Hexagonal Image Quilting for Texture Synthesis

David Kuri
OVGU Magdeburg
Universitätsplatz 2
39114 Magdeburg, Germany
david.kuri@st.ovgu.de

Elena Root
Volkswagen AG
Berliner Ring 2
38440 Wolfsburg, Germany
elena.root@volkswagen.de

Holger Theisel
OVGU Magdeburg
Universitätsplatz 2
39114 Magdeburg, Germany
theisel@isg.cs.ovgu.de

ABSTRACT

The synthesis of textures of arbitrary size from smaller samples is a much-noticed problem in the field of computer graphics. While the proposed solutions deliver very good results for regular and near-regular textures, the synthesis of irregular textures is in need of improvement. In this paper, the well-known Image Quilting algorithm is analyzed and its idea is enhanced by replacing the square shape of the patches by a hexagonal shape. In addition, rotation and mirroring of patches are introduced. A penalty map is used to enforce even usage of source data and transformations to make feature repetition less noticeable and improve synthesis from multiple source images. This leads to considerably better results for complex textures like wood, smoke or water waves.

Keywords

Texture Synthesis, Image Quilting, Hexagonal Tiling, Rotation

1 INTRODUCTION

In the field of computer graphics, textures are used to describe the visual properties of a surface. For realistic results, it is often convenient to use a scan or photograph of a real-world surface and map it to a virtual object. When more source data is required than is available, a bigger texture can be synthesized from a small sample. This paper analyzes the popular Image Quilting algorithm [1] and improves it in several aspects to overcome some of its shortcomings.

1.1 Previous Work

Many algorithms have been proposed to synthesize a texture of arbitrary size from a source texture, using various fundamentally different approaches. This allows for a basic categorization of texture synthesis algorithms into three classes:

Statistics-based: Many of the first algorithms for texture synthesis were based on image statistics. In his works on texture discrimination, Bela Julesz [2] introduced a new model for the human observer's perception of texture. According to Julesz, two images are perceived as being the same texture when some appropriate set of image statistics matches. This idea was applied in various publications on textures synthesis. Heeger and Bergen [3] used Laplacian and steerable image pyramids to iteratively modify a noise image until the pyramids' histograms match closely. This method delivers convincing results for many stochastic textures, but fails to capture distinct structures. De Bonet [4] proposed a similar method that delivers satisfying results for a wider range of textures, but requires fine-tuned threshold parameters and therefore sacrifices usability. Other algorithms were proposed by Zhu et al. [5] and Simoncelli and Portilla [6], but the above-mentioned problems regarding structures could not be solved completely.

Pixel-based: These algorithms synthesize a texture one pixel at a time. Arguably the first algorithm to use this idea was proposed by Popat [7]. In his approach, conditional probability functions are derived from the source texture and used to synthesize a new pixel in the output texture. While able to produce comparatively good results at that time, the algorithm is prone to 'growing garbage'. Paget and Longstaff [8] solved this problem by synthesizing the low frequencies first and gradually adding high frequency detail. The main problem with pixel-based algorithms being speed, Wei and Levoy [9] used tree-structured vector quantization to improve the performance of pixel-based approaches considerably.

Patch-based: The first patch-based texture synthesis algorithms were proposed by Xu, Go and Shum [10] and Liang et al. [11]. These algorithms copy whole patches from the source texture and paste them into the result, smoothing the overlapping edges with cross-edge filtering. While solving the issues with capturing structures in statistics-based algorithms and the lack of speed common to pixel-based methods, these algorithms introduced their own set of problems like implausible blending and noticeable repetition. The Im-

