https://doi.org/10.2352/ISSN.2470-1173.2019.6.MAAP-484

CNN Based Parameter Optimization for Texture Synthesis

Jiangpeng He[1], Kyle Ziga[1], Judy Bagchi[2], Fengqing Zhu[1]

[1] School of Electrical and Computer Engineering
Purdue University
West Lafayette, IN, USA

[2]Dzine Steps LLC
Spring, TX

Abstract

Texture synthesis is the process of generating a large texture image from a small texture sample. The synthesized image must appear as though it has the same underlying structural content as the input texture sample. However, most texture synthesis methods require the user to tune parameters for different input or provide feedback to the system to achieve satisfactory results. To make texture synthesis approaches more efficient and user friendly, we propose a fully automatic method to select a set of suitable parameters for texture synthesis that can be applied on commonly used textures. Our method uses Convolutional Neural Network (CNN) to predict the optimal parameters for texture synthesis based on image quilting algorithm [1]. Our method showed satisfactory results on different types of textures.

Introduction

Texture synthesis plays an important role in variety of applications related to computer vision, computer graphic and image processing. Some examples may be texture reconstruction in image inpainting, content generation for texture mapping. In addition, with the growth of augmented and virtual reality applications, an efficient and automatic method for texture synthesis is indispensable. Texture synthesis is the process of generating a large texture image from a small texture sample. The synthesized image must appear as though it has the same underlying structural content as the input texture sample. Thus, for most textures, simply tiling the sample texture will not create the realistic appearance of the enlargement. Instead it should look as though it was created by the same underlying process as shown in Figure 1. In this way, only input texture samples are required to generate large size synthesized texture. Nowadays, most methods of texture synthesis require user to carefully set parameters [1] or provide feedback to the system in order to generate satisfactory results [2]. Removing the need of manual interaction while producing comparable results, will significantly improve practicability of existing texture synthesis methods.

Texture synthesis methods can be classified into pixel-based methods and patch-based methods. Efros and Leung [3] proposed the first pixel-based texture synthesis method by using non-parametric sampling. Their method begins with a single seed pixel and grows the synthesized texture from that starting location. The conditional distribution of each pixel in the output texture given all its neighbors synthesized so far is estimated by searching the sample image and finding all similar neighborhoods. The center pixel of the chosen neighborhood is then taken as the new synthesized pixel value in the output and the algorithm moves on to the next pixel location. This algorithm works well for a wide range of textures, but since it generates only one pixel at a time and then do a full searching again on input sample to synthesize the next pixel, the process is extremely slow. It can take several hours for a high large texture output, which is not practical at all. In addition, the pixel-based method usually loses the global structure and orientation. Although Wei and Levoy [4] later proposed to accelerate this process by using tree-structured vector quantization, it is still hard to implement in real-time.

The patch-based algorithm is motivated by the fact that for most complex textures, very few pixels have a choice of values that can be assigned to them. This is because during the synthesis process, most pixels have their values totally determined by what has been synthesized so far. Therefore, the unit of synthesis should be a patch instead of a single pixel. This kind of method finds and copies an entire patch instead of a single pixel from the input texture. Patches are placed into the output image and the challenge is to create a seamless transition from one patch to another. In [5] the boundary artifacts are removed by blending the transition areas. Feathering, or blurring is used across the patch boundaries in order to create smooth transitions. Efros and Freeman [1] proposed a patch-based texture synthesis method called image quilting. In their method, neighboring patches are slightly overlapped and minimum error boundary cut is performed to combine the neighboring patches more naturally. To avoid repetition, for each patch in output image, a similarity metric is computed in the overlapping regions for patches in input sample and is used to select one patch that meet the criteria as the next synthesized patch. The minimum boundary cut is then further extended in [6], irregular patches are used without a constant patch size in order to find the optimal seam between patches. A graph cut approach is used to determine the optimal patch seam for any given region of the output texture.

