Implementation of Image Quilting for Texture Synthesis and Transfer

Louis Schlessinger
lschlessinger@wustl.edu

Abstract

This project considers the implementation of image quilting for texture synthesis and transfer. The focus, however, is on the image quilting algorithm itself. The influence of texture synthesis parameters and input texture types, as well as run time, on the qualitative and quantitative performance of the image quilting algorithm are analyzed. This report also looks at the texture transfer results. It was found that on certain types of structured textures, satisfying results can be obtained after much hand-tuning of the block size parameter.

1 Introduction

The texture synthesis problem consists in creating a synthesized image larger than the given input texture. Rather than recreate an entire physical image, generating a new view of that image can be very useful in various domains such as image editing, computer graphics and post-production of films. Additionally, the texture transfer problem consists in placing a small input texture on a given a target image. This is also very useful as it allows for new images to be synthesized by cutting and pasting a source texture on any given object, which may also be used in the same domains. The method analyzed in this paper is an older one from 2001, but still gave impressive results [2].

2 Background & Related Work

The naivest way of solving the texture synthesis problem is using tiling, which is just copying the entire original texture to a desired size. This does not give satisfying results unless the texture is perfectly symmetric. However, the first approach was to match the statistics of the input texture, which was done by [4]. This stochastic texture synthesis method worked very well for stochastic textures, but not other types of textures. One reason for this is that the statistics did not capture relationships well over different possible orientations and scales. Following this stochastic approach, many pixel-based synthesis methods were

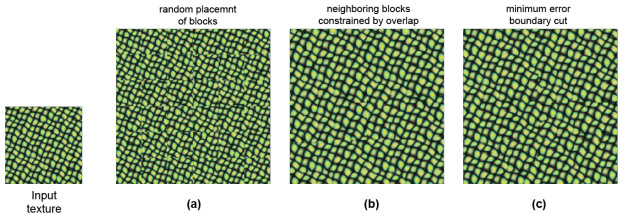

Figure 1: Quilting texture. This shows the reproduced results of [2]. Blocks are sampled from the input texture and joined together to create a larger target image. (a) shows a naive random sampling approach. (b) shows an improved sampling method taking the neighborhood overlap into account. (c) has the same sampling as (b), but takes the min-cut of that overlapping boundary.

developed including [1], using non-parametric sampling. They are very simple and easy to implement algorithms that enforce local statistics, which have given somewhat satisfying results. However, their run times are very poor and may search for pixel values that could have already been predicted. Following this, the patch-based texture synthesis algorithms are explored to overcome those difficulties, including [2].

More recently, deep learning approaches have proven to be a very effective way of generating texture. [3] showed that this could be done using convolutional neural networks. After that, other approaches within deep learning such as using generative models for texture synthesis have shown themselves to be very successful. The Spatial GAN [5] demonstrated how GANs could be used for texture synthesis.

3 Proposed Approach

In this section I will describe the patch-based texture synthesis algorithm, as well as the modification of this algorithm to include texture transfer. In the Image Quilting section, three patch-based texture synthesis approaches, including random placement of blocks, neighboring blocks constrained by overlap, and minimum error

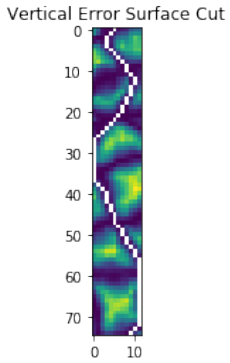

Figure 2: Vertical minimum boundary cut of the error surface in the image quilting algorithm.

boundary cut will be described. After that, the actual texture synthesis algorithm will be given. In the last section, the texture transfer method will be explained.

3.1 Image Quilting

The simplest and naivest way of performing patch-based image synthesis is to randomly select patches from the original input texture and place them onto a target image of a larger size. We will define the set of all possible overlapping patches to be S_B and a user-specified patch size to be B. Each patch will be of size $B \times B$. This can be done by looping from left to right, top to bottom (rendering order) and placing randomly sampled blocks from S_B next to each other. This yields unsatisfying results as seen in Figure 1(a).

An improved method is to consider the overlap of neighboring blocks. In rendering order, we compute the patches in S_B that minimize the overlap error, which is defined to be the L2-norm of the overlapping regions. A patch within some error tolerance ([2] uses a tolerance of 0.1) of the overlap error is randomly selected. The top left patch is initialized randomly. Figure 1(b) shows how this improved the synthetic texture results. The new block is placed partially overlayed on the previous one(s) and the cut to choose which points belong in the new block and those previously placed are the exact middle (half of the overlap size).