 ISBN 978-80-86943-70-1

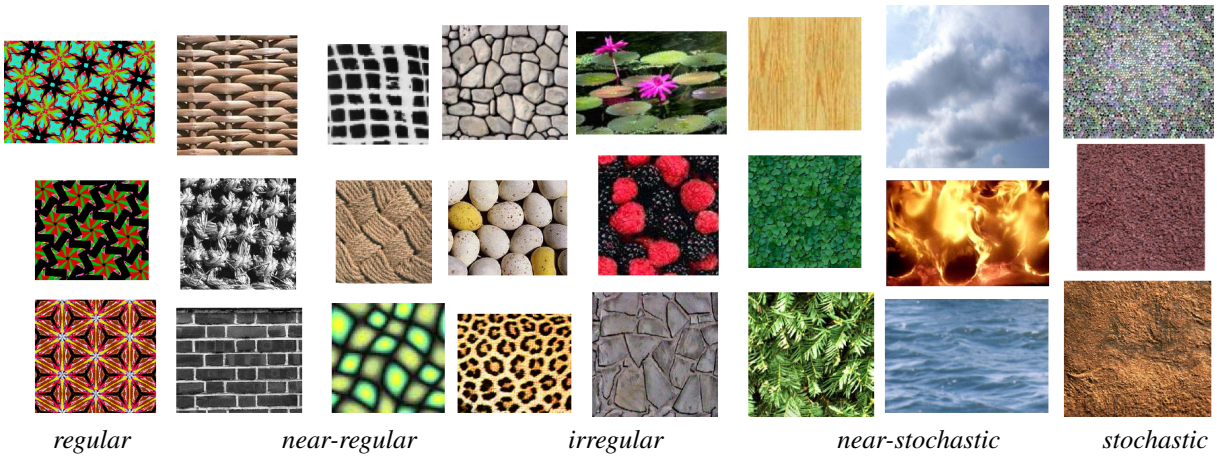


Figure 1: Texture spectrum based on [13]. Textures are classified anywhere from regular to stochastic. The proposed improvements target mainly irregular and near-stochastic textures.

age Quilting algorithm by Efros and Freeman [1] improved the blending by using a cut through the overlap region for the which the difference in pixel values is minimal. Kwatra et al. [12] don't use a fixed patch size, but apply a graph cut technique to find optimal patch regions. Beyond translations, they also suggest using other transformations like rotation and scaling and provide some results for the use of rotation by multiples of 90°and mirroring.

1.2 Texture classification

Lin et al. [13] suggest a classification of textures on an axis between *regular* and *stochastic*, as seen in Figure 1. **Regular textures** are perfectly periodic patterns, where the pixel values repeat in equal intervals without any variation. With real-world textures this is very uncommon and slight variations in pixel values due to fabrication inaccuracy or measurement noise are almost always present. Therefore, these textures are called **near-regular textures**.

In the center of the spectrum are **irregular textures**. They are periodic at their core but feature some stochastic component, may it be a non-periodic deformation or a considerable random variation in color.

Due to the focus of their work, Lin et al. do not elaborate on the stochastic end of the spectrum. For the purpose of this paper, **stochastic textures** are defined as textures that consist mainly of random noise, although this noise may only affect the brightness of each pixel with a strongly biased color hue. **Near-stochastic textures** feature defining structures, but with random variation and no perceptible repetition.

1.3 Motivation

As shown in Lin et al. [13], patch-based algorithms produce very good results for regular and near-regular textures. Convincing results for pure stochastic textures can be achieved using statistics-based methods as described by Simoncelli and Portilla [6]. In contrast, the synthesis of near-stochastic and irregular textures admits of improvement.

One of the most prominent approaches for near-stochastic and irregular textures is the Image Quilting algorithm [1]. Despite the good results, the underlying idea is very simple and leaves enough room for improvements.

Like most patch-based algorithms, Image Quilting is suitable for the left part of the texture spectrum as shown in Figure 1. However, when synthesizing near-stochastic or irregular textures like wood, smoke or water waves the algorithm's results quickly deteriorate. Noticeable repetition or vanishing of distinct features decrease similarity to the source texture.

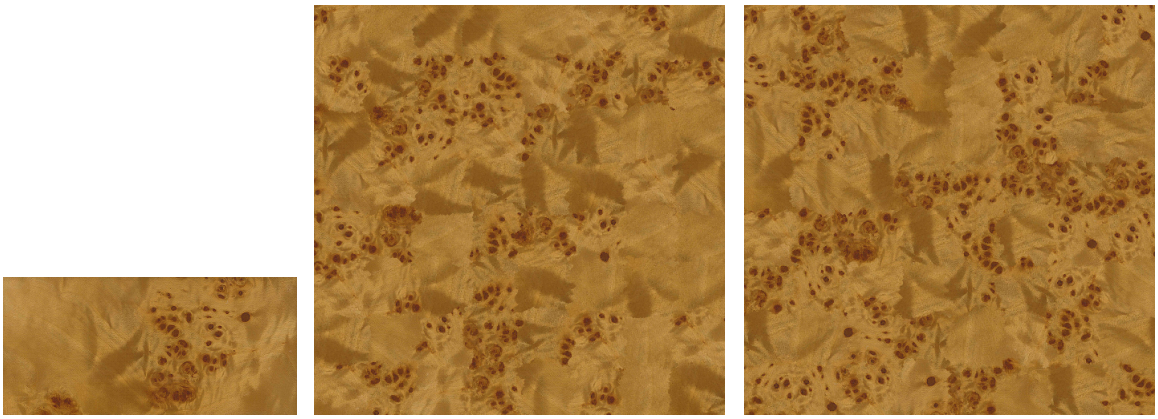
In this paper, the Image Quilting algorithm is analyzed and improved to provide better results for near-stochastic and irregular textures. The perfect result would be a method that delivers convincing results for all texture types alike. When looking at the existing approaches, this seems unrealistic. Therefore, a deterioration of results for the other texture types is acceptable if it benefits the results for near-stochastic and irregular textures.

