Deep Correlations for Texture synthesis

OMRY SENDIK and DANIEL COHEN-OR
Tel-Aviv University

Example-based texture synthesis has been an active research problem for over two decades. Still, synthesizing textures with non-local structures remains a challenge. In this paper, we present a texture synthesis technique that builds upon convolutional neural networks and extracted statistics of pre-trained deep features. We introduce a structural energy, based on correlations among deep features, which capture the self-similarities and regularities characterizing the texture. Specifically, we show that our technique can synthesize textures that have structures of various scales, local and non-local, and the combination of the two.

Categories and Subject Descriptors: I.3.7 [**Computer Graphics**]: Image manipulation—*Texturing*

Additional Key Words and Phrases: texture synthesis, autocorrelation, neural networks

ACM Reference Format:

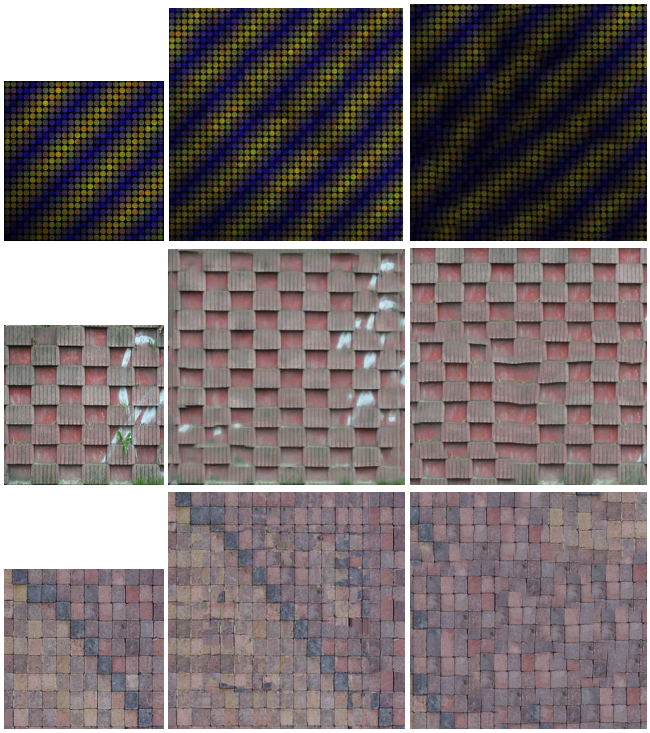
Omry Sendik and Daniel Cohen-Or. Deep Correlations for Texture synthesis.

1. INTRODUCTION

Example-based texture synthesis has been an active research area in the last two decades. Many methods have been developed for synthesizing a texture that is visually similar to a given input texture sample [Wei et al. 2009]. Parametric methods [Heeger and Bergen 1995] deal well with stationary and homogeneous textures, but are limited in their ability to analyse and synthesize structures. Non-parametric methods [Efros and Leung 1999; Wei and Levoy 2000; Efros and Freeman 2001] can handle small scale structures by employing patches large enough to capture the local structures. Similarly, optimization-based methods [Wexler et al. 2004; Kwatra et al. 2005] can mainly handle structures that can be represented within the patch size. Still, synthesizing textures with non-local structures is challenging.

Recently, Gatys et al. [2015] presented a new type of texture synthesis technique which is based on convolutional neural networks and extracted statistics of pre-trained deep features. In this paper, we present a texture synthesis technique that builds upon their approach, and can deal with structured textures. The premise of our work is that textures are inherently characterized by strong self-similarities and regularities. The core idea of our work is a *structural energy* that captures these regularities. We introduce a structural matrix, which represents *deep correlation* among deep features. Figure 1 shows the results of our synthesis algorithm applied to structural textures of various scales. The results are compared side by side with the result of a recent advanced patch-based optimization [Kaspar et al. 2015]. Note how the non-local structures of the input exemplar, as well as the local ones, are synthesized well by our deep correlations.

The deep correlation that we introduce is closely related to autocorrelation which reflects regularities in a signal. However, the deep correlation that we introduce is tuned to be more sensitive to the underlying structure of the texture. Figure 2 shows a comparison between the autocorrelation matrix and the structural matrix



(a) (b) (c)

Fig. 1: Synthesis of textures of various structural scales. (a) the texture exemplars; (b) our results with deep correlations, and (c) results with the patch-based optimization of Kaspar et al. [2015].

of the deep correlation. Clearly, the structural matrix better reflects the structural regularities. We show that based on these structural matrices of deep correlations we can synthesize a given texture, possibly with regular structures, by optimization, where the objective function is driven by the energy of the structural matrix of the texture sample.

An important property of the structural matrices is that they do not explicitly enforce specific image content (edges, corners, etc.) to appear in specific locations. The gist of our approach is that the structural energy does not encode image coordinates, but relative offsets among features, enabling generating textures that are visually and structurally similar to the exemplar. Unlike previous methods where the scale of a patch determines the scale of the target structure, here, our method has no parameters to control the scale of the target structure. Specifically, as shown in Figure 1 our technique can synthesize textures that have structures of various scales, local and non-local, and the combination of the two (note the small scale tiles of various sizes in the three examples and the non-local diagonal patterns or the graffiti in the middle row).

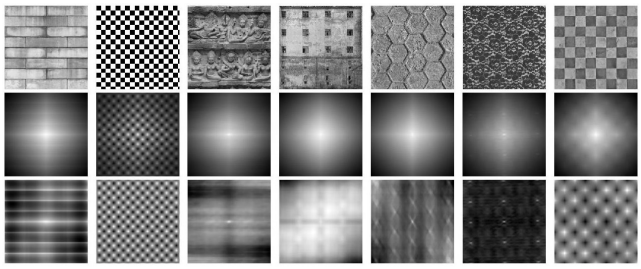


Fig. 2: **Structural Matrix vs. Autocorreleation matrix**. The upper row shows the images on which the correlations were calculated. The middle row shows their respective autocorrelation matrices while the third row shows their structural matrices.

2. BACKGROUND

2.1 Parametric Texture Synthesis

Texture analysis and synthesis is an intriguing problem which has drawn wide attention in the past two decades [Wei et al. 2009]. Early works aimed to analyse textures with respect to human perception, studying the features and their statistics. The prominent work by Heeger and Bergen [1995] involved iteratively matching histograms across image scales between the sampled texture and the synthesized one. Portilla and Simoncelli [2000] improved the synthesis performance using joint statistics by treating structures in the texture exemplar. Starting with random noise, they alternate between matching the sample statistics of steerable (tunable) filters to those of the exemplar and reconstruct a texture from these filters by optimizing the statistics of the output to match those of the exemplar.

To improve the reproduction of structured textures, De Bonet [1997] generated and sampled from a joint distribution of the texture features across scales. In our work, we do not design filters, but rather use the features pre-learned by a neural network, and present a novel correlation target among these features to analyse the textures and their structures.

Galerne et al. [2011] targeted at synthesizing weakly structured textures by enforcing the Fourier power spectrum of images using a random phase approach. They synthesize new textures by randomizing the Fourier phase of the source image.

2.2 Non-Parametric Texture Synthesis

Patch-based techniques bypassed the need to analyse the sample texture, and directly synthesize the texture and composed it by combining patches taken from the sample texture. The earlier works [Efros and Leung 1999; Wei and Levoy 2000; Efros and Freeman 2001] used Markov Random Fields, where the probabilities are taken from the exemplar texture. Later patch-based optimization techniques, where all pixels are simultaneously synthesized were developed [Wexler et al. 2004; Kwatra et al. 2005; Lefebvre and Hoppe 2005; 2006; Risser et al. 2010; Darabi et al. 2012]. These patch-based optimization techniques became the state-of-the-art in texture synthesis due to their simplicity and consequently their speed. They limit randomness for preserving regularity. In particular, Risser et al. do it via an analysis of the initial structure in the exemplar while Lefebvre et al. allow the user to control it manually. Furthermore, these techniques have a larger scope since the patches capture local structures well. However, as identified by Liu et al. [2004] they do not capture non-local structures. In their work, Liu et al. developed a method to track and explicitly learn near-regular structures with a user-assisted lattice extraction.

2.3 Gram-based Texture Synthesis

Recently, Convolutionl Neural Networks (CNN) stormed in and enabled tasks which were previously extremely difficult. One of the major break-throughs enabled by CNNs is their alleviation of the need to intricately design filter banks or image features. The "deep" image features are implicitly learned by means of huge neural networks trained to solve complex tasks.

