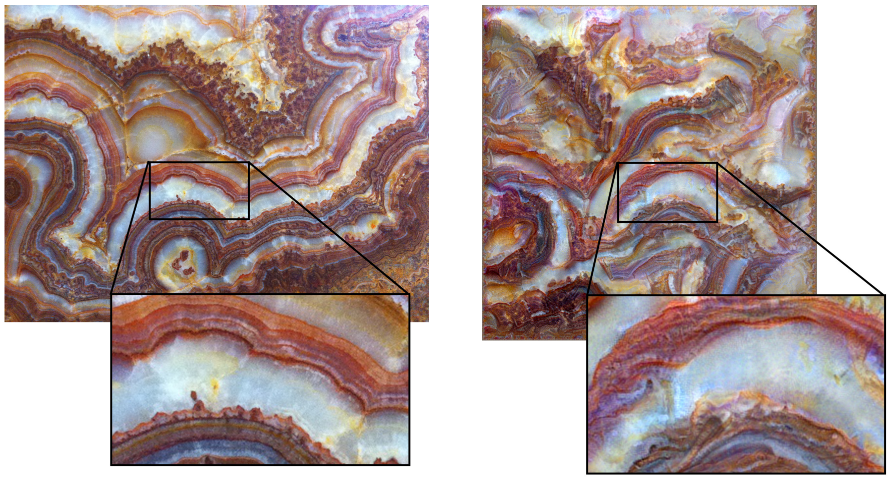
# High-Resolution Multi-Scale Neural Texture Synthesis

Xavier Snelgrove
Toronto, Ontario, Canada
xs@wxs.ca



(a) Source texture (b) Synthesized image

**Figure 1: High resolution texture synthesis matching CNN texture statistics at 5 image scales**

## ABSTRACT

We introduce a novel multi-scale approach for synthesizing high-resolution natural textures using convolutional neural networks trained on image classification tasks. Previous breakthroughs were based on the observation that correlations between features at intermediate layers of the network are a powerful texture representation, however the fixed receptive field of network neurons limits the maximum size of texture features that can be synthesized.

We show that rather than matching statistical properties at many layers of the CNN, better results can be achieved by matching a small number of network layers but across many scales of a Gaussian pyramid. This leads to qualitatively superior synthesized high-resolution textures.

## CCS CONCEPTS

• **Computing methodologies → Texturing**;

## KEYWORDS

texture synthesis, neural networks, Gaussian pyramid

**ACM Reference Format:**
Xavier Snelgrove. 2017. High-Resolution Multi-Scale Neural Texture Synthesis. In *SA '17 Technical Briefs: SIGGRAPH Asia 2017 Technical Briefs, November 27–30, 2017, Bangkok, Thailand.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3145749.3149449

## 1 INTRODUCTION

There have been recent significant improvements in the quality of example-based texture synthesis techniques by taking advantage of intermediate representations in a convolutional neural network (CNN) trained to classify images [Gatys et al. 2015b].

Correlations between features maps at these intermediate layers, represented by a Gram matrix, turn out to be a powerful representation of texture. By synthesizing new images whose Gram matrix is close to that of an exemplar image (for instance via gradient descent), we get images with similar texture.

However, this only works well when the semantically significant features in the image are at the correct scale for the network, and in practice the receptive field of a feature at an intermediate layer for common CNN architectures is relatively small [Luo et al. 2016]. The popular VGG architectures from Simonyan and Zisserman [2014], used by Gatys et al. and others, are trained on 224×224 pixel images, in which relevant features will be quite a bit smaller.

So, given a high-resolution source image, for optimal results an artist must scale down that image until the pixel-scale of the features of interest match the receptive field of the appropriate semantic layer of the network. This limits the resolution of the rendered image, and further breaks down for source images with textures at multiple scales. The artist must choose to capture one scale of texture at the expense of the other.

In this work we propose a multi-scale neural texture synthesis approach, in which the optimization simultaneously matches textures at every layer of a Gaussian pyramid [Adelson et al. 1984]. The artist no longer needs to trade off between image resolution and texture fidelity, and further can synthesize textures with statistics at multiple scales. For neural texture synthesis this issue has been tackled by exploiting side-effects of the optimization process [Gatys et al. 2016b], however despite the long history of multi-scale pyramid methods in texture synthesis[Han et al. 2008; Lefebvre and Hoppe 2005; Portilla and Simoncelli 2000], as far as the authors know this is the first work using these in the context of neural texture synthesis.

