An Improved Texture Synthesis Algorithm Using Morphological Processing with Image Analogy

Jiang Ni Henry Schneiderman

CMU-RI-TR-04-52

October 2004

Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

Abstract

Texture synthesis is an important technology for graphics and animation. This paper proposes a novel method based upon the multi-resolution neighborhood matching technique pioneered by Wei and Levoy. Wei and Levoy's simple but powerful method is effective at synthesizing a wide range of textures. However, it has a tendency to produce synthesis artifacts in textures containing distinctive structural elements with large degrees of irregularity, such as stones in a stone wall. We propose several improvements designed to overcome these artifacts: (1) Morphological operations to emphasize key aspects of structure such as edges followed by image analogy to undo the effects of morphology (2) Combining non-causal neighborhoods with causal neighborhoods (3) Appropriate weighting of the parent and current level in the multi-resolution pyramid. Experimental results demonstrate that these modifications improve the quality of synthesis for textures containing irregularly structured components.

1 Introduction

In recent years, texture synthesis has become an active research area in computer vision and computer graphics. The idea of texture synthesis was closely related to the idea of image-based rendering, which synthesizes novel views of the world using the sample images captured from the real world, instead of recreating the entire physical world from scratch. In texture synthesis, given a sample texture image, the algorithms will use the information extracted from this input to synthesize new images that belong to the same texture category but do not look identical to the input image. Such techniques could be extended to a lot of interesting applications such as texture mapping, texture transfer, occlusion fill-in, etc., which are useful in entertainment industries such as movies or video games. Also, texture synthesis is a good way to verify texture analysis methods, which aim to understand the fundamental mechanisms of textures and could be used for texture classification, texture segmentation, etc.

Texture synthesis is often formulated using a statistical model. Some methods use a parametric texture model. These include Cross and Jain's MRF model [2], Heeger and Bergen's pyramid-based model [7], Zhu's FRAME model [13], Portilla and Simoncelli's parametric model based on joint statistics of wavelet coefficients [11], etc. Other methods use non-parametric models to resample the input texture image one pixel at a time to get an output image. These include De Bonet's Laplacian pyramid sampling [3], Efros and Leung's single-scale sampling [5], Wei and Levoy's multi-resolution approach [12], etc. A related group of resampling methods sample patches instead of individual pixels. These include Xu et al.'s mosaic [6], Liang et al.'s patch-based sampling [10], Efros and Freeman's image quilting [4], Kwatra et al.'s graph-cut approach [9], etc. Ashikhmin[1] presented a modified patch-based version of Wei and Levoy's algorithm which takes into account the relative position of the pixel and its neighborhood pixels.

Patch-based resampling methods, especially the graph-cut approach [9], generally seem more effective in terms of their empirical performance. By copying entire patches, many edges and corners are preserved producing a result that is visually appealing. One potential downside of this approach is that it fails to obey various constraints that are implicit in the texture, such as preserving the convexity of texture elements. Another problem can be a discontinuous seam between patches. Also, such synthesized textures can often appear too similar to the original input texture. Pixel-based resampling techniques, on the other hand, have the potential to generate a wider range of textures and naturally avoid seam artifacts.

Wei and Levoy [12] published their pixel-based non-parametric texture synthesis algorithm (the WL algorithm) in 2000. What makes it attractive is that they combine multi-resolution pyramids with non-parametric neighborhood matching. Their simple but powerful model can synthesize a wide range of textures successfully. However, the algorithm has a tendency to produce synthesis artifacts in textures containing distinctive structural elements with large degrees of irregularity, such as stones in a stone wall. We propose several modifications designed to overcome these artifacts, and present improved empirical results on a range of example textures.

In Section 2, we describe the WL algorithm. In Section 3, we analyze some limitations of the algorithm. In Section 4, we introduce our modifications to that algorithm. Section 5 gives some example empirical results and Section 6 gives a summary of the overall approach described in this paper.

2 Wei and Levoy's Texture Synthesis Algorithm

The WL algorithm performs texture synthesis in a multi-resolution fashion. It uses a Gaussian pyramid (Fig. 1) to decompose the image into several scales. Starting with random white noise, it synthesizes each level of Gaussian pyramid of the output image from coarse to fine. During the synthesis of each level, the value

of each pixel is determined by comparing its L-shaped causal neighborhood (Fig. 2a) with all the candidate neighborhoods in the training data. The input pixel with the most similar neighborhood using the sum of squared differences (SSD) distance metric, is assigned to the output pixel.

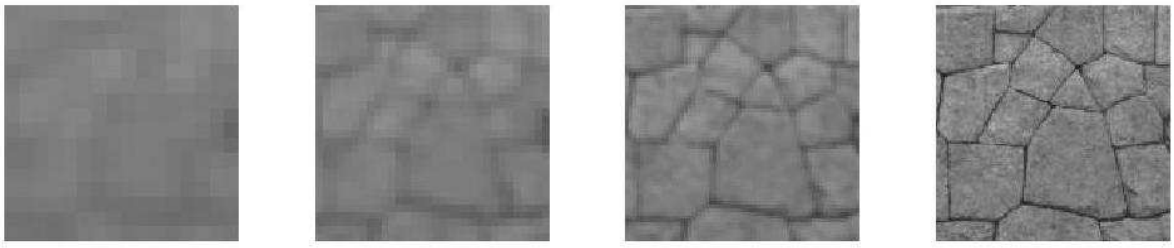
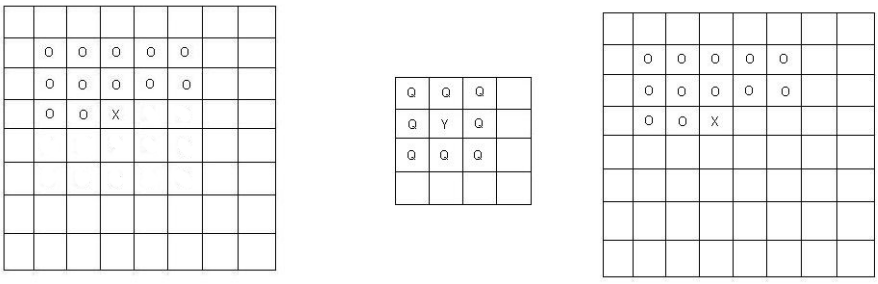


