A Comparison Study of Four Texture Synthesis Algorithms on Regular and Near-regular Textures

Wen-Chieh Lin James H. Hays Chenyu Wu Vivek Kwatra*
Yanxi Liu

CMU-RI-TR-04-01

January 2004

School of Computer Science
Carnegie Mellon University

College of Computing*
Georgia Institute of Technology

Abstract

In this report, we compare the performance of four texture synthesis algorithms on synthesizing regular and near-regular textures. Our results show that near-regular texture synthesis remains a challenging problem. This is because a near-regular texture demonstrates both global regularity and local randomness in its texture pattern. It is difficult to preserve both properties in the synthetic textures. The comparison indicates that a specially-designed texture synthesis algorithm that respects the nature of near-regular textures can produce more faithfully synthesized textures than general purpose state of the art synthesis algorithms.

1 Introduction

Textures are conventionally classified as either regular or stochastic textures[5]. However, many real-world textures fall somewhere in-between these two extremes. Most textures, along with regular and stochastic textures, form a texture spectrum on which the structure patterns vary continuously towards randomness(Figure 1). Ideally, a good texture synthesis algorithm should be able to handle all types of textures on the spectrum; however, the performance of existing texture synthesis algorithms varies on different types of textures. Moreover, the performance of a synthesis algorithm is usually judged by examining the results visually, which could be subjective and inconsistent among different people. To better evaluate the synthesis results, an objective and consistent criterion in addition to the visual inspection would be very helpful.

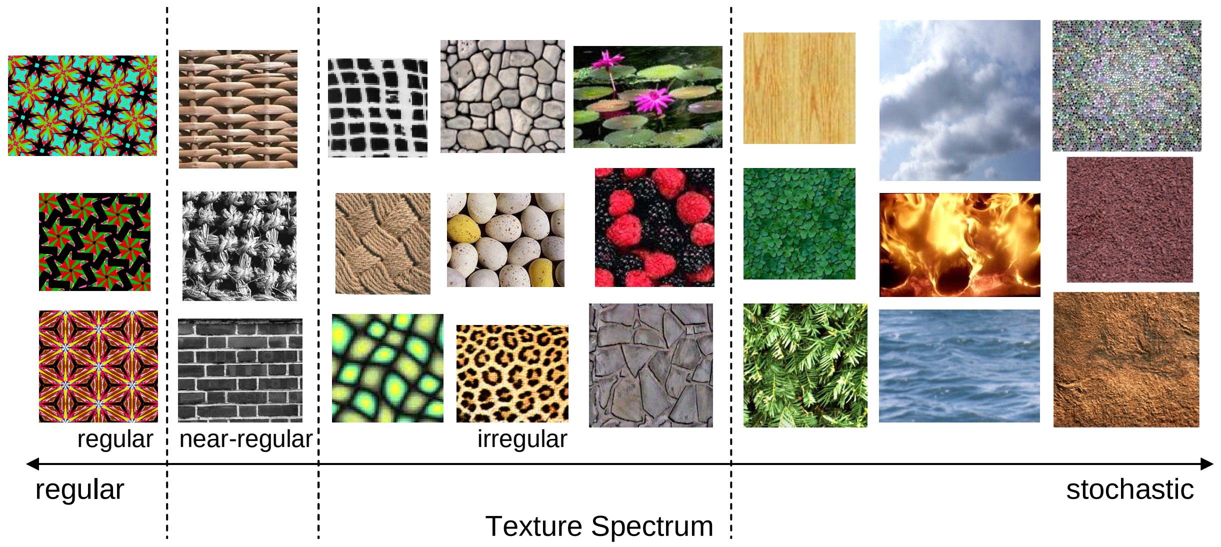


Figure 1: A texture spectrum on which textures are arranged according to the regularity of their structural variations, where irregular textures refers to geometrically-irregular near-regular textures.

The goal of this report is to compare the performance of texture synthesis algorithms in a more objective and consistent way. To satisfy this goal, testing samples should cover an appropriate scope of textures on the spectrum, and more importantly, the testing samples should also have a consistent property so that it is easy to verify whether the property is faithfully preserved in the synthesis process. For these reasons, we consider two particular types of textures in this report: regular and near-regular textures[13].

