INCORPORATING LONG-RANGE CONSISTENCY IN CNN-BASED TEXTURE GENERATION

Guillaume Berger & Roland Memisevic
Department of Computer Science and Operations Research
University of Montreal
guillaume.berger@umontreal.ca, memisevr@iro.umontreal.ca

ABSTRACT

Gatys et al. (2015a) showed that pair-wise products of features in a convolutional network are a very effective representation of image textures. We propose a simple modification to that representation which makes it possible to incorporate long-range structure into image generation, and to render images that satisfy various symmetry constraints. We show how this can greatly improve rendering of regular textures and of images that contain other kinds of symmetric structure. We also present applications to inpainting and season transfer.

1 INTRODUCTION

There are currently two dominant approaches to texture synthesis: non-parametric techniques, which synthesize a texture by extracting pixels (or patches) from a reference image that are resampled for rendering (Efros & Leung, 1999; Kwatra et al., 2003), and parametric statistical models, which optimize reconstructions to match certain statistics computed on filter responses (Heeger & Bergen, 1995; Portilla & Simoncelli, 2000). Recently, the second approach has seen a significant advancement, after Gatys et al. (2015a) showed that a CNN pre-trained on an object classification task, such as ImageNet (Russakovsky et al., 2015), can be very effective at generating textures. Gatys et al. (2015a) propose to minimize with respect to the input image a loss function, that measures how well certain high-level features of a reference image are preserved. The reference image constitutes an example of the texture to be generated. The high-level features to be preserved are pair-wise products of feature responses, averaged over the whole image, referred to as the "Gramian" in that work. In Gatys et al. (2015b), the same authors show that by adding a second term to the cost, which matches the content of another image, one can render that other image in the "style" (texture) of the first. Numerous follow-up works have since then analysed and extended this approach (Ulyanov et al., 2016; Johnson et al., 2016; Ustyuzhaninov et al., 2016).

As shown in Figure 1, this method produces impressive results. However, it fails to take into account non-local structure, and consequently cannot generate results that exhibit long-range correlations in images. An example of the importance of long-range structure is the regular brick wall texture in the middle of the figure. Another example is the task of inpainting, where the goal is to fill in a missing part of an image, such that it is faithful to the non-missing pixels. Our main contribution is to introduce a way to deal with long-range structure using a simple modification to the product-based texture features. Our approach is based on imposing a "Markov-structure" on high-level features, allowing us to establish feature constraints that range across sites instead of being local. Unlike classical approaches to preserving spatial structure in image generation, such as Markov Random Fields and learning-based extensions (Roth & Black, 2005), our approach does not impose any explicit local constraints on pixels themselves. Rather, inspired by Gatys et al. (2015a), it encourages consistency to be satisfied on high-level features and on average across the whole image. We present applications to texture generation, inpainting and season transfer.

2 THE ARTISTIC STYLE ALGORITHM

2.1 SYNTHESIS PROCEDURE

Given a reference texture, x, the algorithm described in Gatys et al. (2015a) permits to synthesize by optimization a new texture $\hat{x}$ similar to x. To achieve this, the algorithm exploits an ImageNet

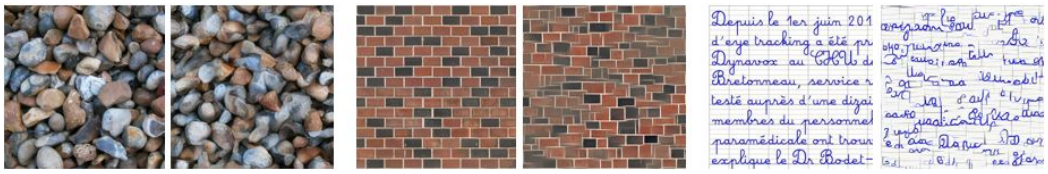


Figure 1: Reference image (*left*) and generated texture (*right*) using the procedure described in Gatys et al. (2015a).

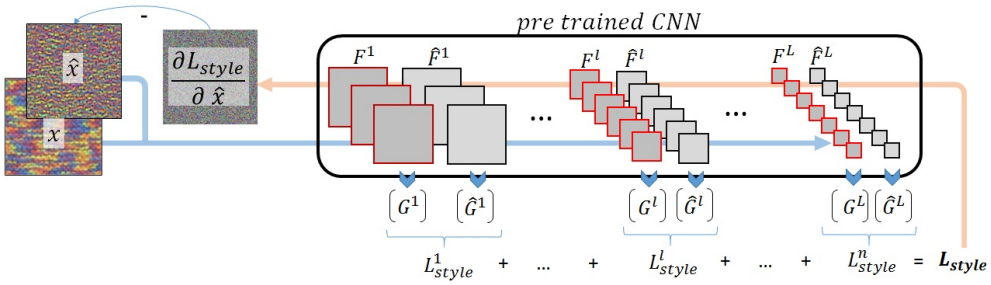


Figure 2: Summary of the texture synthesis procedure described in Gatys et al. (2015a). We use a VGG-19 network Simonyan & Zisserman (2014) as the pre-trained CNN.

pre-trained model to define metrics suitable for describing textures: "Gram" matrices of feature maps, computed on top of L selected layers. Formally, let N^l be the number of maps in layer l of a pre-trained CNN. The corresponding Gram matrix G^l is a $N^l \times N^l$ matrix defined as:

$$G^l_{ij} = \frac{1}{M^l} \sum_{k=1}^{M^l} F^l_{ik} F^l_{jk} = \frac{1}{M^l} \langle F^l_{i:}, F^l_{j:} \rangle \tag{1}$$

where $F^l_{i:}$ is the i^{th} vectorized feature map of layer l, M^l is the number of elements in each map of this layer, and where $\langle \cdot, \cdot \rangle$ denotes the inner product. Equation 1 makes it clear that G^l captures how feature maps from layer l are correlated to each other. Diagonal terms, G^l_{ii} are the squared Frobenius norm of the i^{th} map $\left\| F^l_{i:} \right\|^2_F$, so they represent its spatially averaged energy. We will discuss the Gramians in more detail in the next paragraph. Once the Gram matrices $\{G^l\}_{l \in [1,L]}$ of the reference texture are computed, the synthesis procedure by Gatys et al. (2015a) amounts to constructing an image that produces Gram matrices $\{\hat{G}^l\}_{l \in [1,L]}$ that match the ones of the reference texture. More precisely, the following loss function is minimized with respect to the image being constructed:

$$\mathcal{L}_{style} = \sum_{l=1}^{L} w_l \left\| \hat{G}^l - G^l \right\|^2_F = \sum_{l=1}^{L} w_l \mathcal{L}^l_{style} \tag{2}$$

where w_l is a normalizing constant similar to Gatys et al. (2015a).

The overall process is summarized in Figure 2. While the procedure can be computationally expensive, there have been successful attempts reported recently which reduce the generation time (Ulyanov et al., 2016; Johnson et al., 2016).