2 ALGORITHM

This section provides a quick overview of the Image Quilting algorithm as proposed by Efros and Freeman and suggests several improvements to achieve better results for the targeted texture types. For a complete definition of Image Quilting refer to [1].

2.1 Image Quilting

The Image Quilting algorithm divides the output image into equal blocks that overlap by a certain amount of



(a) (b) (c)

Figure 2: From left to right: original image and Image Quilting results with 0.01 and 0.2 error tolerance. High error tolerance avoids repetition to some extent, but also leads to more visual inconsistencies.

pixels. For each of these blocks, the input texture is searched for a set of blocks for which the error in the overlap region satisfies some error constraint. In the original implementation, the error is computed as the sum of squared distances (*L2* norm) between between RGB color values. Each block for which the error is at most 1.1 times the best possible error (i.e. the error of the best-matching block) is added to the set. From this set, one of the blocks is chosen at random.

An error surface is calculated for the overlap region of the previously chosen blocks and the new one. A minimum cost path through this surface is calculated and used as the boundary of the new block, which is then pasted onto the result image.

The original algorithm uses a dynamic programming approach to find a path through the error surface. For a vertical overlap between two blocks B_1 and B_2 with the error surface e defined as the error for every point in the overlap region, the cumulative error is computed as:

$$E_{i,j} = e_{i,j} + min(E_{i-1,j-1}, E_{i-1,j}, E_{i-1,j+1}) \quad (1)$$

The minimum error of the last row in E can be used to trace back and find the shortest path. The calculation of a path through a horizontal overlap follows the same principal.

This approach, while being exceedingly fast and producing adequate results most of the time, is not guaranteed to find the optimal path. As can easily be seen from the formula, the gradient of all paths found is constant in the first dimension. This may not be true for the optimal path. Throughout this paper, the Dijkstra algorithm [14] is used as a substitute in all self-provided images. The boundary cut could be further improved, as described by Long and Mould [15].

2.2 Rotation and Mirroring

When using Image Quilting to produce a texture that is larger than the input, some regions will inevitably occur more than once. This becomes a major problem when dealing with near-stochastic and irregular textures that have *non-repeating distinct* features.

One simple approach is to identify these distinct features by some measure and prevent multiple use of the particular image regions. This would lead to an artificial lowering of the frequency of these features, changing the appearance of the result and potentially decreasing the similarity to the input texture.

A better solution is to include variations of the source image. Rotation and mirroring can be used to make the repetition of features less noticeable for a human observer, as suggested by Kwatra et al. [12]. To integrate this into the Image Quilting algorithm, a number of rotated and / or flipped images are taken into account in addition to the original source image. The number of rotations can be chosen freely. The complexity of the algorithm increases linearly with the number of images, i.e. $O(n)$, where n is the number of images taken into account.

As can be seen in Figure 2, the error tolerance greatly affects the distribution of the result. A low error tolerance is beneficial, because only the best-fitting pieces are used in the output image. This in turn leads to an undue preference of image regions with low contrast and without hard edges over image regions that show distinct features, but possibly produce a higher error. If few such regions exist in the source data, the repetition becomes visible as seen in Figure 2 b).

Using a higher error tolerance alleviates this problem stochastically and gives the algorithm a bias towards randomness. This has both positive and negative effects. It leads to a more even distribution of source image regions in the output, but also globally increases

the error in the result and is not a reliable solution. Even high values can't prevent noticeable repetition altogether.

To enforce an optimal exploitation of the available source data (both original and transformed), we introduce a penalty map. This map stores the previous uses of an image region with the respective transformation encoded as gray levels. For every possible combination of rotation angle and mirroring, a penalty map is created. These maps are initially black, i.e. filled with zeros, because no source region in any transformation has been used. For every block that is chosen during the algorithm, the source image region is lightened in the penalty map for the corresponding rotation and mirroring values. During the calculation of the error values, the penalty map is queried and, if the the region in question has been used before (i.e. the value in the penalty map is greater than zero), the error is increased. This is implemented as follows:

- When a block is found, a blurred dot is additively drawn to the map at the center point of the chosen source region. The blur radius should roughly correspond to the size of the blocks.
- After the error e in the overlap region is summed up, the pixel at the center point of the source region in question is queried. The new error e' is calculated as $e' = e + p\,e\,c^2$, where c is the gray-scale value of the pixel in range $[0,1]$ and p is a user-defined parameter to adjust the penalty map influence.

The introduced penalty maps have another beneficial side effect. While it is trivial to use the Image Quilting algorithm to synthesize a texture not from one but from multiple source textures, the result is often unsatisfactory due to one source being overused. Penalty maps encourage an even usage of image regions *across images*. This means that if the requested result image requires more than the provided source data, all the available source data is used evenly on both image and image region level.

