LONG RANGE CONSTRAINTS FOR NEURAL TEXTURE SYNTHESIS USING SLICED WASSERSTEIN LOSS

Liping Yin[*,**], *Albert Chua*[**]

CMSE Department, Michigan State University[*],
Department of Mathematics, Michigan State University[**]

ABSTRACT

In the past decade, exemplar-based texture synthesis algorithms have seen strong gains in performance by matching statistics of deep convolutional neural networks. However, these algorithms require regularization terms or user-added spatial tags to capture long range constraints in images. Having access to a user-added spatial tag for all situations is not always feasible, and regularization terms can be difficult to tune. Thus, we propose a new set of statistics for texture synthesis based on Sliced Wasserstein Loss, create a multi-scale method to synthesize textures without a user-added spatial tag, study the ability of our proposed method to capture long range constraints, and compare our results to other optimization-based, single texture synthesis algorithms.

Index Terms— Neural Texture Synthesis, Sliced Wasserstein Distance, Convolutional Neural Networks

1. INTRODUCTION

1.1. Background on exemplar-based texture synthesis

Texture synthesis has been of interest to researchers for over half a century starting with [1]. The goal is to take a reference texture and use statistical information about the reference texture to generate a new, different texture with similar properties as the reference texture. While this problem is inherently interesting as a computer vision task, the literature from this subfield of research also provides us with a better understanding of how humans perceive texture.

Until recently, many methods, like [3, 4], have used statistical information of wavelet coefficients for texture generation. However, these methods failed to produce believable synthesis for images with complex structures, which suggested more statistical information was needed to capture the essence of a texture. Neural texture synthesis, which involves the use of deep neural networks to generate textures, started with the work of Gatys et. al. in [7, 8]. The authors used the mean squared error between gram matrices of VGG19 feature maps as a loss function. While the results in [8] were state-of-the-art compared to previous work at the time of publication, the proposed method has trouble capturing long range constraints.

Numerous papers have imposing long range constraints to synthesized images for single texture synthesis [9, 10, 11, 12, 13], which generates one texture per synthesis process. However, these methods generally require tunable regularization parameters, do not fully capture long range constraints, or require multiple hours to synthesize one image. Other approaches include universal texture synthesis algorithms [17, 18, 19], which train on a database of textures to reduce synthesis time after training; however, the focus of this paper will be on single texture synthesis.

1.2. Background on Sliced Wasserstein Loss

Suppose that layer ℓ of an L layer convolutional neural network has N_ℓ channels and M_ℓ pixels in each channel. We denote the feature vector located at pixel m as $F_m^\ell \in \mathbb{R}^{N_\ell}$.

The authors of [16] propose using a different set of statistics instead of the mean squared error between gram matrices. In a manner similar to [3, 15], the authors match the distributions between feature maps, but these feature maps used in [16] are feature maps of VGG19 [6].

With respect a network architecture, let p^ℓ and $\hat{p}^\ell$ be the probability density functions associated with the set of feature vectors $\{F_m^\ell\}$ and $\{\hat{F}_m^\ell\}$. Since our network is discrete, we assume that the probability density functions are always an average of Dirac delta distributions of the form

$$p^\ell(x) = \frac{1}{M_\ell}\sum_{m=1}^{M_\ell}\delta_{F_m^\ell}(x). \tag{1}$$

Let $V \in \mathbb{S}^{N_\ell}$ be a random direction on the unit sphere of dimension N_ℓ. For the purpose of this paper, the Sliced Wasserstein Distance between two distributions of features is of the form

$$\mathcal{L}_{\text{SW},\ell}(p^\ell, \hat{p}^\ell) = \mathbb{E}_V[\mathcal{L}_{\text{SW1D}}(p_V^\ell, \hat{p}_V^\ell)], \tag{2}$$

where

$$p_V^\ell := \{\langle F_m^\ell, V\rangle\} \tag{3}$$

is a set consisting of batched projections of the feature maps F_m^ℓ onto V; if we make a vector P_V^ℓ consisting of the elements of p_V^ℓ, the $1D$ Sliced Wasserstein Loss is the 2-norm between sets of sorted projections:

$$\mathcal{L}_{\text{SW1D}}(p_V^\ell, \hat{p}_V^\ell) = \frac{1}{\text{len}(P_V^\ell)}\left\|\text{sort}(P_V^\ell) - \text{sort}(\hat{P}_V^\ell)\right\|_2^2 \tag{4}$$

and the full Sliced Wasserstein loss over all the layers is

$$\mathcal{L}_{\text{SW}}(I_1, I_2) = \sum_{\ell=1}^{L}\mathcal{L}_{\text{SW},\ell}(p_{V,I_1}^\ell, p_{V,I_2}^\ell), \tag{5}$$

for images I_1 and I_2, respectively. For practical applications, one uses a loss of the form

$$\mathcal{L}_{\text{SW}}(I_1, I_2) = \sum_{\ell=1}^{L} w_i\mathcal{L}_{\text{SW},\ell}(p_{V,I_1}^\ell, p_{V,I_2}^\ell), \tag{6}$$

where w_i are weight terms that set to zero for layers that are not used. We will use this formulation for the rest of the paper.

https://github.com/liping1005/LongRangeSlicedWasserstein. Additional examples and comparision for our proposed synthesis algorithm are given.

We would like to thank Dr. Matthew Hirn for helpful comments and suggestions.

This work was supported in part through computational resources and services provided by the Institute for Cyber-Enabled Research at Michigan State University.

1.3. Theoretical justifications for SW loss

Pitie et al. [5] showed that Sliced Wasserstein Distance satisfies:

$$\mathcal{L}_{\text{SW}}(p, \hat{p}) = 0 \implies p = \hat{p}.$$

The same does not hold for other losses used for texture synthesis, such as the gram matrix loss. Thus, using a Sliced Wasserstein-based loss should capture more stationary statistics compared to the traditional Gram Loss.

1.4. Our contributions

While theoretically sound, the method proposed in [16] cannot effectively capture long range constraints unless a user-added spatial tag is added to guide synthesis. We propose a new set of statistics for single texture synthesis based on Sliced Wasserstein Loss to capture long range constraints without any supervision or hyperparameter tuning. The proposed set of statistics displayers competitive results compared to other single texture synthesis algorithms, and we augment our synthesis results via a coarse-to-fine multi-scale procedure.

2. TEXTURE SYNTHESIS ALGORITHM

Instead of simply matching distributions via slicing over the channel dimension of the feature maps, we purpose matching more statistics in a very simple way. Consider a set of feature maps $F^\ell \in \mathbb{R}^{H_\ell \times W_\ell \times N_\ell}$. In [16], one unravels each $H_\ell \times W_\ell$ feature map, and projects the feature vector associated to each pixel onto direction V in Eq. (5).

To add another loss term, reshape the feature maps into H_ℓ different $W_\ell \times N_\ell$ feature vectors, F^ℓ_H (with $F^\ell_{H,n}$ being a vector of all n^{th} pixels of each feature vector), and project them onto $V_{H_\ell} \in \mathbb{S}^{H_\ell}$. Analogous to Eq. (3), for the distribution p^ℓ_H associated to feature vectors $\{F^\ell_{H,n}\}$, define

$$p^\ell_{V_{H_\ell}} = \{\langle F^\ell_{H,n}, V_{H_\ell} \rangle\}. \tag{7}$$

The corresponding additional loss term is

$$\mathcal{L}_{\text{SW},H}(I_1, I_2) = \sum_{\ell=1}^{L} w_i \mathcal{L}_{\text{SW},\ell}\left(p^\ell_{V_{H_\ell}, I_1}, p^\ell_{V_{H_\ell}, I_2}\right). \tag{8}$$

Intuitively, this loss term accounts for alignment in an image by slicing over the dimension for the height of the feature maps rather than the dimension for the channel of the feature maps, and our new loss function is

$$\mathcal{L}_{\text{Slicing}}(I_1, I_2) = \mathcal{L}_{\text{SW}}(I_1, I_2) + \mathcal{L}_{\text{SW},H}(I_1, I_2), \tag{9}$$

which is the sum of Eq. (5) and Eq. (8).

Denote the feature map extraction from VGG19 as Extract(I). Start with a reference image I_{ref} and a white noise I_{WN} and run for M epochs. The implementation for slicing synthesis is the same as in [16] for Eq. (5). For the additional loss term in equation Eq. (8), the number of batched projections is H_ℓ.