2.2 Why Gram matrices work

Feature Gram matrices are effective at representing texture, because they capture global statistics across the image due to spatial averaging. Since textures are static, averaging over positions is required and makes Gram matrices fully blind to the global arrangement of objects inside the reference image. This property permits to generate very diverse textures by just changing the starting point of the optimization. Despite averaging over positions, coherence across multiple features needs to be preserved (locally) to model visually sensible textures. This requirement is taken care of by the off-diagonal terms in the Gram matrix, which capture the co-occurence of different features at

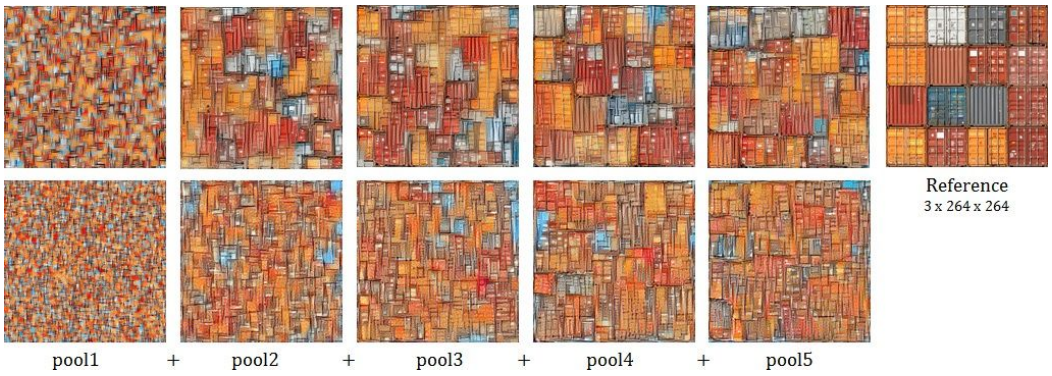


Figure 3: Exploiting Gram matrices of feature maps as in Gatys et al. (2015a) (1^{st} *row*) or only the squared Frobenius norm of feature maps (2^{nd} *row*) for increasingly deep layers (from *left* to *right*).

a single spatial location. Indeed, Figure 3 shows that restricting the texture representation to the squared Frobenius norm of feature maps (i.e. diagonal terms) makes distinct object-parts from the reference texture encroach on each other in the reconstruction, as local coherence is not captured by the model. Exploiting off-diagonal terms improves the quality of the reconstruction as consistency across feature maps is enforced (on average across the image).

The importance of local coherence can be intuitively understood in the case of linear features (or in the lowest layer of a convolutional network): when decomposing an image using Gabor-like features, local structure can be expressed as the relative offsets in the Fourier phase angles between multiple different filter responses. A sharp step-edge, for example, requires the phases of local Fourier components at different frequencies to align in a different way than a blurry edge or a ridge (Morrone & Burr, 1988; Kovesi, 1999). Also, natural images exhibit very specific phase-relationships across frequency components in general, and destroying these makes the image look unnatural (Wang & Simoncelli, 2003). The same is not true of Fourier amplitudes (represented on the diagonals of the Gramian), which play a much less important role in the visual appearance (Oppenheim & Lim, 1981). In the case of deeper representations, the situation is more complex, but it is still local co-occurrence *averaged over the whole image* that captures texture.

Unfortunately, average local coherence falls short of capturing long-range structure in images. Spatial consistency is hard to capture within a single filter bank, because of combinatorial effects. Indeed, since Gram matrices capture coherence at a single spatial location, every feature would have to be matched to multiple transformed versions of itself. A corollary is that every feature would have to appear in the form of multiple transformed copies of itself in order to capture spatial consistency. However, this requirement clashes with the limited number of features available in each CNN layer. One way to address this is to use higher-layer features, whose receptive fields are larger. Unfortunately, as illustrated in Figure 3, even if using layers up to $pool5$ whose input receptive field covers the whole image[1] (first row, last column), the reconstruction remains mainly unstructured and the method fails to produce spatial regularities.

3 Modeling spatial co-occurences

To account for spatial structure in images, we propose encoding this structure in the feature self-similarity matrices themselves. To this end, we suggest that, instead of computing co-occurences between multiple features within a map, we compute co-occurences between feature maps F^l and *spatially transformed* feature maps $T(F^l)$, where T denotes a spatial transformation. In the simplest case, T represents local translation, which amounts to measuring similarities between local features and other neighbouring features. We denote by $T_{x,+\delta}$ the operation consisting in horizontally translating feature maps by δ pixels and define the transformed Gramian:

$$G^l_{x,\delta,ij} = \frac{1}{M^l} \langle T_{x,+\delta}\left(F^l_{i:}\right), T_{x,-\delta}\left(F^l_{j:}\right) \rangle \tag{3}$$

[1]For this experiment, the image size is 264×264, which is also the size of the $pool5$ receptive field.

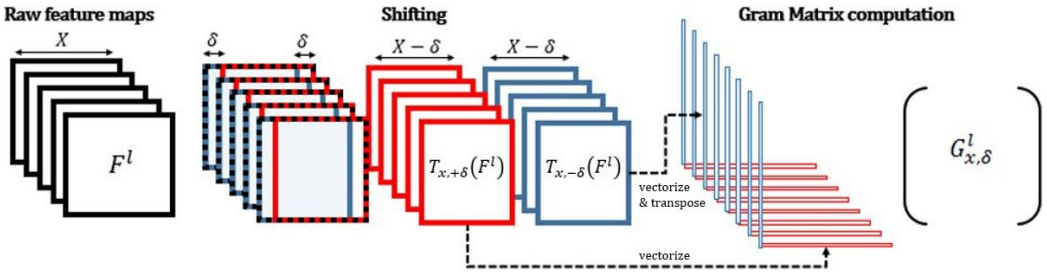


Figure 4: Computing the shifted Gram matrix for a given layer with feature maps of width X.

where $T_{x,-\delta}$ performs a translation in the opposite direction. As illustrated in Figure 4, the transformation in practice simply amounts to removing the δ first or last columns from the raw feature maps. Therefore, the inner product now captures how features at position (i, j) are correlated with features located at position $(i, j + \delta)$ in average. While Figure 4 illustrates the case where feature maps are horizontally shifted, one would typically use translations along both the x-axis and the y-axis. Our transformed Gramians are related to Gray-Level Co-occurrence Matrices (GLCM) (Haralick et al., 1973) which compute the unnormalized frequencies of pixel values for a given offset in an image. While GLCMs have been mainly used for analysis, some work tried to use these features for texture synthesis (Lohmann, 1995). Usually, GLCMs are defined along 4 directions: $0°$ and $90°$ (i.e. horizontal and vertical offsets), as well as $45°$ and $135°$ (i.e. diagonal offsets). In comparison, our method does not consider diagonal offsets and captures spatial coherence on high-level features, making use of a pre-trained CNN, instead of working directly in the pixel domain.

With this definition for transformed Gram matrices, we propose defining the loss as: $\mathcal{L} = \mathcal{L}_{style} + \mathcal{L}_{cc}$, where *cc* stands for *cross-correlation*. Like $\mathcal{L}_{style}$, $\mathcal{L}_{cc}$ is a weighted sum of multiple losses $\mathcal{L}^l_{cc,\delta}$ defined for several selected layers as the mean squared error between transformed Gram matrices of the reference texture and the one being constructed:

$$\mathcal{L}^l_{cc,\delta} = \frac{1}{2}\left(\mathcal{L}^l_{cc,x,\delta} + \mathcal{L}^l_{cc,y,\delta}\right) = \frac{1}{2}\left(\left\|\hat{G}^l_{x,\delta} - G^l_{x,\delta}\right\|^2_F + \left\|\hat{G}^l_{y,\delta} - G^l_{y,\delta}\right\|^2_F\right) \tag{4}$$

Although this amounts to adding more terms to a representation that was already high-dimensional and overparametrized, we found that these additional terms do not hurt the diversity of generated textures. Indeed, the new loss remains blind to the global arrangement of objects. In fact, there exists a specific situation where our approach is strictly equivalent to computing Gram matrices of deeper layers: with linear activations and "one-hot" convolution kernels[2], deeper layers would simply contain translated versions of lower feature maps. In that case, computing Gram matrices of deeper layers would permit to directly capture cross-correlation statistics in lower ones. Nevertheless, this situation is very unlikely when using a pretrained CNN, as the network probably learned operations more useful than simple translations during its supervised training.

