}) \tag{2}$$

with the discriminator's costs $-J^{(D)}(\boldsymbol{\theta}^{(D)}, \boldsymbol{\theta}^{(G)})$.

GAN training: In practice, neural networks are used to represent the generator G and the discriminator D. The training of G and D mathematically corresponds to finding the parameters $(\boldsymbol{\theta}^{(D)}, \boldsymbol{\theta}^{(G)})$ that represent a local optimum of both objectives $J^{(D)}$ and $J^{(G)}$. To improve the capability of the network to learn the local ingredients of the reference image, we randomly extract patches with a fixed size of $n \times n$ pixels from the reference image, which are then fed into the iterative GAN training procedure. During this procedure, G and D are trained in an alternating fashion (see [3], [18] for additional details). The training of the generator G relies on the input set $Z = \{z_0, \ldots, z_k\}$ of k samples, where at each spatial position of the tensor $z_i \in \mathbb{R}^{m \times n \times d}$ a d-dimensional vector drawn from the distribution p_z is used. The generator then transforms the z_i into images with $x'_i = G(z_i)$, where $x'_i \in \mathbb{R}^{n \times n \times 3}$. Furthermore, the k patches $X = \{x_0, \ldots, x_k\}$ with $x_i \in \mathbb{R}^{n \times n \times 3}$ from the training image and k fabricated patches $X' = \{x'_0, \ldots, x'_k\}$ serve as input for the training of the discriminator D. Here, the samples $x_i \in X$ are



(a) (b)

Fig. 2. Generation of huge textures by stitching individual tiles. Tiles are generated from overlapping parts of the input noise (a). Thus, generated tiles must be cropped before they can be stitched together (b) (shown as 1D case).

randomly drawn patches in the reference image. While we only used a single reference image in the scope of this work to obtain patches for the GAN training, a database containing multiple images could be used instead. For further details, we refer to the work by Jetchev et al. [18]. Furthermore, additional information such as normals, height maps, etc. can be incorporated by adding them as separate channels to the GAN input. After the training, we only need the generator network for the subsequent steps to generate textures and the discriminator can be discarded.

Synthesis of large output textures: After training the GAN as described in the previous section, we use the generator G to create new, large terrains. In principle, the generator is able to synthesize output textures of any size. However, bigger textures also exhibit an increased memory consumption which might limit the system in practical situations. We therefore exploit the local influence of the input noise to the output to generate an arbitrary large output in subsequent steps. A completely parallel generation process is also possible but unnecessary due to the high generation speed (e.g. well under a minute for typical terrain size).

In order to synthesize huge texture maps, the z_i that are passed to the generator are in $\mathbb{R}^{m \times l \times d}$, where $m \times l$ denotes the spatial resolution (e.g. 384×384) and d its depths. Each d-dimensional column of the z_i is sampled from a uniform distribution with a value range of $[0, 1]$. When m and l are large (e.g. > 64 on current hardware) the memory requirements for a single forward pass through the generator network are not tractable anymore. Therefore, we split the generation

Fig. 3. Splitting a texture into low-frequency and high-frequency components.

process into several forward passes using overlapping subsets extracted from z_i. An overlap of 2 in each direction is a good compromise between quality and efficiency. Furthermore, the whole input is also padded by a border used as overlap from the border tiles, which allows a homogeneous handling of all the tiles (see Figure 2 (a)). Each subset is passed through G leading to an individual image. These images might not be stitched together yet, because they contain information resulting from the overlap of the subsets. Instead, they need to be cropped around the border, which removes repetitions of features and results in stitchable edges (Figure 2 (b)).

While the seams are in fact not perfect as the overlap is usually below the total influence range of each input value, these imperfections are hardly visible and vanish completely in the later processing. For later use, the tiling factor (the ratio between the output image and the input image) should also be stored.

Splitting into detail maps: Our algorithm relies on splitting both the GAN output texture and the reference image into their high-frequency and low-frequency parts so that the low-frequency component of the GAN output and the high-frequency component of the reference image can be used for detail mapping.

To compute the low frequency map, the synthesized terrain texture is transformed into Fourier space where the low frequencies can be selected. For this purpose, choosing a Gaussian kernel is appropriate, which results in a soft border of the frequency selection and also allows for a straightforward implementation by convolving the texture in the primal domain with the same kernel. While the image size of the result remains unchanged, the image has a greatly reduced entropy which is why it can be down-sampled substantially without loosing information.

In contrast, the extraction of the high-frequency components of the reference image is performed by selecting the high frequencies in Fourier space with an kernel complementary to a Gaussian. This can be efficiently implemented in the primal domain by subtracting the image convolved with a Gaussian kernel from its unconvolved version, thanks to the additive properties of Fourier transformation. This also means that both textures can later efficiently be recombined in the shader by a simple addition.