Figure 1: **A Gaussian pyramid. From left to right is the direction of from coarse to fine. Each level of the pyramid is resized so that different levels can be compared more easily.**

The algorithm involves the following steps:

1. The input image Ia is decomposed into a Gaussian pyramid Ga.
2. The output image Is is initialized as random white noise, and also decomposed into a Gaussian pyramid Gs.
3. Each level of Gs is processed with histogram equalization with the corresponding level of Ga.
4. For each level in the Ga and Gs, from coarse to fine:
   (a) Neighborhood definition
   At the coarsest level, define the neighborhood as the L-shaped causal neighborhood around the pixel (Fig. 2a).
   At any other level, define the neighborhood as the concatenated vector consisting of the square non-causal neighborhood of the parent level and the L-shaped causal neighborhood of the current level (Fig. 2b).
   (b) Training
   Collect all such neighborhoods for the current level of Ga, such that its full extent of the neighborhood is located within the image boundary.
   (c) Synthesis using nearest neighbor matching
   Assume the output image wraps around. For each pixel in the current level of the Gs, extract its neighborhood and compare it with all the candidates in the training set. From the training set, find the pixel whose neighborhood has the smallest SSD error with the query neighborhood, and put this input pixel at the current position inside Gs.
5. After all the levels of the pyramid are synthesized, the finest level is output as the synthesized image Is.

The WL algorithm implementation uses tree-structured vector quantization (TSVQ) to accelerate neighborhood matching. This only improves the speed of the algorithm, but does not help to improve the quality of the synthesis and may degrade it. Since our purpose is to improve the quality of the synthesis, we do not use TSVQ acceleration.



(a) **(b)**

Figure 2: **(a) A 5x5 causal neighborhood used in the coarsest level of the pyramid. (b) A 5x5 causal neighborhood from current level combined with a 3x3 non-causal neighborhood from the parent level. (In our experiment, we actually used 9x9 causal neighborhoods and 5x5 non-causal neighborhoods.)**

3 Analysis of the WL Algorithm

The WL algorithm is effective at synthesizing a wide range of textures. However, the algorithm has a tendency to produce synthesis artifacts in textures containing distinctive structural elements with large degrees of irregularity (See Fig. 7b). Below we examine several issues that seem to contribute to these artifacts.

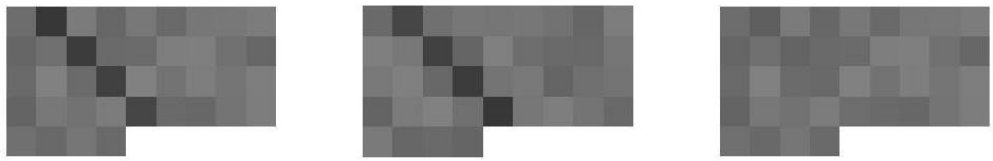
3.1 Nearest Neighbor Matching Using SSD

The WL algorithm uses a nearest neighbor (NN) method for neighborhood matching. As with all NN methods, it is guaranteed to perform well as the quantity of training data approaches infinity. The classification error can be no worse than twice the Bayes (optimal) rate, if we view neighborhood matching as a classification problem. However, due to limited training data, NN can be subject to over-fitting. Moreover, there is a known mismatch between the SSD criterion and human perception. It has long been observed in image compression that differences in image quality do not always correlate well with the SSD. In particular, visually important features, such as edges and corners, are under-emphasized by the SSD criterion. For a simple example, in Fig. 3a, among the 40 pixels in the neighborhood, the dark diagonal line given by 4 pixels is much more visually salient than the remaining 36 pixels. SSD sums up the squared differences of the total 40 pixels, and the contribution of the 36 pixels dominates over that of the 4 pixels. If Fig. 3a is compared with 3b and 3c respectively, SSD(Fig3a, Fig3b) = 6444, and SSD(Fig3a, Fig3c) = 6400. Therefore, SSD will choose 3c as the closer match although in fact 3b is much more similar to the original neighborhood. This shows SSD can give suboptimal matches.

3.2 Causal Neighborhoods

Image synthesis is inherently constrained to the raster ordering of an image and must proceed in a causal fashion. Causality, however, limits a synthesized pixel from depending on all its neighbors. Such an assumption overlooks strong dependencies and will degrade the overall results.

Also, the WL algorithm has an assumption that the output image wraps around and handles edges toroidally. This tileability is well preserved across the left and right borders. However, the top and bottom borders do not tile well because the top rows do not appear in the causal neighborhood of the bottom



(a) **(b)** **(c)**

109	55	123	103	120	106	119	120	124
102	112	60	106	107	126	127	113	103
106	129	107	65	129	115	126	117	126
100	120	114	124	70	106	104	116	125
112	105	118	106					

106	70	112	119	118	119	114	102	125
127	120	65	101	128	110	106	105	118
121	128	104	60	116	121	101	112	116
100	123	129	112	55	119	126	116	105
122	103	106	101					

109	95	123	103	120	106	119	120	124
102	112	100	106	107	126	127	113	103
106	129	107	105	129	115	126	117	126
100	120	114	124	110	106	104	116	125
112	105	118	106					

(d) **(e)** **(f)**

Figure 3: **(a), (b) and (c) are three neighborhoods. (d), (e) and (f) show the pixel values of the above neighborhoods.**

part. This is undesirable because they two will appear in the same non-causal neighborhood and be compared with the candidate non-causal neighborhoods when the next level is being synthesized. This conflict between the two part will lead to undesirable behavior under SSD matching.