Regular textures are simply periodic patterns where the color/intensity and shape of all texture elements are repeating in equal intervals. That is, a texture element is a unit tile in a regular texture, which can be synthesized by tiling the space with the unit tile[1]. An example of regular textures is wallpaper. In the real-world, however, few textures are exactly regular. Most of the time, the textures we see in the real-world are near-regular, such as cloth, basket, windows, brick walls,

[1]The tile and texture element refer to the same concept and will be used interchangeably in this report.

building columns, carpet, blanket, honeycomb, etc. Near-regular textures can be considered as departures of regular textures in different spaces with different degrees. For example, in brick wall textures, the major departure happens in the color/intensity space as the shape of each brick is regular but the color/intensity may vary. On the other hand, the perspectively-distorted window texture in Figure 2 departs mainly in the geometry space while the color/intensity of each window is almost regular. Figure 1 shows a texture spectrum where the textures are arranged according to the regularity variation in geometry space.

In this report, we will only consider a subclass of near-regular textures: **geometrically-regular near-regular textures**, of which the texture elements only vary in color/intensity but are almost regular in geometry. In other words, the texture elements can be defined by a regular lattice. For simplicity, we will call geometrically-regular near-regular textures ”near-regular textures (NRT)” in the rest of the report.

The global regularity of the geometrically-regular near-regular textures is a consistent property that can be used to evaluate the synthesis results. Moreover, the departures in color/intensity space of near-regular textures provides another property to evaluate synthesis algorithms.

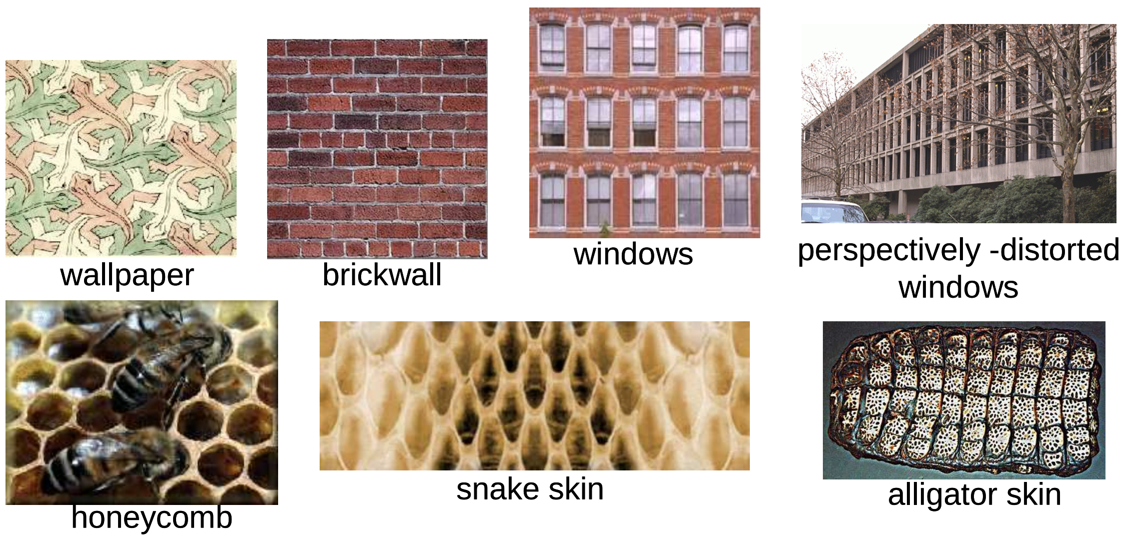


Figure 2: Examples of regular and near-regular textures.

2 Selected texture synthesis algorithms

Work in texture synthesis has achieved impressive results in a variety of cases[1, 2, 3, 4, 5, 6, 7, 9, 14, 15, 16, 17]. These algorithms can be roughly divided into two groups: the statistical-model-based approach[2, 3, 6, 14] and image-based approach[1, 4, 5, 7, 9, 15, 16, 17]. The statistical-model-based approach is closely related to texture analysis and classification in that a statistical model is constructed based on the input textures and this model is then used for texture analysis or synthesis. Due to its statistical nature, this approach performs much better on stochastic textures than on structural textures. This is because the statistical-model-based approach synthesizes textures by re-sampling image pixels from a statistical model, and thus can not guarantee that structures in the structural textures will be preserved–the boundaries of the structures may be blurred or even broken. On the other hand, the image based approach synthesizes textures by directly copying image pixels or patches from the input texture and stitching them together in the synthesized image. Because the image-based approach tries to keep the image pixels untouched as much as possible, image details can be well preserved in the synthesized textures. However, these are local approaches in nature, with no special consideration given to the texture's global structures. In general, the image-based approach performs better on structural textures than the statistical-model-based approach.

