Internal Report 2012-2013-09 June 2013

Universiteit Leiden
Opleiding Informatica

Evaluation

of

Image Quilting algorithms

Pepijn van Heiningen

BACHELOR THESIS

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

Evaluation of Image Quilting algorithms

Pepijn van Heiningen

June 28, 2013

Abstract

Image Quilting is a method for synthesizing new images by stitching patches of an existing image together. It is a very fast and simple algorithm, mainly used for the generation of new textures. Even though it produces impressive results, there are some problems. In this paper we analyse these problems and try to improve the original algorithm, without changing the simplicity of the algorithm or the time it takes to run. These improvements are then compared with the original to see how well they work.

Keywords: Image Quilting, Texture Synthesis

1 Introduction

The first thing that comes to mind when generating a texture is tiling the input image. This leads to a repetition of the texture which does not seem natural to the user. One of the advantages of Image Quilting is that this problem does not occur, because the generated texture will be perceived as exactly the same as the original, but without repetition.

The next thing one might think of is generating textures by using pixel-by-pixel synthesis. (see Efros and Leung [3] and Ashikhmin [10]). This produces decent results, but to generate a single pixel the entire input image has to be searched, or it only works well for stochastic images.

Another way of synthesizing textures was described by Xu et al [4]. First the image was tiled with the original texture, subsequently random blocks were taken from the image and placed at random positions in the generated texture with alpha blending. This method can be used for stochastic textures, but it fails for more structered ones.

Work was done by Wu et al. [9] trying to improve texture synthesis by using Curvilinear Feature Matching, which matches features from the input image instead of selecting by color.

Other work was done using the strong Markov Random Field model by Paget [7], which produced decent results for stochastic textures.

Even neural networks can be used to synthesize textures as shown by Slot, Kornatowski and Debiec [5]. There, random blocks were selected from an input image with space between them, which is filled in by a Cellular Neural Network.

A good overview of the state of example-based image synthesis can be found in [8]

Image Quilting was first described by Efros and Freeman [1]. It takes small patches from an existing texture and stitches them together to generate new textures. The patches are overlapped with eachother and a random block within a certain score is chosen.

Image Quilting is useful for a variety of different applications. Instead of recreating an entire world (or its textures), a sample from the real world can now be used and a larger texture can be generated. This could decrease the bandwidth of online applications, where only the original image would be sent, and then generated.

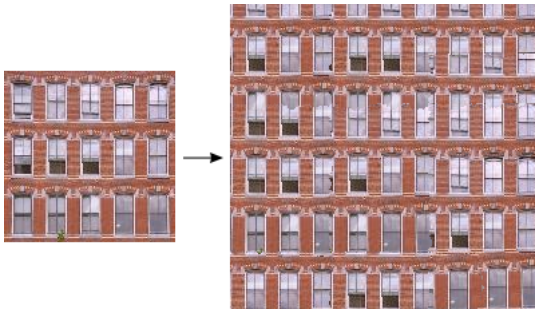

Figure 1: Image Quilting example

2 Original Algorithm

The Image Quilting algorithm can be described as follows:

- First a random patch is chosen from the input texture. For the next blocks, we introduce an overlap to calculate how well they fit together.
- We search the input image, and randomly choose a block within some error tolerance.
- Finally we calculate the minimum cost path through the overlap (with for example Dijkstra's algorithm [2]) and paste the new block onto the generated texture. Repeat this until you are done.

Graph cuts can also be used to find a minumum cost path through the overlap, as shown by Kwatra et al. [6]

The size of a block is controlled by the user, the error in the overlap of blocks is calculated by using the $L2$ norm on pixel values. An example is shown in Figure 1 .

This algorithm can be extended for use as a texture transfer algorithm. Now, instead of only looking at the overlap we also compare the current block with the block at the target image. The error is then calculated as the weighted sum of the overlap error and the error between the source block and target block. The weight is controlled by a parameter α. To further improve results, we iterate over the image several times, decreasing the block size by a third each time. The number of iterations N is set between $N = 3$ and $N = 5$, and α is set to $\alpha_i = 0.8 \cdot \frac{i-1}{N-1} + 0.1$. A result can be seen in figure 2.

In the original paper describing Image Quilting some of the drawbacks of the algorithm are mentioned, namely excessive repetition and mismatched boundaries.

3 Improvements

The original algorithm selects blocks randomly from within a certain error range of the best block (the block that has the minimum error in overlap). Because of this, the error between the overlapping surfaces will not always be minimal.

