Synthesis of textures in image processing

Roman V. Grigorii

June 5, 2017

1 Introduction

Texture synthesis in image processing has been a field of significant interest in the days of quickly expanding usage of computer graphic interfaces. Early uses of texture synthesis were in texture storage applications where it was desired to save and use only a small portion of a texture from which synthesis would be possible. Today, some common applications include image filling, the process which fills up sections of an image with a set texture, texture mapping, where 2D textures are patched over 3D objects, among many others. With an enormous effort for developing robust techniques in texture synthesis in the 1990's and early 2000's researches have paved way for future algorithms which adopt in some way the techniques presented in these early works. Indeed, recent research that has set the new state of the art for syntheses techniques regards the synthesis results of early methods as standard points of comparison.

In this paper I discuss some of the most significant contributions to the field of texture image synthesis and processing. I start with the dated approaches to texture synthesis and discuss the more recent developments in the field. Furthermore, I develop several of the techniques which I introduce in order to evaluate their performance and develop points of comparison for some of the approaches. The significance of the methods I implement is their simplicity in concept while producing results of great quality. Due to this simplicity and the possibility for applying these methods real time, I plan to adopt them for synthesis of haptic textures.

2 Synthesis at large

In the field of texture image processing there are two approaches into which the task of synthesis is generally divided into. These are parametric and non-parametric methods. A typical parametric texture synthesis technique assumes a level of stochasticity present in a sample texture image which can be described using distribution parameters. Therefore all textures are treated as a random fields which when passed through a number of constraint filters result in a texture that matches statistical measurements of some model texture. Textures are perceived as one when a given set of statistical properties are equal. This method of synthesizing texture has been motivated heavily by perceptual qualities of human vision

with some algorithms discussed later directly adapting filters observed to take place in human visual information processing. There is therefore a need for understanding of human perceptual process in order to motivate the placement of various filters and understanding the statistical interaction of filtered components of the image during texture processing. Unfortunately, without clear perceptual motivation, basic parametric methods lack in synthesis quality. The strength of the parametric approach lies he fact that it focuses on constructing a texture model described by a set of parameters which require minimal storage space. Moreover, having a texture model means that it can be applied iteratively to filter a random set of pixel values rather than sampling pixel values directly from the original image.

The second approach for texture synthesis is called non-parametric, which does not rely on any parameters or statistical constraints in order to synthesize a texture. Instead this method relies on the assumption that a given texture can be synthesized by some variation of Markov Chain sampling. This means that having picked some unit window of a given texture we can form a new texture by placing element samples from a given texture next to each other to fill the desired space. No texture model is used in the synthesis stage and parts of the original image are directly copied into the new one. Non-parametric process is extremely robust in synthesis since generally the most important parameter to control is the window size which sets the amount of local features that will be copied directly over. No new pixel values are introduced unless feathering techniques are utilized (discussed later). This method does not consider the perceptual process of processing texture and can be applied in real time texture synthesis applications due to the low computational cost. Unlike the parametric method, non-parametric techniques have shown great synthesis results early on in the field of image processing. Modeling a texture is a very difficult process which is generally computationally expensive with parametric based techniques therefore developing steadily with improvements in computer processing speeds.

2.1 Early parametric synthesis methods

Parametric texture synthesis techniques are based on the idea that an image texture can be broken down into subcomponents which can then be parametrized by various distribution. There are several reasons to why this break down is necessary, with one of the primary motivations for this decomposition is arising from evidence that early visual processing completes similar operations[1, 2]. A set of bandpass filters, isotropically oriented and varying in scale, are applied on the image to extract these low level representations. Many research works use the statistical parameters, typically referred to as marginal statistics, of pixel intensity probability distribution in these images to synthesize textures [3, 4]. The work of [4] in particular uses maximum entropy principle to find band placement, process these distributions, and extract useful parameters.