3.3 Multi Resolution

The neighborhood matching technique is based on the Markov assumption that in a texture image every pixel only depends on its neighborhood. Theoretically, the size of the neighborhood should be at least as large as the size of the largest regular texture structure. However, this makes the computation very expensive because of high dimension if the image is represented in a single resolution. The multi-resolution approach is a more natural way to solve this problem, because in the coarsest level, even very small neighborhoods can cover a large part of the image.

Also, the multi-resolution method is very intuitive. In the coarse level, it captures the global structure of the texture, such as long-range repetitions, and synthesizes a low-resolution version of the output texture that is blurry and lacks details, yet giving the correct global structure. Then from coarse to fine, it adds details onto the predefined structure, to make it better and better. The low-resolution structures serve as a guide laying out the foundation of the image as details are added.

The success of multi-resolution depends upon its execution. In the WL implementation, the image structure often seems to change during the coarse to fine progression; that is, the final result looks very different from the synthesis of the first coarse level (See Fig. 4). The reason is very simple: in the concatenated neighborhood containing both the current level and the parent level, the numbers of pixels from the two sources are different! For example, if the parent level non-causal neighborhood has 5x5=25 pixels, and the current level causal neighborhood has (9x9-1)/2=40 pixels, in the SSD the 40 current-level pixels will dominate the 25 parent-level pixels. Therefore, the parent level contributes less weight to the SSD and may not fully forward the global structure to the next level.

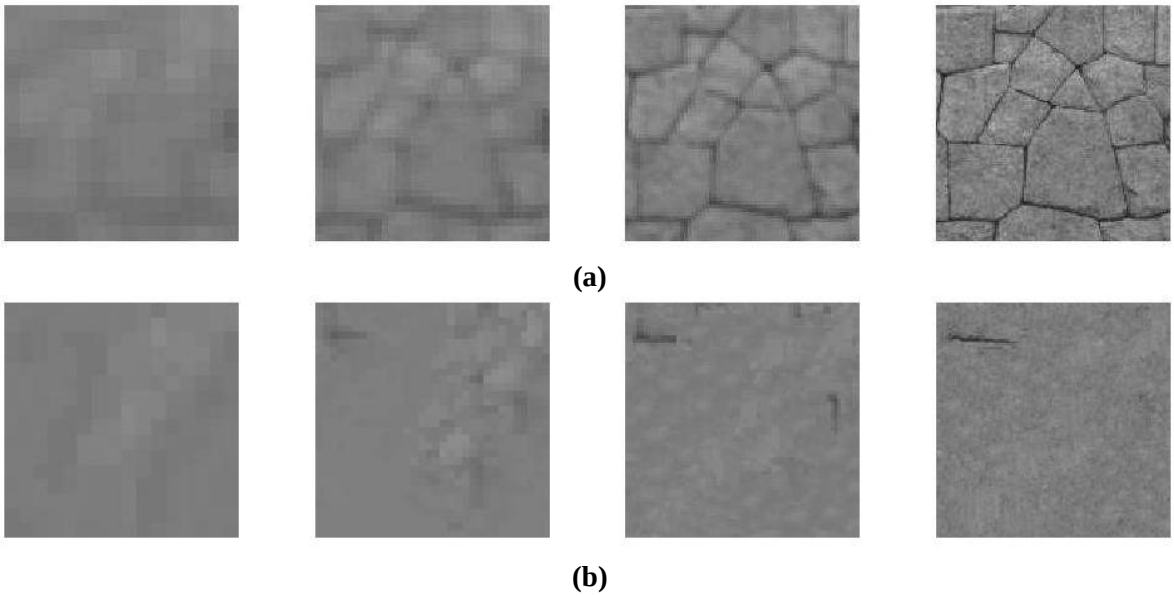


Figure 4: (a) is the Gaussian pyramid of the input stonewall texture. (b) is the synthesized Gaussian pyramid of output texture. The coarsest level vaguely gives hints of two dark lines in the reverse diagonal direction. However, these hints are not picked up in the coarse-to-fine process.

4 The Improved Algorithm

Based on the problems that were discussed in the previous section, we propose some modifications to the original algorithm. They are explained one by one as follows.

4.1 Morphological Processing And Image Analogy

What motivated us to begin the research on this topic was the failure of synthesis of the stonewall texture (Fig. 4). The boundaries of the stones are lost in the output image. The challenge in synthesizing this texture is that a majority of the pixels form the bodies of the stones and are plain in appearance. Only a few salient pixels form edges at the boundaries between the stones.

To balance the weights of plain pixels and edge pixels, we used morphological operations such as erosion or dilation, depending on whether the edges are relatively dark pixels or bright pixels. Such morphological operations will increase the width of the edges so that they can count more in the SSD. The steps can be described as follows. (Fig. 5)

1. Input image Ia is eroded (or dilated) to increase the width of the edges, resulting in image Sa.
2. Sa is processed with WL multiresolution synthesis algorithm, to synthesize an output Ss.
3. Use the image analogy concept [8], where three input images A, A', B are given to produce an output image B' such that this new "analogous" image B' relates to B in "the same way" as A' relates to A. Here A is Sa, A' is Ia, B is Ss, and we compute the B' which is the final output texture Is. We use image analogy in single resolution, by matching the concatenated vector of neighborhoods from B (non-causal) and B' (causal) to those candidate neighborhoods from A and A'.