As an improvement, one might suggest to pick the best block and as a result reduce the error

Figure 2: Texture Transfer example

in the overlap. A drawback of this approach is the fact that copying occurs; multiple blocks from the input image are selected together because the error between them is 0. This will lead to the algorithm first copying the whole, or a part of the original image, and only then start picking blocks in the way it was designed.

Another disadvantage is the fact that if the best block is chosen, more blocks will be repeated. These problems can be fixed by keeping track of which blocks were already used, and applying a penalty for using them again. To determine whether a block has already been used, we keep a list of all the top left pixels of blocks that were already synthesized, and if the current block is within a certain range from a used block (we call this the block reuse range), a penalty is given.

Because the number of blocks used ranges from 25 to about 250 there is no noticeable difference in the time used to run the algorithm.

Block reuse range	Max. error	Avg. Maximum Error	Avg. Error	Avg. number of repetitions
2	61593	59048	31555	1.56
4	62189	59538	32999	2.44
8	63035	60965	33815	3.64
16	93022	80730	37767	6.96

Table 1: Modified Algorithm: Block reuse range

4 Parameter Analysis

The goal is of course to generate images with these algorithms that are indistinguishable from real images. It is however very hard to determine whether a certain image looks good to a user or not. Because of this finding a measurement to compare the images that were generated by the algorithms is also difficult.

Since the block size is the only variable that is changeable by the user, the range and the penalty for blocks have to be fixed. In order to get optimal results from the improved algorithm, I conducted a parameter analysis.

First I tested the range for reusing blocks. The changed texture synthesis algorithm was run 25 different times, and the results were averaged. In table 1 you can see the results. The maximum error, average maximum error, average error, and the number of repetitions all increase if you increase the block reuse range. A block reuse range of 2 clearly gives the best results, but we have to take into consideration that the average number of repetitions is not the same as the amount of repeated elements in the generated image.

A block reuse range of 2 is still used for the rest of this paper. This is done because the synthesized images that mainly show repetition are the ones with small blocksizes.

Subsequently I tested the penalty values. The algorithm was run 25 times over 5 different images. You can see the results in table 2. You can see the average error is very close, it almost looks independant of the penalty. The number of repetitions does show a pretty clear image however. You can see that the average repetitions slowly decreases. This is most likely due to the fact that with small errors, blocks are more likely to be repeated. A higher error makes sure that block will not come back again. In the next section we will see whether the modified algorithm performs better with these settings than the original algorithm.

Penalty	Avg. Max. Error	Avg. Error	Avg. rep.
1.05	59448	31819	1.968
1.1	57967	31358	1.912
1.2	59018	31483	1.848
1.3	58470	31476	1.872
1.4	58990	31465	1.8
1.5	58408	31476	1.824
1.6	58750	31459	1.928
1.7	57398	31083	1.848
1.8	57823	31004	1.768
1.9	58798	30916	1.76
2.0	58703	31360	1.832

Table 2: Modified Algorithm: Penalty table

5 Results

To compare the original and the modified algorithm, I decided to conduct two different tests:

- A comparison of the maximum and average error that occurred
- A survey was held to compare the images.

As one of the measurements I propose the usage of the average error between the overlapping surfaces. If one part of the overlap is in a completely different shade than the other the algorithm will not produce good-looking results, seeing as both blocks were taken from the same image and will not fit well together.

Also the maximum error is one of the criteria used to compare images. Even one block that does not fit well will convince a viewer that the image he/she is looking at was generated by an algorithm.

Since the error values are the criteria the algorithm uses, they are also used for the comparison of images. Because the average error might not be the best way to test algorithms, we decided to also hold a survey under 26 participants.

Image	Max. error	Avg. Maximum Error	Avg. Error	Avg. number of repetitions	Block size
1	63302	59276	32313	1.648	40
2	19662	15547	7878	4.348	25
3	62253	56634	32336	1.852	40
4	15757	10882	2961	4.504	20
5	115966	88749	52118	2.984	50
Average:	55388	46217,6	25521,2		

Table 3: Modified Algorithm: Texture Synthesis Results

Image	Max. error	Avg. Maximum Error	Avg. Error	Block size
1	62885	59166	33609	40
2	18079	15355	7718	25
3	74327	68931	41689	40
4	15938	10623	2872	20
5	117629	88434	51277	50
Average:	57771,6	48501,8	27433	

Table 4: Original Algorithm: Texture Synthesis Results

5.1 Comparison of texture synthesis

We picked five different images from the testset to test our algorithm on. Some of them suffered from the problems mentioned in the original paper, others are the ones that performed best.