An important contribution in parametric texture synthesis techniques was introduced by Portilla et. al. who have used cross-correlations of of joint-density representation of texture pixel intensities as one of the constraints to capture the realism of a wider set of texture than before [5]. In other words, in addition to identifying the parameters of relevant distributions their correlations across the image are also used for synthesis. Portilla group used "steerable pyramid" which decomposes an image by varying the direction and width of band filters and is commonly utilized as a foremost tool in image processing operations [6]. The coefficients

of these transforms are then correlated in order to extract a deeper level in interdependencies of local texture densities. In particular, a cross-correlations of high-frequency bands with low-frequency bands were shown to retain a good amount of repeated texture sections, or the regularity of the image, and preserve edges and corners.

A point to make here is that the algorithms of early parametric synthesis begin this process by using random noise field of set size which is then iteratively filtered to match the marginal statistics of a given texture image. The iterative process is necessary, since correlation based metrics are difficult to implement in a simple functional form. Due to iterative nature of posed problem synthesis of a wider set of textures can become very computationally expensive. In early parametric techniques results varied drastically and strength of this method lay in representing stochastic over regular textures. The reason for this is that a perfectly stochastic texture can be synthesized given the distribution parameters alone. With even some regularity even quasi-stochastic approaches tend to fail. Therefore, much of early efforts in parametric synthesis has been dedicated to developing more robust techniques for modeling periodic textures.

3 Modern parametric approaches

In modern days of texture analysis and synthesis there is a heavy reliance on neural network as means of computing texture unique parameters from which synthesis is possible. In particular convolution based neural networks have seen wide spread use in image recognition software [7–10] and have been found to perform well when adapted for texture synthesis[11]. The idea is to pose the problem as one of classification where each layer in a CNN represents groups of filtered images called "feature-maps" [12]. The features of activation matrix is then used to iteratively synthesize texture in order to match the same activation response of a CNN. This class of texture generation networks are branded "generative networks" and have seen large success in rendering both regular images and textures. [13, 14]. Some interesting applications of these include transfer of style, where the lighting and general style of on image is posed onto another image through a feed-forward process. [15]

It has been noted that the features of NN hidden layers resemble filter banks used in parametric synthesis. The difference is in that these filters are non-linear unlike ones adapted by Silmoncelli et al, and the generated texture models do not necessarily mimic the perceptual qualities of vision. A general process of synthesis begins by passing a sample texture through a CNN followed by extraction of Gram matrices of features in some number of it's layers. A loss function defined as the difference between the Gram matrices of the synthesized image and the original image is used to iteratively filter the initial noise image to match the layer features of the original sample. A perfect synthesis is achieved when Gram matrices of original and synthesized image are the same.

Synthesized texture images using CNN are state of the art examples of synthesis techniques. A drawback to these approaches, however, is the computational cost which arises from the extremely large number of parameters over which synthesis occurs. Gram matrices can be very large making the parameter sets be thousands of features long ([13] have worked with close to one million parameters in syntheses). The issue of speed is currently being widely investigated by research community interested in CNNs. Great news is that

any findings made that speed up neural network layer evaluation transfer directly to texture synthesis methods.

3.1 Non-parametric texture models and computation optimization

Non-parametric texture synthesis relies on sampling regions of a given texture, which I will henceforth refer to as "the seed", to fill up some region of desired size. The most simple non-parametric approach involves randomly sampling $n \times m$ sized sections of seed texture image and tile them side by side to fill up some region. As one can imagine this is a highly problematic approach because the contact lines are usually very apparent in the resulting image. However, some work was done in image synthesis by method of tiling in getting rid of these artifacts typical to tiling. One method has approached the removal of the seams through low-pass filtering of the image at regions of contact to get rid of high frequency components due to abrupt transitions. This introduces some blurring, and there is a clear trade-off between natural look of the texture image and it's quality.