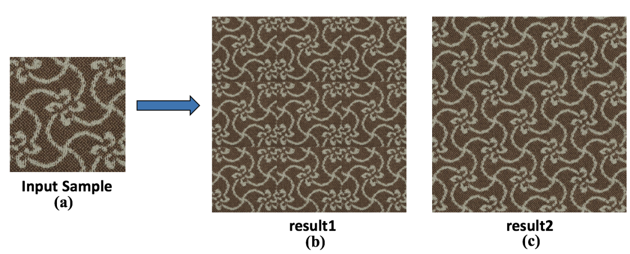

Figure 1. *An example of tiling the input texture (a) to generate result (b) and synthesized texture (c) with proper parameters.*

The patch based method works well and fast for certain types of texture while some textures are extremely difficult to synthesize no matter which method is applied. Since this is a parameter-based method, the user need some understanding of the algorithm to select a set of different parameters for different input textures in order to achieve satisfactory result. This process can be tedious, burdensome to untrained users, and difficult to adapt the method to real life applications.

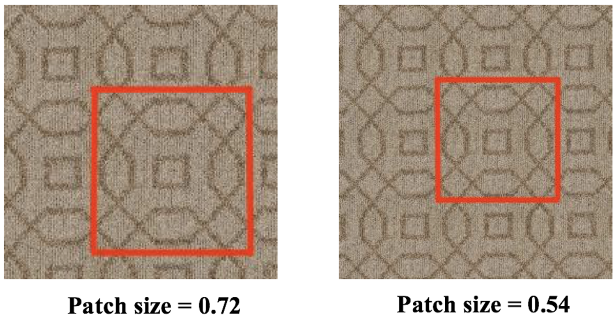

Figure 2. *An example of the same texture with different patch size due to different pattern scales.*

In [2], the authors proposed a non-parametric synthesis method using texture classification. A sample texture is classified into three categories: stochastic, regular and irregular based on local binary pattern. A set of starting parameters is provided based on the texture class. Users does not need to manually provide a set of parameters but they still need to give feedback to the system in terms of the satisfaction of the synthesis results to decide if a different set of parameters should be used to achieve better results. Although this semi-automatic method removes the need for users to understand and set parameters, it still involves manual interaction. In addition, since the parameters such as patch size vary a lot due to different textures and different pattern scales, the provided starting parameters cannot fit all textures from the same class. As a result, the user may still need to perform many trials to get a satisfactory result.

In this work, our goal is to develop a method that can automatically determine a set of parameters for texture synthesis given an input sample using the image quilting method. However, like most texture synthesis methods, the image quilting method requires the user to tune parameters for different input samples to achieve a satisfactory result. The challenge of finding a set of suitable parameters comes from the fact that the parameters, in particular the patch size, varies a lot due to different texture types. Moreover, even the same texture sample with different scales require different patch size for synthesis. Since these parameters are related to their underlying texture features such as scales and structures, we propose to use Convolutional Neural Networks (CNN) to learn the connections between the synthesis parameters and texture features. As a result, our method can predict a set of suitable parameters for texture synthesis for different input textures.

Training Data

There are three parameters in the image quilting method: patch size, size of overlap region and error tolerance for overlap region. The patch size is defined as a percentage of the input image size, the overlap is defined as a percentage of the patch size and the error tolerance is the difference between patch overlap regions based on a selected error metric . We found that these parameters, especially the patch size is strongly related to features of the input texture such as pattern scales. For example, the same textures with different pattern scales will result in different patch size as shown in Figure 2, the patch size must be on the order of the fundamental repeated element. In addition, the patch size also varies a lot due to different types of textures. Therefore, we propose to use CNN to extract the pattern related feature of input sample and automatically determine the suitable parameters corresponding to different types of texture and also different pattern scales.

Training Sample Generation

Since the number of different textures we can use in the training process is limited, we first generate a large synthesized texture and then extract small blocks from it. Each small block can be used as a training sample. The specific steps are: (1) Manually set the best parameters for the image quilting method to generate a good synthesized texture and use it as the ground truth; (2) Define a set of sizes for the training sample texture; (3) Extract small blocks from the synthesized result with different scales from step (2). As shown in Figure 3, the input sample is of size 700×700, and the synthesized texture is with size 1500×1500. We then extract from it small blocks with pre-defined scales. In step (2), we need to ensure that each training sample can be synthesized back to the ground truth so that each small block contains the complete texture information.

