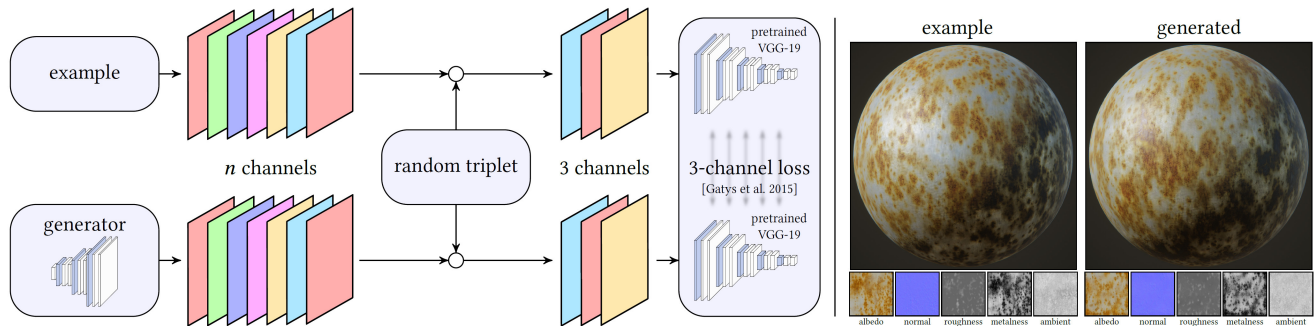


# Passing Multi-Channel Material Textures to a 3-Channel Loss

Thomas Chambon  
Unity Technologies

Eric Heitz  
Unity Technologies

Laurent Belcour  
Unity Technologies



**Figure 1:** We train a material texture generator with multiple channels such as *albedo*, *normal*, *roughness*, *metalness* and *ambient occlusion* by passing random channel triplets to the 3-channel loss proposed by Gatys et al. [2015].

## ABSTRACT

Our objective is to compute a textural loss that can be used to train texture generators with multiple material channels typically used for physically based rendering such as *albedo*, *normal*, *roughness*, *metalness*, *ambient occlusion*, etc. Neural textural losses often build on top of the feature spaces of pretrained convolutional neural networks. Unfortunately, these pretrained models are only available for 3-channel RGB data and hence limit neural textural losses to this format. To overcome this limitation, we show that passing random triplets to a 3-channel loss provides a multi-channel loss that can be used to generate high-quality material textures.

## CCS CONCEPTS

• **Computing methodologies** → **Rendering.**

### ACM Reference Format:

Thomas Chambon, Eric Heitz, and Laurent Belcour. 2021. Passing Multi-Channel Material Textures to a 3-Channel Loss. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Talks (SIGGRAPH '21 Talks)*, August 09-13, 2021. ACM, New York, NY, USA, 2 pages. <https://doi.org/10.1145/3450623.3464685>

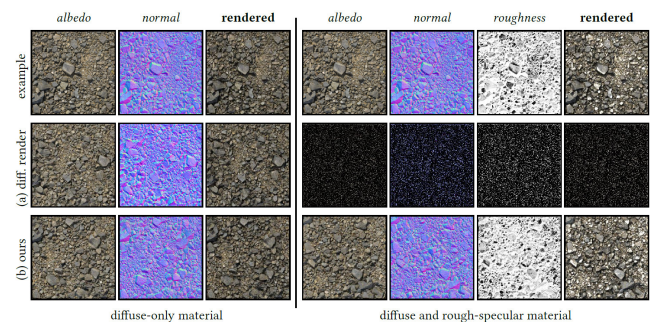
## 1 INTRODUCTION

A neural textural loss allows for generating textures by image optimization [Gatys et al. 2015] or training generative models [Ulyanov et al. 2016]. Typically, the loss is computed from the statistics of the feature activations in pretrained Convolutional Neural Networks (CNNs) such as VGG-19 [Simonyan and Zisserman 2015]. These pretrained CNNs are mainly available for RGB inputs, i.e. a 3-channel formats. This is limiting for material textures used in physically

based rendering that have multiple channels such as *albedo*, *normal*, *roughness*, *metalness*, *ambient occlusion*, etc. We thus investigate how a 3-channel loss can be applied to  $n$ -channel textures.