A different approach introduced by [16] handles this problem by remapping pixel values at the seams in order to remove the seam pattern. This remapping is referred to as "cat mapping" and is given by the set of equations:

$$x^{l+1} = (x^l + y^l) mod(n)$$

$$x^{l+1} = (x^l + 2y^l) mod(m)$$

where the window over which the remapping takes place is of size $n \times m$. This mapping belongs to the group of deterministic chaos transformations and being an ergodic process it is a good transformation shown to work well in preservation of stochasticity in texture images. The remapping procedure preserves all pixels from a given region generating a stochastic distribution over the entire image size. To retain some local features present in the seed a block-wise remapping can take precedence over pixel-wise as described by above equations. To determine the block size the paper reports a constraint that allowed parametrization of the entire process of block remapping with the size of original seed image in mind. The constraint is as follows:

$$c_l <= \frac{mn}{gh} < c_h$$

where $m \times n$ is the size of the block and $g \times h$ is the size of seed image. The research work reports .5 and .75 as good values for texture synthesis procedure. These are tunable parameters, along with m and n variables.

The authors have admitted that some boundaries can still be visible post processing even after multiple iterations of cat mapping. The suggested solution is to remove pixels at the boundaries and use pixel wise sampling based on neighborhood pixels values of removed pixels. Although I have not observed any literature that applies this procedure to fix the seams as a fore-front procedure I imagine that this would also work well enough that remapping of pixel values would not be necessary. However, chaos mosaic procedure allows for a level of randomness introduced to the synthesis that ensures that is often lacking in non-parametric approaches.

As I have mentioned, non-parametric synthesis problems are often approached by assuming that texture can be modeled a Markov Random Field. This means that any given

pixel intensity values is determined by it's closest pixels forming a set that is defined as the neighborhood of the pixel. Given a texture seed of defined size, each pixel u has a set of the pixels in it's neighborhood $r(u)$ that can sampled to determine the conditional probability $P(u|r(u))$ distributions for the intensity values of the original pixel. A technique proposed in [17] which adapts this direction in synthesis implements an algorithm in which a new pixel value u' is determined by finding what neighborhood of pixels $p(u) \in I_i$ where I_i is the initial seed image of size $g \times h$ most closely resembles $p(u') \in I_f$ where I_f is the final synthesized image. An example of a simple equation which determines similarity of two regions of an image is given by:

$$argmin_{u'} \sum_{n=1}^{N} \sum_{m=1}^{M} (p(u') - p(u))^2$$

but as one can imagine, some modifications to this equation can be applied such as the distance metric used depending on the class of textures at hand. The assumption being made by utilizing this equation is that it acts as a good heuristic for sampling pixels under the conditional probability distribution. In other words, since the conditional distribution is not continuous this provides a robust method for sampling new pixel values from the seed image.

Using this metric an entire image can be scanned in order to find a neighborhood closest to the one of a given pixel, and set the new pixel value to the value of one in the neighborhood as $u' = u$. Tunable parameters in this scheme is the neighborhood size and the weighting distribution of the neighborhood pixels. It is common to apply a Gaussian weight in order to let the closest pixel values to the pixel in question be more deterministic of it's value than the ones further away in a given window. The resulting synthesized texture samples are able to retain the average pixel value of the seed image while producing new region of texture unobserved in the original. Given the correct size of what is defined as the "neighborhood" a regularity of a given image can be very well preserved.

In order to test the workings of this algorithm I have written a synthesis algorithm based upon it. The results of this synthesis method through my own implementations can be observed in Figure 1. Notable accuracy of periodicity from sample 1 and 2 is observed. No local feature of synthesized images is extracted directly from the seed. All of the roughness is reflected in pixel by pixel copying.