## 2 GRAM MATRIX TEXTURE SYNTHESIS

Gatys et al. [2015a] introduced the idea of using the Gram matrix as a spatially invariant representation feature correlations, which has been used in many works since. [Gatys et al. 2016a; Johnson et al. 2016; Saito et al. 2016; Sendik and Cohen-Or 2017].

Their approach is as follows: they take a vectorized source image $\vec{x}$, and feed it into a CNN (in their case, VGG-19 [Simonyan and Zisserman 2014]), computing the activations at each layer of the network.

Defining the number of feature maps for layer $l$ as $N_l$, with vectorized size $M_l$, they define the *feature matrix* for layer $l$ as $F_l \in \mathbb{R}^{N_l \times M_l}$, where $F^l_{jk}$ shows the activation at layer $l$ of the $j^{\text{th}}$ feature in position $k$.

They define the Gram matrix at layer $l$ as $G_l \in \mathbb{R}^{N_l \times N_l}$, the inner product between the vectorized feature map $i$ and $j$ in layer $l$ summing across every neuron in that map. We slightly modify their definition to normalize the representation

$$G^l_{ij} = \frac{1}{M_l\, N_l} \sum_k F^l_{ik}\, F^l_{jk} \tag{1}$$

This Gram matrix parameterizes a texture at a particular layer of the CNN. Entry $G_{ij}$ is proportional to how often feature $i$ and feature $j$ are active in the same spatial location. We can determine the difference between the same layer of a source image and synthesized image by taking the squared Frobenius norm of the difference between the source image's Gram matrix $\hat{G}^l$ and the synthesized image's Gram matrix $G^l$

$$E_l = \sum_{i,j} \left(\hat{G}^l_{ij} - G^l_{ij}\right)^2 \tag{2}$$

In order to synthesize a new image Gatys et al. directly minimize the sum of these terms across multiple layers:

$$\mathcal{L}(\hat{\vec{x}}, \vec{x}) = \sum_{l=0}^{L} w_l\, E_l \tag{3}$$

with $w_l$ a selectable weighting factor. They use the L-BFGS optimizer [Zhu et al. 1997] with analytic gradients since every operation is differentiable.

## 3 MULTI-SCALE TEXTURE SYNTHESIS

Natural textures may contain regular structure at multiple scales. Portilla and Simoncelli [2000] showed very good texture synthesis with a bank of multi-scale linear filters called a "steerable pyramid".

Gatys' work can be seen as an extension of (and indeed credits) Portilla and Simoncelli's, but instead of a multi-scale *linear* filter-bank uses a *non*-linear multi-scale filter bank that is given by the CNN. This is one reason why Equation 3 sums across many layers of the CNN.

However even at the highest layers of the CNN the effective receptive field is relatively small [Luo et al. 2016]. What's more, the degree of abstraction increases with higher levels of the CNN so it is not a purely multi-scale representation but a multi-scale multi-semantic representation.

All this conspires to lead to Gatys' approach not being suited to high-resolution images, which are effectively images with long-range spatial dependencies between pixels [Berger and Memisevic 2016].

We propose resolving this by disentangling the semantic and spatial terms in the optimization. We pick only one or two intermediate layers of VGG-19, but we feed in many layers of a Gaussian pyramid [Adelson et al. 1984]. Each layer of a Gaussian pyramid is formed by blurring and downsampling the previous layer.

We can now modify the equations of Section 2 to incorporate scale. Let the feature matrix for the $s^{\text{th}}$ scale octave of the Guassian pyramid and $l^{\text{th}}$ CNN layer as $F_i k^{l,s}$, and the corresponding Gram matrix be

$$G^{l,s}_{ij} = \frac{1}{M_l\, N_l} \sum_k F^{l,s}_{ik}\, F^{l,s}_{jk} \tag{4}$$

We can now modify Equations 2 and 3 to include terms for both layers and scales.

$$E^s_l = \sum_{i,j} \left(\hat{G}^s_{ij} - G^s_{ij}\right)^2 \tag{5}$$

and

$$\mathcal{L}(\hat{\vec{x}}, \vec{x}) = \sum_{s=0}^{S-1} v_s \sum_{l=0}^{L-1} w_l\, E^s_l \tag{6}$$

where $S$ is the number of octaves. In practice we only set $v_s$ (octave weights) and $w_l$ (CNN layer weights) to values of either 0 or 1, uniformly weighting all layers and scales of interest. Refer to Figure 2 for a block diagram of our system.

## 4 EXPERIMENTAL RESULTS

For our experiments we use the VGG-19 architecture, but replace the max-pooling layers with average-pooling layers, following Gatys et al. and further use their re-scaled VGG-19 weights[1] which normalize the average activation of every feature map across images to 1.