While we focus on translation for most of our results, we shall discuss other types of transformation in the experiments Section.

4 Experiments

In our experiments, we exploit the same normalized version of the VGG-19 network[3] (Simonyan & Zisserman, 2014) as in Gatys et al. (2015a;b) and layers *conv1_1*, *pool1*, *pool2*, *pool3*, and *pool4* are always used to define the standard Gram matrices. In our method, we did not use *conv1_1* to define cross-correlation terms, as the large number of neurons at this stage makes the computation of Gram matrices costly. Corresponding δ values for each layer are discussed in the next paragraph.

[2]To match our translated Gramians, the only non-zero component would be δ or $-\delta$ shifted from the center.
[3]available at http://bethgelab.org/media/uploads/deeptextures/vgg_normalised.caffemodel.

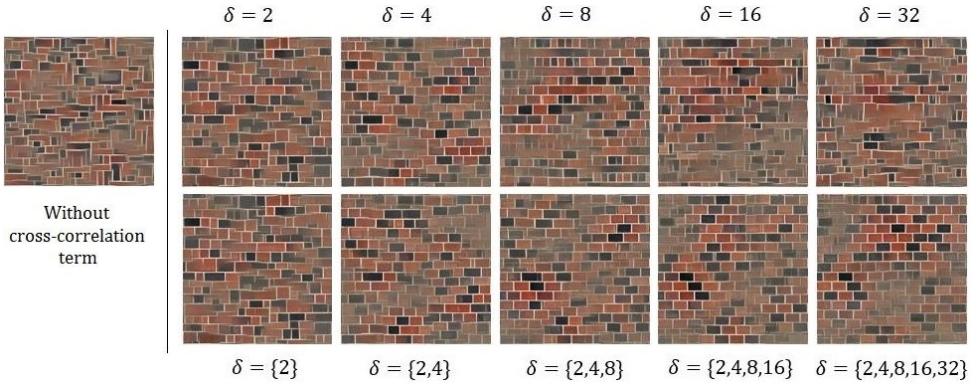


Figure 5: $Pool2$ reconstruction using different values for δ in $\mathcal{L}^{pool2}_{cc,\delta}$. ($1^{st}$ *column*): Without cross-correlation terms. (*Other columns*): Using a single cross-correlation term with a fixed δ value (1^{st} *row*) or using multiple cross-correlation terms with distinct δ values (2^{nd} *row*).

Finally, our implementation[4] uses Lasagne (Dieleman et al., 2015). Each image is of size 384×384. Most textures used as references in this paper were taken from *textures.com* and *pixabay.com*.

4.1 Experiments with translation-Gramians

The δ parameter is of central importance as it dictates the range of the spatial constraints. We observed that the optimal value depends on both the considered layer in the pre-trained CNN and the reference texture, making it difficult to choose a value automatically.

For instance, Figure 5 shows generated images from the brick wall texture using only the *pool2* layer with different δ configurations. The first row depicts the results when considering single values of δ only. While $\delta = 4$ or $\delta = 8$ are good choices, considering extreme long-range correlations does not help for this particular texture: a brick depends mostly on its neighbouring bricks and not the far-away ones. More precisely, a translation of more than 16 pixels in the *pool2* layer makes the input receptive field move more than 64 pixels. Therefore $\delta = 16$ or $\delta = 32$ do not capture any information about neighbouring bricks. Unfortunately, this is not true for all textures, and $\delta = 16$ or $\delta = 32$ might be good choices for another image that exhibits longer structures.

Searching systematically for a δ configuration that works well with the reference texture being considered would be a tedious task: even for a very regular texture with a periodic horizontal (or vertical) pattern, it is hard to guess the optimal δ values for each layer (for deeper ones in particular). Instead, we propose to use a fixed but wide set of δ values per layer, by defining the cost to be: $\mathcal{L}^l_{cc} = \sum_k \mathcal{L}^l_{cc,\delta_k}$. A potential concern is that combining many loss terms can hurt the reconstruction or the diversity of generated textures. Figure 5 (second row) shows contrarily that there is no visual effect from using δ values that are *not* specifically useful for the reference texture being considered: the rendering benefits from using $\delta \in \{2, 4, 8\}$ while considering bigger values ($\delta \in \{2, 4, 8, 16, 32\}$, e.g.) does not help, but does not hurt the reconstruction either. We found the same to be true for other textures as well, and our results (shown in the next Section) show that combining the loss terms is able to generate very diverse textures. A drawback from using multiple cross-correlation terms per layers, however, is computational. We found that in our experimental setups, adding cross-correlation terms increases the generation time by roughly 80%.

As a guide-line, for image sizes of roughly 384×384 pixels, we recommend the following δ values per layer (which we used in all our following experiments): $\{2, 4, 8, 16, 32, 64\}$ for *pool1*, $\{2, 4, 8, 16, 32\}$ for *pool2*, $\{2, 4, 8, 16\}$ for *pool3*, and $\{2, 4, 8\}$ for *pool4*. The number and the range of δ values decrease with depth because feature maps are getting smaller due to 2×2 pooling layers. This configuration should be sufficient to account for spatial structure in any 384×384 image.