A pixel by pixel texture synthesis approach is usually very computationally expensive. A lot of computational time is spent on searching for the closest neighborhood in the original image. Computation for a 100x100 seed image in MATLAB took 30 seconds but this time would increase by a power function, making computation of larger textures impossible to do efficiently. One approach to fix this problem adopted in the field is to select a neighborhood of a pixel which is close enough to the one at hand and not the closest from the seed. In other words, the problem becomes about finding a neighborhood that is different to the neighborhood of an empty pixel by some parameter ϵ. In my experience this tends to work worse than the initial method for the periodic textures since the range of possible lowest distance values vary dramatically and therefore some regions get synthesized better than others. The parameterization also increases in difficulty and I have found that ϵ values range dramatically over various texture samples depending on the pixel variance. For this approach, a normalization of the distance function is in order to account for this.

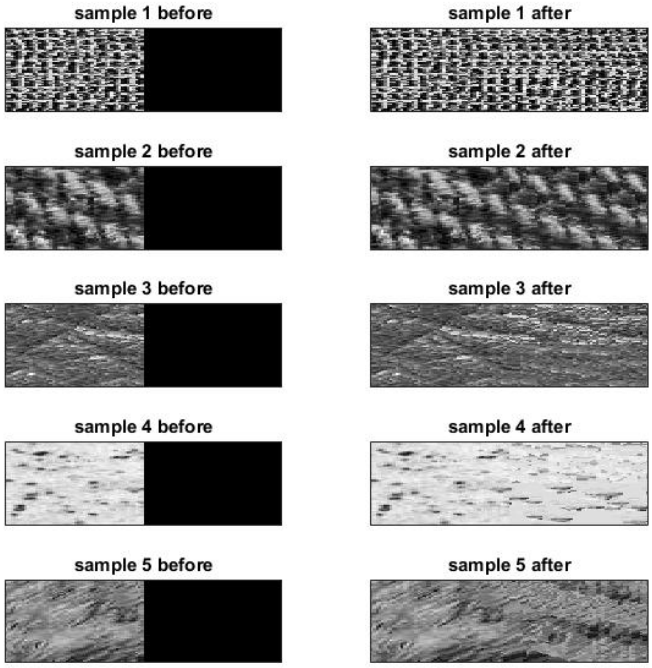

Figure 1: Several sample textures generated using the Non-parametric model. A 40x40 neighborhood window was used to fill a 100x100 image

A very robust approach in the time optimization problem is to project the high dimensional $m \times n$ neighborhood space into a lower dimensional one using PCA (principal component analysis). This reduction of dimensions of a given neighborhood highly decreases the computational time at the small cost of come computational inaccuracies. Several researchers have utilized the subspace representation of texture images [18, 19] with a reported $\tilde{9}9\%$ variance retention after an order of magnitude in dimensionality reduction and therefore approximately a 99% accuracy in computation of neighborhood similarity. Another approach to this problem is to project the entire $g \times h$ seed texture into a subspace and apply sampling techniques in this space [20]. To test the workings of this technique more closely I have reduced the dimensionality of a 512×512 sample image by a factor of 32 retaining 3.3 % of the original dimensions. The results are shown in Figure **??**. As one can observe, a lot of the original image quality is retained while a severe decrease in dimensionality has been completed. This operation would significantly decrease the computational cost of pixel by pixel texture synthesis and allow for real-time texture synthesis important in computer graphics.

Yet another technique for non-parametric based texture synthesis, inherently faster than a pixel by pixel synthesis approach, is patch-based sampling [19] . In this approach a texture is grown by randomly selecting patches from seed texture image and placing them into an empty canvas with some overlap over one another. In order to select patches that have the best transitions in pixel intensity from one to the next, the difference between pixel values at