Recently, Gatys et al [2015] presented a solution which employs CNNs to synthesize textures. They defined an inter-feature loss, which involves calculating the inner product between sets of feature vectors and encoding them in Gram matrices. The deep features were extracted from the texture exemplar using a pre-trained CNN (VGG-19) [Simonyan and Zisserman 2014]. Their method synthesizes a target texture by iteratively optimizing an initial noise image using an objective which enforces the synthesized texture to have a similar Gram matrix to the one of the exemplar texture. Following Gatys et al., Li and Wand [2016] combined MRF priors with CNN to significantly improve synthesis quality. For global structure consistency, they imposed explicit layout constraints through a "content" image.

Since our method builds upon Gatys' and uses their method as a reference, we elaborate on this Gram-based method.

Let I denote the target texture and $\widetilde{I}$ the exemplar texture. Extracting the Gram matrix involves running an image I or $\widetilde{I}$ forward through a neural network, which yields the deep features. We denote the n^{th} feature channel of the l^{th} layer by $f^{l,n}$.

Using the Gram matrix, between n and the n' feature channel

$$G^l_{n,n'} = \sum_{q,m} f^{l,n}_{q,m} f^{l,n'}_{q,m}, \tag{1}$$

where q and m are the indices running across the $Q \times M$ feature map. We denote the loss associated with the Gram matrices as E^l_{Grm} which is given by

$$E^l_{Grm} = \frac{1}{(2QM)^2} \sum_{n,n'} \left(G^l_{n,n'} - \widetilde{G}^l_{n,n'} \right)^2, \tag{2}$$

where $\widetilde{G}$ is the Gram matrix extracted from $\widetilde{I}$.

The total Gram loss, summing over the various selected layers which take part in this loss is given by

$$E_{Grm} = \sum_l w^G_l E^l_{Grm}, \tag{3}$$

where w^G_l are hyper parameter weights which allow setting the various Grams' relative influence.

The Gram-based optimization successfully synthesizes textures with visually similar characteristics. However, the synthesized textures ignore the structural form. The method we present deals with a larger scope of textures that may include structures.

Aittala et al. [2016] have successfully applied Fourier-domain priors within texture synthesis. Liu et al. proposed to augment the Gram-based loss by incorporating low frequency constraints to allow the synthesis of textures having large scale regularity. This is achieved by adding spectral information, using the images' Fourier Transform, in a loss function. We believe that by controlling the structure of deep features rather than the pixel values directly, we can tolerate minor pixel differences between objects with similar semantics. For example, the correlations of the series of statues as

in Figure 8 are more significant in feature space rather than in image space (or a linear transformation of it, i.e., its Fourier Transform).

3. DEEP CORRELATIONS

Textures by definition contain repeated and similar patterns and often have structural form. To allow synthesis of a broad scope of textures, we present, a novel energy term, denoted by E_{DCor} whose role is to direct the optimization process towards preserving regularity. As we shall see, it can be used to synthesize textures with or without the Gram loss. Our deep correlation energy term is motivated by the fact that the Gram matrix only makes use of inter-feature correlations whereas image structures are represented by the intra-feature correlations. Based on this insight, we define the set of deep correlation matrices at different layers by:

$$R^{l,n}_{i,j} = \sum_{q,m} w_{i,j} f^{l,n}_{q,m} f^{l,n}_{q-i,m-j}. \tag{4}$$

where $i \in [-Q/2, Q/2]$ and $j \in [-M/2, M/2]$ or $R^{l,n} \in \mathbb{R}^{Q \times M}$

This amounts to shifting $f^{l,n}_{q,m}$ by i pixels vertically and j pixels horizontally and applying a point-wise multiplication across the overlapping region, weighted by the inverse of the total amount of overlapping regions. That is:

$$w_{i,j} = \left[(Q - |i|)\,(M - |j|) \right]^{-1}. \tag{5}$$

Based on the above deep correlation matrix, we define the feature structural loss for the l^{th} layer by

$$E^{l}_{DCor} = \frac{1}{4} \sum_{i,j,n} \left(R^{l,n}_{i,j} - \widetilde{R}^{l,n}_{i,j} \right)^2. \tag{6}$$

To run a backpropagation iteration using this loss, we derive it

$$\begin{aligned}
\frac{\partial E^{l}_{DCor}}{\partial f^{l,n}_{q,m}} &= \frac{\partial}{\partial f^{l,n}_{q,m}} \left[\frac{1}{4} \sum_{i,j,n} \left(R^{l,n}_{i,j} - \widetilde{R}^{l,n}_{i,j} \right)^2 \right] \\
&= \frac{1}{2} \sum_{i,j} w_{q-i,m-j} f^{l,n}_{q-i,m-j} \left(R^{l,n}_{i,j} - \widetilde{R}^{l,n}_{i,j} \right) + \\
&\qquad \frac{1}{2} \sum_{i,j} w_{q+i,m+j} f^{l,n}_{q+i,m+j} \left(R^{l,n}_{i,j} - \widetilde{R}^{l,n}_{i,j} \right) \\
&= \left[\sum_{i,j} w_{q+i,m+j} f^{l,n}_{q+i,m+j} \left(R^{l,n}_{i,j} - \widetilde{R}^{l,n}_{i,j} \right) \right] \\
&= \left[w \cdot f^{l,n} \circledast \left(R^{l,n} - \widetilde{R}^{l,n} \right)^T \right]_{q,m},
\end{aligned}$$

where $\circledast$ denotes a convolution.

Similar to the Gram loss, we weigh the various layers' DCor losses, to yield the total DCor loss

$$E_{DCor} = \sum_{l} w^{D}_{l} E^{l}_{DCor}. \tag{7}$$

We emphasize that the normalization of the correlations is key, as it provides equal influence to each offset (see Figure 2) and therefore improves the SNR of the matrix. Empirically, we found that the normalized deep correlations better preserve the fine or less noticeable structures. In Section 6 we show that the deep correlations without normalization aid in preserving structure compared to using the Gram loss alone. We also show that using the deep correlations (with normalization) improves the structure preservation even further and grants it the ability to preserve finer structures.

We stress that our deep correlation is targeted at structured textures which are approximately stationary , and thus they are a function of offsets. This is demonstrated in Figure 3, where different crops of a larger structural texture are shown. Their autocorrelation matrices, which were calculated directly on the pixel values (after converting them to gray scale in this illustration) convey similar patterns. Minimizing the difference between the source and target textures can then generate various output textures with similar structure.

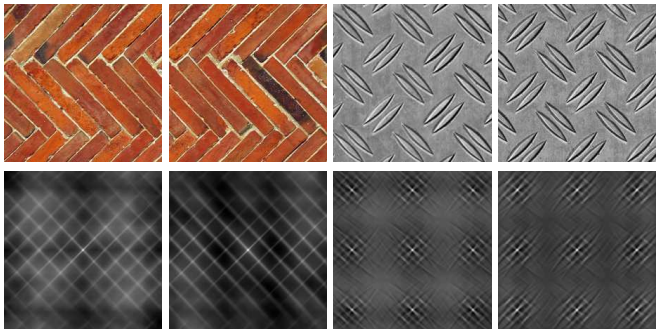


Fig. 3: This figure illustrates that various examples with similar structures but different pixel contents have very similar autocorrelation matrices.

3.1 Encouraging Diversity

Optimization has a tendency to reproduce the input texture when the texture representation is over-complete. In the following we present two methods that we developed to alleviate this tendency and encourage diversity.

By standard definition, the correlation matrix of a given image I has the same size as the input image. Thus, in case the exemplar texture dimension is not equal to the one of the synthesized texture, their corresponding correlation matrices are not of the same size and the expression $(R - \widetilde{R})$ is ill-defined.

When the exemplar image $\widetilde{I}$ is exactly the size of the target image I, calculating $\widetilde{R}$ involves shifting $f^{l,n}_{q,m}$ by up to $Q/2$ and $M/2$ in each direction and applying a point-wise multiplication across the overlapping regions, weighted by the total overlapping area. However, with these overlapping weights (Equation 5) we can generate a matrix $\widetilde{R}$ with larger shifts yielding a larger matrix. This extended matrix $\widetilde{R}$ can be thought of as the deep correlation matrix of some larger input example $\widetilde{I}$. The weighted overlaps allow the output to extend up to $(2Q-1) \times (2M-1)$. We may exploit this insight in order to synthesize an output larger than the input, effectively enforcing the output to be different than the input. Synthesis of a result which is larger than $(2Q-1) \times (2M-1)$ requires expanding the correlation matrices. In the results presented in Section 6 we make use of toroidal expansions, by applying a modulus operation on the image coordinates.

Figure 4 shows two images (upper row) and their corresponding deep correlation matrices (lower row): the left one was generated from a large texture by shifting it by $Q/2$ and $M/2$ in each direction

which yields a correlation matrix equal in its size to the input image. The right correlation matrix was generated from a cropped version of the same input, while shifting by more than $Q/2$ and $M/2$, in order to achieve a larger correlation matrix. As can be seen, the two matrices are quite similar, and have similar structural features at the same relative locations.