Our first attempt was inspired by previous work that generates material textures from RGB photographs examples [Aittala et al. 2016]. They use a differentiable renderer to light the material textures and create an RGB render that can be passed to an RGB loss. The material textures can then be optimized via gradient descent by backpropagating gradients through the differential renderer. As shown in Figure 2-(a), we found this approach to be unstable with non-diffuse materials, especially sharp speculars. Indeed, in addition to texture synthesis, the optimizer also needs to solve a challenging inverse rendering problem. Aittala et al. [2016] report using additional priors and considerable engineering efforts.

Fortunately, we can take advantage of explicitly provided material channels and avoid solving a difficult inverse rendering problem if we find a simpler way to pass  $n$  channels to a 3-channel loss. We tested different more or less elaborated ideas such as computing a partial component analysis, training a  $n$ -to-3 channel encoder, etc. In the end, we found that the best solution consists of choosing random triplets in the  $n$  channels. It provides a surprisingly simple and well-founded approach with stable outcome shown in Figure 2-(b).



**Figure 2:** (a) Passing a differentiable render to a 3-channel loss works with diffuse materials but becomes unstable with rough speculars. (b) Our approach is robust and stable.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

SIGGRAPH '21 Talks, August 09-13, 2021, Virtual Event, USA

© 2021 Copyright held by the owner/author(s).

ACM ISBN 978-1-4503-8373-8/21/08.

<https://doi.org/10.1145/3450623.3464685>

## 2 OUR MULTI-CHANNEL TEXTURAL LOSS

*The 3-channel loss.* We build upon the 3-channel textural loss introduced by Gatys et al. [2015]. They define the textural distance between two RGB images  $I_3$  and  $\tilde{I}_3$  as the MSE between the Gram matrices of the activations produced by the images in the  $L$  layers of a pretrained CNN:

$$\mathcal{L}_{3\text{-channel}}(I_3, \tilde{I}_3) = \sum_{l=1}^L \frac{1}{N_l^2} \|G^l - \tilde{G}^l\|^2, \quad (1)$$

where  $G^l$  and  $\tilde{G}^l$  are the Gram matrices of the  $N_l$  deep features extracted from respectively  $I$  and  $\tilde{I}$  at layer  $l$  in the pretrained CNN. In our experiments, we use a pretrained VGG-19 [Simonyan and Zisserman 2015].

*Combining multiple 3-channel losses.* Accounting for more than 3 channels can be done by adding multiple 3-channel losses applied on the different maps. For instance, in Figure 3-(a) we optimize for the sum of two 3-channel losses computed for the *albedo* and the *roughness* triplets separately. This produces a texture whose *albedo* and *normal* look realistic separately but do not match together because the correlations between the albedo and the normal have not been accounted for.

*Our  $n$ -channel loss.* In order to account for all the inter-channel correlations, we define the loss between two  $n$ -channel images  $I_n$  and  $\tilde{I}_n$  as the expectation of the 3-channel loss over *all* the possible triplets:

$$\mathcal{L}_{n\text{-channel}}(I_n, \tilde{I}_n) = \mathbb{E} \left[ \mathcal{L}_{3\text{-channel}}(\text{triplet}(I_n), \text{triplet}(\tilde{I}_n)) \right], \quad (2)$$

where  $\text{triplet}(I_n)$  is a 3-channel image whose channels are chosen randomly among the  $n$  channels of  $I_n$ . As shown in Figure 3-(b),  $\mathcal{L}_{n\text{-channel}}$  preserves inter-channel correlations. The generated textures have the same feature at the same places channel-wide. The downside is a direct evaluation of  $\mathcal{L}_{n\text{-channel}}$  requires evaluating  $\mathcal{L}_{3\text{-channel}}$  for all possible triplets and averaging the results. A material of  $n$  channels hence requires  $n^3$  different evaluations, which is untractable in practice.