First both algorithms were run on these 5 images and the best block size was determined. The same block size is used for each image in both algorithms. For the modified algorithm we used the parameter settings described in the previous section. The original algorithm has a error-range of 1.1. A block is randomly selected from the blocks with errors between the minimal error and 1.1 times the minimal error.

Then both algorithms were run 250 times per image to compare the maximum, average maximum and average error. In the results of the modified algorithm (Table 3) you can find the average number of repetitions in the range.

The results from the two algorithms are extremely close. On some images the original algorithm works better, on others the modified algorithm outperforms the original. Only on the third image (see Appendix) we see a big decrease of the average error. If we look at the average of all runs, we see a average error decrease of 7%. This is entirely because of the third image.

5.2 Comparison of texture transfer

For texture transfer, only one image was chosen as comparison image. This is due to the fact that running the texture transfer part of the algorithm takes a lot longer than the synthesis. It takes roughly 2 hours for an image to be generated. Both algorithms were run 5 times, and the results were averaged. The algorithm was started with a blocksize that was a sixth of the original image's width and height. Then the blocksize was decreased by a third as described in the original algorithm. I found that increasing the overlap from a sixth of the blockwidth and height to a fourth improved results.

If we compare the results, we see a big advantage for the original algorithm in both the average and maximum error. This is probably because the top left part of the target image is almost the same color of grey, which causes the algorithm to try to find the best solution, and selects blocks with a penalty.

Looking at the results from the texture transfer, you can see that more runs would likely not have changed the result, since these runs are very close together. Since there is no random factor in the modified algorithm, we see results that are incredibly close together. In the original algorithm we see only a slight variation in the maximum and average errors, due to the random picking of blocks.

Run	Max. error	Avg. Error	No. of rep.
1	24733818	2379230	12
2	24733818	2379234	12
3	24733818	2379239	12
4	24733818	2379231	12
5	24733818	2379220	12
Average	24733818	2379231	12

Table 5: Modified Algorithm: Texture Transfer Results

Run	Max. error	Avg. Error
1	18979220	1721931
2	18977494	1721980
3	18978696	1721943
4	18978168	1721928
5	18977980	1721966
Average	18978312	1721950

Table 6: Original Algorithm: Texture Transfer Results

5.3 Survey

For the survey 10 different images were chosen and generated with equal blocksizes. (see Appendix) These were subsequently added to a survey where people could pick the image they preferred. The order of the images was mixed, to balance the results. The final survey was filled in by 25 people. The results are in table 7.

5.3.1 Texture Synthesis

The results from the survey agree with the results from the average and maximum error. There also seems to be little difference if humans compare the images. The output of the original algorithm is picked as many times as the output of the modified algorithm.

Image	% Modified	% Original	Block Size
1	68,0	32,0	40
2	40,0	60,0	25
3	60,0	40,0	40
4	48,0	52,0	20
5	44,0	56,0	50
6	40,0	60,0	15
7	56,0	44,0	50
8	68,0	32,0	37
9	41,7	58,3	50
10	69.6	30,4	65
Average	53,53	46,47	

Table 7: Texture Synthesis: Survey results

5.3.2 Texture Transfer

For comparing the Texture Transfer part of the algorithm, 3 different images were used (see the Appendix). You can see the results in table 8.

Now we see completely different results compared with the average and maximum error. Looking at the first image, the average error was 30% higher, but 79% of respondents preferred the modified algorithm over the original. On the third image all participants of the survey selected output of the modified algorithm as the best one.

Image	% Modified	% Original
1	79,2	20.8
2	91,7	8.3
3	100	0

Table 8: Texture Transfer: Survey results

6 Conclusion

If we look at texture synthesis, selecting the best block produces similar results compared to selecting a random block within a margin. Looking at texture transfer there is a big difference in the average error between the modified and orignal algorithm, but the survey suggests the opposite. Since the photorealisticness of the image is the measure for a good image, we can conclude that selecting the best block works better for texture transfer. This is most likely to

7 Future Work

I feel that, even though the original algorithm produces some impressive results, there are still improvements to be made. The other drawback of the algorithm is mismatched boundaries. If that can be dealt with it would be a great improvement over the original. Also the block reuse range is now a static variable, independant of the block size. If a small block-size is selected, the boundaries for repetition of blocks will be much stricter than with larger blocks. A block reuse range depending on the block size could be a topic of further research.