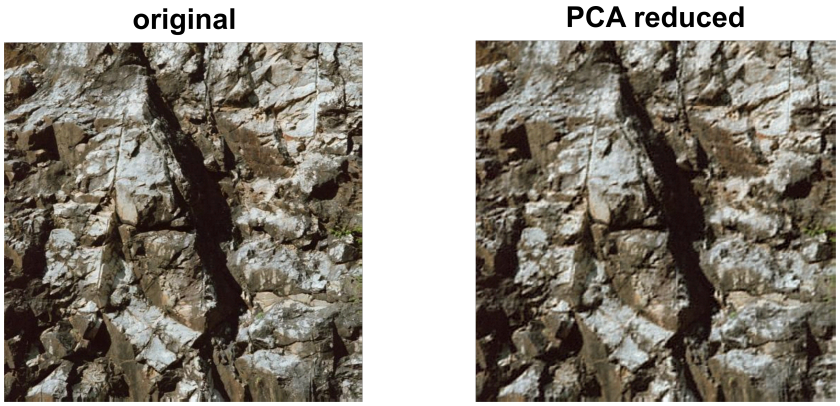

Figure 2: The original sample image along side one that that had 96% of pixel dimensions removed, while retaining only 32 out of 1024 dimensions which provide most variance in the given image.

the overlap regions is sought to be minimized similar to the neighborhood based optimization in the pixel based synthesis. The algorithm I adapted in my work uses random sampling of patches from original texture, and seeks to find a patch that is is some ϵ euclidean distance at the overlap region from it's neighboring patch. Alternatively the entire seed image can be searched for the most similar neighborhood. The distance metric can be modified depending on prior knowledge of texture type. To speed up this process a large ϵ can be selected at the cost of quality in the intersection regions. To fix the regions of overlap where the seems are apparent, a technique called "feathering" can be applied to smooth the transition between bordering patches.

In my implementation of the patch-based sampling I have applied feathering at the borders where I gradually increased the intensity of the region of patch being added and decreased the intensity of the region of texture already present and adding them together to generate the final image. I define the distance between two overlapping regions as

$$d = (\frac{1}{wh}\sum_{i=1}^{w}\sum_{j=1}^{h}(u_{ij} - u'_{ij})^2)^{1/2}$$

Where the size of the overlapping region is $w * h$, and u and u' are pixel intensity values of the two regions from the seed and new patch of image respectively. The results of my implementation are presented in Figures **??** where I used a 100x100 window to generate 512x512 images. I show the results of my algorithms when feathering was applied as well as when it was not. Notice that in general the feathering technique does blend the edges well together, and provides very smooth transitions that add to the realism of the texture. Of particular note is sample 5, which shows no local similarity to the seed image and creates patterns of it's own based on the optimization method. Patch based texture synthesis has shown drastic improvement in speed which can be further improved with the described PCA technique.

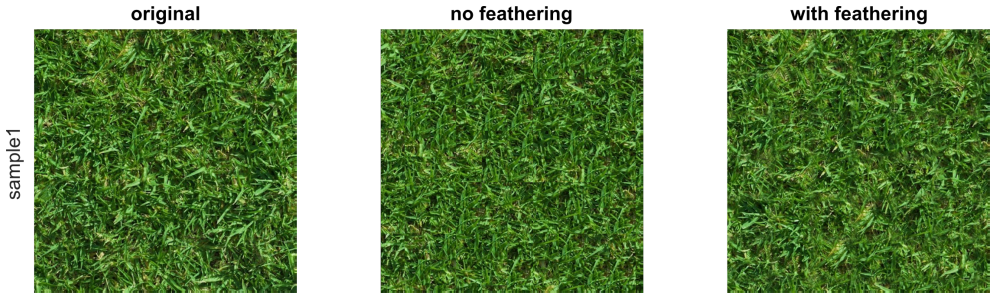

Figure 3: An image of grassy texture being rendered using patch sampling. Notice the slightly blurred section of the feather synthesis result due to feathered stitching.