Due to the subtraction, the dynamic range of the result can be too high to be stored in the same data type. To avoid data overflows, we compress the dynamic range again by dividing each color by 2 and adding neutral gray, which can efficiently be reversed in the shader and does not diminish the visual quality noticeably.

Color correction: Besides being slightly blurred and containing artifacts typically produced by neural network based approaches, the images generated by the GAN also deviate from the input image in color tone. If the characteristics of the result should closely match the ones of the input image, an additional color correction might be necessary.

In this regard, we found that a simple color shift yields good results in many cases. The mean color of both the input and the generated image are computed and each pixel color of the generated image is shifted by the difference of these means. Clipping to the dynamic range of the data type might be necessary here.

Rendering: Rendering with the detail maps is straight forward. Both textures are bound to the pixel shaders. Texture coordinates can either be computed from the vertex position (as the low-frequency map spans the whole terrain) or explicitly stored, which adds the advantage of reducing stretching artifacts at steep parts of the terrain. As the high-frequency map is tiled over the terrain, the texture coordinates can just be scaled by the tiling factor. Then, the dynamic range compression of the high-frequency map is reversed and both textures are added. Additional texturing techniques such as texture splatting can directly be applied and potentially available resources like normal or displacement maps can be reused without modification, as they usually only depend on the local features of the terrain (glossmaps are one example that could not be reused).

IV. RESULTS

We implemented the detail map generation in Python, based on the GAN implementation published by Jetchev et al. [18]. The training was performed with the Theano GPU interface on a nVidia GeForce GTX 1080, which resulted in a training time of about 4 hours per image.

As the determination of a success measure for the training of a GAN is still under investigation, a visual check of the generated samples is required and the generator weights $\boldsymbol{\theta}^{(G)}$ that fabricate visually pleasing outputs are stored.

Synthesis: We collected various terrain textures from popular web resources (poliigon.com, freestocktextures.com and textures.com). Figure 4 shows the results of the texture synthesis. For most textures, we obtained decent results regarding the coarse characteristics of the texture. However, the output appears often somewhat grainy and blurred. Most problems occur with large features as present in e.g. regular textures, which the GAN cannot preserve due to the use of a limited filter size during the training. Increasing the filter size (e.g. by switching to SGAN6 or SGAN7) makes the training more expensive but can often be avoided by simply downscaling the input image (e.g. from 1024^2 to 512^2). Furthermore, our proposed color correction is a simple yet effective method of matching the output tone to the input.

The shortcomings of the synthesized images are mostly corrected by the detail transfer (Figure 4). By taking the low-frequency characteristics from the GAN output and the

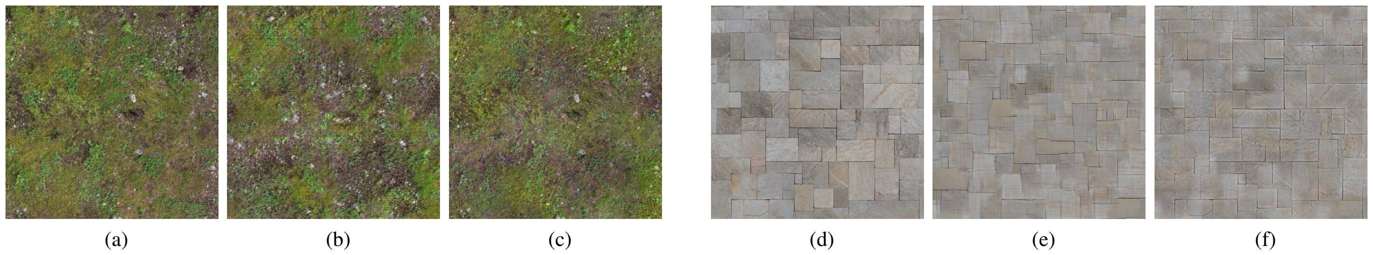


Fig. 4. The input texture (a,d), the texture synthesized with the GAN (b,e) and the result of our technique (c,f) are shown for textures containing a forest ground and tiles. The close-up inspection reveals that the texture generated with our technique is similar to the input texture. When viewed from far, it is similar to the texture synthesized with the GAN. Note that handling the tile texture is particularly challenging due to the high-frequency bandwidth of the features but the results are still acceptable.

high-frequency component directly from the input image, we can produce plausible variations of the reference texture that contain both the coarse-scale characteristics as well as fine details within the texture. However, the approach is limited in cases where features are composed of a wide range of frequencies, e.g. a brown stick on brown ground may look greenish if placed into a grass context. In practice, these errors are notable but mostly not conspicuous.