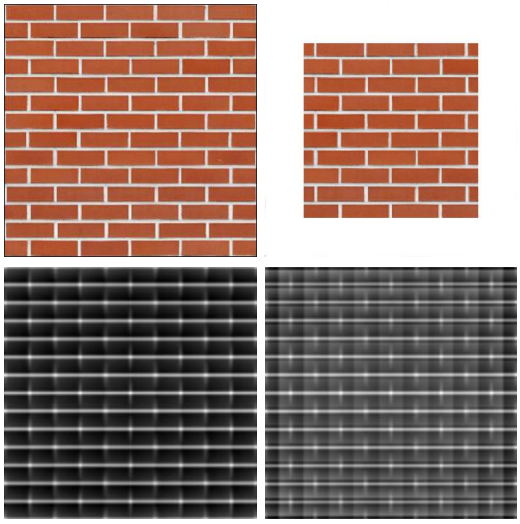


Fig. 4: The extended deep correlation matrix generated from a smaller exemplar can have similar structure as the one generated from a larger exemplar.

The second mechanism for encouraging diversity is an additional loss function that penalizes for synthesizing deep features which are similar to those of the exemplar:

$$E^l_{Div} = \frac{1}{2}\sum_{q,m,n}\left(f^{l,n}_{q,m} - \widetilde{f}^{l,n}_{q,m}\right)^2. \quad (8)$$

We weigh different layers' diversity losses, to yield the final diversity loss

$$E_{Div} = \sum_l w^D_l E^l_{Div}. \quad (9)$$

In Figure 5 we show the generation of different results using our diversity control methods.

4. SMOOTHNESS PRIOR

To enhance the synthesized textures and make them more visually pleasing, we add another loss term. Commonly, [Johnson et al. 2016] a smoothness loss is defined as a total variation term. We found that such a term tends to over-smooth the texture since it is oblivious to the edges. Instead, we propose an edge preserving loss. Our smoothness term does not penalize in cases where at least one adjacent pixel's value is similar. Like a soft-min loss, we wish that our loss penalizes only in cases where none of the neighbouring pixels have a value which is similar to the pixel under consideration. This should also alleviate the common checker artefacts which CNNs tend to produce due to the unbalanced filters and strides. Our smoothness term is given by,

$$E^l_{smooth} = \frac{1}{2\sigma}\sum_{i,j,n}\log\sum_{\delta i,\delta j}\exp\left[-\sigma\left(f^{l,n}_{i,j} - f^{l,n}_{i+\delta i,j+\delta j}\right)^2\right], \quad (10)$$

where $\delta i, \delta j$ are the neighbourhood indices.

This term preserves the sharpness along directions perpendicular to edges. It functions by returning a value which is approximately the minimal value out of the summed differences. Hence, it is sufficient that at least one neighbour pixel is similar for the loss not to penalize.

Similarly to the other losses, we weigh different layers' smoothness losses, to yield the final smoothness loss

$$E_{smooth} = \sum_l w^S_l E^l_{Smooth}. \quad (11)$$

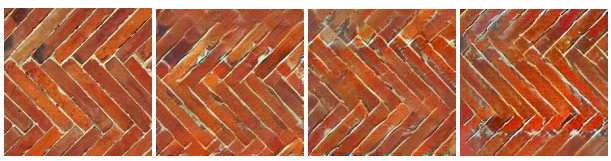


Fig. 5: A structural texture on the left and three different synthesized results. These results were obtained by the diversity mechanism that extends deep correlation matrices.

5. STRUCTURAL TEXTURE SYNTHESIS

Given a texture exemplar $\widetilde{I}$, the synthesis of a texture I is the result of an optimization process executed in a convolutional network which involves forward and backward passes of I through the CNN. First, a single forward pass on $\widetilde{I}$ generates $\widetilde{G}^l_{n,n'}$, $\widetilde{R}^{l,n}_{i,j}$ and $\widetilde{f}^{l,n}_{q,m}$ which are then used to calculate the total loss function E. Given the Gram and deep correlation matrices of both I and $\widetilde{I}$, the loss can be calculated and derived. The total loss we make use of is given by,

$$E = \alpha E_{DCor} + \beta E_{Grm} + \eta E_{Div} + \gamma E_{Smooth}. \quad (12)$$

However, to demonstrate the descriptive power of the deep correlations, we first show its performance in synthesizing textures without the Gram term. That is:

$$E_d = \beta E_{DCor} + \eta E_{Div} + \gamma E_{Smooth}. \quad (13)$$

Figure 9 displays sample textures and the synthesized textures with and without the Gram term. The results of applying the DCor term without the Gram term are surprisingly good. The Gram term alone cannot handle the structures. Note, however, as shown in column (b), that the combination of the two terms yields better results.

To summarize, the optimization scheme is illustrated in Figure 6 (the details about the convolutional network shall be provided in the following section). In each forward pass, $G^l_{n,n'}$, $R^{l,n}_{i,j}$, diversity and smoothness terms are generated at prescribed layers $\{l\}$. After each forward pass, updated features are computed for updating the loss and its gradients, which are then fed to the backward pass and back-propagated, up to the output layer, to correct and optimize I. The forward pass on the left (in gray background) is applied only once, to generate the target Gram and DCor matrices. The synthesis algorithm is summarized in Algorithm 1

6. EXPERIMENTAL RESULTS

We tested our synthesis scheme on various textures most of which, but not all, contain structures, and compared its performance to those of Gatys et al. [2015], Portilla and Simoncelli [2000] and

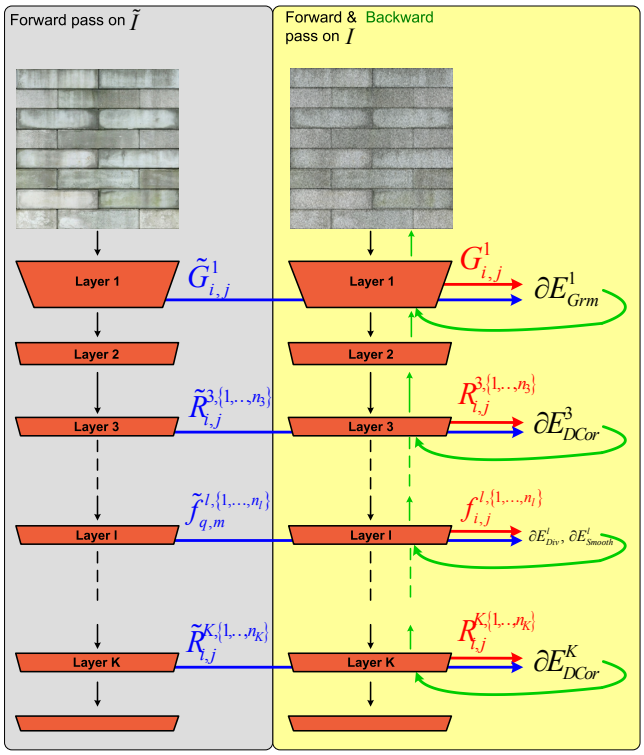


Fig. 6: **Texture Synthesis Scheme**. The block on the left with the gray background is executed only once on the exemplar texture to yield the reference matrices. The rectangle with the yellow background consists of the forward and backward iterations. Green arrows are executed during backward iterations only.

Algorithm 1 Texture Synthesis scheme

1: **Input**: image $\widetilde{I}$, weights w_l^G, w_l^D, w_l^S
2: **Forward pass:** Compute $\widetilde{G}^l_{n,n'}, \widetilde{R}^{l,n}_{i,j}, \widetilde{f}^{l,n}_{q,m}$
3: **while** (not converged) **do**
4: **Forward pass:** Compute $G^l_{n,n'}, R^{l,n}_{i,j}, f^{l,n}_{q,m}$
5: Compute $\frac{\partial E}{\partial f^{l,n}_{q,m}}$
6: **Backward pass:** Back propagate and update I

the advanced patch-based optimization of Kaspar et al [2015]. We used images from the CG-Textures dataset [2005] and the Brodatz dataset [Valkealahti and Oja 1998]. We implemented our algorithm using MatConvnet [Vedaldi and Lenc 2014] as our CNN framework. As part of the MatConvnet project, a pre-trained model for VGG-19 is provided. VGG-19 is a 19-layer network, consisting of only small 3-by-3 filters. The network consists of 16 convolution layers and three fully connected layers. No retraining was applied and the provided VGG-19 CNN was used as is, with no changes except in the pooling layers. Following the recommendation in [Gatys et al. 2015], we found the results to be more visually pleasing when the pooling is an average pooling rather than a max pooling one. Hence, all of the pooling layers in our CNN were switched to average pooling ones. Since VGG-19 was trained on images with 224x224 pixels, we rescaled our inputs accordingly. For the Gram based style loss, we made use of the layers 'pool1', 'pool2', 'pool3' and 'pool4' with equal weights for each layer which sum to one. The deep correlation and diversity losses are applied to

	Layer names	Layer weights	Loss weights	Misc.
E_{Grm}	$pool1, pool2,$ $pool3, pool4$	$0.25, 0.25,$ $0.25, 0.25$	α=0.5	
E_{DCor}	$pool2$	1	$\beta = 0.5\times10^{-4}$	
E_{Div}	$pool2$	1	$\eta = -1\times10^{-4}$	
E_{Smooth}	$conv1$	1	$\gamma = -0.75\times10^{-3}$	σ=1×10^{-3}

Table I. : The parameters that were used in our experiments

the 'pool2' layer only. The smoothness loss is computed at the 'conv1' layer only. The choices of the various loss weights were $\alpha = 0.5$, $\beta = 0.5 \times 10^{-4}, \eta = -1 \times 10^{-4}$ and $\gamma = -0.75 \times 10^{-3}$. Table I summarizes all of the algorithm's hyper parameters. In our smoothness term, we make use of 4-neighborhood pixels for calculating E_{Smooth}.