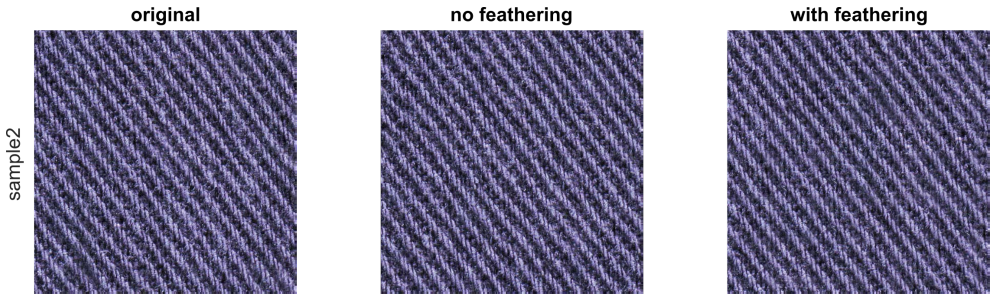

Figure 4: A set of successful synthesis results for an image of denim texture.

4 Closing remarks

As was mentioned, modern approaches to texture synthesis rely on the complexity of neural network for extraction of non-linear filters used for synthesis hidden within the layers of the networks. Parametric synthesis results are often compared to those produced by non-parametric methods because non-parametric methods have seen early successes over parametric methods. It is only in recent times that parametric synthesis has shown to work well for a wide class of textures spanning beyond simple stochastic samples.

Neural networks are especially good when applied for texture mapping and style mapping purposes implemented by applying layers of the network in synthesis problems. It is clear that most of research in the field of texture synthesis will be approached from the paradigm of machine learning at least in the near future. I expect that a lot of time will be dedicated to solving the problem of efficiency of these approaches by reduction of high-dimensional parameter spaces extracted from neural layers.

It would be a fascinating research objective to bridge the neural net feature set-up and the perceptual process. Neural nets are of course only similar to the neural networks in the human brain by virtue of the name, but it appears that such non-trivial processes as visual processing and synthesis must have some similarities.

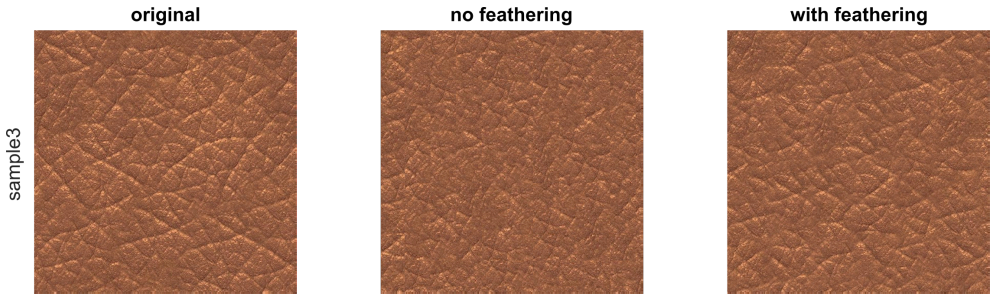

Figure 5: Synthesized leather image. Notice that even though the same window was used for both synthesis results, feathering allows for blending that seems to capture larger feature to a better extent.

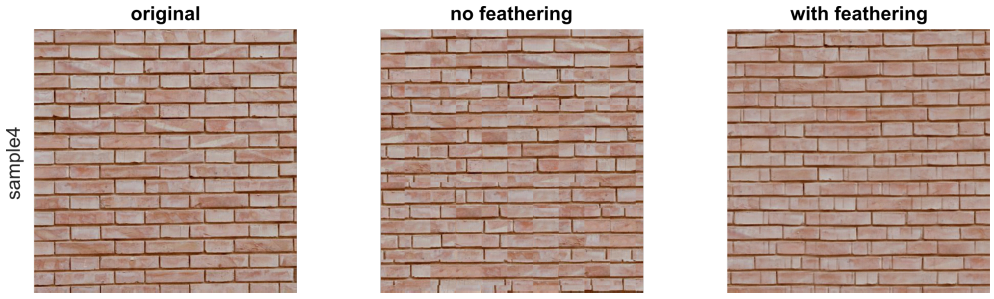

Figure 6: A synthesizes brick texture. Feathering is very apparent in this example due to clear lines separating regular sections of the image.