*Stochastic evaluation.* To overcome this problem we proceed as shown in Figure 1. Instead of evaluating  $\mathcal{L}_{3\text{-channel}}$  on all the possible triplets, we only evaluate it on a *single* triplet that is randomized for each batch during learning. In other words, we compute a stochastic estimate of Equation (2):

$$\hat{\mathcal{L}}_{n\text{-channel}}(I_n, \tilde{I}_n) = \mathcal{L}_{3\text{-channel}}(\text{triplet}(I_n), \text{triplet}(\tilde{I}_n)). \quad (3)$$

This evaluation is fast, practical and does not change the optimum of the  $L^2$  minimization because the estimate is unbiased. Furthermore, it remains robust because the randomness induced by the stochastic evaluation is similar to the natural randomness of stochastic gradient descent and well-handled by state-of-the-art optimizers.

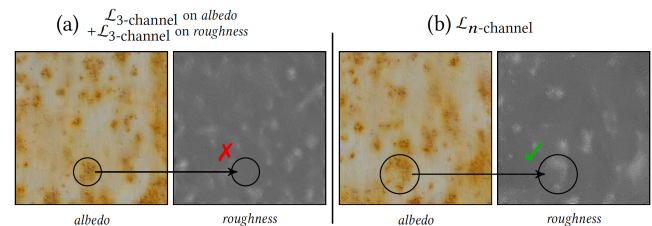
## 3 TRAINING AND RESULTS

We use our loss as a drop-in extension of a 3-channel loss to train generative architectures of  $n$  rather than 3 channels. We train a mono-texture [Ulyanov et al. 2016] and a multi-texture [Li et al. 2017] generators, which we adapted to output  $n$  channels. We use

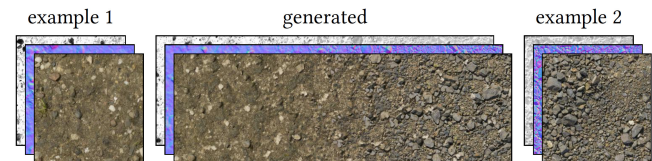
$\hat{\mathcal{L}}_{n\text{-channel}}$  for sole loss function without further priors or regularization terms and we train with the Adam optimizer. Note that we use a vanilla implementation of the 3-channel loss of Gatys et al. [2015]. Several improvements to this loss have been published and implementing them would directly benefit to our  $n$ -channel extension as well. Figure 1 shows a result generated by our mono-texture generator. Figure 4 shows a result generated by our multi-texture generator. These generative architectures are capable of producing arbitrarily-large texture at inference time with variation (no verbatim copying of the exemplar). Our supplemental material shows further results with different sets of channels on various textures.

## 4 CONCLUSION

We have proposed a simple approach to extend existing 3-channel textural losses to  $n$  channels. The main idea is to span all the possible inter-channel correlations thanks to a stochastic evaluation that can be implemented in a single line of code. With this approach we hope to bring a vast literature of neural texture synthesis approaches to material texture synthesis without further efforts.



**Figure 3: (a) Optimizing multiple 3-channel losses on separate triplets produces textures that look realistic independently but that are not correlated, i.e. the spatial features do not match. (b) Our loss preserves inter-channel correlations since it optimizes for all the possible channel combinations.**



**Figure 4: We train a multi-texture generator that allows for interpolating material textures. We trained the same architecture for 16 textures that includes these 2 examples.**

## REFERENCES

- Miika Aittala, Timo Aila, and Jaakko Lehtinen. 2016. Reflectance Modeling by Neural Texture Synthesis. *ACM Trans. Graph.* 35, 4, Article 65 (2016), 13 pages.
- Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. 2015. Texture Synthesis Using Convolutional Neural Networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1 (NIPS'15)*. 262–270.
- Yijun Li, Chen Fang, Jimei Yang, Zhaowen Wang, Xin Lu, and Ming Hsuan Yang. 2017. Diversified texture synthesis with feed-forward networks. In *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*. 266–274.
- Karen Simonyan and Andrew Zisserman. 2015. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *International Conference on Learning Representations*.
- Dmitry Ulyanov, Vadim Lebedev, Andrea Vedaldi, and Victor Lempitsky. 2016. Texture Networks: Feed-Forward Synthesis of Textures and Stylized Images. In *Proceedings of the 33rd International Conference on Machine Learning - Volume 48 (ICML'16)*. 1349–1357.