[4] available at https://github.com/guillaumebrg/texture_generation

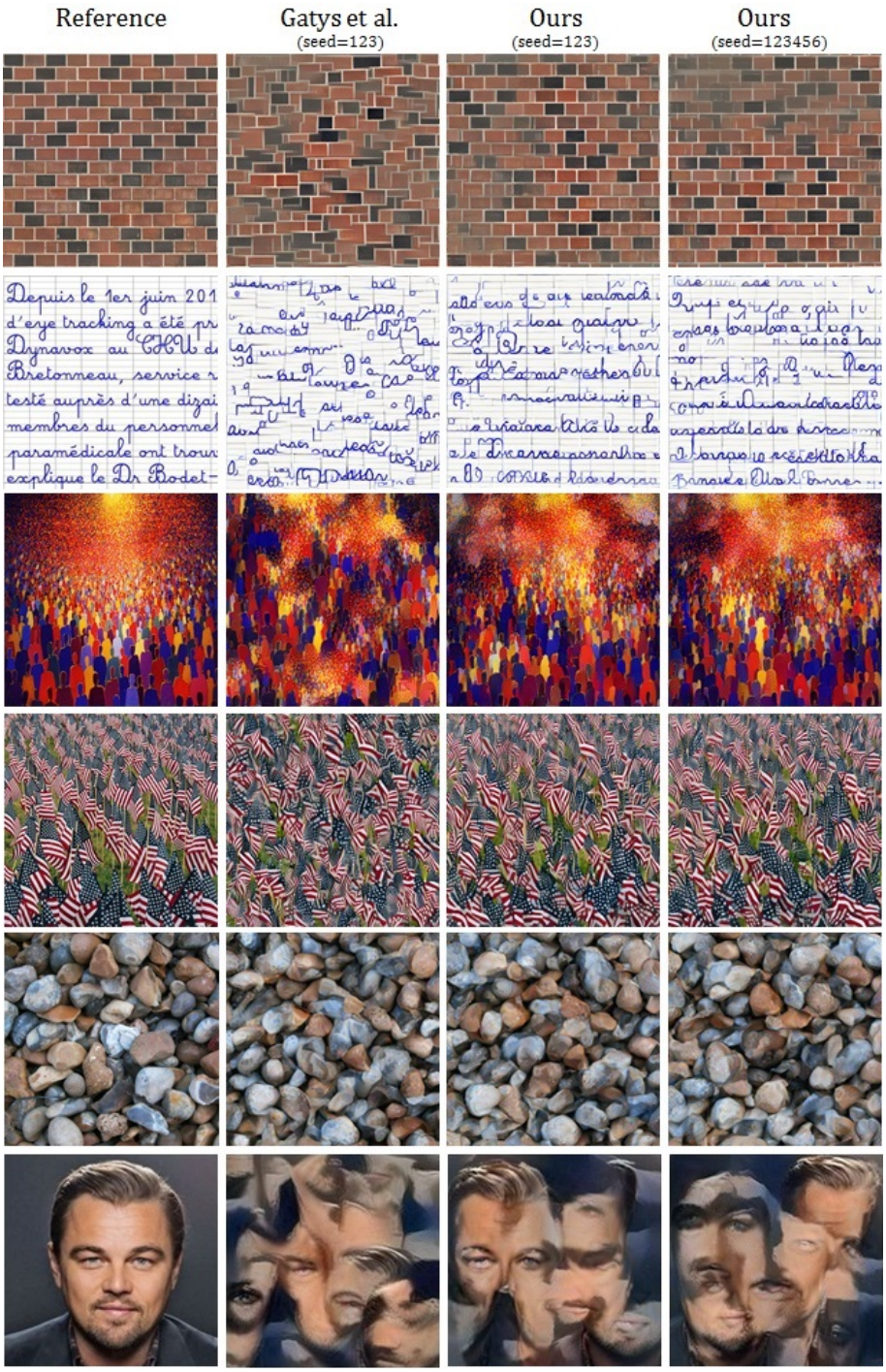


Figure 6: Some results of our approach compared with Gatys et al. (2015a). Only the initialization differs in the last two columns. Further results are shown in the supplementary material.

4.2 Synthesis of structured texture examples

Figure 6 shows the result of our approach applied to various structured and unstructured textures. It demonstrates that the method is effective at capturing long-range correlations without simply copying the content of the original texture. For instance, note how our model captures the depth aspect of the reference image in the third and fourth rows.

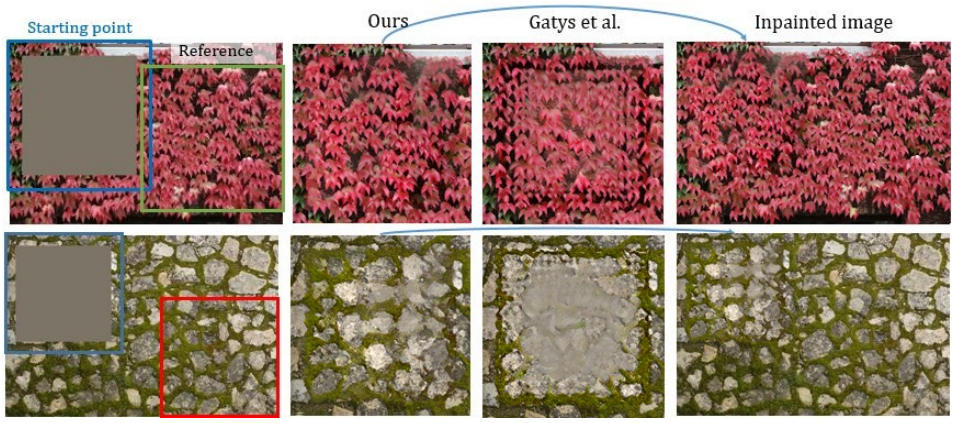


Figure 7: Texture generation applied to in-painting. More in-painted images can be found in the supplementary material.

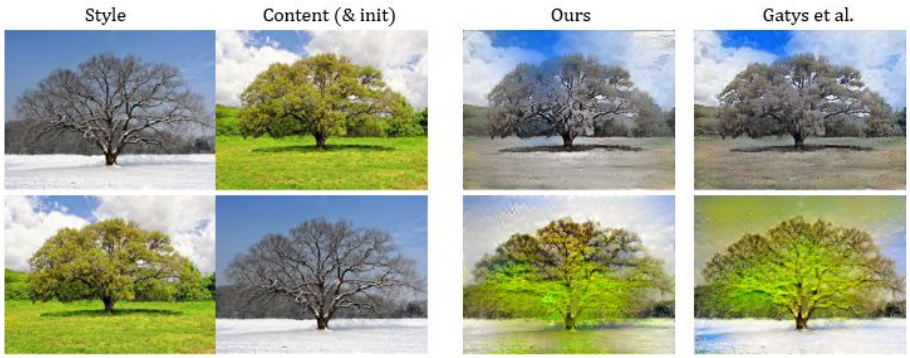


Figure 8: Season transfer examples.

The problem of synthesizing near-regular structures is challenging because stochasticity and regularity are adversarial properties (Lin et al., 2006). Non-parametric patch-based techniques, such as Efros & Freeman (2001), are better suited for this task because they can *tile*[5] the reference image. On the other hand, regular structures are usually more problematic for parametric statistical models. Nevertheless, the two first rows of Figure 6 demonstrate that our approach can produce good visual results and can reduce the gap to patch-based methods on these kinds of texture.

Even if the reference image is not a texture, the generated images in the last row (Leonardo Dicaprio's face) provide a good visual illustration of the effect of translation terms. In contrast to Gatys et al., our approach preserves longer-range structure, such as the alignment and similar appearance of the eyes, hair on top of the forehead, the chin below the mouth, etc. Finally, when the reference texture is unstructured (fifth row), our solution does not necessarily provide a benefit, but it also does not hurt the visual quality or the diversity of the generated textures.

4.3 Inpainting application

Modelling long-range correlations can make it possible to apply texture generation to inpainting, because it allows us to impose consistency constraints between the newly rendered region and the unmodified parts of the original image.

To apply our approach to texture inpainting, we extracted two patches from the original image: one that covers the whole area to inpaint, and another one that serves as the reference texture. Then, approximately the same process as for texture generation is used, with the following two

[5]Copy and paste patches side by side.