We use valid-mode convolution for all convolutional layers, which is important for boundary effects, especially when synthesizing images of a different scale than the source image (so with a different ratio of boundary to internal pixels).

We find good results in general using 5 octaves in our Gaussian pyramid, and matching simultaneously against layers `block1_pool`

[1]Available at https://github.com/leongatys/DeepTextures

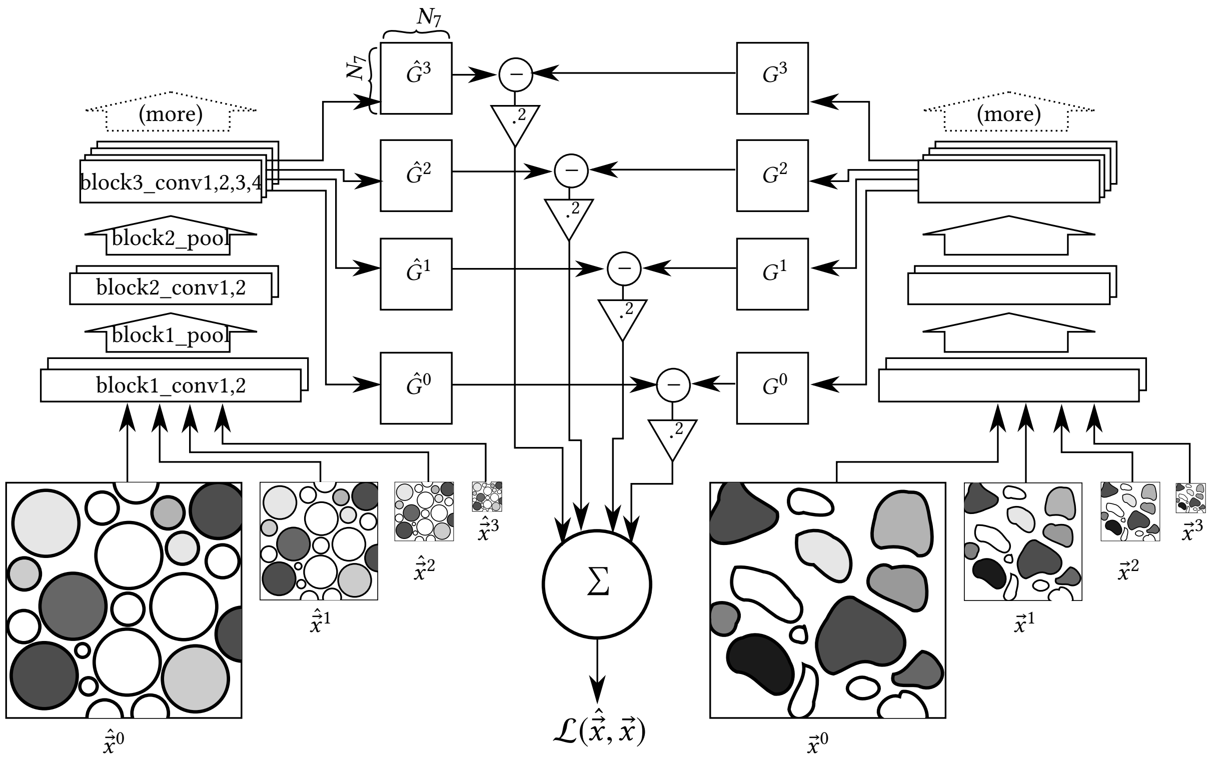


**Figure 2: Block diagram of our system. The left-hand-side shows the source image, with Gram matrices of feature correlations being extracted for VGG-19 layer `block3_conv2` across 4 distinct spatial scales. On the right is the current state of the optimization $\vec{x}$, whose feature correlation matrices are extracted in the same fashion. The sum of the squared difference of these matrices corresponds to the loss function $\mathcal{L}$ which is minimized by gradient descent on $\vec{x}$.**

and `block3_conv2` (the 3rd and 8th layers). Using both a low level layer and an intermediate layer sometimes allows the optimization to escape local optima that it would get stuck in using only the higher level layer.

Figure 3 shows an example result. The source texture in 3a shows texture at multiple scales. At the macro scale it is dominated by regions of blue paint and rusty-red wall, with red streaks in the blue regions. At the micro-scale we see that the red region has a rough texture, whereas the blue region has a smooth texture, and a distinct rounded "paint-chipping" shape at the border between the regions. All of these effects are captured in the synthesized image 3b. Using Gatys et al. we find that some of the micro-scale properties are captured, but the macro-scale red/blue alternation is almost completely missing. Further we can see strong artefacts around the bright blue point in the corner. Refer to Figure 4 for many more results.