For the optimization process, we tested with both our own plain vanilla gradient descent energy minimization, as well as the BFGS algorithm with bounded constraints [Byrd et al. 1995] and found that the BFGS algorithm yields better results. The image I is initialized as white Gaussian noise with a standard deviation of $\sigma_N = 1 \times 10^{-3}$.

The CNN forward/backward passes and BFGS solver are computationally intensive. The algorithm takes about 15 minutes to generate a 224×224 output image using unoptimized Matlab code on an NVIDIA GeForce GTX 780. The memory consumption is dominated by the need to load the neural network to memory. In our implementation we use VGG-19 which required roughly 1GB of RAM.

6.1 Effect of the various losses

In Figure 7 we show the effect of the smoothness term by synthesizing a texture with exactly the same settings but with various relative strength to the smoothness term γ. The left most image is the input and to its right is the result in which the smoothness is set to the weakest strength. Note the checker artifacts which are frequently present in CNN based image synthesis. As we increase the strength to stronger values, the checkers which are shown in box (A) diminish. On the right most end, we apply the strongest smoothing. The upper row of results demonstrates that edge boundaries are still kept sharp. Moreover, since the smoothness term favours edges, we can observe that new edges start to appear, see boxes (B) to (D).

In Figure 8 we compare between our deep correlation and the unweighted correlation (i.e., does not use the overlapping weights in Equation 5). The left most column presents the input images and the second shows our results using the deep correlation. The third column from the left shows our results using the unweighted correlation. Note that both correlations preserve the general structure of the input better than using only a Gram loss, which is shown in the right most column. Note in the upper row images that the vertical alignment of tiles is preserved significantly better than the Gram loss alone. However, using the unweighted correlations fails to preserve the horizontal tiles placements. The deep (normalized) correlations help revealing these structures within the image which leads to a better reproduction of the results. A similar effect can be observed in the bottom row in which the deep correlations help preserving the vertical borders between the two stacks of statues.

In Figure 9 we present examples of textures synthesized without the Gram loss. Column (a) shows the input texture. In column (b) we use the Gram loss only (without the deep correlations). In

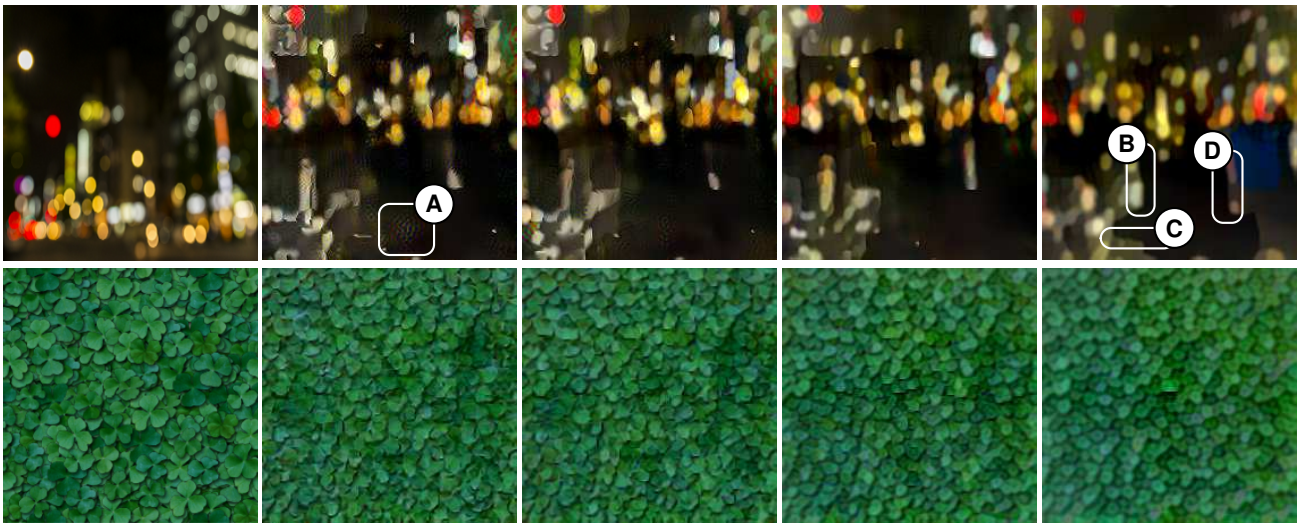


Fig. 7: A comparison of our results as a function of γ. The images from left to right are the input images and the synthesized images with $\gamma = -0.75 \times 10^{-5}$, $\gamma = -0.75 \times 10^{-4}$, $\gamma = -0.75 \times 10^{-3}$ and $\gamma = -0.75 \times 10^{-2}$. Box A shows the checker artifacts which the smoothness loss has prevented. Boxes B to D show the edges which have appeared once the smoothness strength has been increased.

column (c) we set $\alpha = 0$ and by such effectively remove the Gram loss, leaving only the DCor loss. Column (d) presents results of our deep correlation applied together with the Gram loss. In the upper most row, one may note that the general structure is still preserved in column (c), but some of the fine structures and colors are not as plausable as the ones in column (d).

We deliberately synthesize textures with and without regular structures (see the three images in the bottom of the figure). As we can see in column (c), although there are no significant autocorrelations, the deep correlations contain significant information about the texture. Nevertheless, these results show that the deep correlation and the Gram loss are complementary and together provide a holistic solution to both regular and stochastic texture synthesis.

6.2 Comparison with other algorithms

The results in Figures 10 and 11 depict the input images, the Self-tuning algorithm by Kaspar et al [2015], Gatys et al [2015], Portilla and Simoncelli [2000] and our results, respectively. The images in these figures are all taken from the CG-Textures [2005] and the Brodatz dataset [1998]. Running both Portilla and Simoncelli's algorithm and Kaspar et al. algorithm was done using their default parameters, as provided online. For Gatys' algorithm [2015], we made use of the same parameters as they have prescribed in their paper . Clearly the use of the DCor loss preserves the regular structures in these textures. For example, the order of tile placement, in the left most image, is well preserved. Note that our algorithm detects and synthesizes multiscale textures without any parameter tuning. In Figures 10 and 11 we employed our two methods for encouraging diversity, as described in Section 3.1

In Figure 12 we present a comparison between our method and the method by Liu et al. [2016], which builds upon Gatys' approach and attempts to improve it by imposing structural constraints. Boxes (A) to (G) in the figure, emphasize where the structure preservation is unsatisfactory.

In Figures 13 to 15 we present more of our algorithm's results, including some failure cases.

Following Kaspar et al. [2015], we performed an analysis they call texture sequences. This may be regarded as a "stress test" that examines the synthesis quality of an algorithm. The idea is to repeatedly take the synthesized texture output as an input, and observe the gradual degradation of the texture quality through the iterations. The results of our tests are presented in Figures 16 and 17 side by side with those of Kaspar et al. [2015] and Portilla and Simoncelli [2000]. As can be noted, our synthesis preserves well the essence of the texture without introducing many errors. We attribute this by virtue of the fact that small errors that are not systematic nor having a structure, have little affect over our correlation loss. On the other hand, systematic errors, such as checker artifacts and boundary blur, have some correlation that gradually adds up.

7. DISCUSSION AND CONCLUSIONS

In this work we presented texture synthesis based on deep correlations among pre-trained CNN features. We showed that the deep correlations capture the regularities in the texture without encoding the content or the feature positions. The deep correlation targets structural textures, where state-of-the-art methods typically struggle. In particular, our approach excels in handling structures of multiple scales, without any parameter tuning. Moreover, we have shown that the quality of the synthesis is high in the sense that there is only little degradation in the texture reproducibility as shown by a series of stress tests.