In the next algorithm, assume that the goal is to synthesize an image the same size as the reference image without any loss of generality.

The settings for the slicing loss are to use the first 12 layers of VGG19 for calculating $\mathcal{L}_{\text{SW}}$ and the first two convolutions (after the ReLU) in each convolution block for for calculating $\mathcal{L}_{\text{SW},H}$. The L-BFGS optimizer [2] is used for optimization with a learning rate of $\eta = 1$.

Algorithm 1: Synthesis Algorithm

1: Set I_{WN} as variable to be updated by optimizer.
2: **for** $k = 1, \ldots, M$ **do**
3: Calculate Extract(I_{WN}).
4: Calculate Extract(I_{ref}).
5: Calculate $\mathcal{L}_{\text{Slicing}}(I_{\text{WN}}, I_{\text{ref}})$.
6: Backpropogate and update I_{WN}.
7: **end for**
8: Return updated I_{WN} as synthesized texture.

2.1. Comparison with Other Methods

To test our proposed algorithm, we compare our results with algorithms that have similar runtime and computational complexity. A comparison with Heitz. et. al.[1] (without a spatial tag) and with gram matrices using a spectrum constraint for generating simple 256×256 textures[2] with long range constraints are given in Fig. 1.

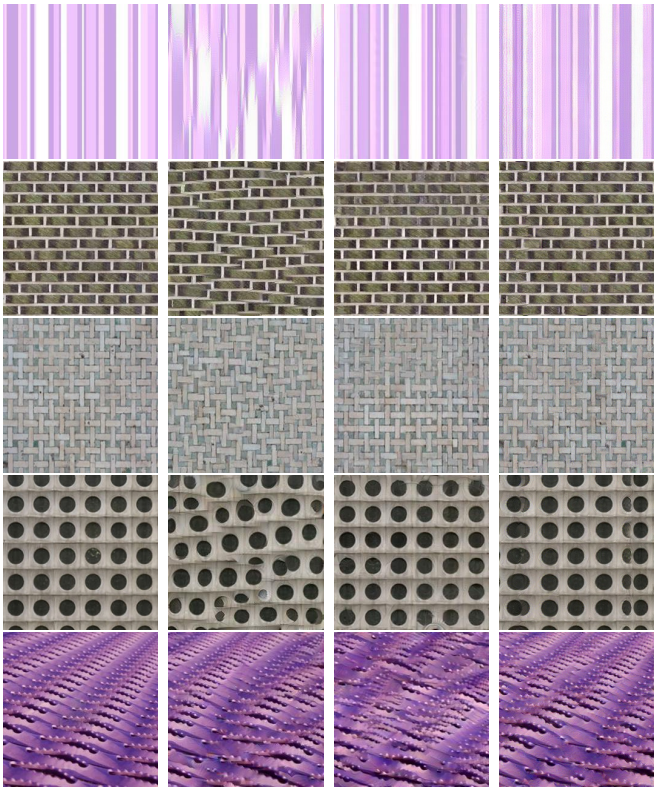

Fig. 1. Comparison of results for pseudoperiodic textures. **Left:** Reference. **Mid Left:** SW Loss. **Mid Right:** Spectrum. **Right:** Using Eq. (9) [Ours].

From the results in Fig. 1, using Eq. (9) yields long range constraints for synthesizing pseudoperiodic images. The results are similar to [9]. However, note that the synthesis using Eq. (9) yields less faithful synthesis relative to [9] in some cases, such as in the fourth

[1] We use the author's TensorFlow implementation, which is a previous commit in https://github.com/tchambon/A-Sliced-Wasserstein-Loss-for-Neural-Texture-Synthesis.

[2] For all the experiments in this paper, our texture sources were the following: the DeepCorr Github Page and the DTD texture database.

row, but can also yield better synthesis compared to [9] in other cases, such as in the fifth row. One possible reason for the failure of using Eq. (9) is using Eq. (8) captures only horizontal stationary statistics for each tensor of VGG19 feature maps, but not vertical stationary statistics.

However, many textures have nonstationary components, so a comparison with more complex textures is in Fig 2.