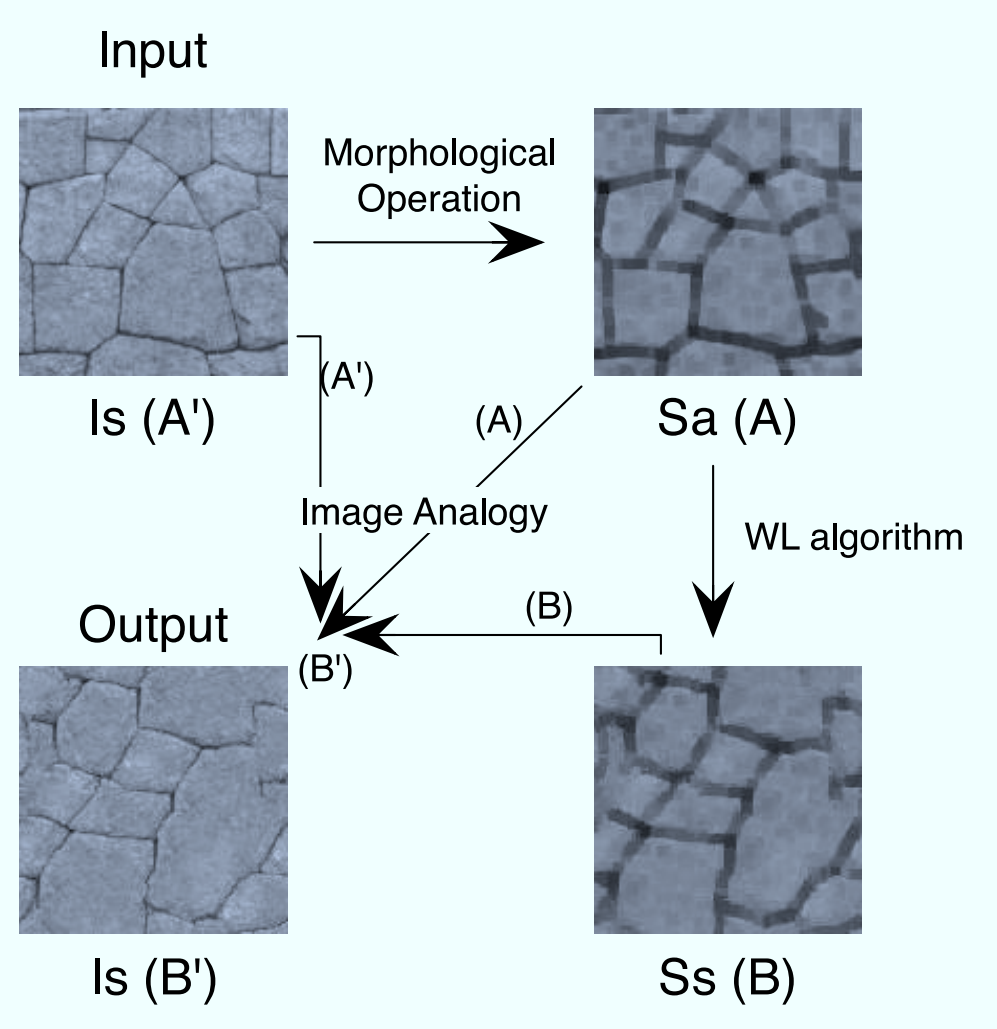


Figure 5: **The image analogy step. Ia is the input texture image. It was processed with morphological operation to enhance the edges, to be image Sa. Sa was used as the input of WL algorithm to synthesize an output image Ss. Then Ia, Sa and Ss are used as the three input of the image analogy step, to get the final output texture image Is.**

In the morphological operation, the structure element is chosen as a 5x5 square. Currently it is chosen by hand. In future, we may select the size of the structure element automatically based on the image histogram analysis, i.e., adjusting the size of the structure element such that in the image histogram the peak corresponding to 'figure' will have a comparable size with the peak corresponding to 'background'.

This modification is only effective for those textures that consist of a lot of similar but irregular objects. For other kinds of textures, this modification may not be necessary.

4.2 Non-causal Neighborhoods

For each level of the pyramid, we reformulate synthesis as a two step process.