Experiments were performed using the Keras [Chollet et al. 2015] neural networks framework. We used the Scipy [Jones et al. 2001] implementation of L-BFGS-B [Zhu et al. 1997]. The code for these experiments is available online[2].

## 5 DISCUSSION

This work has shown the power of using multi-scale representations in conjunction with neural texture synthesis. Gatys et al. can be seen as a special-case of this work for a single scale octave $S = 1$. As such, much of work building on their approach would also be applicable to ours.

A natural extension would be to attempt multi-scale style transfer, following [Gatys et al. 2015a] but using our objective function to represent the "style loss".

We use a simple optimization process to synthesize our images. Other work has shown that feed-forward neural networks can be trained to approximate this optimization [Johnson et al. 2016] so it would be interesting to attempt those methods with our multi-scale objective. It is likely that some similar multi-scale approach would be required for the architecture of that network as otherwise the same receptive-field issues would apply for that feed-forward network.

Finally, combing our work with other approaches for finding larger structures in images such as by finding symmetries as in [Berger and Memisevic 2016; Sendik and Cohen-Or 2017] would also be interesting.

[2]http://github.com/wxs/subjective-functions

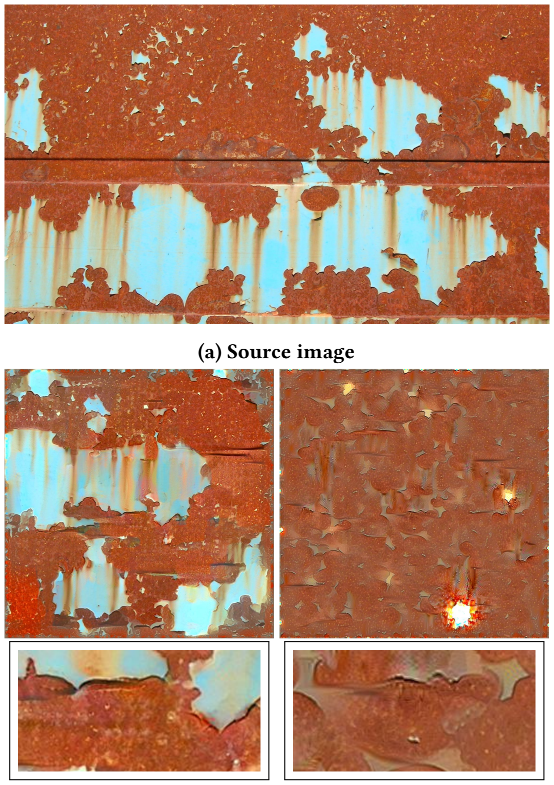


**Figure 3: Our results on a photograph with texture at multiple scales. Zoom in to see the small-scale surface texture within the red and blue areas.**