2.3 Hexagons

With difficult textures like the wood texture used in Figure 2, the best possible block sometimes still leads to a visible crack or error. This is aggravated by both a high error tolerance value and a high penalty map influence, which is giving preference to less-used, but possibly higher-error blocks. On the other hand, as seen in section 2.2, lowering these values can lead to noticeable and unwanted repetition of distinct features and is therefore not an option.

When looking closely at the results, one thing becomes obvious: an underlying pattern of equal squares was used to construct the images. This regular grid structure further emphasizes the (not easily avoidable) error. By using a different, less obvious structure the error could be concealed without nominally being lowered.

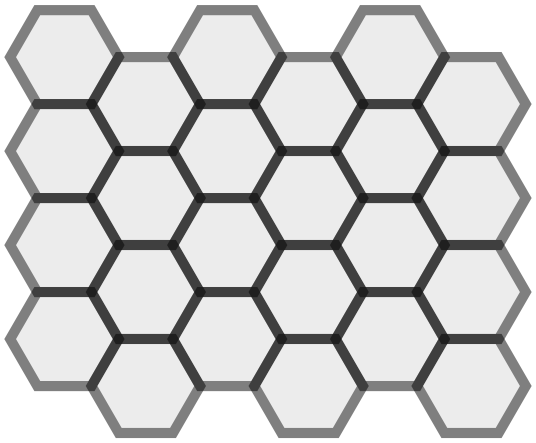


Figure 3: Hexagonal tiling and the resulting overlap regions.

The idea of the Image Quilting algorithm can in theory be used with any tessellation of the plane. For the sake of simplicity, a regular tessellation (i.e. a tessellation using only congruent regular polygons) should be used. The three regular tessellations of the plane are triangular, square and hexagonal tessellation [16]. Squares were used in the original Image Quilting algorithm. Of the remaining two, hexagons were chosen as the more 'interesting' shape. Triangle-based Image Quilting may have its own benefits that are yet to be analyzed.

The hexagonal shape breaks up the all too common rectangular pattern. Overlapping areas do not only occur horizontally or vertically, but also at a slanted angle. Together with the arbitrary number of rotations, this better represents the structure of complex textures and leads to improved results. A hexagon cannot exceed the borders of a source image. The size of the hexagons is set by the user and should be chosen dependent on the size of the features present in the source data.

A positive side-effect of the hexagonal structure can be observed when synthesizing some near-regular or regular textures. If a texture is horizontally uniform in an area bigger than the overlap region of two horizontally adjacent squares, this can lead to a shift that breaks up the regular structure of the texture. The hexagons used in our variation are more 'interlocked' than the squares, making this case less likely to happen. For an example, see section 3.1.

2.4 Parameters

The algorithm is controlled by various parameters, which are described and summed up in this section.

The size of the result is controlled by the *vertical* and *horizontal hexagon count*. These parameters should be chosen generously, so that the desired pixel size can be cut from the result.

The *size* parameter controls the overall size of the hexagons. This value depends solely on the source data and should be big enough to capture the relevant structures, but not bigger.

The *overlap* parameter determines by how much pixels the hexagons overlap each other. A bigger overlap can lead to better results, but also increases computational cost. A value of $\frac{1}{6}$ of the hexagon diameter, similar to $\frac{1}{6}$ of the block size suggested by Efros and Freeman [1], is a good starting point.

The influence of the penalty map can be controlled with the *penalty factor p*. If the result shows noticeable repetition, this factor should be increased. Depending on the implementation details, especially the brightness of the blurred dot described in section 2.2, this value can vary widely. In our implementation, values between 100 and 800 were used, with a brightness of 10% in the center of the blurred dot.

During each step of the algorithm, a set of possible hexagons is assembled from which one is chosen randomly. The *error tolerance t* determines which hexagons are added to the set. Only those hexagons for which $e \leq min(e) * (1+t)$ are taken into consideration. Efros and Freeman [1] suggested a value of 0.1. The penalty map alleviates the need for a high error tolerance. A value of 0.01 was used in most of our tests.

Additional transformations are controlled by the integer *directions* parameter d and the boolean *flip* parameter. The first controls the number of rotations, each by a multiple of $\frac{360}{d}$ degree, while the second determines whether each image should additionally be mirrored horizontally. The total number of images taken into account equals d with disabled or $2d$ with enabled flip. Additional transformations should only be used when the source data requires them.

3 RESULTS

The algorithm was tested with a wide range of source textures. The algorithm is compared to the original Image Quilting algorithm using the images provided in [1]. In addition, we also used some complex textures that are very hard to synthesize with existing algorithms.

While the proposed modifications of the Image Quilting algorithm can improve results, there are textures that require only part of the features or none at all. This can also be seen in the following comparison.

3.1 Comparison

Some results of the comparison with the original Image Quilting algorithm can be seen in Figure 4. For the targeted texture types, i.e. irregular and near-stochastic textures, our algorithm shows improvement.