In this comparison study, we consider the image-based synthesis algorithms because we are interested in regular and near-regular texture synthesis. In particular, we compare the graph cuts approach[7], near-regular texture synthesis[13], the patch-based approach[9], and the regularized patch-based approach developed by one of the authors. We briefly describe these algorithms here. Readers who are interested in these algorithms are encouraged to read the original papers for details.

2.1 Graph cuts texture synthesis

Kwatra et al.[7] demonstrate a very effective general texture synthesis algorithm. Texture is synthesized by overlaying the entire input texture onto the synthetic texture at various offsets and using a graph cut algorithm to find the optimal region to add to the synthetic texture. The graph cut algorithm avoids the need for a fixed, a-priori patch size, and scales well to any dimension (such as video). However, for near regular textures the choice of offsets is as important as finding low-error seams. If the input texture is copied onto the synthesized texture at an offset that is inconsistent with the periodicity of the texture, any selection of seams will still violate the global regularity of the texture. Kwatra et al. describe patch placement algorithms which do a fair job of finding low error offsets. The error is defined as the sum of squared difference (SSD) between the pixels in the overlapping region of the input texture and the texture being synthesized. They treat the input texture as a template and compute the correlation between the template and the texture being synthesized to find the low error offsets. The minimum error (or maximum correlation) offsets often, but not always, correspond to the offsets preserving the periodicity of the input texture.

2.2 Near-regular texture synthesis

Liu et al.[13] propose a texture synthesis algorithm for geometrically-regular near-regular textures[2]. The basic idea is to utilize the translational symmetry property[10][11] of a near-regular texture to find the underlying lattice structure of the texture patterns and locate the texture elements. These tiles represent the smallest parallelogram-shaped region on a regular texture that can reproduce the texture pattern under the texture's translation subgroup. For a regular texture, only

[2]Although the name of the synthesis algorithm is near-regular texture synthesis, the algorithm only deals with geometrically-regular near-regular textures.

one tile is needed for recovering the full texture. For a near-regular texture, one needs a set of tiles collected by sampling the input texture in a principled manner to preserve both the geometric regularity and color/intensity variations in the input texture[12]. The tiles in the tile set have roughly the same size and shape but varied color/intensity. The output texture is synthesized by randomly picking a tile from the tile set and pasting the tile to the synthesizing image with overlapping on lattice points. Dynamic programming and image blending techniques are applied to the overlapping regions to stitch the tiles.

2.3 Patch-based texture synthesis

Liang et al.[9] develop a patch-based synthesis algorithm. The basic idea of the algorithm is to synthesize textures by directly copying image patches from the input texture. The major difference from other image-based approaches is that they apply a modified approximate nearest neighbor technique to speed up the search for the best matched patch. With this improved search speed, the algorithm can run in real-time and reach similar image quality as other image-based synthesis algorithms. The image feathering technique is used in the patch-based synthesis approach to blend the overlapping regions of patches. This might blur the overlapping region slightly compared to the dynamic programming technique used in near-regular texture synthesis or the graph cut technique in the graph cuts synthesis approach.

Patch placement in the patch-based approach is very different from that in near-regular texture synthesis. In the patch-based approach, the patch is rectangular and the patches are pasted in a scan-line order. Since the patch size and placement offset are arbitrarily defined by a user, they may not match the lattice structure of the input near-regular texture.

2.4 Regularized patch-based texture synthesis

We develop a regularized patch-based texture synthesis algorithm to deal with near-regular textures[3] in which each texture element may not be well circumscribed by a parallelogram. We allow the parallelograms on a regular lattice to be deformed to quadrilaterals so that the texture elements can be separated by the deformed lattice. In other words, we deform a geometrically-irregular near-regular texture to a geometrically-regular near-regular texture. We then apply a modified patch-based approach to synthesize the geometrically-regular texture. Our modification to the patch-based approach allows patches to be pasted along the lattice axis direction and allow the patch shape to be a parallelogram rather than a rectangle. The patch-size and lattice construction vectors are provided by a user who identifies the underlying lattice structure of the input near-regular texture. A synthesized inverse deformation is used to warp the synthesized regular texture to a near-regular texture. We describe the details of the algorithm in the appendix.