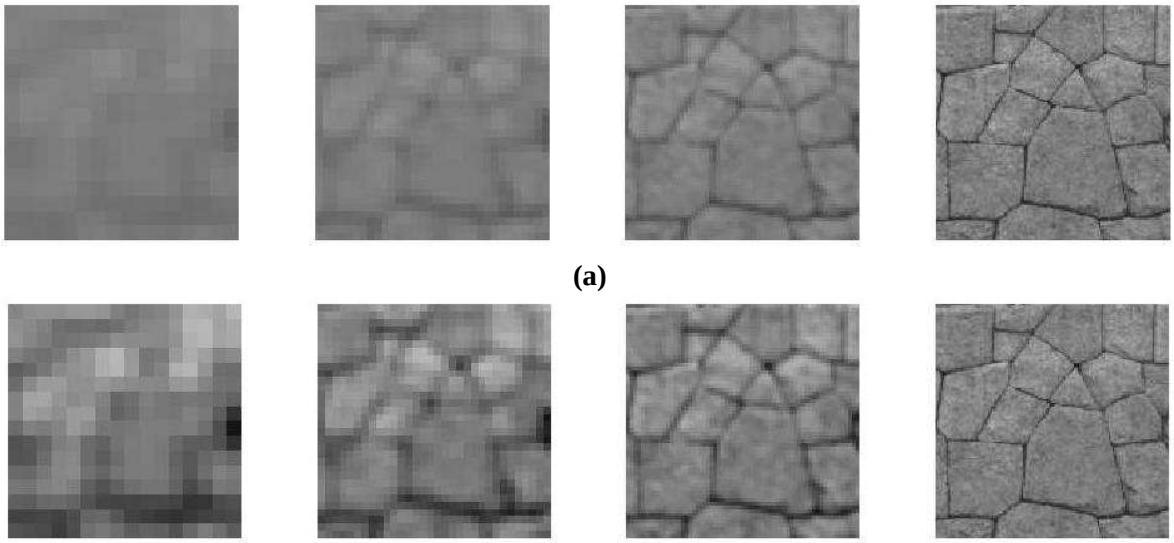
1. First, we synthesize this level using causal neighborhoods. When the bottom part is being synthesized, non-causal neighborhoods should be used so that the already synthesized top part can participate into the synthesis process and give guide to the bottom part since they are supposed to be tiled seamlessly.

2. Then we repeat the synthesis process using different non-causal neighborhoods within the same level. This step can refine the synthesized results slightly because non-causal neighborhoods give complete information of the surroundings of each pixel.

4.3 Weighted Multiresolution Matching

To maintain a consistent image structure throughout the coarse-to-fine process, we use weighted neighborhood matching. The goal of the weighting is to overcome the uneven numbers of pixels from the current level and the parent level. In the total SSD we give the parent level more weight to balance it with the current level. In our experiment, we give the weights to the parent level and the current level as 2:1 to approximately balance the 25 and 40 pixels in the SSD.

In the coarse level, because it is a blurred version of the image, it often has a much smaller dynamic range of pixel values. Therefore, it also makes the parent level actually less weighted. Also, sometimes because of the blurriness, the coarsest level is nearly featureless (See Fig. 6). So we adjusted the dynamic range of each level to be the same as the finest level to overcome this problem.



(a)

(b)

Figure 6: The Gaussian pyramid of the input stonewall texture image. (a) is the pyramid without adjusting dynamic range. The coarsest level is almost featureless because of the blurriness. (b) shows the pyramid with adjusted dynamic range, giving more details in the coarsest level.

5 Results

We tested our algorithm on many different textures, including: stonewall, pebbles, beads, pine-shoots, peas and leaves. All these textures share one common feature, that is, they all consist of many similar but irregular objects. The WL algorithm does not perform well on these textures, but our modifications produce good results on these.

We did two sets of experiments for comparison. The first set of experiments intends to show the improvements by different combinations of our modifications over the WL algorithm (See Fig. 7). The second set of experiments compares our results with state-of-the-art approaches in the texture synthesis field (See Fig. 8 and Fig. 9).

5.1 First Set of Experiments

The first set of experiments shows different combinations of our modifications, by adding one modification each time. The three combinations include: (1)"Weighted matching", (2) "Weighted matching" + "Non-causal neighborhoods", (3) "Weighted matching" + "Non-causal neighborhoods" + "Morphological processing and image analogy". We compare these three combinations with the WL algorithm and Ashikhmin's algorithm [1]. Ashikhmin's algorithm is another modified version of the WL algorithm which usually produces discontinuity between patches. (See Fig. 7)

For each texture, we showed six images. They are: (a) the input image, (b) the output of the WL algorithm [12], (c) the output of Ashikhmin's algorithm [1], (d) using the modification of "Weighted matching" on the WL algorithm, (e) using "Weighted matching" + "Non-causal neighborhoods" on the WL algorithm, and (f) using "Weighted matching" + "Non-causal neighborhoods" + "Morphological processing and image analogy" on the WL algorithm.

The results show that our modifications do improve the quality of the WL texture synthesis, in that the blurriness produced by the WL algorithm is reduced by the modifications. Moreover, our modifications do not have the discontinuity problem that Ashikhmin's algorithm has. Among the three modifications, it seems that "Morphological processing and image analogy" has the greatest effect, which improves the synthesis quality dramatically.

In this set of experiment, each texture image is 128x128 grayscale, including both input and output images. In both the WL algorithm and our approach, the Gaussian pyramids contain 4 levels, with the sizes as 16x16, 32x32, 64x64 and 128x128. Neighborhood size is chosen as 9x9 and parent level has neighborhood size of 5x5. We produced the results of Ashikhmin's algorithm by using Hertzmann's code (http://mrl.nyu.edu/projects/image-analogies/lf/), with the option "Ashikhmin search method" being selected.

5.2 Second Set of Experiments

In order to compare our algorithm with state-of-the-art approaches in the texture synthesis area, we did the second set of experiments. We compare the our results of combining all three modifications, with Efros and Freeman's image quilting algorithm [4] and Kwatra et al.'s Graph-cut approach [9]. These two state-of-the-art approaches are both patch-based. It is said that patch-based texture synthesis approaches usually can avoid the blurriness problem which most pixel-based approaches have. This time, all output images are 256x256 grayscale, and the input texture images are the same as those in the first set of experiment, which are all 128x128 grayscale.