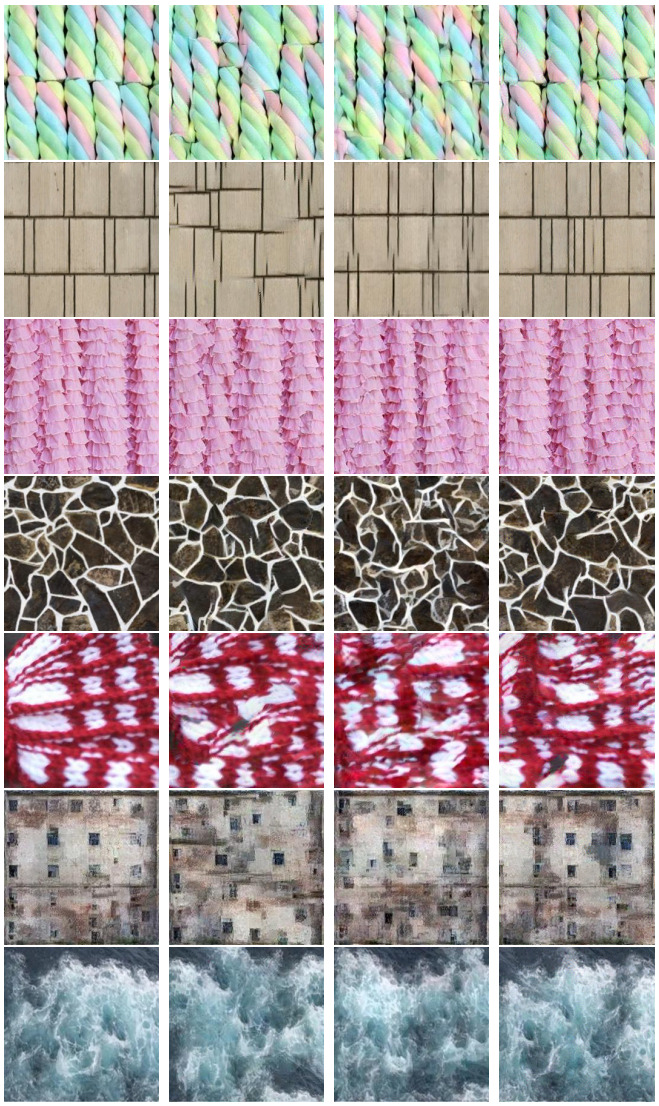

Fig. 2. Comparison of results for more complex textures. **Left:** Reference. **Mid Left:** SW Loss. **Mid Right:** Spectrum. **Right:** Using Eq. (9) [Ours].

From the results of Fig. 2, it is apparent that our proposed algorithm yields better synthesis results on textures with nonstationary components.

To better quantify our results, a set of 34 textures (the textures will be provided in the project page after review) was compiled for a quantitative study between the original SW loss, the spectrum constraint, and our proposed method in Table 1. Most of the textures in this set have long range constraints, some pseudoperiodic and some with nonstationary components. The textures from this set are also used throughout for the visual comparisons in the paper. In the table, SW stands for the method using the original SW Loss, Spec. stands for using a spectrum constraint, and GT stands for the Ground Truth.

We chose the following quantitative metrics: LPIPS [20], FID [21], crop-based FID (this done by taking sixty-four 128×128 crops of the reference texture and synthesized texture for each exemplar. The FID score is calculated between these two sets of images. For the ground truth case, a different set of crops of the reference is used) similar to [18], KID [22], and crop-based KID (c-KID) score. For FID and KID based scores, the implementation from [23] is used.

Table 1. Quantitative Comparison

Method	LPIPS	FID	c-FID	KID	c-KID
Ours	**0.437**	107.220	**71.938**	-0.014	**0.073**
SW	0.454	101.768	78.683	**-0.016**	0.083
Spec.	0.447	**99.615**	78.250	**-0.016**	0.083
GT	0	0	18.069	-0.025	0

From the table, our results are competitive and our proposed set of statistics *did not require searching for a proper hyperparameter to get competitive results*. Note that our results for FID and c-FID compared to other works because FID is a biased estimate [24] and our sample count is lower.