[3]Strictly speaking, the algorithm can handle a near-regular texture which has both geometric variation and color/intensity variation. Since we only compare the synthesis performance on geometrically-regular near-regular textures in this study, the capability of the algorithm to handle the geometrically-irregular textures would not be addressed in this report.

	Graph cuts	**Near-regular synthesis**	**Regularized patch-based**	**Patch-based**
Patch shape/size	input texture image	tile shape/size from translational symmetry analysis	user identified lattice, quadrilateral patch	user defined, rectangular patch patch
Patch placement	random or maximal correlation locations	lattice points	lattice points	patch grids
Patch stitching	graph cut & blending	dynamic programming & blending	image-feathering	image-feathering

Table 1: Summary of four synthesis algorithms. A *patch* is a 2D sample of neighboring pixels extracted from the input texture image. *Patch shape/size* refers to how the shape and size of the region are determined when extracting image pixels from the input texture, while *patch placement* and *stitching* refer to how the patches are placed and stitched in the synthesized texture.

To conclude this section, we summarize these four algorithms in Table 1. We compare the patch shape/size determination, patch placement, and patch stitching methods used in these algorithms, where a *patch* is a 2D sample of neighboring pixels extracted from the input texture. *Patch shape/size* refers to how the shape and size of the region are determined when extracting image pixels from the input texture, while *patch placement* and *stitching* refer to how the patches are placed and stitched in the synthesized texture.

3 Results

We compare the synthesis results of these four algorithms on regular textures (Figure 3-7) and near-regular textures (Figure 8-45). For the convenience of comparison, we summarize the synthesis performance of these four algorithms on regular textures in Table 2 and near-regular textures in Table 3. These two tables record whether the global regularity is preserved in the synthesized texture. In addition, we include the synthesis results by the *image quilting* approach[4] for six textures on their website[4](Figure 16, 18, 22, 27, 35, 36). Among the six tested textures, two synthesized results of image quilting preserve the global regularity(Figure 22 and 27).

From Table 2, we find that all four algorithms can handle the regular textures very well, but they perform quite differently on near-regular textures, as shown in Table 3. It may not be surprising that the near-regular texture synthesis algorithm performs best among the four because it is specially designed for near-regular texture synthesis. The graph cuts approach and patch-based approach do not preserve the global regularity well most of the time. This is because these two algorithms do not explicitly model the underlying lattice structure of the input texture. It is interesting to compare the patch-based approach and the regularized patch-based approach. Because the latter utilizes a user-specified lattice of the input texture in the synthesis process, it preserves the global regularity

[4]http://www.cs.berkeley.edu/ efros/research/quilting.html.

Textures	Description	**Graph cuts**	**Near-regular synthesis**	**Regularized patch-based**	**Patch-based**
Figure 3	wallpaper	✓	✓	✓	✓
Figure 4	wallpaper	✓	✓	✓	✓
Figure 5	wallpaper	✓	✓	✓	✓
Figure 6	jigsaw puzzle	✓	✓	✓	✓
Figure 7	pavement tiles	✓	✓	✓	✓
success rate	preserved/total	100%	100%	100%	100%

Table 2: Comparison of whether four algorithms preserve global regularity in regular textures, where ✓denotes that regularity is preserved.

much better than the patch-based approach.

Although the graph cuts approach does not utilize the lattice structure in the synthesis process, it still performs well on a few near-regular textures and much better than the patch-based approach(Table 3). The reason for this is that the algorithm incorporates a correlation technique to determine the best pasting location so that the underlying periodicity, if it exists, can be preserved. This correlation-based patch placement works well on regular textures and some near-regular textures; however, for near-regular textures in which the color/intensity of texture elements is not regular, the correlation technique can not guarantee the regularity to be maintained globally. This is especially true when the input texture contains an interlocked structure, like woven fabric or brick walls (Figure 23, 28, 34, 36, 37, 38, 40, 41). Figure 46 shows several synthesis results of a brick wall texture by different patch placement settings in the graph cuts. This example shows that the correlation technique can not work well on discovering the regularity of near-regular textures in this brick wall texture. Synthesis of the same texture by the near-regular synthesis approach is shown in Figure 36.