Limitations. The results presented in the paper show that synthesizing structured textures with CNN has its merits, and the ease of possibly adding more loss terms has more potential to further improve the scope and quality. However, in our research we found a number of limitations to this approach. First, the synthesis is rather slow as we discussed in the previous section. Second, the results typically suffer from a quality degradation along the image boundaries. These problems are attributed to CNN convolutions which are ill-defined in those regions (the convolution at the image boundary usually uses either pixel padding or mirroring to fill the missing pixels.) To alleviate this issue, we configured our algorithm to syn-

Fig. 8: A comparison of texture synthesis results with deep correlations, unnormalized correlations and a Gram loss only. From left to right we show the input images, the results achieved using deep correlations, the results using unnormalized correlations and the results using the Gram loss only.

thesize results which are 10% larger than the final output size and then crop out the center of the result.

In our research we learned that the Gram loss harms the quality of the results when synthesizing outputs larger than the exemplar. When one scales an input image, different filters become activated in the CNN by virtue of the fact that the filters are not scale invariant. Hence, different features become dominant and the Gram matrix changes its structure. Thus, it is improper to use a Gram matrix extracted from an image of a particular size and then impose an output result, of different size, to have a similar Gram matrix.

Future research. We believe that for improving some of the limitations mentioned above and further extend this work, we can attempt to retrain networks for explicitly coping with near-regular and multi-scale textures. Similarly, inhomogeneous textures may also benefit from CNN retraining. For inhomogeneous textures such as the broken tiles, in which the correlations present no structure, we believe retraining with the correlation matrix as an optimization target may yield features which detect correlations even for such inputs.

REFERENCES

AITTALA, M., AILA, T., AND LEHTINEN, J. 2016. Reflectance modeling by neural texture synthesis. *ACM Transactions on Graphics 35,* 4.

BYRD, R. H., LU, P., NOCEDAL, J., AND ZHU, C. 1995. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing 16,* 5, 1190–1208.

DARABI, S., SHECHTMAN, E., BARNES, C., GOLDMAN, D. B., AND SEN, P. 2012. Image Melding: Combining inconsistent images using patch-based synthesis. *ACM Transactions on Graphics (TOG) (Proceedings of SIGGRAPH 2012) 31,* 4, 82:1–82:10.

DE BONET, J. S. 1997. Multiresolution sampling procedure for analysis and synthesis of texture images. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*. ACM Press/Addison-Wesley Publishing Co., 361–368.

EFROS, A. A. AND FREEMAN, W. T. 2001. Image quilting for texture synthesis and transfer. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*. ACM, 341–346.

EFROS, A. A. AND LEUNG, T. K. 1999. Texture synthesis by non-parametric sampling. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*. Vol. 2. IEEE, 1033–1038.

GALERNE, B., GOUSSEAU, Y., AND MOREL, J.-M. 2011. Random phase textures: Theory and synthesis. *IEEE Transactions on image processing 20,* 1, 257–267.

GATYS, L., ECKER, A. S., AND BETHGE, M. 2015. Texture synthesis using convolutional neural networks. In *Advances in Neural Information Processing Systems*. 262–270.

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*. ACM, 229–238.

JOHNSON, J., ALAHI, A., AND LI, F. 2016. Perceptual losses for real-time style transfer and super-resolution. *CoRR abs/1603.08155*.

KASPAR, A., NEUBERT, B., LISCHINSKI, D., PAULY, M., AND KOPF, J. 2015. Self tuning texture optimization. In *Computer Graphics Forum*. Vol. 34. Wiley Online Library, 349–359.

KWATRA, V., ESSA, I., BOBICK, A., AND KWATRA, N. 2005. Texture optimization for example-based synthesis. *ACM Transactions on Graphics (ToG) 24,* 3, 795–802.

LEFEBVRE, S. AND HOPPE, H. 2005. Parallel controllable texture synthesis. In *ACM Transactions on Graphics (ToG)*. Vol. 24. ACM, 777–786.

LEFEBVRE, S. AND HOPPE, H. 2006. Appearance-space texture synthesis. *ACM Transactions on Graphics (TOG) 25,* 3, 541–548.

LI, C. AND WAND, M. 2016. Combining markov random fields and convolutional neural networks for image synthesis. *arXiv preprint arXiv:1601.04589*.

LIU, G., GOUSSEAU, Y., AND XIA, G. 2016. Texture synthesis through convolutional neural networks and spectrum constraints. *CoRR abs/1605.01141*.

LIU, Y., LIN, W.-C., AND HAYS, J. 2004. Near-regular texture analysis and manipulation. In *ACM Transactions on Graphics (TOG)*. Vol. 23. ACM, 368–376.

PORTILLA, J. AND SIMONCELLI, E. P. 2000. A parametric texture model based on joint statistics of complex wavelet coefficients. *International journal of computer vision 40,* 1, 49–70.

RISSER, E., HAN, C., DAHYOT, R., AND GRINSPUN, E. 2010. Synthesizing structured image hybrids. In *ACM Transactions on Graphics (TOG)*. Vol. 29. ACM, 85.

SIMONYAN, K. AND ZISSERMAN, A. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

VALKEALAHTI, K. AND OJA, E. 1998. Reduced multidimensional co-occurrence histograms in texture classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence 20,* 1, 90–94.

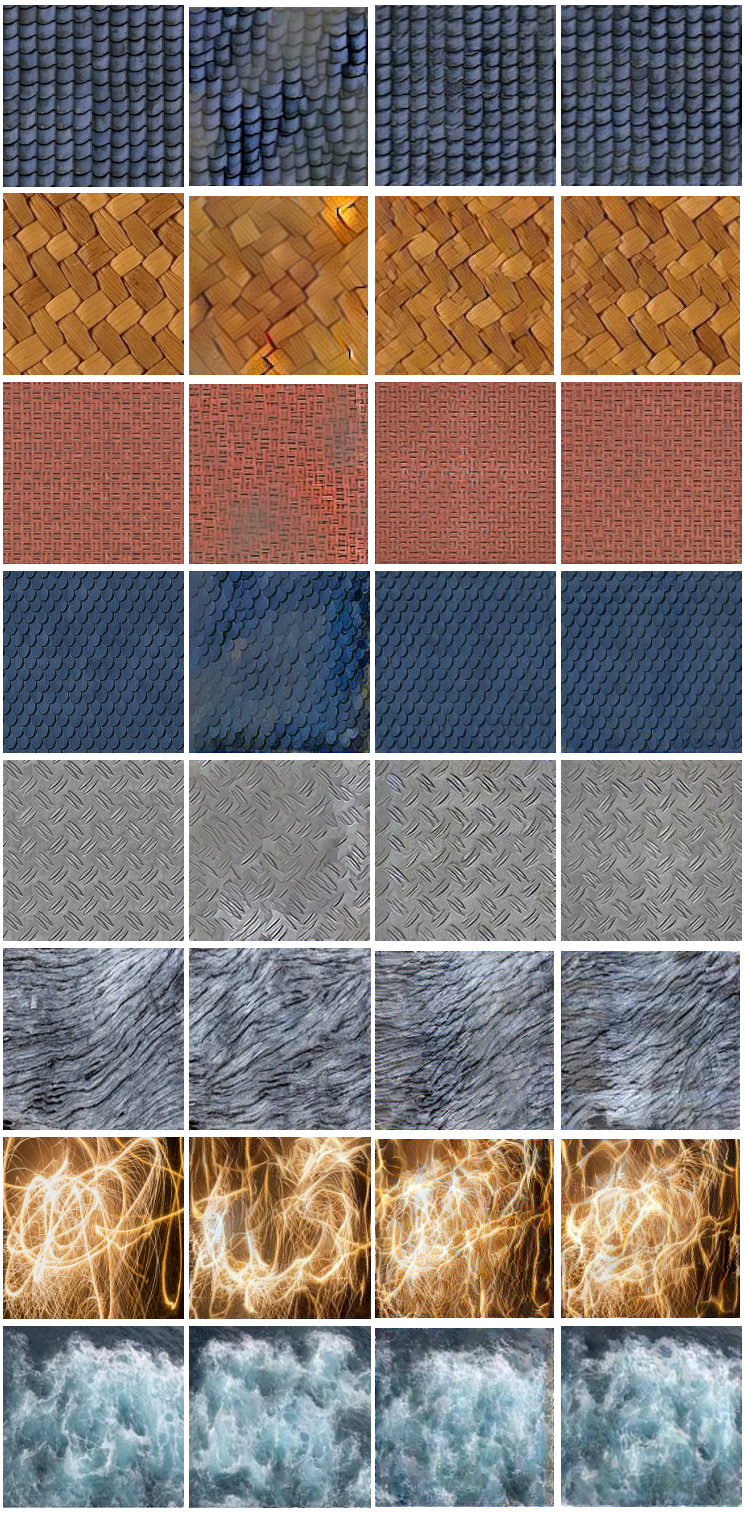
VEDALDI, A. AND LENC, K. 2014. Matconvnet-convolutional neural networks for matlab. *arXiv preprint arXiv:1412.4564*.