3. IMPROVEMENTS VIA A MULTI-SCALE APPROACH

To ameliorate deficiencies in synthesis quality, we propose augmenting our synthesis process with a multi-scale procedure in a manner identical to [12, 14, 15]. For the multi-scale algorithm at K scales, let $I_{\text{ref},i}$ be the reference image downsampled by a scale factor of 2^i with $i = 0, \ldots, K$, and define the upsampling operator as $\text{Upsample}(I)$. Lastly, define the output of Algorithm 1 using the notation $I_{\text{Synthesis}} = \text{SWSynthesis}(I_{\text{input}}, I_{\text{ref}})$, where I_{input} is the input to be optimized via backpropogation, I_{ref} is the reference texture, and $I_{\text{Synthesis}}$ is the output after synthesis using Algorithm 1.

Algorithm 2: Multi-scale Synthesis Algorithm

1: Initialize $I_{\text{Synthesis}}$ as a white noise that is the same size as the reference texture downsampled by 2^K.
2: **for** $i = 0, \ldots, K$ **do**
3: $\quad I_{\text{Synthesis}} \leftarrow \text{SWSynthesis}(I_{\text{Synthesis}}, I_{\text{ref},K-i})$.
4: $\quad I_{\text{Synthesis}} \leftarrow \text{Upsample}(I_{\text{Synthesis}})$.
5: **end for**
6: Return $I_{\text{Synthesis}}$ as the synthesized texture.

In Fig. 4, note the small improvements in edge generation and general structure when using $K = 1$ compared to $K = 0$. The intuition for why it works is that it generates the details of the texture in a coarse-to-fine way; the initial scale generates the general color and macro-scale features and additional scales add on fine-grain details in an image.

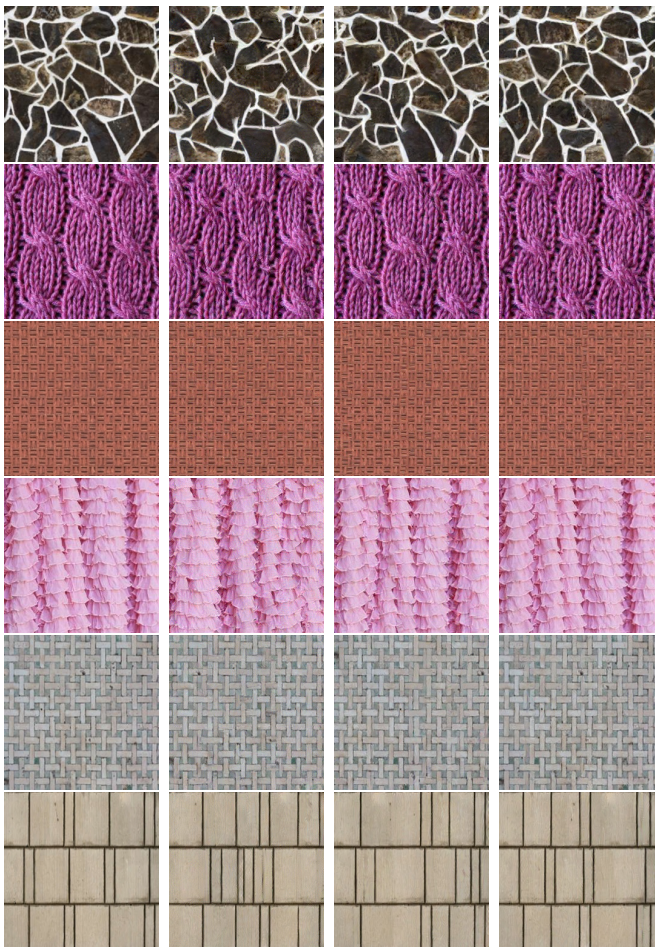

Fig. 3. Multi-scale procedure at different scales. **Left:** Reference. **Mid Left:** $K = 0$. **Mid Right:** $K = 1$. **Right:** $K = 2$.

However, it is possible to create replica textures for larger values of K. Of the 34 images generated for the experiments, there were four repetitions when $K = 2$ for 256×256 images. See Fig 4. below for an example. Based on results for generating large textures from [12], we believe the number of scales one can use before generating repetitions depends on the resolution of the image. Our tests found no repetitions when $K = 1$ for 256×256 images.