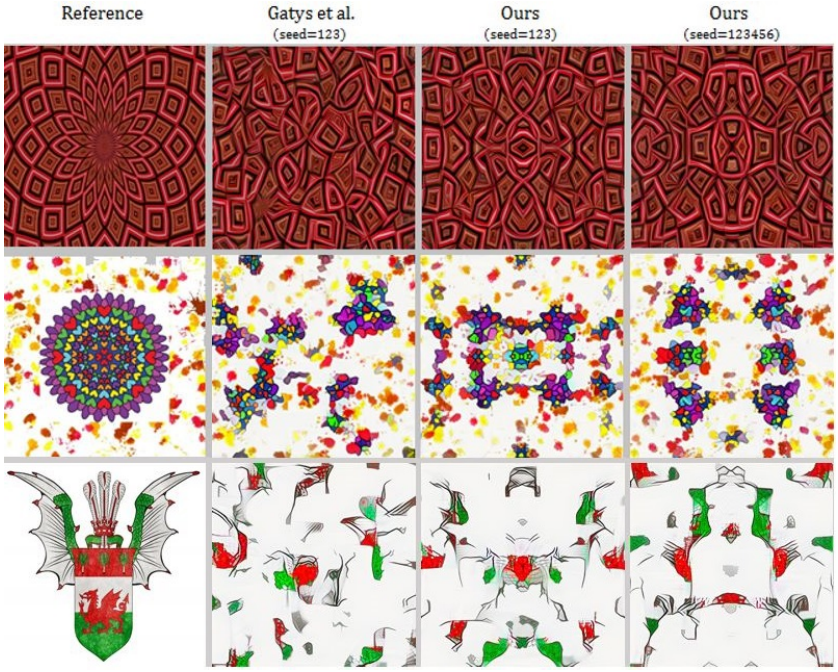


Figure 9: Generation of abstract symmetric textures.

modifications: First, instead of random noise, the optimization starts from the masked content patch (the one to inpaint) showing a grey area and its non-missing surrounding. Second, we encourage the borders of the output to not change much with respect to the original image using an L_2 penalty. We apply the penalty both in the Gatys et al. rendering and in ours. Some inpainted images are shown in Figure 7. As seen in the figure, our solution significantly outperforms Gatys et al. in terms of visual quality. Further results are shown in the supplementary material.

4.4 Season transfer

Figure 8 shows the result of applying our approach to a style transfer task, as in Gatys et al. (2015b): transferring the "season" of a landscape image to another one. On this task, the results from our approach are similar to those from Gatys et al. (2015b). Nevertheless, in contrast to the Gatys et al. results, our approach seems to better capture global information, such as sky color and leaves (bottom row), or the appearance of branches in the winter image (top row).

4.5 Incorporating other types of structure

While we focused on feature map translations in most of our experiments, other transformations can be applied as well. To illustrate this point, we explored a way to generate symmetric textures using another simple transformation. To this end, we propose flipping one of the two feature maps before computing the Gram matrices: $G^l_{lr,ij} = \langle F^l_{i,:}, T_{lr}\left(F^l_{j,:}\right)\rangle$. Here T_{lr} corresponds to the left-right flipping operation, but we also considered up-down flipping of feature maps: $\mathcal{L}^l = \mathcal{L}^l_{style} + \mathcal{L}^l_{lr} + \mathcal{L}^l_{ud}$.

As can be seen in Figure 9, in contrast to Gatys et al., the additional loss terms capture which objects are symmetric in the reference texture, and enforce these same objects to be symmetric in the reconstruction as well. Other kinds of transformation could be used, depending on the type of property in the source texture one desires to preserve.

5 CONCLUSION

We presented an approach to satisfying long-range consistency constraints in the generation of images. It is based on a variation of the method by Gatys et al., and considers spatial co-occurences of local features (instead of only co-occurences across features). We showed that the approach permits to generate textures with various global symmetry properties and that it makes it possible to apply texture generation to in-painting. Since it preserves correlations across sites, the approach is reminiscent of an MRF, but in contrast to an MRF or other graphical models, it defines correlation-constraints on high-level features of a (pre-trained) CNN rather than on pixels.

REFERENCES

Sander Dieleman, Jan Schlter, Colin Raffel, et al. Lasagne: First release., August 2015. URL http://dx.doi.org/10.5281/zenodo.27878.

A. A. Efros and T. K. Leung. Texture synthesis by non-parametric sampling. In *Proceedings of the International Conference on Computer Vision-Volume 2*, ICCV '99, pp. 1033–, 1999.

Alexei A. Efros and William T. Freeman. Image Quilting for Texture Synthesis and Transfer. *Proceedings of SIGGRAPH 2001*, pp. 341–346, August 2001.

L. A. Gatys, A. S. Ecker, and M. Bethge. Texture synthesis using convolutional neural networks. In *Advances in Neural Information Processing Systems 28*, 2015a. URL http://arxiv.org/abs/1505.07376.

L. A. Gatys, A. S. Ecker, and M. Bethge. A neural algorithm of artistic style. *CoRR*, abs/1508.06576, 2015b. URL http://arxiv.org/abs/1508.06576.

R. Haralick, K. Shanmugam, and I. Dinstein. Texture features for image classification. *IEEE Transactions on Systems, Man, and Cybernetics 3 (6)*, 1973.

David J. Heeger and James R. Bergen. Pyramid-based texture analysis/synthesis. In *Proceedings of the 22nd Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1995*, pp. 229–238, 1995.

J. Johnson, A. Alahi, and L. Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. *CoRR*, abs/1603.08155, 2016. URL http://arxiv.org/abs/1603.08155.

P. Kovesi. Image features from phase congruency. *Videre*, 1(3):2–27, 1999.

V. Kwatra, A. Schdl, I. Essa, G. Turk, and A. Bobick. Graphcut textures: Image and video synthesis using graph cuts. *ACM Transactions on Graphics, SIGGRAPH 2003*, 22(3):277–286, 2003.

Wen-Chieh Lin, James Hays, Chenyu Wu, V. Kwatra, and Yanxi Liu. Quantitative evaluation on near regular texture synthesis. In *CVPR '06*, volume 1, pp. 427 – 434, June 2006.

G. Lohmann. Analysis and synthesis of textures: A co-occurrence-based approach. *Computers and Graphics, 19(1)*, pp. 29–36, 1995.

M. C. Morrone and D. C. Burr. Feature detection in human vision: A phase-dependent energy model. *Proceedings of the Royal Society of London B: Biological Sciences*, 235(1280):221–245, 1988.

A. V. Oppenheim and J. S. Lim. The importance of phase in signals. *Proceedings of the IEEE*, 69(5):529–541, 1981. ISSN 0018-9219.

J. Portilla and E. P. Simoncelli. A parametric texture model based on joint statistics of complex wavelet coefficients. *Int. J. Comput. Vision*, 40(1):49–70, 2000.

S. Roth and M. J. Black. Fields of experts: A framework for learning image priors. In *IEEE Conf. on Computer Vision and Pattern Recognition*, volume 2, pp. 860–867, June 2005.

O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

D. Ulyanov, V. Lebedev, A. Vedaldi, and V. Lempitsky. Texture networks: Feed-forward synthesis of textures and stylized images. In *International Conference on Machine Learning (ICML)*, 2016.