The final approach that is proposed is to relax the constraint of the middle edge cuts and allow for the cut between two patches to be not straight. This allows for a better approximation of the edges in the texture. Before the actual placement of the patch, an error surface is computed, which is defined to be the squared difference in the overlap region. A minimum cost path is found using dynamic programming. This defines the new boundary cut. The results of this can be seen in Figure 1(c). The minimum cut of a vertical surface error can be seen in Figure 2.

3.2 Texture Transfer

The image quilting algorithm can be naturally extended to the task of texture transfer. Given an input texture and a target image, the input texture can be transferred to the target image using correspondence maps. The simple modification is to change the way that patches are selected by including another term in the error function. The block selection cost is now a weighted sum of α times the original block overlap error and $1-\alpha$ times the correspondence error, which is a squared difference between the correspondence map $\vec{C}$ of the patch in the current target and all possible texture correspondences. These can be defined to be the image intensities.

4 Experimental Results

The results of the texture synthesis process can be seen in Figure 3. In my analysis of the image quilting algorithm, I looked at the block size, the type of input texture, the error tolerance, the size of the overlap, and the run time. The user-specified parameter block size greatly influenced the quality of the results. This can be seen in Figure 5. The block size could range from 1 pixel to the minimum of the height and width of the target. The authors in the original paper did not report these sizes, thus making it very difficult to reproduce the exact results and required much tuning. I found that the size of the block must be large enough so that it captured the structures in the texture adequately. It also seemed that because the blocks were square, the blocks had trouble with very asymmetric features and those that had nonlocal dependencies. For most textures that was not an issue though.

The type of input texture also greatly affected the results. The method performed very well on the easier type of both very stochastic and very regular textures. However, even a random patch-based algorithm would likely have done not much worse. The improvement in results came in the near-irregular and near-stochastic regime. These textures are slightly harder, but the image quilting algorithm gave very satisfying results provided the right block size. The algorithm had some unnatural looking synthesized images in the irregular part of the texture spectrum.

The authors of the original paper chose an error tolerance of 0.1 when choosing the next patch. I experimented with this parameter and found that around that value, it did not significantly affect the results depending on the texture. It seemed as though decreasing the tolerance gave higher fidelity images but gave fewer novel types of results.