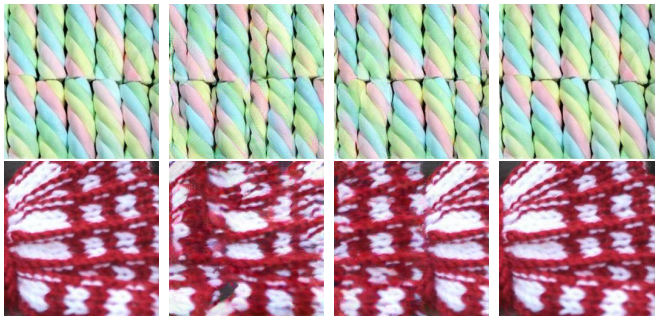

Fig. 4. Progression of synthesis that lead to repetitions. **Left:** Reference Texture. **Middle Left:** $K = 0$. **Middle Right:** $K = 1$. **Right:** $K = 2$.

In the Fig. 5, a small ablation study is done to test the effectiveness of Eq. (9). The multi-scale approach is applied without the additional loss term in Eq. (9). From our results, the multi-scale approach by itself is not enough to fully capture nonstationary statistics or enforce long range constraints. That is to say, the loss term added in Eq. (9) is the main reason for improvements in synthesis quality.

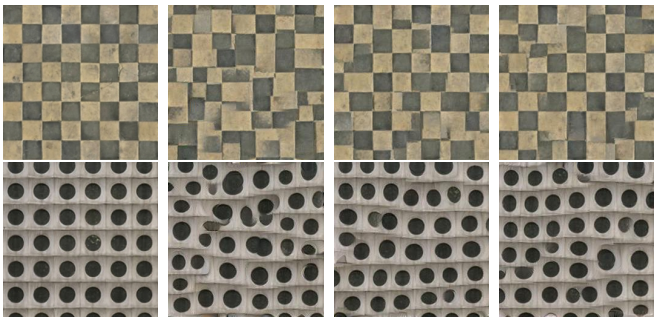

Fig. 5. Results with SW Loss. **Left:** Reference. **Mid Left:** $K = 0$. **Mid Right:** $K = 1$. **Right:** $K = 2$.

3.1. Additional comparisons

Our proposed algorithm with $K = 1$ is compared with the synthesis using SW, a gram matrices with a spectrum constraint, our proposed method with $K = 0$, and the multi-scale method from [12], which uses gram matrices with a spectrum constraint. In Fig. 6. we provide synthesis results, and we provide a quantitative study in Table 2.

Table 2. Comparison of Various Synthesis Methods

Method	LPIPS	FID	c-FID	KID	c-KID
K = 0	0.437	107.220	71.938	−0.014	0.073
SW	0.454	101.768	78.683	−0.016	0.083
Spec.	0.447	99.615	78.250	−0.016	0.083
Gonthier	0.415	77.569	67.728	**−0.018**	0.067
$K = 1$	**0.381**	**67.118**	**53.908**	**−0.018**	**0.044**
GT	0	0	18.069	−0.025	0

4. CONCLUSIONS

We present a modification of texture synthesis via Sliced Wasserstein Loss that has the ability to add long range constraints without user-added spatial tags or other forms of supervision. Our additional loss term can be thought of as a regularization term, but careful hyperparameter tuning was not needed.

Future work would involve testing the dependence between number of scales and synthesis quality for a large dataset of textures. One important question is the following: for each image size, is there a simple way to determine when we have used too many scales in our multiscale algorithm without testing on a database of textures? Additionally, one could consider neural style transfer. Slicing over all three dimensions of VGG feature maps yielded repetitions in our algorithm when $K = 0$. However, for style transfer, it is possible that this would lead more faithful style transfer.