A limitation of the near-regular texture synthesis algorithm is that the input texture sample must contain at least two complete tiles so that the underlying lattice and the tile set are well defined. If no complete tile exists in the texture, the algorithm can not produce good results. This is the situation in Figure 31, where the input texture contains two overlapping of periodic patterns (squares and hexagons), but the sampling area is too small to have a complete tile that covers a full period of the composed periodic pattern[5]. The near-regular texture synthesis algorithm preserves the square pattern because it demonstrates stronger periodicity, but the hexagon net is discontinuous between squares. Another failure example of the near-regular texture synthesis algorithm is the thin line jigsaw puzzle texture in Figure 33. In this case, the input texture does not contain a complete tile either. This can be observed from the upper-right image in which the pattern in the circled region only appears once in the texture.

[5]In fact, it is not guaranteed that a two-dimensional pattern composed of two periodic two-dimensional patterns remains to be a periodic pattern.

Textures	Description	**Graph cuts**	**Near-regular synthesis**	**Regularized patch-based**	**Patch-based**
Figure 8	punched card	✓	✓	✓	×
Figure 9	hexagonal net	✓	✓	✓	✓
Figure 10	metal	×	✓	✓	×
Figure 11	ceramic tiles	✓	✓	✓	×
Figure 12	fish tiles	✓	✓	✓	×
Figure 13	wall	✓	✓	✓	×
Figure 14	squares	×	✓	✓	×
Figure 15	pavement tiles	×	✓	×	×
Figure 16	cans	✓	✓	✓	×
Figure 17	swirl	✓	✓	✓	×
Figure 18	basket	×	✓	✓	×
Figure 19	fabric	✓	✓	✓	✓
Figure 20	fabric	✓	✓	✓	✓
Figure 21	fabric	✓	✓	✓	✓
Figure 22	knotted mat	✓	✓	✓	×
Figure 23	pie	×	✓	×	×
Figure 24	fabric	✓	✓	✓	×
Figure 25	toothpastes	✓	✓	✓	✓
Figure 26	windows	✓	✓	✓	×
Figure 27	windows	✓	✓	✓	×
Figure 28	fabric	×	✓	✓	×
Figure 29	basket	✓	✓	×	×
Figure 30	fabric	✓	✓	N/A	N/A
Figure 31	squares&hexagons	×	×	N/A	N/A
Figure 32	mosaic	×	✓	N/A	N/A
Figure 33	jigsaw puzzle	✓	×	N/A	N/A
Figure 34	fabric	×	✓	N/A	N/A
Figure 35	cracker	×	✓	N/A	N/A
Figure 36	brick wall	×	✓	N/A	N/A
Figure 37	brick wall	×	✓	N/A	N/A
Figure 38	brick wall	×	✓	N/A	N/A
Figure 39	brick wall	✓	✓	N/A	N/A
Figure 40	brick wall	×	✓	N/A	N/A
Figure 41	brick wall	×	✓	N/A	N/A
Figure 42	carpet	✓	✓	N/A	N/A
Figure 43	rug	×	✓	N/A	N/A
Figure 44	rug	×	✓	N/A	N/A
Figure 45	cans	×	✓	N/A	N/A
success rate	preserved/total	53%	95%	86%	23%

Table 3: Comparison of whether four algorithms preserve global regularity in near-regular textures, where ✓denotes that regularity is preserved, and × denotes that not preserved.

4 Conclusion

In this report, we compare the performance of four texture synthesis algorithms on regular and near-regular textures. Because of the global regularity property of the regular and near-regular textures, we are able to provide a more consistent and objective criterion to evaluate the synthesis results.

This comparison shows that near-regular texture synthesis remains a challenge for several state-of-the-art algorithms, such as the graph cuts[7], patch-based[9], and image quilting methods[4]. The results from the regularized patch-based approach and the near-regular texture synthesis approach show that the global regularity of the near-regular textures can be better preserved if the synthesis algorithm analyzes the underlying lattice structure and uses this information in the synthesis process. In fact, the synthesis results by the near-regular texture synthesis algorithm demonstrate that both the global regularity and local randomness of a near-regular texture can be faithfully preserved.