Neither the original algorithm nor our modified approach were able to produce convincing results for (1). The source texture does not show enough to clearly imply a pattern. The colors seem to be randomly distributed. On the other hand, the texture shows strong regularity in terms of shape. The Image Quilting algorithms are not able to capture these structures accurately.

(2), (11) and partly (6) show shadows and highlights that imply strong directional light. Our approach perhaps could have produced better results than the original algorithm if the textures were evenly lit. However, under the given conditions, rotation would lead to implausible lighting in the result image and severely deteriorate the perceived quality. For (6), rotation is arguably a valid choice because the structure is chaotic enough and the flawed lighting is not immediately obvious. For the other two we decided to not use rotation. The results for (2) and (11) are virtually unchanged.

In (3), the Image Quilting result show obvious repetition, which in our result is prevented to some degree thanks to an even distribution of different rotations. On the downside, the structure is not preserved as well as in the comparison image. We believe though that a set of parameters exists for which the structure is not distorted.

For (4), the original result was not perfect but our approach indeed seems to produce more visible inconsistencies. Better results could probably be achieved when experimenting with different shape and overlap sizes.

In (5), there is indeed more repetition in our result, but it is also closer to the source image. In the Image Quilting result, small portions of the available data with low contrast were used frequently. This is exactly the problem described in section 2.2. With our approach, the similarity in distribution can be controlled with the penalty factor p. Another example for this problem of the Image Quilting algorithm can be seen in (12). The feature distribution in our result closely matches the distribution in the source image, leading to a considerably better result while still avoiding noticeable repetition. Likewise, our results for (8) and (10) show less repetition and more even distribution compared to the Image Quilting results. For (9), the result was already very good with the original algorithm and couldn't be improved any further.

(7) is a good example of the hexagonal structure's benefit for regular and near-regular textures, as described in section 2.3. In the Image Quilting result, the cans