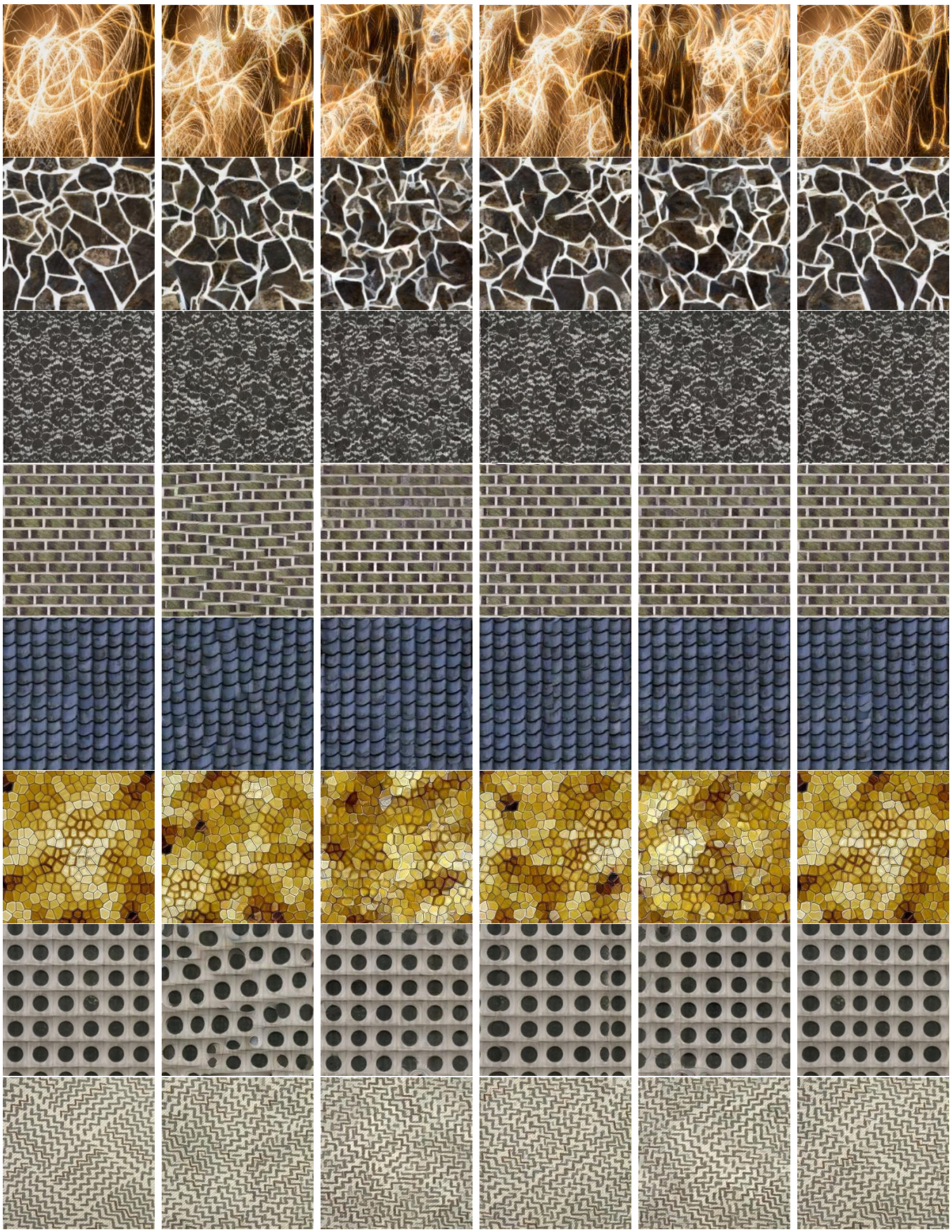

Fig. 6. Additional visual comparison of results. **First Column:** Reference. **Second Column:** SW Loss. **Third Column:** Spectrum Constraint. **Fourth Column:** K = 0 (Ours). **Fifth Column:** Gonthier. **Sixth Column:** K = 1 (Ours).

5. REFERENCES

[1] Bela Julesz. "Visual pattern discrimination," in *IRE Transactions on Information Theory, vol. 8, no. 2, pp. 84–92, 196*, 1962.

[2] Dong C. Liu and Jorge Nocedal. "On the limited memory BFGS method for large scale optimization," in *Mathematical Programming, vol. 45, pp. 503–528*, 1989.

[3] D. J. Heeger and J. R. Bergen. "Pyramid-based texture analysis/synthesis," in *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques, ACM*, 1995, pp. 229–238.

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

[5] Pitie, Francois and Kokaram, Anil C and Dahyot, Rozenn "N-dimensional probability density function transfer and its application to color transfer," in *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, 2005.