References

[1] A. A. Efros and W. T. Freeman. Image Quilting for Texture Synthesis and Transfer. In SIGGRAPH ′01, 2001.

[2] E. W. Dijkstra. A note on two problems in connexion with graphs. In Numerische Mathematik 1: 269271, 1959.

[3] A. A. Efros and T. K. Leung. Texture synthesis by non-parametric sampling. In International Conference on Computer Vision: 1033–1038, 1999.

[4] Y. Xu, B. Guo and H. Y. Shum. Chaos mosaic: Fast and memory efficient texture synthesis. Technical Report MSR-TR-2000-32, 2000.

[5] K. Slot, L. Kornatowski, Piotr Debiec. Fast texture synthesis with cellular neural network-based patch stitching. In International Journal of Circuit Theory and Applications, 2012.

[6] V. Kwatra et al. Graphcut Textures: Image and Video Synthesis Using Graph Cuts. In SIGGRAPH ′03, 2003.

[7] R. Paget. Strong markov random eld model. In IEEE Transactions on Pattern Analysis and Machine Intelligence (Volume: 26, Issue 3), 2004. 408413.

[8] L. Wei et al. State of the Art in Example-based Texture Synthesis. In Eurographics, 2009.

[9] Q. Wu, Y. Yu. Feature matching and deformation for texture synthesis: 364–367. In SIGGRAPH ′04, 2004.

[10] M. Ashikhmin. Synthesizing natural textures. In ACM Symposium on Interactive 3D Graphics: 217–226, 2001.

Appendix

The ordering of the images is as follows: on odd numbers the left image is generated by the modified algorithm, on even numbers the right.

Figure 3: Image 1

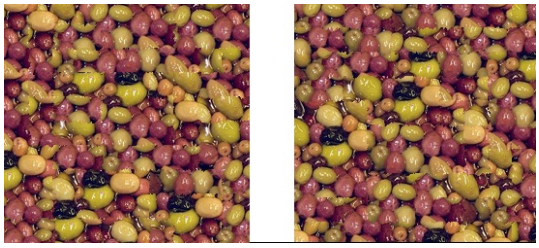

Figure 4: Image 2

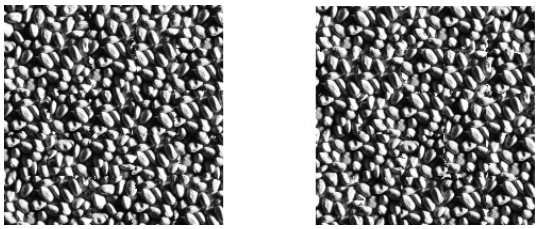

Figure 5: Image 3

Figure 6: Image 4

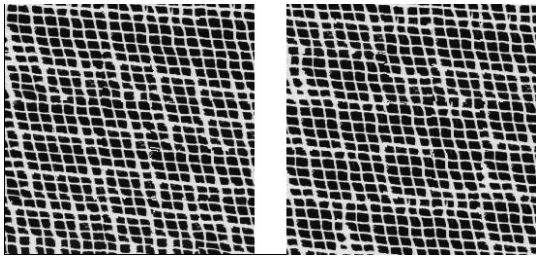

Figure 7: Image 5

Figure 8: Image 6

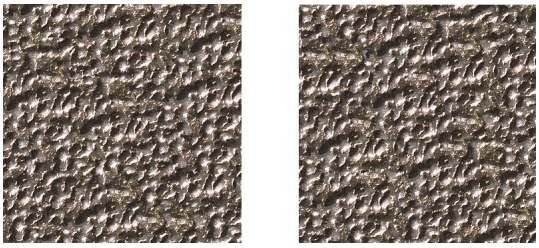

Figure 9: Image 7

Figure 10: Image 8

Figure 11: Image 9

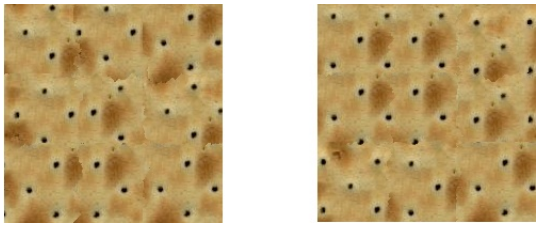

Figure 12: Image 10

For Texture Transfer the left image is generated by the modified algorithm, the right by the original.

Figure 13: Image 10

Figure 14: Image 10

Figure 15: Image 10