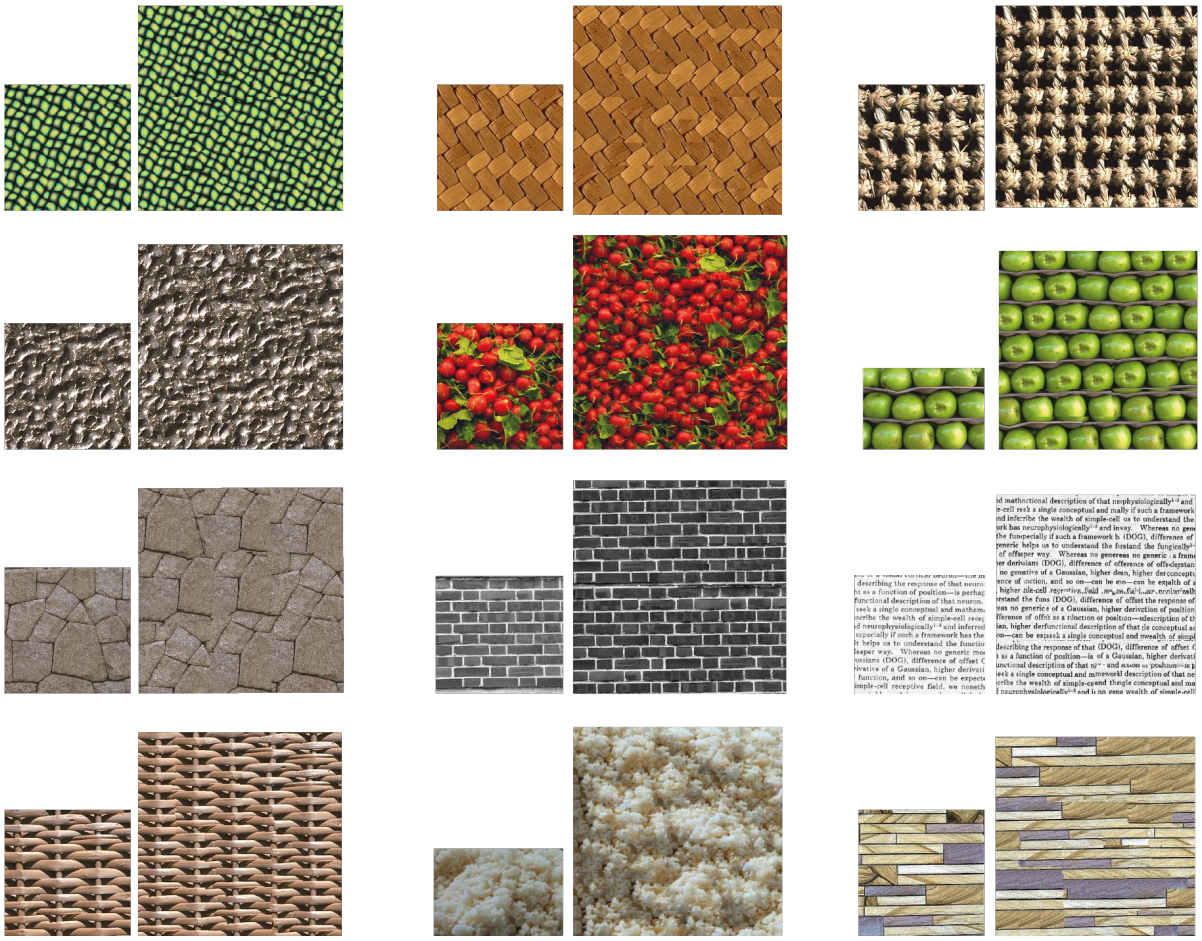

Figure 3: Image quilting synthesis results. For each pair of images, the left is the original one, and the right is the synthesized one.

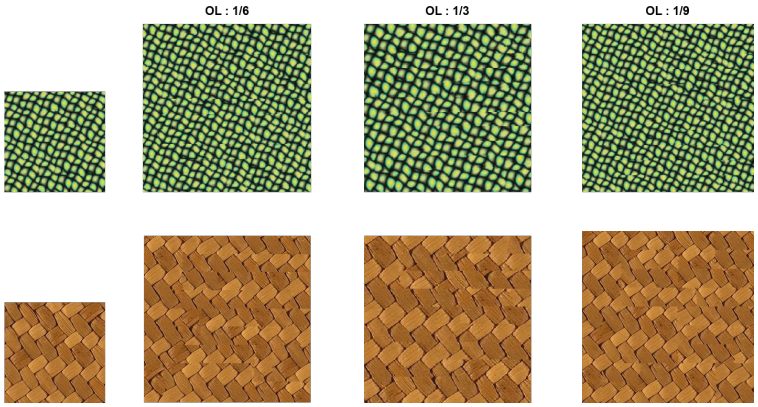

Figure 4: Image quilting synthesis results for varying overlap sizes. The far-left is the input texture, and those to the right are the synthesized ones. OL is the fraction of the block size to consider. The authors in the original paper used $1/6$.

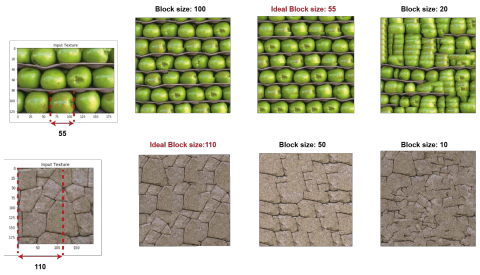

Figure 5: Texture synthesis varying block sizes. The far left image is the original texture and the right ones are synthesized.

Figure 6: Texture transfer result

The overlap size was also suggested to be $1/6$ of the block size. Some results can be seen in Figure 4. I found that, as expected, decreasing the overlap size made the results much worse. I also found that in some images, slightly decreasing or increasing the overlap size improved results. This additional parameter seemed to be well chosen for most types of structures, at least for a starting point.

Lastly, I found that the only method that significantly impacted the run time was the block size. As was to be expected, I found a quadratic dependence on the block size. None of the other parameters seemed to greatly influence the run time, all of which were around 20s - 100s on my machine.

Some results of the texture transfer method can be seen in Figure 6. Although this was not the aim of this report, I found that the results of the original paper hard to reproduce.

5 Conclusion

This paper gave an analysis of the image quilting algorithm and its quantitative and qualitative performance. This patch-based synthesis algorithm works remarkably well but required quite a bit of hand-tuning of parameters to get visually-pleasing results. Even after implementing it, which was not trivial to implement as the authors stated, the results seemed to be slightly worse than those of the authors. Given all of that, it was still a very impressive algorithm on certain types of textures.

Acknowledgments

I would like to acknowledge [2] for providing textures and images.

References

[1] A. Efros and T. Leung. Texture synthesis by non-parametric sampling. 1999.

[2] A. A. Efros and W. T. Freeman. Image quilting for texture synthesis and transfer. 2001.

[3] L. A. Gatys, A. S. Ecker, and M. Bethge. Texture synthesis using convolutional neural networks. 2015.

[4] D. J. Heeger and J. R. Bergen. Pyramid-based texture analysis/synthesis. 1995.

[5] J. N, B. U., and V. R. Texture synthesis with spatial generative adversarial networks. 2017.