Mostly observing a high quality of the resulting textures, we believe that this detail transfer can also be a valuable tool outside the scope of terrain texturing.

Detail mapping: Figure 5 compares the conventional tiling to detail mapping using our automatically generated maps. Note how the tiling artifacts are greatly reduced while the overall image quality remains the same. We also implemented an interactive demo which allows the user to explore a terrain from various angles and to switch between naive tiling and detail mapping. For typical perspectives, both close by and far away parts of the terrain are visible at the same time, which demonstrates the overall good results on any scale.

Features in low-frequency map: Providing only a single input image to the GAN prohibits it from learning meaningful global features, which are simply not available in the training data. Even though the large terrain texture synthesized by the GAN is different everywhere, its lower frequencies contain mostly noise like features. They are, however, still computed from the input image and plausible when combined with the higher frequencies taken from the reference image. Figure 6 shows the low-frequency components for various materials. Their statistical distributions arise directly from the neural network and there is no straightforward way to extract them directly from the input image, which justifies the GAN training.

If more global structures are desired, these can be added by using approaches like texture splatting [4]. This is completely independent from our approach which focuses on tiling artifact removal rather than global detail generation. Due to the orthogonality of the approaches, they can be directly combined.

Additional data channels: High-quality scenes require multiple texture layers such as normal maps [19], displacement maps, gloss maps or ambient occlusion maps. These maps typically semantically depend on the diffuse color map. In order to synthesize these additional maps in combination with the diffuse color channels, we adapted the method of Jetchev [18] to jointly train the network using diffuse color channels with additional data modes. This is achieved by stacking additional detail maps on top of the diffuse color channels resulting in a d-channel image where $d = 6$, when the combination of a diffuse color map and its normal map is used. The additional channels increase training time and also the memory requirements of the network. For the six-channel image containing diffuse and normal information the time consumed per epoch increased by a factor of 2 taking roughly 240 seconds instead of 120 with a batch size of 64. The results are shown in Figure 7.

V. Conclusion

We present a novel approach to automatically generate detail maps from a single input image using generative adversarial networks. This technique is especially suitable for low-budget productions where costly work by artists or hardware availability might be limited. As our detail mapping is fast regarding both implementation and execution, this approach can be used to automatically improve the visual quality of the output without requiring more resources. It can also be combined with almost any other terrain texturing method which can be used to improve problems not addressed by our method such as the addition of meaningful global features. The detail transfer from original to generated texture could furthermore become a valuable tool for other texture synthesis approaches.

Our proposed method might be extended in multiple directions. While being a fundamental building block of our approach, the implementation of the neural network is largely independent from the rest of our pipeline. Being a very active field of research, it can be expected that texture synthesis algorithms will advance greatly in the future. Those improvements can then be directly transferred to our approach. Furthermore, with the structure of the terrain as input, a neural network could be used to automatically generate low-frequency details that align with terrain features. However, for truly global details, different materials are usually desirable, which can readily be achieved with approaches like texture splatting. In addition, the local influence of the input noise of the SGAN can be exploited in many ways. One way would be the

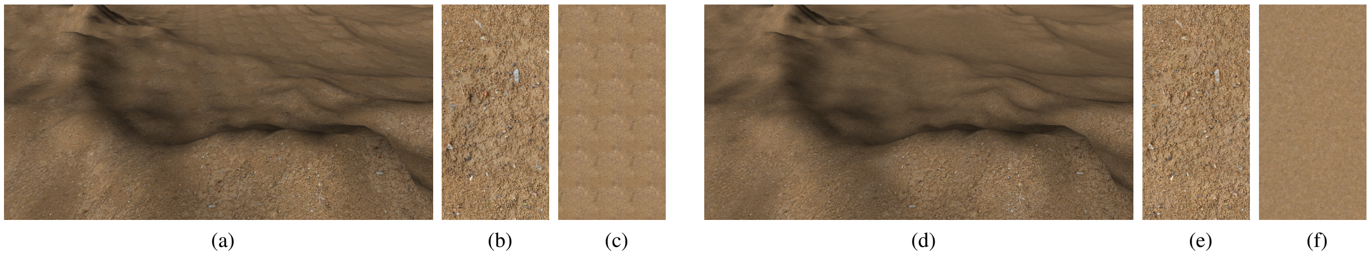


Fig. 5. Naive tiling (left) compared to detail mapping using our automatically generated maps (right). With our approach, the tiling artifacts in distant regions are greatly reduced, while small details in the regions nearby are well preserved.