Acknowledgment

The authors would like to thank Yanghai Tsin for providing his code for an earlier version of the near-regular texture synthesis algorithm. This research is partially funded by NSF grant #IIS-0099597. The regularized-patch-based method was implemented when Wu was an intern at Microsoft Research Asia, jointly advised by Dr. Liu and Dr. Harry Shum.

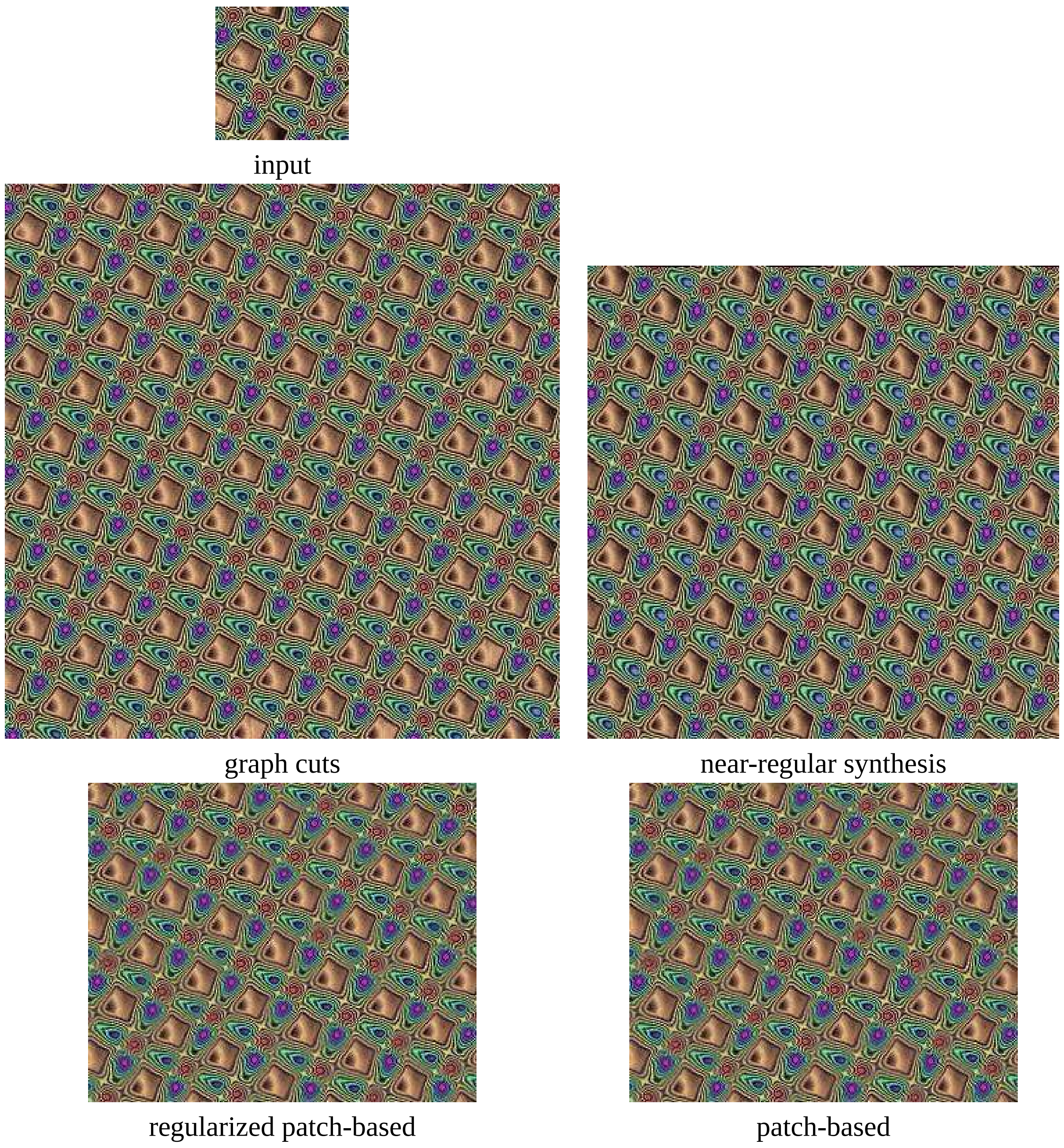


Figure 3: Synthesis results of a wallpaper texture.