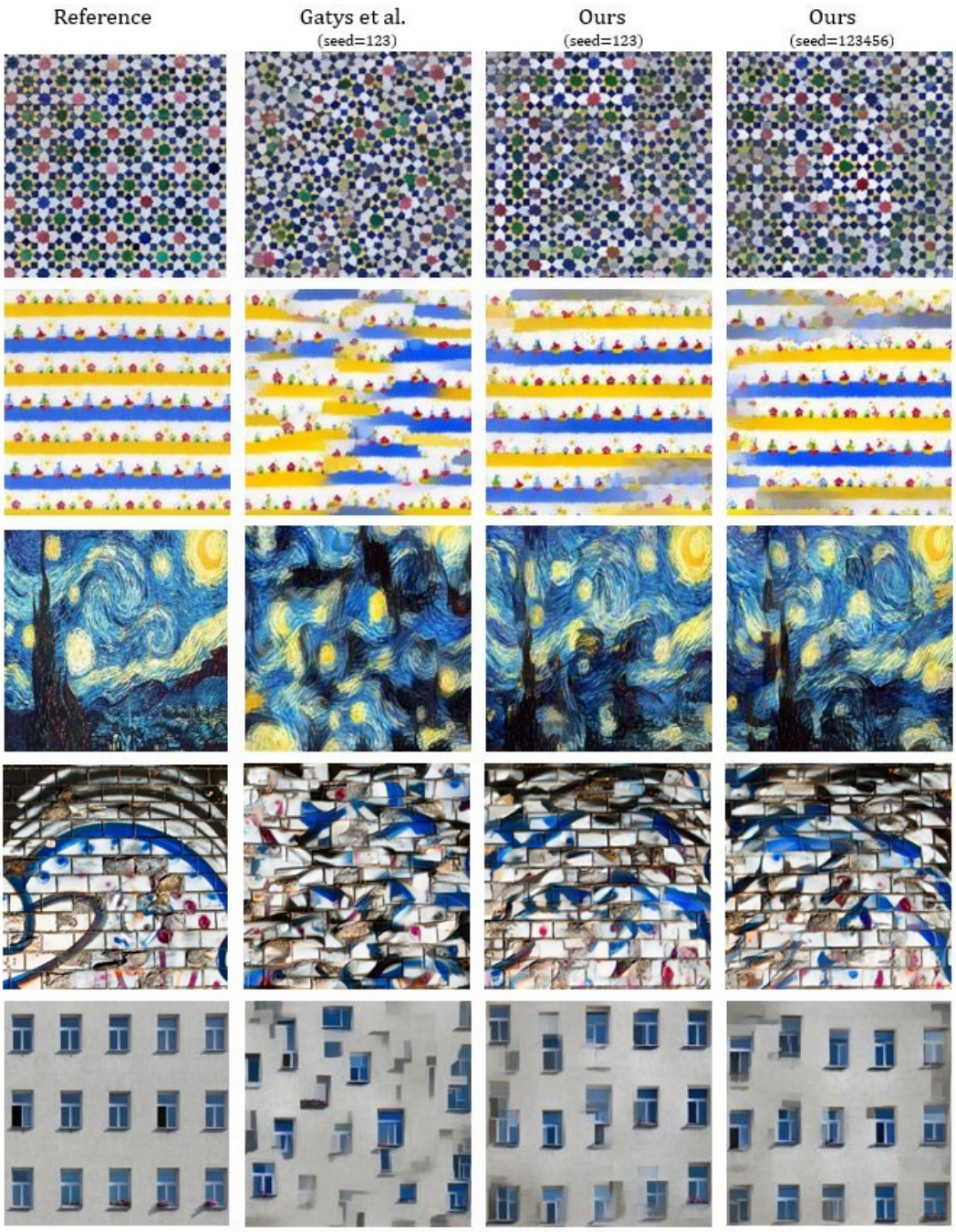
I. Ustyuzhaninov, W. Brendel, L. A. Gatys, and M. Bethge. Texture synthesis using shallow convolutional networks with random filters. *CoRR*, abs/1606.00021, 2016. URL http://arxiv.org/abs/1606.00021.

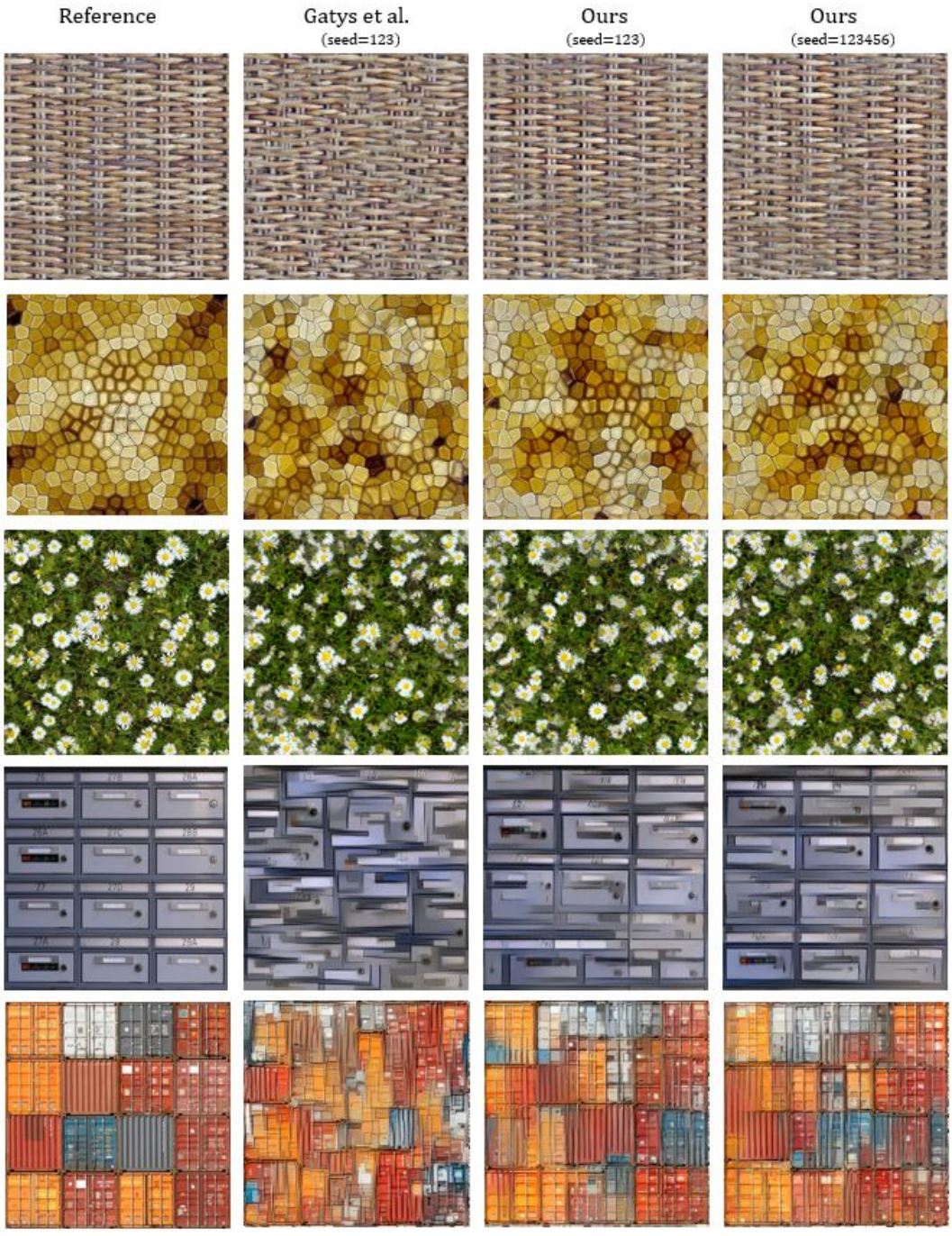
Z. Wang and E. P. Simoncelli. Local phase coherence and the perception of blur. In *Advances in Neural Information Processing Systems 16 [NIPS 2003]*, pp. 1435–1442, 2003.

Ciyou Zhu, Richard H. Byrd, Peihuang Lu, and Jorge Nocedal. L-bfgs-b - fortran subroutines for large-scale bound constrained optimization. Technical report, ACM Trans. Math. Software, 1994.