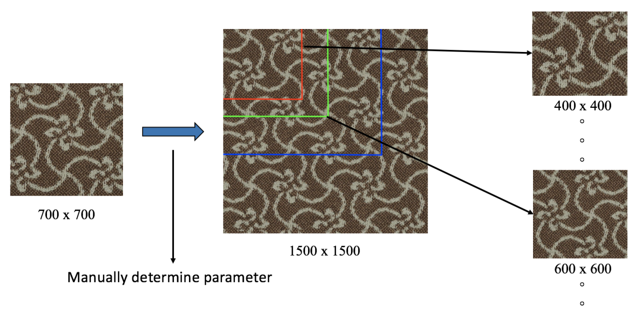

Figure 3. *An example of generating training samples with different scales from the same synthesized texture*

Training Sample Selection

Due to the large number of training samples, it is difficult to manually select a suitable parameter for each sample that can produce good synthesis result. In this work, we proposed a texture image labeling pipeline which can automatically assign each training sample a patch size that can produce a satisfactory result using image quilting algorithm. In addition, since not every training sample can achieve a good result, we need to remove noisy training samples. The following steps describe our method to label a training sample:

1. Given an input texture, assign it with an initial patch size
2. Use the initial patch size parameter to generate a synthesized texture using the image quilting method

3. Compare the synthesized result with the ground truth generated from training samples based on structural similarity index (SSIM) [7]
4. If the value is larger than a threshold, which means the synthesized result is good enough, we will use this patch size as the label of the input training sample. If not, we will increase the patch size and go back to step 2
5. If the patch size reaches 1, which means this input texture cannot achieve a good result, we will remove it from our training sample

Figure 4 shows the flow chart of proposed training sample labeling and removal process. We set the starting patch size to 0.1 and the resolution for each iteration is 0.01. The value of SSIM is between -1 to 1 (low similarity to high similarity), we set different threshold corresponding to different texture types. For stochastic and regular, we set threshold to 0.75. For irregular texture, we set it to 0.5. These thresholds are determined heuristically based on experimental evaluations which showed good overall performance.

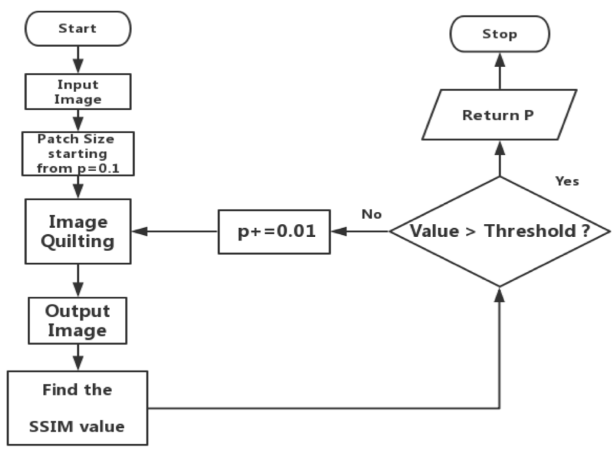

Figure 4. *Overview of proposed method for training sample labeling and selection.*

Proposed Method

Our method is inspired by the structure of the auto-encoder [8]. The encoder part of our system consists of a convolutional neural network, which is used to extract the features of the input texture. We manually assign each training sample an optimal parameter (ground truth), which is obtained by evaluating the SSIM between synthesized result and original texture for a set of different parameters. The output of the encoder is a set of synthesis parameters that the CNN predicts. The decoder then uses the predicted parameters to synthesize the input texture sample usingn the image quilting method. The synthesized textures are then evaluated using SSIM. Figure 5 shows the proposed network structure.