VIJFWINKEL, M. 2005. Cg texture database. [Online; accessed 22-August-2016].

WEI, L.-Y., LEFEBVRE, S., KWATRA, V., AND TURK, G. 2009. State of the art in example-based texture synthesis. In *Eurographics 2009, State of the Art Report, EG-STAR*. Eurographics Association, 93–117.

WEI, L.-Y. AND LEVOY, M. 2000. Fast texture synthesis using tree-structured vector quantization. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*. ACM Press/Addison-Wesley Publishing Co., 479–488.

WEXLER, Y., SHECHTMAN, E., AND IRANI, M. 2004. Space-time video completion. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*. Vol. 1. IEEE, I–120.



(a) (b) (c) (d)

Fig. 9: Results of synthesizing textures with different combinations of losses. (a) shows the input texture. The rest of the columns show the results of (b) Gram loss only by setting $\beta = 0$. (c) without the Gram loss by setting $\alpha = 0$. (d) The deep correlation loss together with the Gram loss.

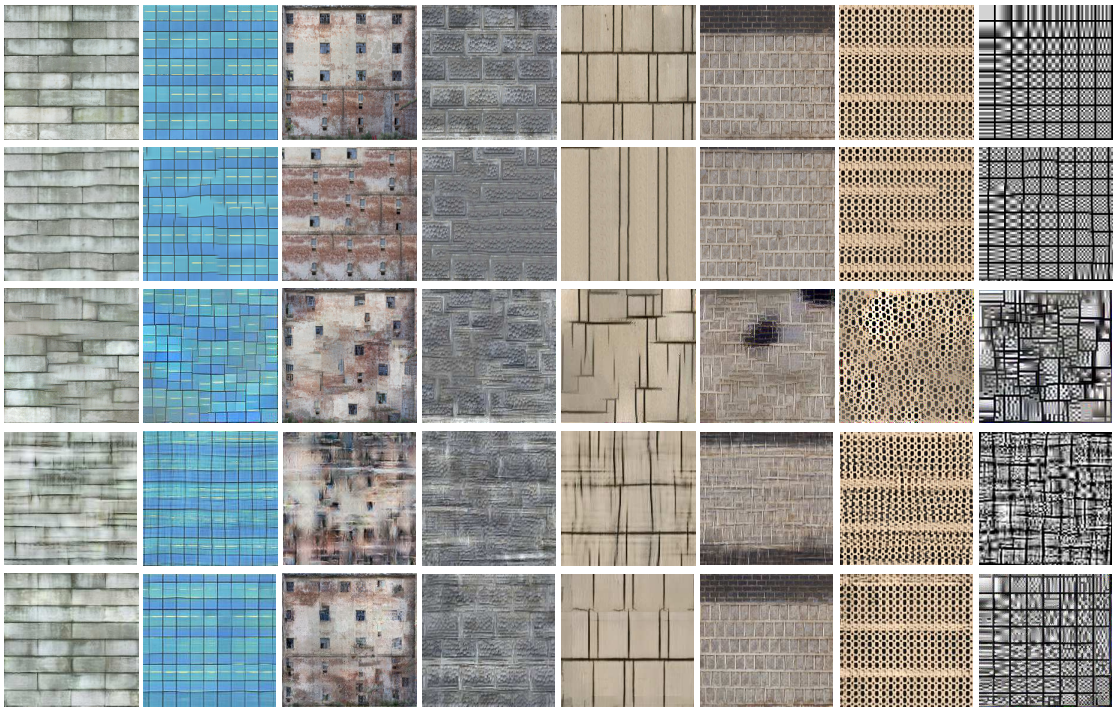


Fig. 10: A comparison of texture synthesis results of various algorithms on the CG-Textures dataset. The rows from top to bottom depict the input images, Kaspar et al's, Gatys et al's, Portilla and Simoncelli's and our deep correlation synthesized results.

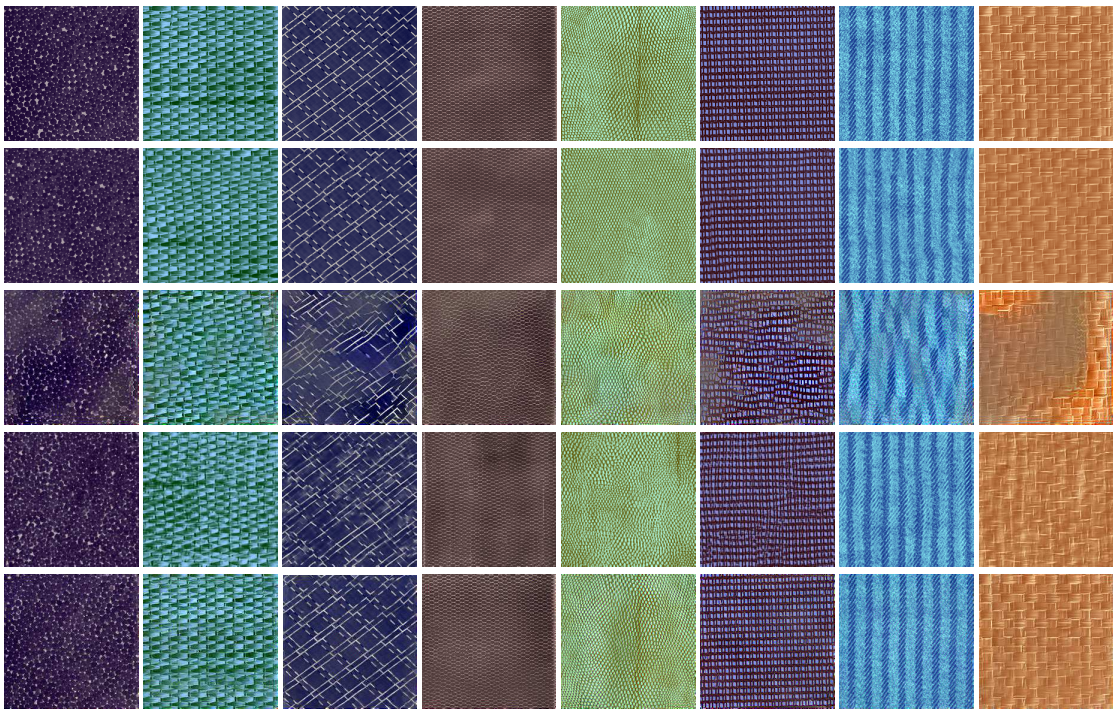


Fig. 11: A comparison of texture synthesis results of various algorithms on the Brodatz dataset. The rows from top to bottom depict the input images, Kaspar et al's, Gatys et al's, Portilla and Simoncelli's and our deep correlation synthesized results.