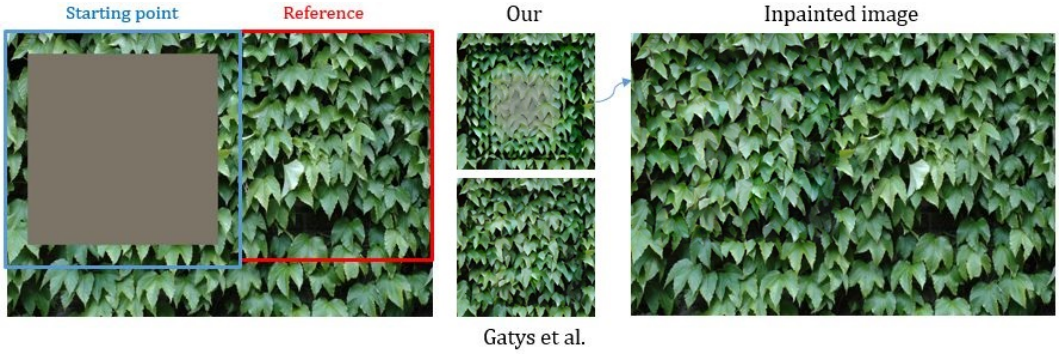
Supplementary material

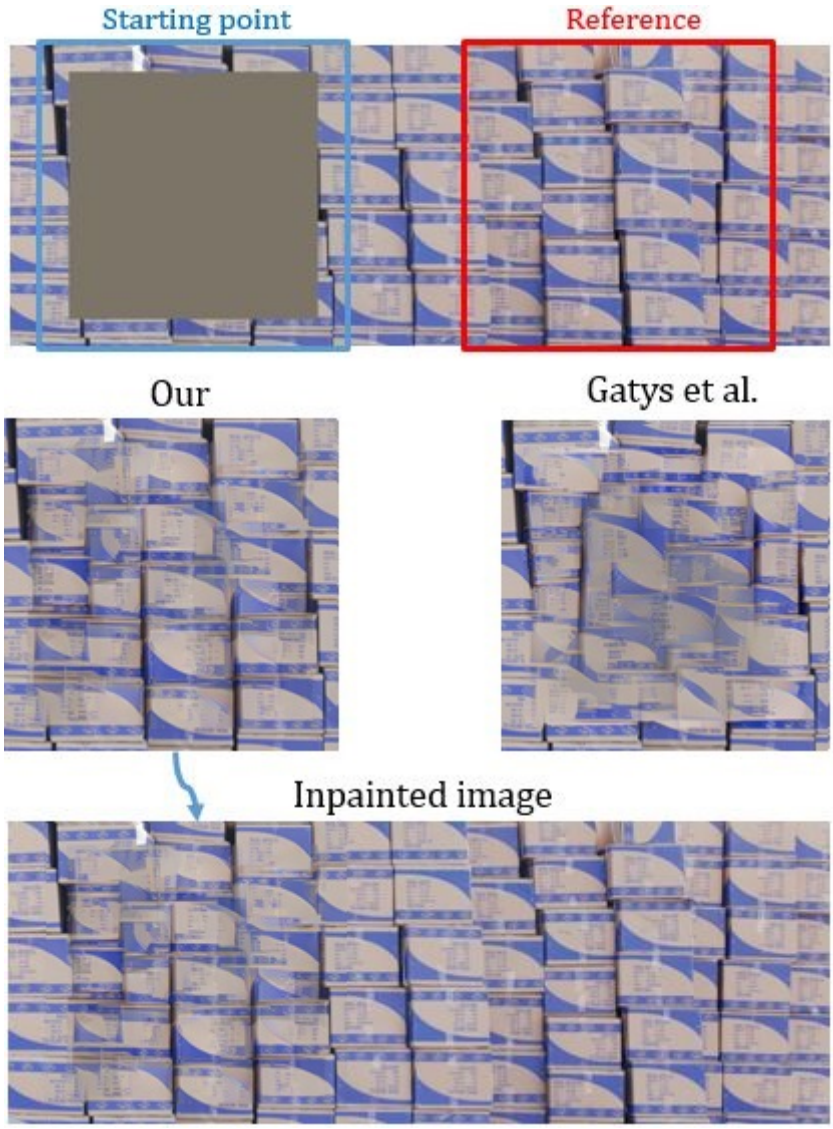
A TEXTURE GENERATION

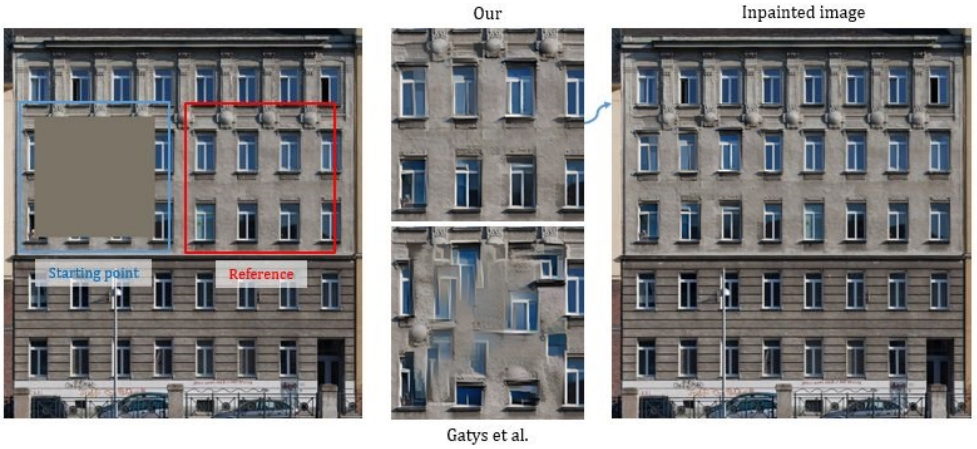



Reference
Gatys et al.
(seed=123)
Ours
(seed=123)
Ours
(seed=123456)

B INPAINTING

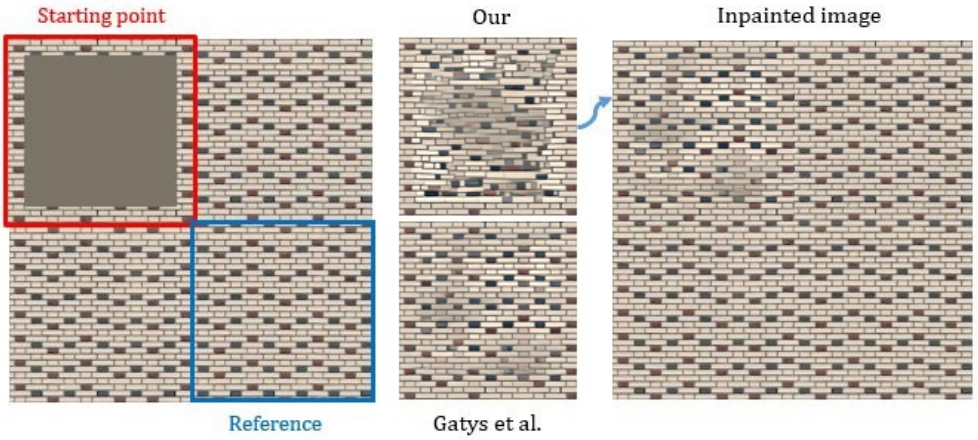





Our
Inpainted image
Starting point
Reference
Gatys et al.


Our
Inpainted image
Starting point
Reference
Gatys et al.


Starting point
Our
Inpainted image
Reference
Gatys et al.