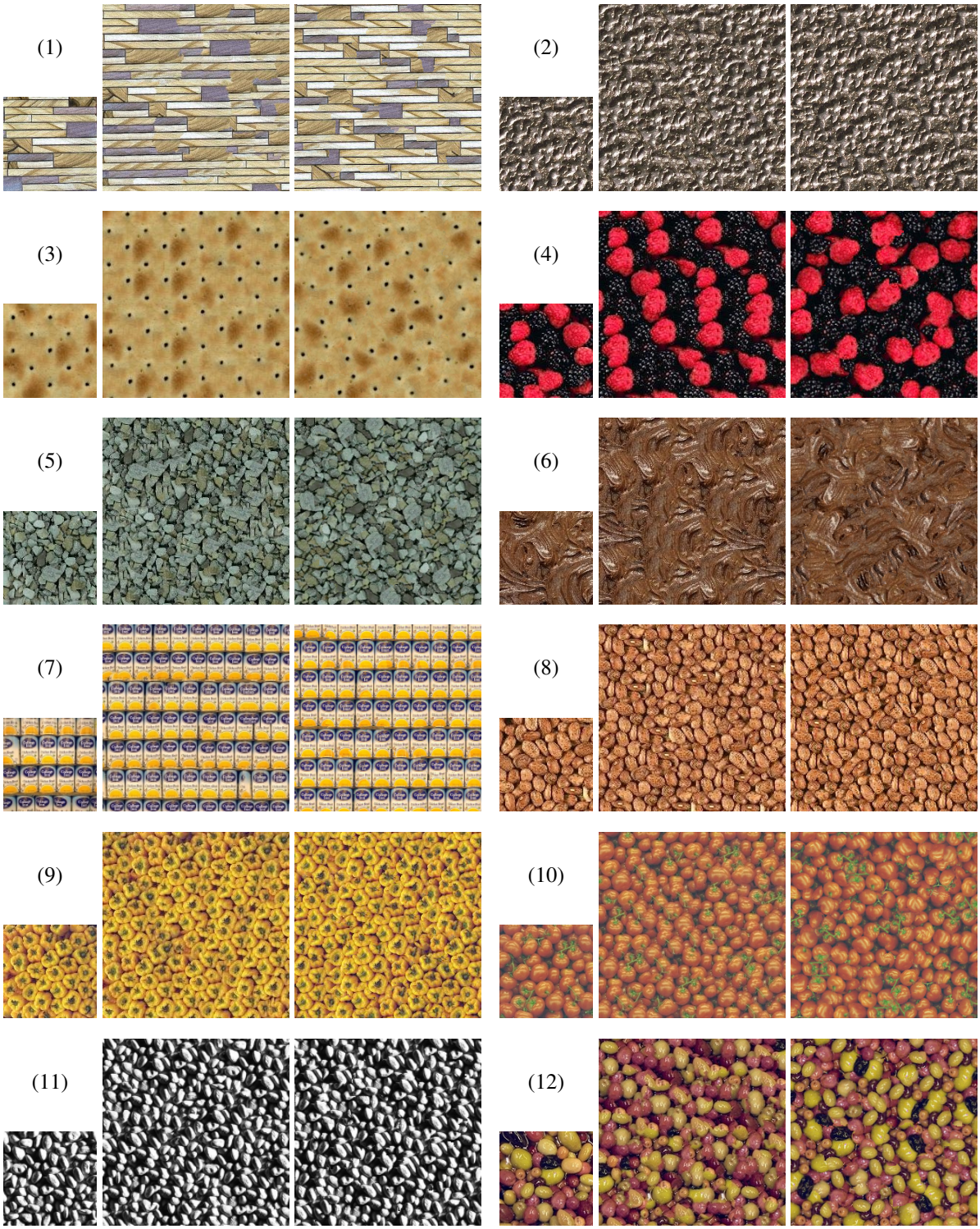


Figure 4: From left to right: comparison of original image, Image Quilting result and Hexagonal Image Quilting result.

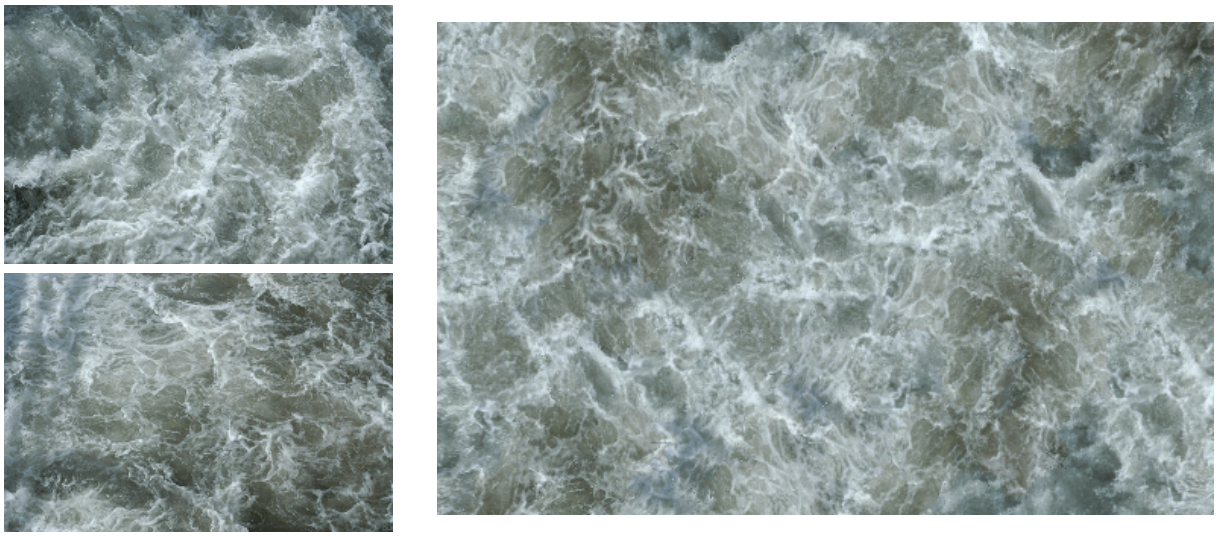


Figure 5: Near-stochastic WATER texture synthesized from two source images. Left: source images, right: result.

are distorted in some places and are either too wide or too narrow. This is due to the fact that each of the cans is relatively homogeneous horizontally. When this homogeneous region is wider than the overlap region, any offset in the homogeneous region will produce a low error and can potentially be chosen for the result, but can lead to visual anomalies. This is what happened to some of the cans. To prevent this, an overlap larger than the homogeneous region would be required. When using hexagons instead of squares, this error does not occur, even for small overlaps.

3.2 Near-stochastic results

For synthesis of near-stochastic texture, we took advantage of the algorithm's ability to synthesize from multiple source images. Two of the results are shown in Figure 5 and Figure 7.

WATER was synthesized from two source images. The pixel count of the result image is roughly doubled compared to the sum of the source images. There are no visible inconsistencies in the image. Repetition is found, but not striking.

For LIGHT COTTONWOOD, we used four source images to produce a texture with tripled pixel count compared to the sum of the source images. First, we synthesized an image using Image Quilting. The only trivial adaption is that not one but multiple source images are used, without penalty maps or any of the enhancements described in this paper. The result shows striking repetition and, as described in section 2.3, reveals the underlying regular structure of patches. For the second result, we used our proposed algorithm with hexagons, rotation and penalty maps. In comparison, our result contains less noticeable repetition. The patch structure is less noticeable and the distribution of features looks plausible concerning the available source data.