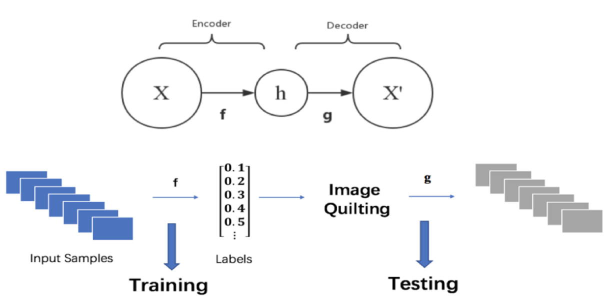

Figure 5. *Proposed network structure.*

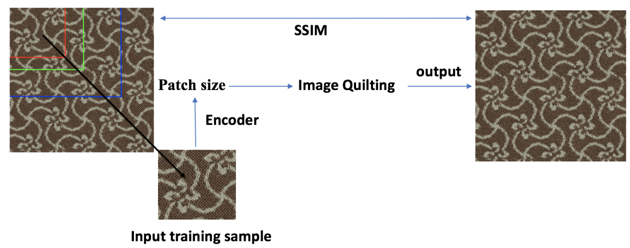

Figure 6. *Illustration of how decoder part of the network works*

Encoder

By encoder we mean the convertion between the input information in one format to another and the new format can represent the specific characteristic of the input. The input to our network is the image data, the output is the best parameter for texture synthesis given the input texture sample. We used VGG-16 as our network and modified the last fully connected layer to output the texture synthesis parameter. There are totally of 10000 texture samples from 20 different textures including all three texture classes. We used 75% for training, 25% for testing. We used Mean Square Error (MSE) as the loss function, defined as

$$Loss = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2 \qquad (1)$$

where $\hat{Y}_i$ is the estimator of CNN and Y_i is the true label. We apply Stochastic Gradient Descent (SGD) optimizer to minimize this loss function.

Decoder

The decoder is used to evaluate whether our network can learn properly the mapping between input texture sample and parameter for texture synthesis as shown in Figure 6. The output of the encoder is a patch size between 0 and 1, which we use as the parameter for image quilting method. We compare the syntheized result with the ground truth generated from training samples by using SSIM.

Experimental Result

Our method is evaluated on real life stochastic, regular and irregular textures. The patch size of the given input texture sample is automatically determined by the encoder-decoder network. Our method works well for most regular and stochastic textures. Figure 7, 8, 9 show example results of the learned texture synthesis parameters and the synthesized texture for stochastic, regular and irregular textures, respectively.

To better understand how our method improves the synthesis result, we compare the learned synthesis parameter with manually determined ground truth parameter range. This means the larger the ground truth range, the easier it is for an untrained user to achieve satisfactory result since the choice of synthesis parameter is less restricted. As shown in Figure 7, the range for stochastic textures is large, so it is easy to get a good synthesized result even without understanding the parameter. However, for regular texture shown in Figure 8, which has a strong pattern, the good parameter range is small. Furthermore, the parameter range varies a lot due to different pattern scales as shown in Figure 2, which is challenging for an untrained user to achieve good synthesis result. For irregular textures, as shown in Figure 9, the range is small. Although our network can predict suitable parameter for texture synthesis, there are still some visual artifacts for this texture sample since the input texture sample has limited information.

Conclusion and Future Work

We proposed a fully automatic texture synthesis method, which can be applied to most commonly found textures in real life applications. We designed an encoder-decoder network that can extract feature of the input texture sample and output the most suitable texture synthesis parameters. We also developed a method to automatically select and label training samples. We showed that our methods works well for stochastic and regular textures by removing user's interaction. As a result, our method improved the efficiency of the image quilting method to enable real life applications without compromising texture synthesis accuracy. However, due to the lack of complete information available for irregular texture samples, we need to explore other synthesis methods that can generate more realistic results for irregular texture samples.

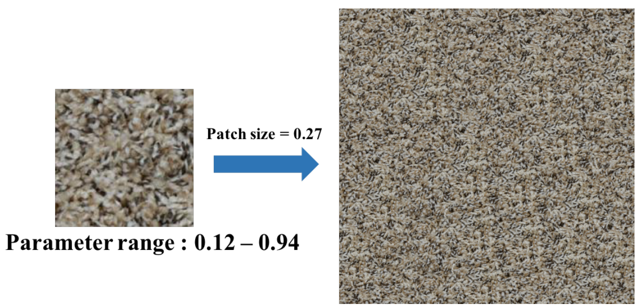

Figure 7. An example result of synthesizing stochastic texture using proposed method.