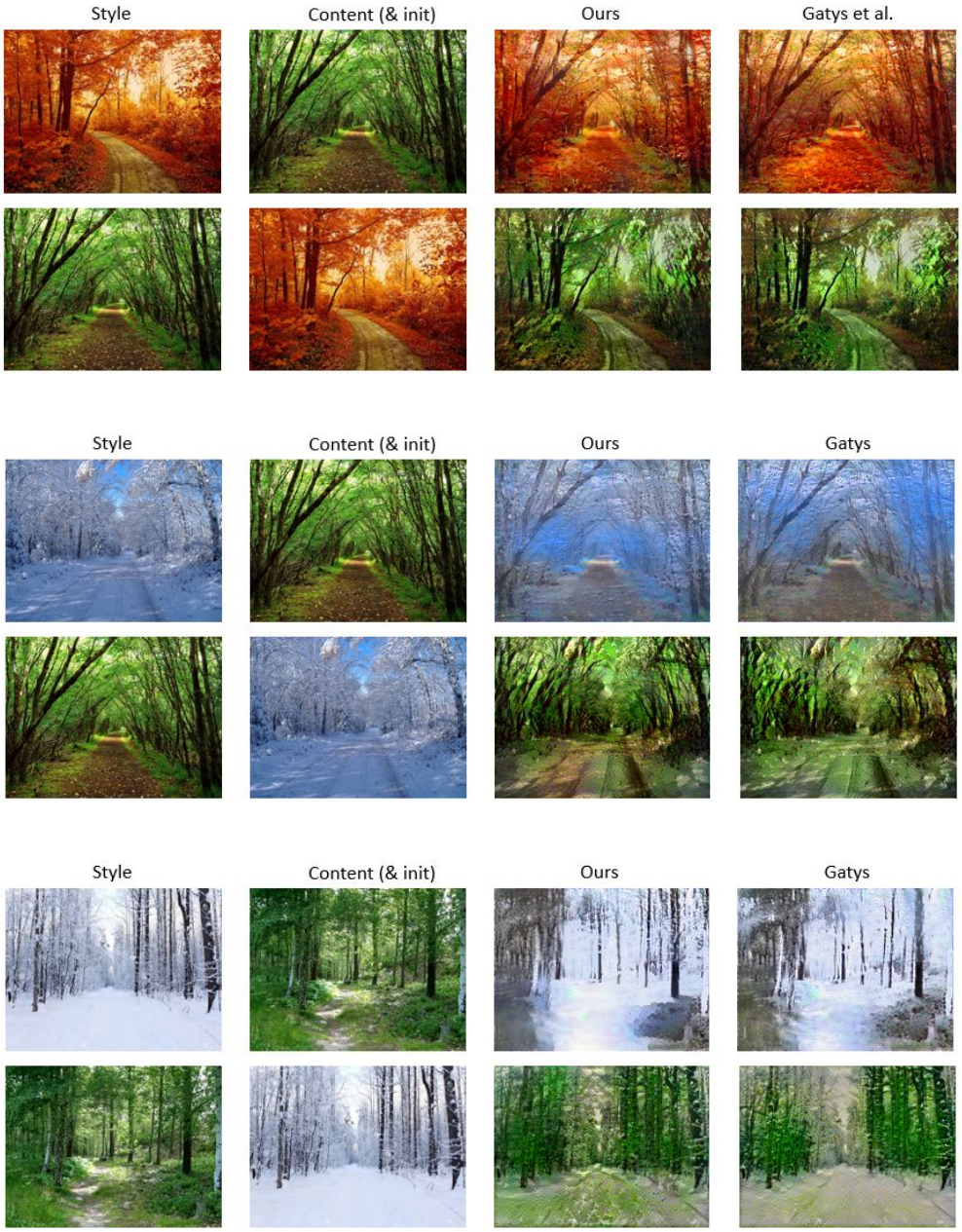


	Gatys et al. loss	
Texture	Gatys et al.	Ours
"red_leaves"	**3.38e-4**	4.72e-4
"floor"	**3.68e-4**	4.73e-4
Supplementary material textures		
"leaves"	**4.08e-4**	5.14e-4
"cargo"	**3.39e-4**	4.73e-4
"building"	8.90e-4	**6.67e-4**
"building2"	**7.19e-4**	7.78e-4
"cargo2"	**1.05e-3**	1.18e-3
"small_bricks"	1.21e-3	**6.82e-4**
"football_team"	**4.56e-3**	5.68e-3

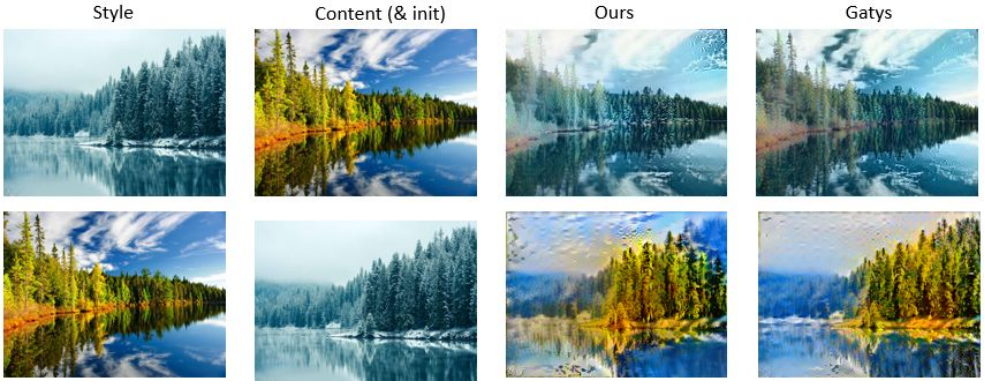
Table 1: Final Gatys et al. loss for all inpainted textures (by order of appearance).

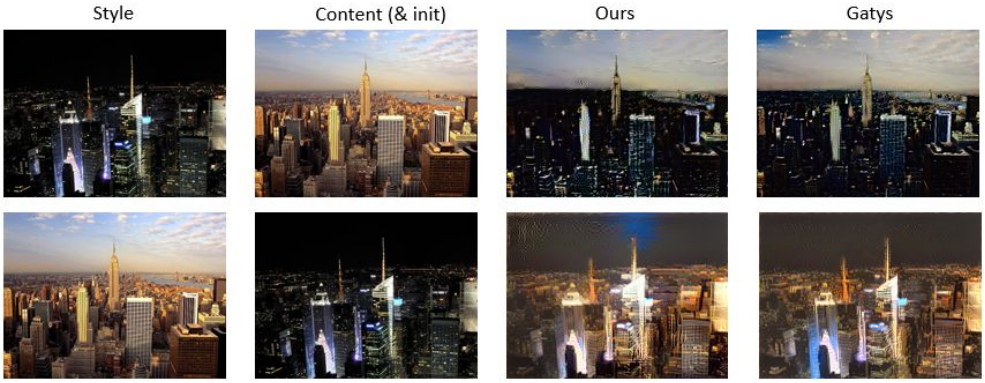
Table 1 reports the final Gatys et al. losses obtained by both approaches for all inpainted textures showed previously. In most cases, the Gatys et al. (2015a) approach converges to a smaller value, which is not surprising since our method optimizes a modified version of the loss. Table 1, combined with the visual aspect of Gatys et al. inpainted renderings, illustrates that only imposing local coherence is not enough to obtain good inpainted textures. Our texture representation seems to be better suited for this task.


Starting point
Reference
Our
Gatys et al.
Inpainted image


Style
Content (& init)
Ours
Gatys et al.
Style
Content (& init)
Ours
Gatys
Style
Content (& init)
Ours
Gatys


Style
Content (& init)
Ours
Gatys


Style
Content (& init)
Ours
Gatys


Style
Content (& init)
Ours
Gatys

C SYMMETRIC TEXTURES