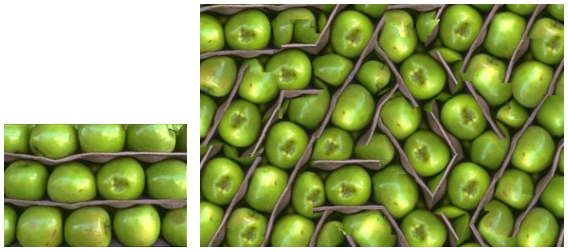


Figure 6: Result with parameters that are not adjusted to the source image. Rotation and a high penalty factor lead to odd behaviour here.

3.3 Performance

Without the additional transformations introduced in section 2.2, the proposed algorithm is nearly as fast as the original algorithm. Depending on the details of the implementation, the hexagonal shape can introduce a small overhead. In our tests, this overhead was negligible.

As stated in section 2.2, each additional rotation contributes to the search space of the algorithm and thereby significantly increases computation time. The duration of the algorithm with multiple of 60° rotations (i.e. six possible directions) compared to the duration with only the original orientation increases by a factor of 6. When mirroring is allowed, the computation time doubles because for every possible patch a mirrored version must be taken into account by the algorithm. Again, no measurable overhead is produced.

3.4 Limitations

The algorithm produces convincing results for most textures, from regular to stochastic, but is not without its limitations.

As mentioned in section 3.1, some textures like (1) in Figure 4 are not handled well by the algorithm. Espe-

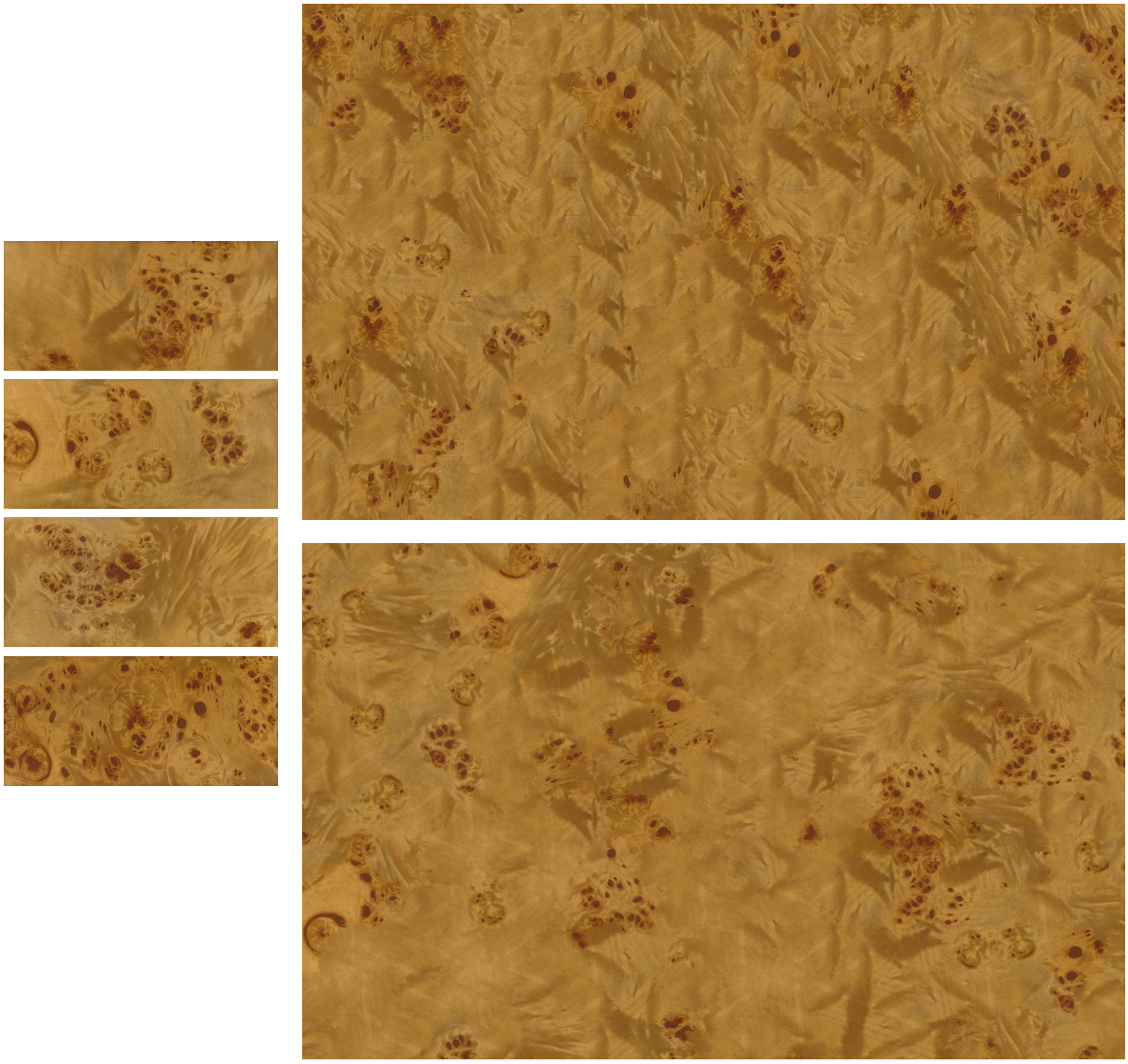


Figure 7: Comparison of near-stochastic LIGHT COTTONWOOD texture synthesis results from multiple source images. Left: source images, right top: result using the Image Quilting algorithm without any of the proposed modifications, right bottom: result using Hexagonal Image Quilting with penalty maps and rotation.