We implemented Efros and Freeman's image quilting algorithm, in which the window size was chosen to be approximately the size of the relevent structures in the texture. That is, the window sizes for stonewall, pebbles, beads, pine-shoots, peas and leaves textures are: 42, 42, 30, 66, 48 and 42 respectively. The width of the overlap edge is 1/6 of the window size. Kwatra et al.'s Graph-cut results were kindly provided by Kwatra. Our own results were produced by combining all three modifications. The Gaussian pyramids for the output images contain 4 levels, with the sizes as 32x32, 64x64, 128x128 and 256x256. Neighborhood size is chosen as 9x9 and parent level has neighborhood size of 5x5. (See Fig. 8 and Fig. 9)

From the comparison, we can see that our approach is almost comparable with these patch-based methods. Sometimes, our approach produces even better results than the other two methods. For example, our approach outperforms Efros and Freeman's image quilting approach on the "stonewall", "pebbles", "beads" and "leaves" textures, because in these cases Efros and Freeman's approach either has broken objects in the output texture or produces a lot of exactly repeating elements. Kwatra et al.'s Graph-cut approach is hard to beat, however, on the "beads" texture, our approach produces fewer "broken" beads in the output image than the graph-cut approach. However, on the "pine-shoots" texture, the two patch-based methods outperforms our approach in that our result produces blurriness in this texture.

6 Conclusion

In this paper, we presented an improved version of Wei and Levoy's texture synthesis algorithm. The modifications include applying morphological operations to add weight to salient edge pixels, using both non-causal neighborhoods and causal neighborhoods, and adjusting the weighting between the parent and current level in the Gaussian pyramid. Experimental results demonstrate that the modifications improve the synthesis results for many textures.

Acknowledgments

We thank Vivek Kwatra for kindly providing us the Graph-cut synthesis results for the input textures.

References

[1] M. Ashikhmin, "Synthesizing Natural Textures," *ACM Symposium on Interactive 3D Graphics*, pp. 217-226, 2001.

[2] G. R. Cross and A. K. Jain, "Markov Random Field Texture Models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 5, pp. 25-39, 1983.

[3] J. S. De Bonet, "Multiresolution Sampling Procedure for Analysis and Synthesis of Texture Images," *SIGGRAPH*, pp. 361-368, 1997.

[4] A. Efros and W. T. Freeman, "Image Quilting for Texture Synthesis and Transfer," *SIGGRAPH*, 2001.

[5] A. Efros and T. Leung, "Texture Synthesis by Non-parametric Sampling," *ICCV*, 1999.

[6] Y. Xu, B. Guo, and H.-Y. Shum, "Chaos mosaic: Fast and memory efficient texture synthesis," *Tech. Rep. MSR-TR-2000-32*, Microsoft Research, 2002.

[7] D. J. Heeger and J. R. Bergen, "Pyramid Based Texture Analysis/Synthesis," *SIGGRAPH*, pp. 229-238, 1995.

[8] A. Hertzmann, C. Jacobs, N. Oliver, B. Curless, D. Salesin, "Image analogies," *SIGGRAPH*, 2001.

[9] V. Kwatra, A. Schödl, I. Essa, G. Turk, and A. Bobick, "Graphcut Textures: Image and Video Synthesis Using Graph Cuts," *SIGGRAPH*, 2003.

[10] L. Liang, C. Liu, Y. Xu, B. Guo, and H.-Y. Shum, "Real-time texture synthesis by patch-based sampling," *ACM Transactions on Graphics*, Vol. 20(3), pp. 127150, 2001.

[11] J Portilla and E P Simoncelli, "A parametric texture model based on joint statistics of complex wavelet coefficients," *Int'l Journal of Computer Vision*, Vol. 40(1), pp. 49-71, 2000.

[12] L. Wei and M. Levoy, "Fast Texture Synthesis using Tree-structured Vector Quantization," *SIGGRAPH*, 2000.

[13] S. C. Zhu, Y. N. Wu and D.B. Mumford, "FRAME: Filters, Random field And Maximum Entropy: Towards a Unified Theory for Texture Modeling," *International Journal of Computer Vision*, Vol. 27(2), pp. 1-20, 1998.