[6] Karen Simonyan and Andrew Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition," in *International Conference on Learning Representations (ICLR)*, May 2015.

[7] Leon Gatys, Alexander S Ecker, and Matthias Bethge. "Image style transfer using convolutional neural networks," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[8] Leon Gatys, Alexander S Ecker, and Matthias Bethge. "Texture synthesis using convolutional neural networks," in *Advances in Neural Information Processing Systems 28*, 2015, pp. 262–270.

[9] Yann Gousseau, Gang Liu, and Gui-Song Xia. "Texture synthesis through convolutional neural networks and spectrum constraints," in *International Conference on Pattern Recognition (ICPR)*, December 2016.

[10] Xavier Snelgrove. "High-resolution multi-scale neural texture synthesis," in *SIGGRAPH Asia 2017 Technical Briefs, SA '17, New York, NY, USA, 2017. Association for Computing Machinery.*

[11] Omry Sendik and Daniel Cohen-Or. "Deep correlations for texture synthesis," in *ACM Trans. Graph., 36(4)*, July 2017.

[12] Yann Gousseau, Nicolas Gonthier, and Saïd Ladjal. "High resolution neural texture synthesis with long range constraints," in *Journal of Mathematical Imaging and Vision 64:478–492*, 2022.

[13] Yang Zhou, Zhen Zhu, Xiang Bai, Dani Lischinski, Daniel Cohen-Or, and Hui Huang. " Non-stationary texture synthesis by adversarial expansion," in *ACM Transactions on Graphics (ToG), vol. 37, ACM, 2018, pp.1-13*, 2018.

[14] V. Kwatra, I. Essa, A. Bobick, and N. Kwatra. 'Wasserstein loss for image synthesis and restoration," in *ACM Transactions on Graphics (ToG), vol. 24, ACM, 2005, pp. 795–802*.

[15] Guillaume Tartavel, Gabriel Peyre, and Yann Gousseau, "Variational Texture Synthesis with Sparsity and Spectrum Constraints," in *Journal of Mathematical Imaging and Vision, 52:124–144*, 2015.

[16] Eric Heitz, Kenneth Vanhoey, Thomas Chambon, and Laurent Belcour. 'A Sliced Wasserstein Loss for Neural Texture Synthesis," in *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2021.

[17] Dmitry Ulyanov, Andrea Vedaldi, and Victor S. Lempitsky. 'Improved Texture Networks: Maximizing Quality and Diversity in Feed-forward Stylization and Texture Synthesis," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, April 2019.

[18] Guilin Liu, Rohan Taori, Ting-Chun Wang, Zhiding Yu, Shiqiu Liu, Fitsum A. Reda, Karan Sapra, Andrew Tao, and Bryan Catanzaro 'Transposer: Universal Texture Synthesis Using Feature Maps as Transposed Convolution Filter," in *arXiv:2007.07243 [cs.CV]*, July 2020.

[19] Morteza Mardani, Guilin Liu, Aysegul Dundar, Shiqiu Liu, Andrew Tao, and Bryan Catanzaro. 'Neural FFTs for Universal Texture Image Synthesis," in *Proceedings of the 34th International Conference on Neural Information Processing Systems (NIPS)*, April 2020.

[20] Zhang, Richard and Isola, Phillip and Efros, Alexei A and Shechtman, Eli and Wang, Oliver 'The unreasonable effectiveness of deep features as a perceptual metric," in *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, 2018.

[21] Heusel, Martin and Ramsauer, Hubert and Unterthiner, Thomas and Nessler, Bernhard and Hochreiter, Sepp 'Gans trained by a two time-scale update rule converge to a local nash equilibrium," in *Advances in neural information processing systems (NIPS)*, 2017.

[22] Bińkowski, M and Sutherland, DJ and Arbel, M and Gretton, A 'Demystifying MMD GANs," in *International Conference on Learning Representations (ICLR)*, 2018.

[23] Parmar, Gaurav and Zhang, Richard and Zhu, Jun-Yan 'On Aliased Resizing and Surprising Subtleties in GAN Evaluation," in *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, 2022.

[24] Chong, Min Jin and Forsyth, David 'Effectively unbiased fid and inception score and where to find them," in *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, 2020.