cially synthesis of textures with low-frequency features (e.g. gradients) will not produce good results. Neither Image Quilting nor the improved version described in this paper is able to globally optimize the result and 'plan ahead' for features that are larger than the shape's size. In addition, the hard cut through the overlap region is not suitable for blending gradients. The output image will show a noticeable tear.

The added features make the algorithm more powerful, but also require careful tweaking. Depending on the source texture, very different parameters may lead to the expected result and using unsuitable values can produce undesired effects as seen in Figure 6. Methods to automatically determine good values for the parameters based on the source texture would greatly benefit the usability of the algorithm.

Various other improvements of the Image Quilting algorithm have been proposed [15][17]. They could be combined with the enhancements described in this paper to further improve the results.

4 CONCLUSION

We proposed an algorithm based on the Image Quilting algorithm by Efros and Freeman. The modifications described in this paper make the algorithm more powerful and can lead to better results for all types of textures.

The introduced transformations can, at the cost of performance and dependent on the source texture, signif-

icantly increase the quality of results. The hexagonal shape yields better results for most near-stochastic textures and can even improve the results for certain near-regular and regular textures.

Acknowledgements: Source images for COTTONWOOD LIGHT are scans of a Volkswagen AG material. Source images for WATER are taken from the texture database at `www.cgtextures.com` maintained by Marcel Vijfwinkel and Wojtek Starak.

5 REFERENCES

[1] Efros, Freeman. Image quilting for texture synthesis and transfer. In *SIGGRAPH 01*, pp. 341–346, 2001.

[2] Julesz. Visual pattern discrimination. In *IRE Transactions on Information Theory 8 (2)*, pp. 84–92, 1962.

[3] Heeger, Bergen. Pyramid based texture analysis/synthesis. In *SIGGRAPH 95*, pp. 229–238, 1995.

[4] De Bonet. Multiresolution sampling procedure for analysis and synthesis of texture images. SIGGRAPH 97, 361-368, 1997.

[5] Zhu, Wu, Mumford. Filters, Random Fields and Maximum Entropy (FRAME): Towards a Unified Theory for Texture Modeling. In *International Journal of Computer Vision 27 (2)*, pp. 108–126, 1998.

[6] Simoncelli, Portilla. Texture characterization via joint statistics of wavelet coefficient magnitudes. In *ICIP 98*, pp. 62–66, 1998.

[7] Popat. Novel Cluster-Based Probability Model for Texture Synthesis, Classification, and Compression. In *Visual Communications and Image Processing*, pp. 756–768, 1993.

[8] Paget, Longstaff. Texture Synthesis via a Nonparametric Markov Random Field. In *IEEE Transactions on Image Processing 7 (6)*, pp. 925–931, 1998.

[9] Wei, Levoy. Fast texture synthesis using tree-structured vector quantization. In *SIGGRAPH 00*, pp. 479–488, 2000.

[10] Xu, Guo, Shum. Chaos Mosaic: Fast and Memory Efficient Texture Synthesis. Technical Report MSR-TR-2000-32, Microsoft Research, April 2000.

[11] Liang, Liu, Xu, Guo, Shum. Real-Time Texture Synthesis By Patch-Based Sampling. Technical Report MSR-TR-2001-40, Microsoft Research, March 2001.

[12] Kwatra, Schödl, Essa, Turk, Bobick. Graphcut Textures: Image and Video Synthesis Using Graph Cuts. In *SIGGRAPH 2003*, pp. 277–286, 2003.

[13] Lin, Hays, Wu, Kwatra, Liu. A Comparison Study of Four Texture Synthesis Algorithms on Regular and Near-regular Textures. Technical Report CMU-RI-TR-04-01, Carnegie Mellon University, January 2004.

[14] Dijkstra. A Note on Two Problems in Connexion with Graphs. In *NUMERISCHE MATHEMATIK 1 (1)*, pp. 269–271, 1959.

[15] Long, Mould. Improved Image Quilting. In *Proceedings of Graphics Interface 2007*, pp. 257–264, 2007.

[16] Grünbaum, Shephard. Tilings and patterns. W. H. Freeman & Co. New York NY, USA, 1986.

[17] O'Brien, Wickramanayake, Edirisinghe, Bez. Image quilting for texture synthesis: a revisit & a variation. In *Information, Communications and Signal Processing 2003 (2)*, pp. 763–767, 2003.