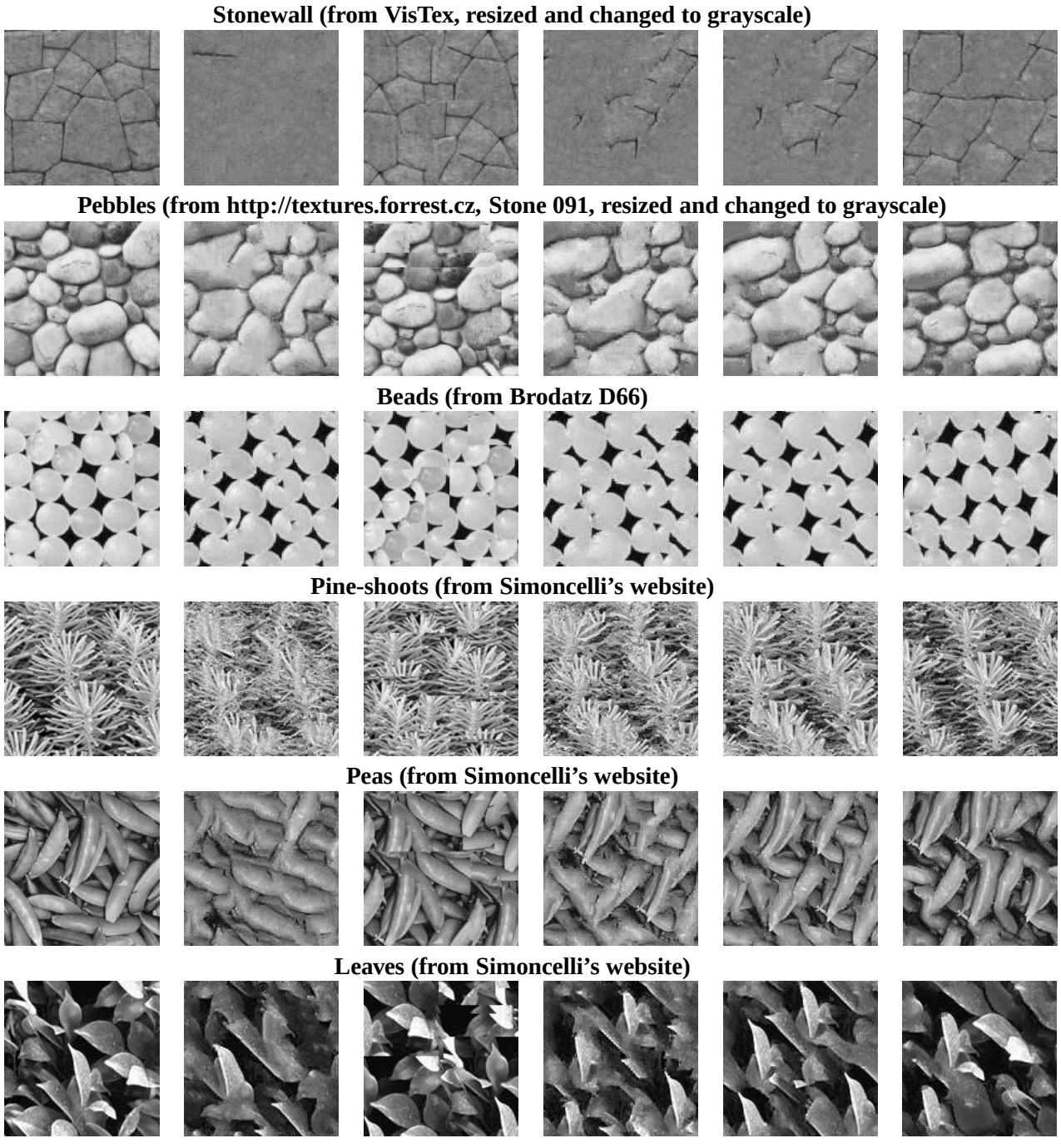


Figure 7: Synthesis results of stonewall, pebbles, beads, pine-shoots, peas and leaves. For each texture, we showed six images. They are: (a) the input image, (b) the output of the WL algorithm, (c) the output of Ashikhmin's algorithm, (d) the output using the modification of "Weighted matching" ("W") on the WL algorithm, (e) the output using "Weighted matching" ("W") + "Non-causal neighborhoods" ("N") on the WL algorithm, and (f) the output using "Weighted matching" ("W") + "Non-causal neighborhoods" ("N") + "Morphological processing and image analogy" ("M") on the WL algorithm.