Photo credit: https://commons.wikimedia.org/wiki/File:Detail_of_rusty_van.JPG

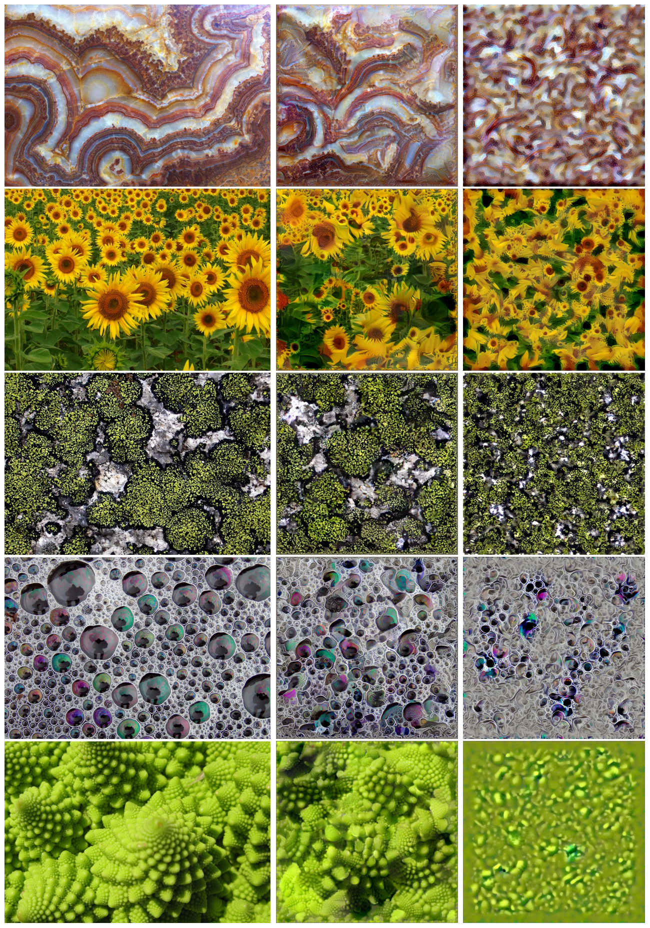


**Figure 4: Comparing our results to Gatys et al. [2015b] on a variety of source images with texture at multiple scales. Generated images are high resolution at** $1024 \times 1024$ **pixels, so best viewed zoomed-in on screen. Our results match against both layers block1_pool and block3_conv2 at 5 scale octaves. Gatys et al. matching layers `conv1_1`, `block1_pool`, `block2_pool`, `block3_pool`, `block4_pool`. Note that source textures are slightly cropped for display here**

Photos all creative commons licensed on Wikimedia Commons by users: HaleiLaihaweadu, MikeLynch, Vik Nanda, Brocken Inaglory, Jacopo Werther

## ACKNOWLEDGMENTS

We would like to thank Laura Lewis, Martin Snelgrove, Claire Freeman-Fawcett, and Severin Smith for useful discussions and for reading drafts of this work.

## REFERENCES

E H Adelson, C H Anderson, and J R Bergen. 1984. Pyramid methods in image processing. *RCA Eng.* 6, 29 (1984), 33–41.
G Berger and R Memisevic. 2016. Incorporating long-range consistency in CNN-based texture generation. (June 2016). arXiv:1606.01286
François Chollet et al. 2015. Keras. https://github.com/fchollet/keras. (2015).
Leon A Gatys, Matthias Bethge, Aaron Hertzmann, and Eli Shechtman. 2016a. Preserving Color in Neural Artistic Style Transfer. (June 2016). arXiv:1606.05897
Leon A Gatys, Alexander S Ecker, and Matthias Bethge. 2015a. A Neural Algorithm of Artistic Style. (Aug. 2015). arXiv:1508.06576
Leon A Gatys, Alexander S Ecker, and Matthias Bethge. 2015b. Texture Synthesis Using Convolutional Neural Networks. *NIPS* cs.CV (2015).
Leon A Gatys, Alexander S Ecker, Matthias Bethge, Aaron Hertzmann, and Eli Shechtman. 2016b. Controlling Perceptual Factors in Neural Style Transfer. (Nov. 2016). arXiv:1611.07865
Charles Han, Eric Risser, Ravi Ramamoorthi, and Eitan Grinspun. 2008. Multiscale texture synthesis. *ACM Transactions on Graphics* 27, 3 (2008), 1.
Justin Johnson, Alexandre Alahi, and Li Fei-Fei. 2016. Perceptual Losses for Real-Time Style Transfer and Super-Resolution. In *Computer Vision – ECCV 2016*. Springer, Cham, Cham, 694–711.
Eric Jones, Travis Oliphant, Pearu Peterson, et al. 2001. SciPy: Open source scientific tools for Python. (2001). http://www.scipy.org/ [Online; accessed May 23, 2017].
Sylvain Lefebvre and Hugues Hoppe. 2005. Parallel controllable texture synthesis. *ACM Transactions on Graphics (TOG)* 24, 3 (July 2005), 777–786.
Wenjie Luo, Yujia Li, Raquel Urtasun, and Richard S Zemel. 2016. Understanding the Effective Receptive Field in Deep Convolutional Neural Networks. *NIPS* (2016).
Javier Portilla and Eero P Simoncelli. 2000. A Parametric Texture Model Based on Joint Statistics of Complex Wavelet Coefficients. *International Journal of Computer Vision* 40, 1 (2000), 49–70.
Shunsuke Saito, Lingyu Wei, Liwen Hu, Koki Nagano, and Hao Li. 2016. Photorealistic Facial Texture Inference Using Deep Neural Networks. (Dec. 2016). arXiv:1612.00523
Omry Sendik and Daniel Cohen-Or. 2017. Deep Correlations for Texture synthesis. *ACM Transactions on Graphics* (April 2017), 1–15.
Karen Simonyan and Andrew Zisserman. 2014. Very Deep Convolutional Networks for Large-Scale Image Recognition. (Sept. 2014). arXiv:1409.1556
Ciyou Zhu, Richard H Byrd, Peihuang Lu, and Jorge Nocedal. 1997. Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization. *ACM Transactions on Mathematical Software (TOMS)* 23, 4 (Dec. 1997), 550–560.