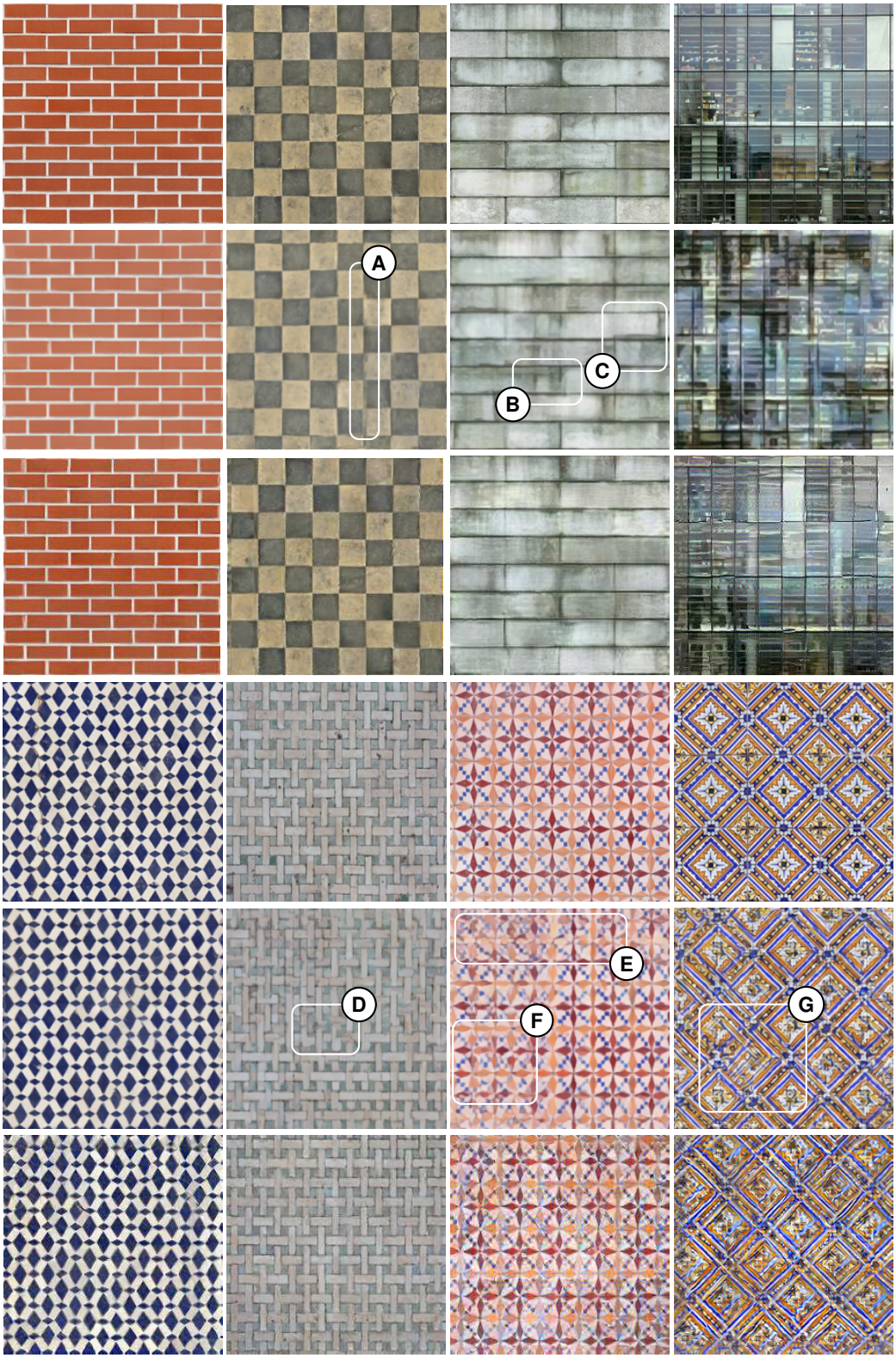


Fig. 12: A comparison of our results to Liu et al. [2016]. The rows from top to bottom depict the input images, Liu et al. and our deep correlation synthesized results.

Fig. 13: A comparison of texture synthesis results of various algorithms. The rows from top to bottom depict the input images, Kaspar et al's results, Gatys et al's results, Portilla and Simoncelli's results and our deep correlation synthesized results

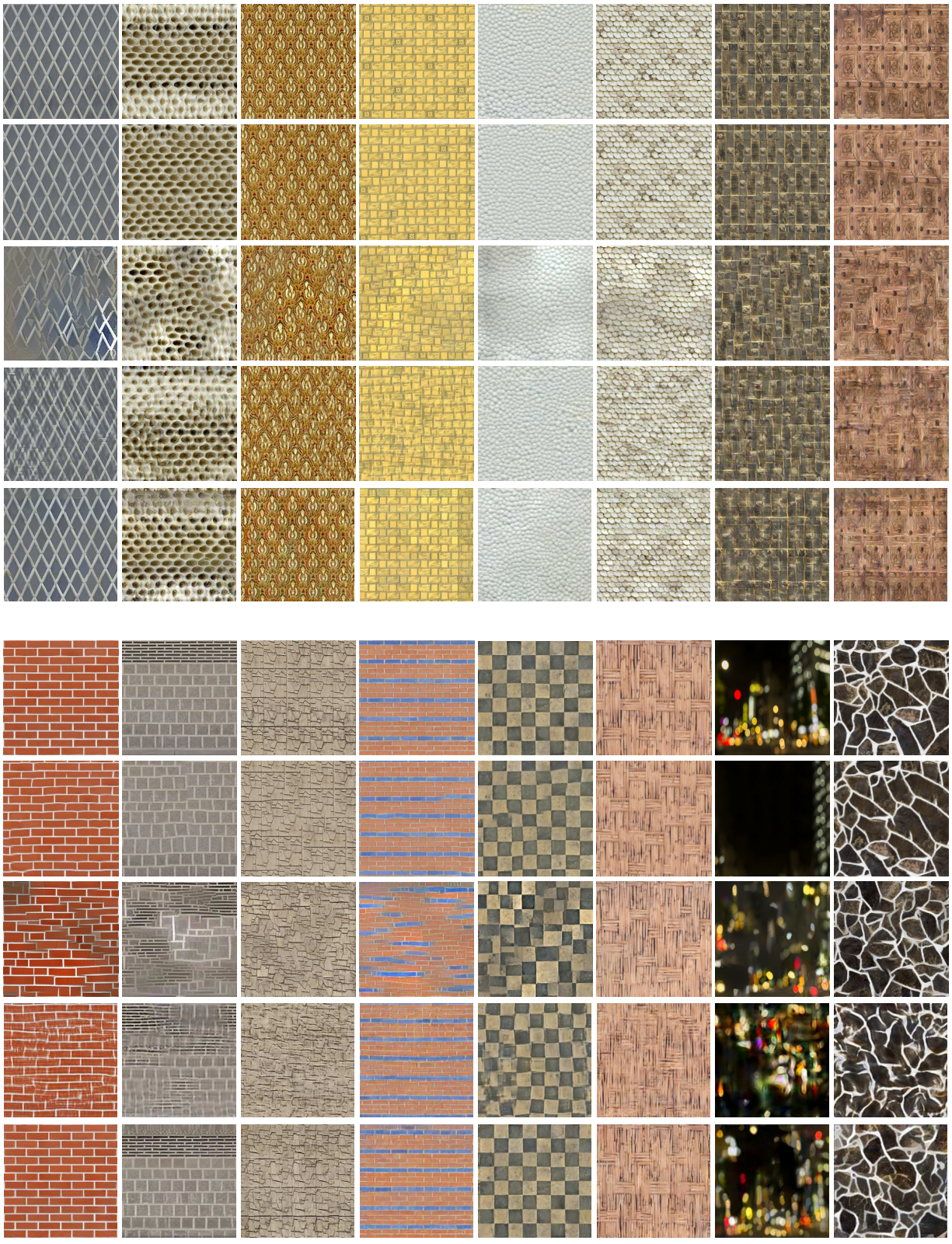


Fig. 14: A comparison of texture synthesis results of various algorithms. The rows from top to bottom depict the input images, Kaspar et al's results, Gatys et al's results, Portilla and Simoncelli's results and our deep correlation synthesized results