Figure 7: A picture of series of radishes synthesized to yield good results. The last image tends to capture some of the roundness of the vegetable, but falls to come repeating patterns.

References

[1] James R Bergen and Edward H Adelson. Early vision and texture perception. *Nature*, 333(6171):363–364, 1988.

[2] Judson P Jones and Larry A Palmer. An evaluation of the two-dimensional gabor filter model of simple receptive fields in cat striate cortex. *Journal of neurophysiology*, 58(6):1233–1258, 1987.

[3] David J Heeger and James R Bergen. Pyramid-based texture analysis/synthesis. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 229–238. ACM, 1995.

[4] Song Chun Zhu, Yingnian Wu, and David Mumford. Filters, random fields and maximum entropy (frame): Towards a unified theory for texture modeling. *International Journal of Computer Vision*, 27(2):107–126, 1998.

[5] Javier Portilla and Eero P Simoncelli. *Texture representation and synthesis using correlation of complex wavelet coefficient magnitudes*. Citeseer, 1999.

[6] William T Freeman, Edward H Adelson, et al. The design and use of steerable filters. *IEEE Transactions on Pattern analysis and machine intelligence*, 13(9):891–906, 1991.

[7] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[8] Steve Lawrence, C Lee Giles, Ah Chung Tsoi, and Andrew D Back. Face recognition: A convolutional neural-network approach. *IEEE transactions on neural networks*, 8(1):98–113, 1997.

[9] Berkman Sahiner, Heang-Ping Chan, Nicholas Petrick, Datong Wei, Mark A Helvie, Dorit D Adler, and Mitchell M Goodsitt. Classification of mass and normal breast tissue: a convolution neural network classifier with spatial domain and texture images. *IEEE transactions on Medical Imaging*, 15(5):598–610, 1996.

[10] Michael Egmont-Petersen, Dick de Ridder, and Heinz Handels. Image processing with neural networks—a review. *Pattern recognition*, 35(10):2279–2301, 2002.

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

[12] Ivan Ustyuzhaninov, Wieland Brendel, Leon A Gatys, and Matthias Bethge. Texture synthesis using shallow convolutional networks with random filters. *arXiv preprint arXiv:1606.00021*, 2016.

[13] Alexey Dosovitskiy, Jost Tobias Springenberg, and Thomas Brox. Learning to generate chairs with convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1538–1546, 2015.

[14] Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Jimenez Rezende, and Daan Wierstra. Draw: A recurrent neural network for image generation. *arXiv preprint arXiv:1502.04623*, 2015.

[15] Dmitry Ulyanov, Vadim Lebedev, Andrea Vedaldi, and Victor Lempitsky. Texture networks: Feed-forward synthesis of textures and stylized images. In *Int. Conf. on Machine Learning (ICML)*, 2016.

[16] Ying-Qing Xu, Baining Guo, Heung-Yeung Shum, et al. Chaos mosaic: Fast and memory efficient texture synthesis. Technical report, Tech. Rep. MSR-TR-2000-32, Microsoft Research, 2000.

[17] Alexei A Efros and Thomas K Leung. Texture synthesis by non-parametric sampling. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 2, pages 1033–1038. IEEE, 1999.

[18] Aaron Hertzmann, Charles E Jacobs, Nuria Oliver, Brian Curless, and David H Salesin. Image analogies. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 327–340. ACM, 2001.

[19] Lin Liang, Ce Liu, Ying-Qing Xu, Baining Guo, and Heung-Yeung Shum. Real-time texture synthesis by patch-based sampling. *ACM Transactions on Graphics (ToG)*, 20(3):127–150, 2001.

[20] Sylvain Lefebvre and Hugues Hoppe. Appearance-space texture synthesis. *ACM Transactions on Graphics (TOG)*, 25(3):541–548, 2006.