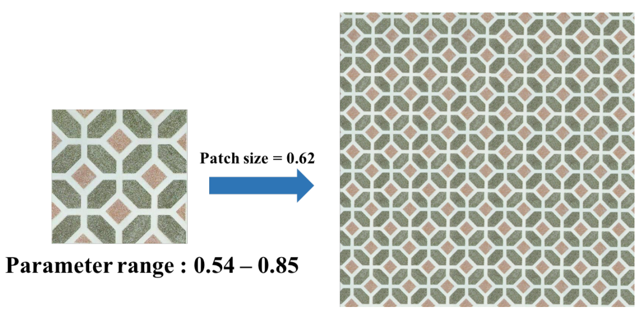

Figure 8. An example result of synthesizing regular texture using proposed method.

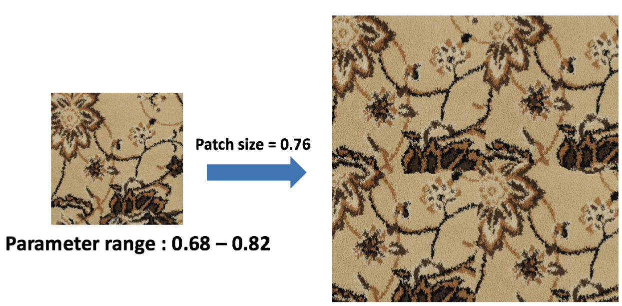

Figure 9. An example result of synthesizing irregular texture using proposed method.

References

[1] A. A. Efros and W. T. Freeman, "Image quilting for texture synthesis and transfer," *ACM SIGGRAPH*, vol. 28, pp. 341–346, August 2001.

[2] K. Ziga, J. Bagchi, J. Allebach, and F. Zhu, "Non-parametric texture synthesis using texture classification," *Electronic Imaging 2017*, vol. 17, pp. 136–141, Feb 2017.

[3] A. A. Efros and T. K. Leung, "Texture synthesis by non-parametric sampling," *The Proceeding of the Seventh IEEE International Conference on Computer Vision*, pp. 1033–1038, September 1999.

[4] L.-Y. Wei and M. Levoy, "Fast texture synthesis using tree-structured vector quantization," *Proceedings of the Conference on Computer Graphics and Interactive Techniques*, vol. 28, pp. 479–488, July 2000.

[5] L. Liang, C. Liu, Y.-Q. Xu, B. Guo, and H.-Y. Shum, "Real-time texture synthesis by patch-based sampling," *ACM Transactions on Graphics*, vol. 20, pp. 127–150, July 2001.

[6] V. Kwatra, A. Schodl, I. Essa, G. Turk, and A. Bobick, "Graphcut textures: Image and video synthesis using graph cuts," *ACM Transactions on Graphics*, vol. 23, pp. 277–286, July 2003.

[7] Z. Wang, A. Bovik, and H. Sheikh, "Image quality assessment: From error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, April 2004.

[8] G. Hinton and R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, pp. 504–507, July 2006.

Author Biography

Jiangpeng He received his BS in Electronic Information Engineering from University of Electronic Science and Technology of China (UESTC) in 2013. He is now working on his PhD at Purdue University West Lafayette in the field of Image Processing.

Kyle Ziga received his BS in Computer Engineering from the Calumet campus of Purdue University in 2015, and MS in Electrical and Computer Engineering from Purdue University, West Lafayette, in 2018. He is now a systems engineer at Raytheon.

Judy Bagchi is founder and CEO of Dzine Steps, a cloud software provider to home builders and independent design centers nationwide. Prior to this Judy held Research & Development Management roles at Hewlett-Packard and Nortel. She is an industry veteran with more than 20 years of experience in the entire business value chain. She has a keen interest in and has *led and participated in various activities supporting Women in Technology.*

Fengqing Zhu is an Assistant Professor of Electrical and Computer Engineering at Purdue University, West Lafayette, IN. Dr. Zhu received her Ph.D. in Electrical and Computer Engineering from Purdue University in 2011. Prior to joining Purdue in 2015, she was a Staff Researcher at Huawei Technologies (USA), where she received a Huawei Certification of Recognition for Core Technology Contribution in 2012. Her research interests include Image processing and analysis, video compression, computer vision and computational photography.