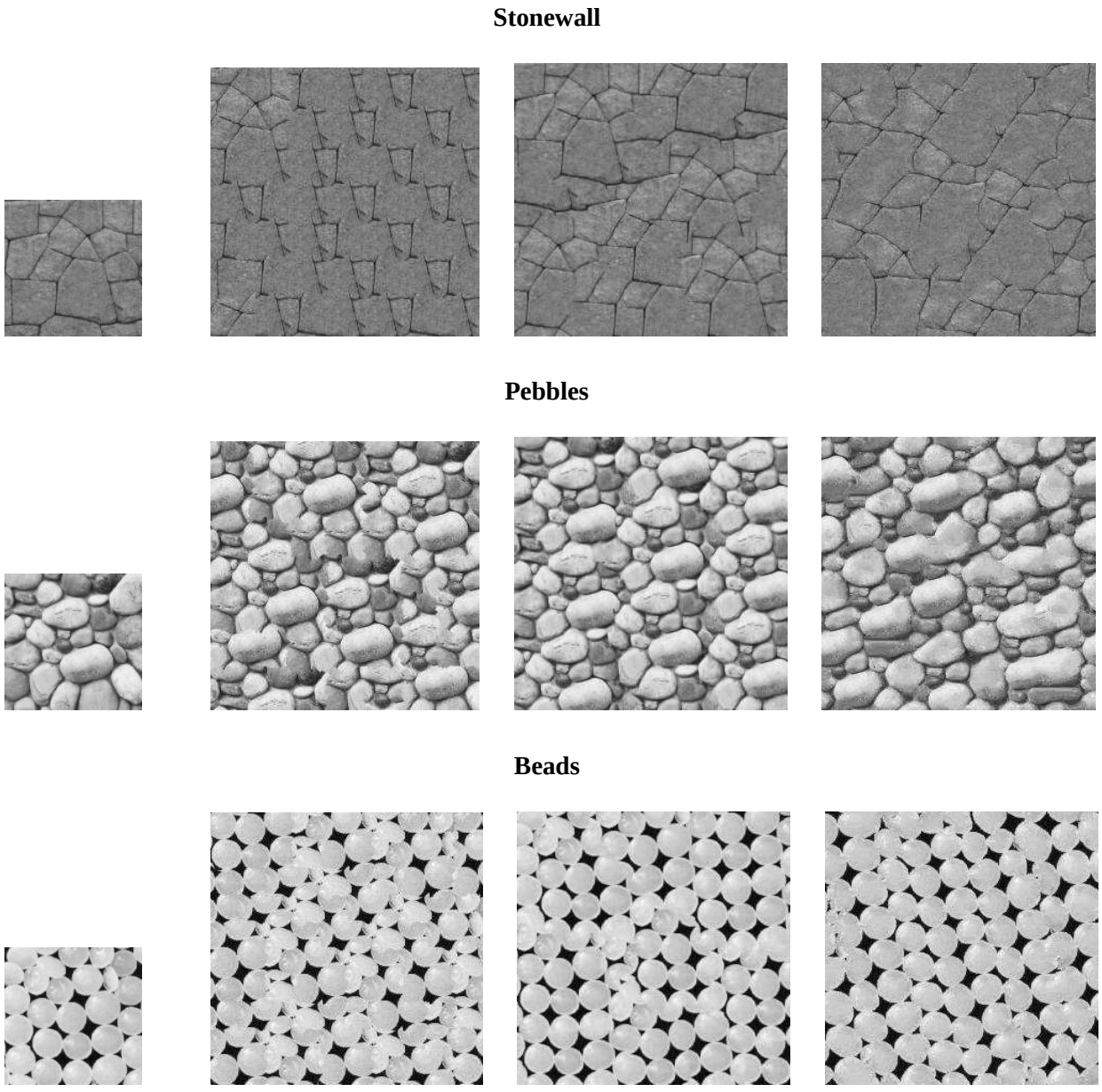


Figure 8: Comparison of our approach with state-of-the-art texture synthesis approaches on stonewall, pebbles, beads textures. Each row has four images: (a) the input image, (b) the output of Efros and Freeman's image quilting approach, (c) the output of Kwatra et al.'s Graph-cut approach, (d) the output of our approach combining all three modifications.

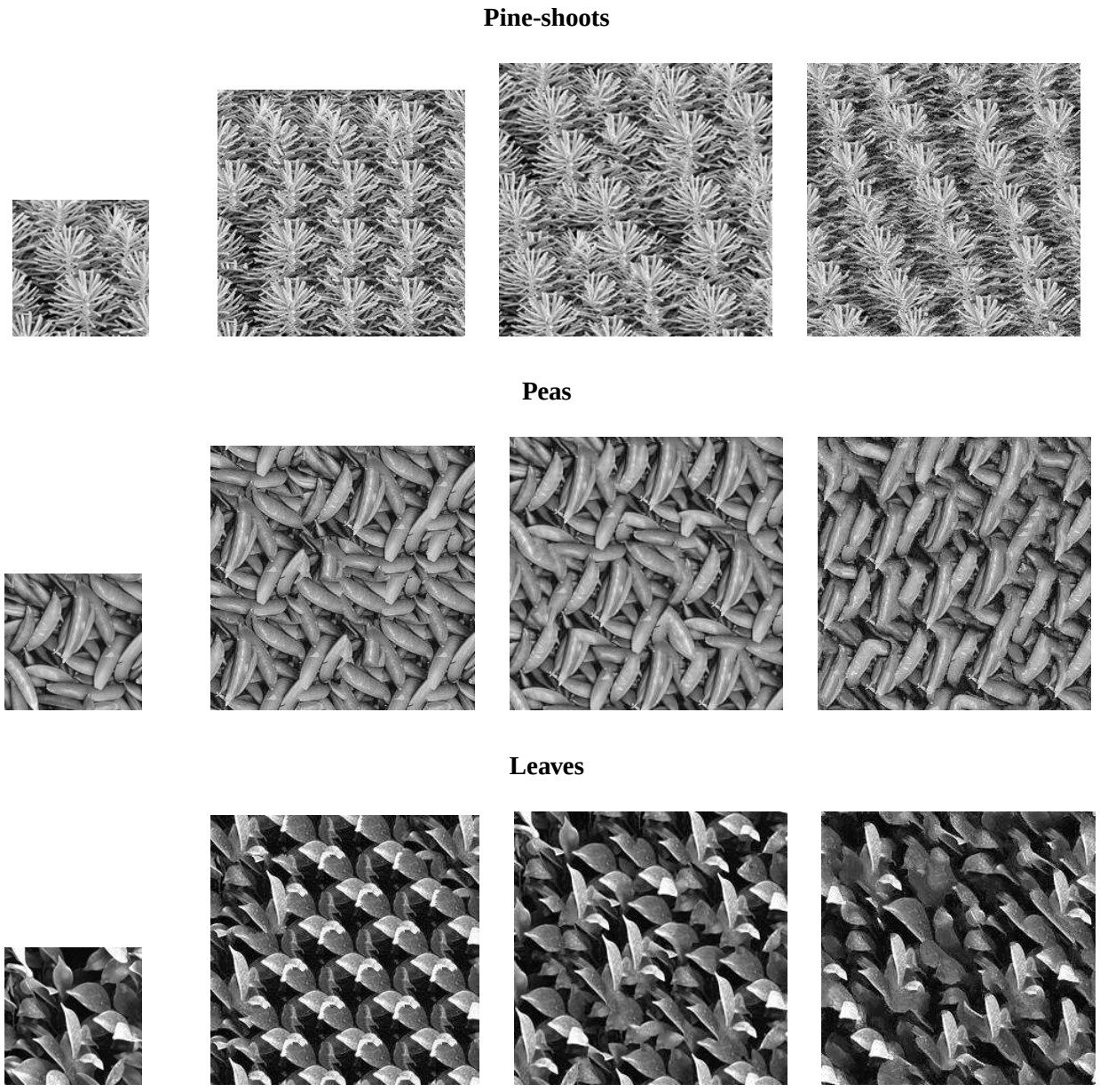


Figure 9: Comparison of our approach with state-of-the-art texture synthesis approaches on pine-shoots, peas and leaves textures. Each row has four images: (a) the input image, (b) the output of Efros and Freeman's image quilting approach, (c) the output of Kwatra et al.'s Graph-cut approach, (d) the output of our approach combining all three modifications.