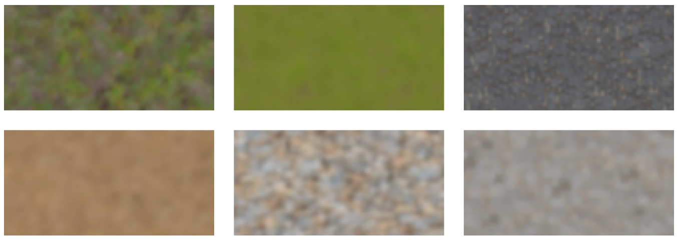


Fig. 6. Comparison of different low-frequency textures. The textures depict forest ground, grass, cobblestone, clay, stone wall and tiles (from top left to bottom right).

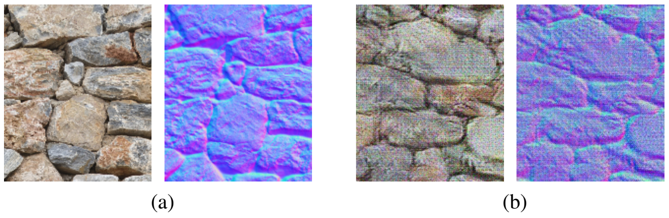


Fig. 7. Generation of multi-modal images. Given the original diffuse map and the normal map as input (a), the GAN generates new plausible versions for the diffuse map and the normal map (b). Other types of information such as height maps can be integrated analogously.

automatic generation of Wang tiles even for large numbers of edge classes.

VI. Acknowledgments

This work was funded by the French-German Research Institute Saint-Louis and the King Abdullah University of Science and Technology (KAUST).

References

[1] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2014, pp. 2672–2680. [Online]. Available: http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf

[2] I. J. Goodfellow, Y. Bengio, and A. C. Courville, *Deep Learning*, ser. Adaptive computation and machine learning. MIT Press, 2016. [Online]. Available: http://www.deeplearningbook.org/

[3] I. J. Goodfellow, "NIPS 2016 tutorial: Generative adversarial networks," *CoRR*, vol. abs/1701.00160, 2017. [Online]. Available: http://arxiv.org/abs/1701.00160

[4] C. Bloom, "Terrain texture compositing by blending in the frame-buffer (aka "splatting" textures)," Nov. 2000. [Online]. Available: http://www.cbloom.com/3d/techdocs/splatting.txt

[5] J. van Waveren, "id tech 5 challenges," in *SIGGRAPH 2009 Course*, 2009.

[6] I. Quilez, "Texture variation v," 2017. [Online]. Available: https://www.shadertoy.com/view/Xtl3zf

[7] M. F. Cohen, J. Shade, S. Hiller, and O. Deussen, *Wang tiles for image and texture generation*. ACM, 2003, vol. 22, no. 3.

[8] Allegorithmic, "Substance designer," 2017. [Online]. Available: https://www.allegorithmic.com/products/substance-designer

[9] J. Portilla and E. P. Simoncelli, "A parametric texture model based on joint statistics of complex wavelet coefficients," *International Journal of Computer Vision*, vol. 40, no. 1, pp. 49–70, Oct 2000.

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

[11] L. A. Gatys, A. S. Ecker, and M. Bethge, "Texture synthesis using convolutional neural networks," in *Advances in Neural Information Processing Systems 28*, May 2015. [Online]. Available: http://arxiv.org/abs/1505.07376

[12] C. Li and M. Wand, "Precomputed real-time texture synthesis with markovian generative adversarial networks," in *European Conference on Computer Vision*. Springer, 2016, pp. 702–716.

[13] C. Beckham and C. Pal, "A step towards procedural terrain generation with gans," *arXiv preprint arXiv:1707.03383*, 2017.

[14] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," *arxiv*, 2016.

[15] E. Gurin, J. Digne, E. Galin, A. Peytavie, C. Wolf, B. Benes, and B. Martinez, "Interactive example-based terrain authoring with conditional generative adversarial networks," *ACM Transactions on Graphics (proceedings of Siggraph Asia 2017)*, vol. 36, no. 6, 2017.

[16] H. Wu, S. Zheng, J. Zhang, and K. Huang, "Gp-gan: Towards realistic high-resolution image blending," *arXiv preprint arXiv:1703.07195*, 2017.

[17] M. O. Vertolli and J. Davies, "Image quality assessment techniques show improved training and evaluation of autoencoder generative adversarial networks," *arXiv preprint arXiv:1708.02237*, 2017.

[18] N. Jetchev, U. Bergmann, and R. Vollgraf, "Texture synthesis with spatial generative adversarial networks," *arXiv preprint arXiv:1611.08207*, 2016.

[19] J. Cohen, M. Olano, and D. Manocha, "Appearance-preserving simplification," in *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '98. New York, NY, USA: ACM, 1998, pp. 115–122. [Online]. Available: http://doi.acm.org/10.1145/280814.280832