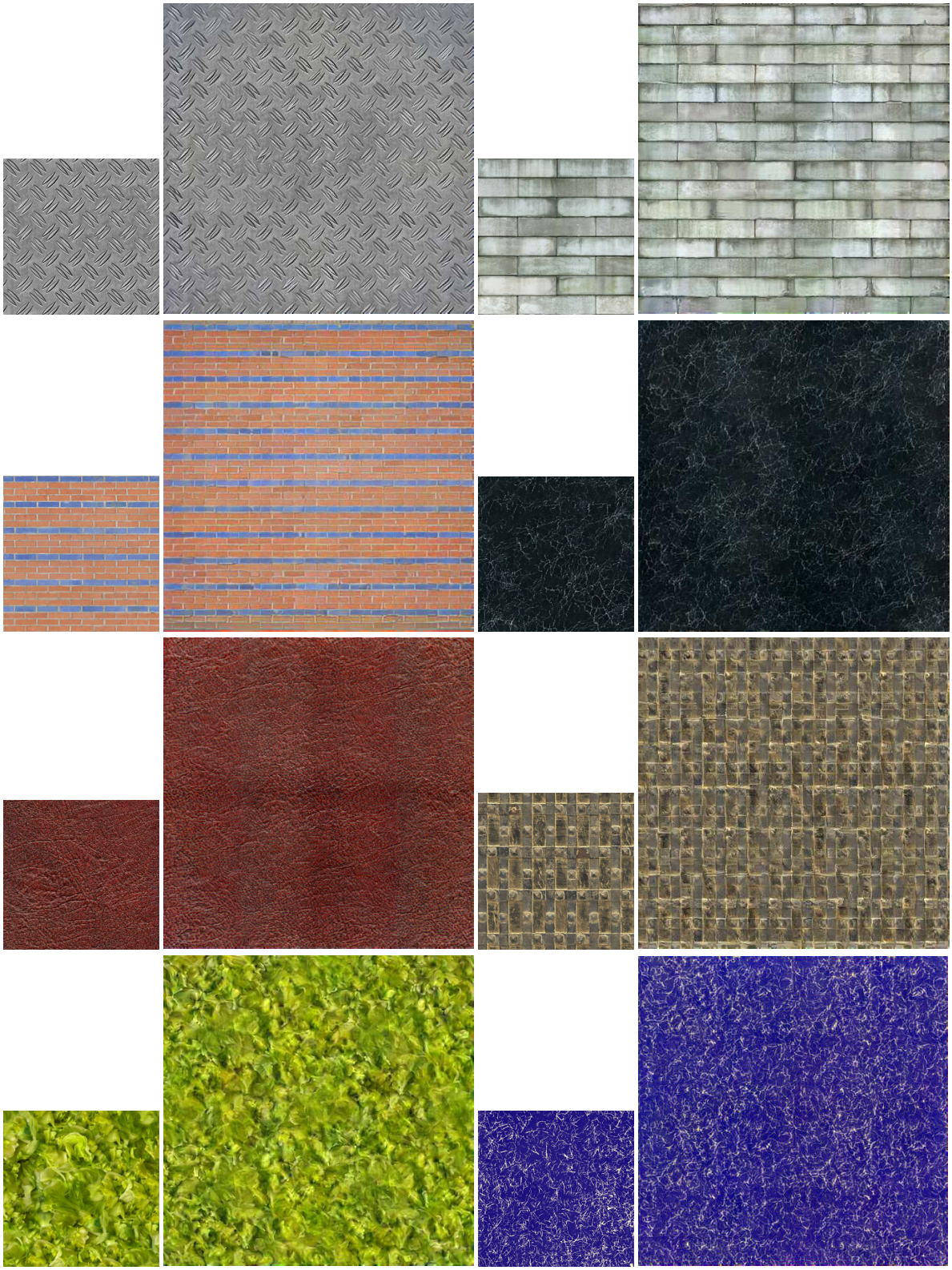


Fig. 15: Results of synthesizing large outputs, applied on images from both the CG-Textures and Brodatz datasets. We show examples of both structured and stochastic textures.

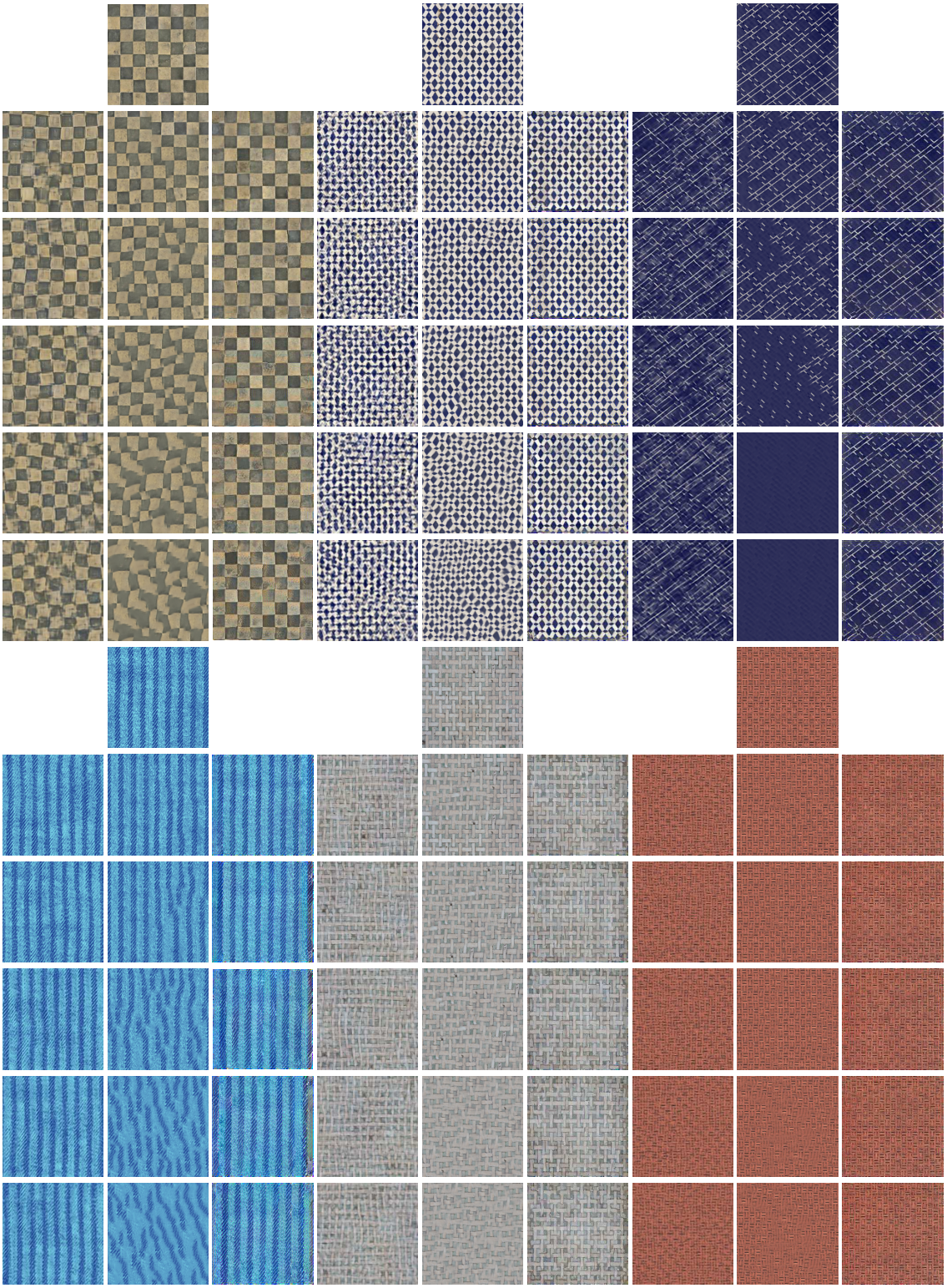


Fig. 16: Stress testing of our algorithm on regular textures: Each iteration uses the synthesized output of the previous iteration as its input exemplar. The upper most row are the input images and beneath it are the results of iterations 1,2,3,7 and 10. Columns left to right are the results of Portilla & Simoncelli, Kaspar et al. and ours, respectively.

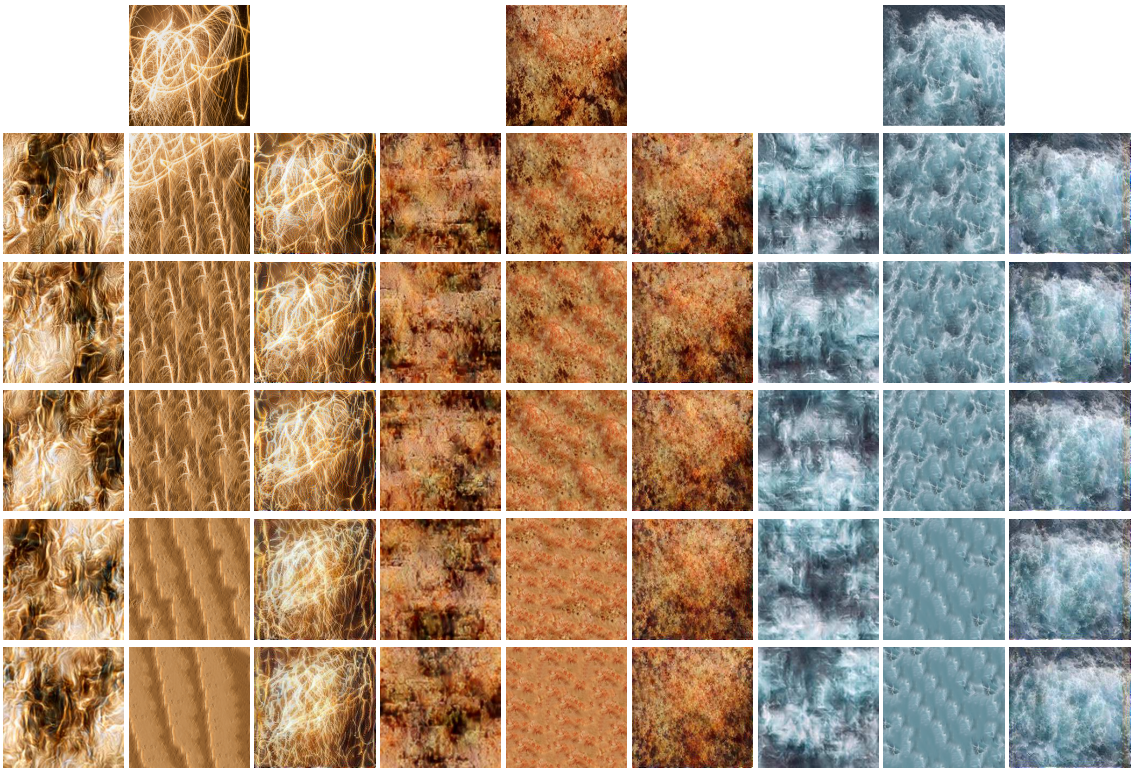


Fig. 17: Stress testing of our algorithm on stochastic textures: Each iteration uses the synthesized output of the previous iteration as its input exemplar. The upper most row are the input images and beneath it are the results of iterations 1,2,3,7 and 10. Columns left to right are the results of Portilla & Simoncelli, Kaspar et al. and ours, respectively.