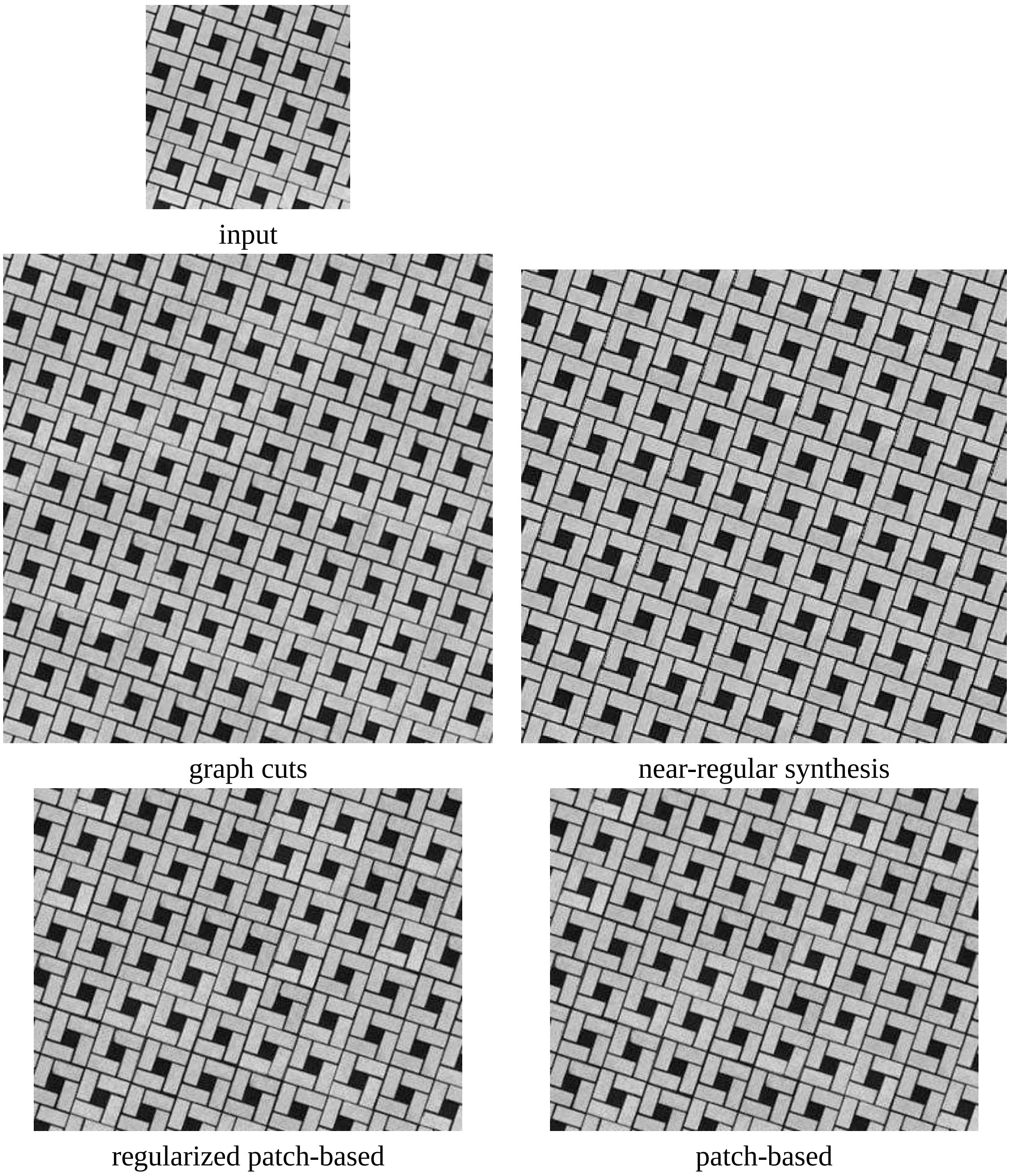


Figure 4: Synthesis results of a wallpaper texture.

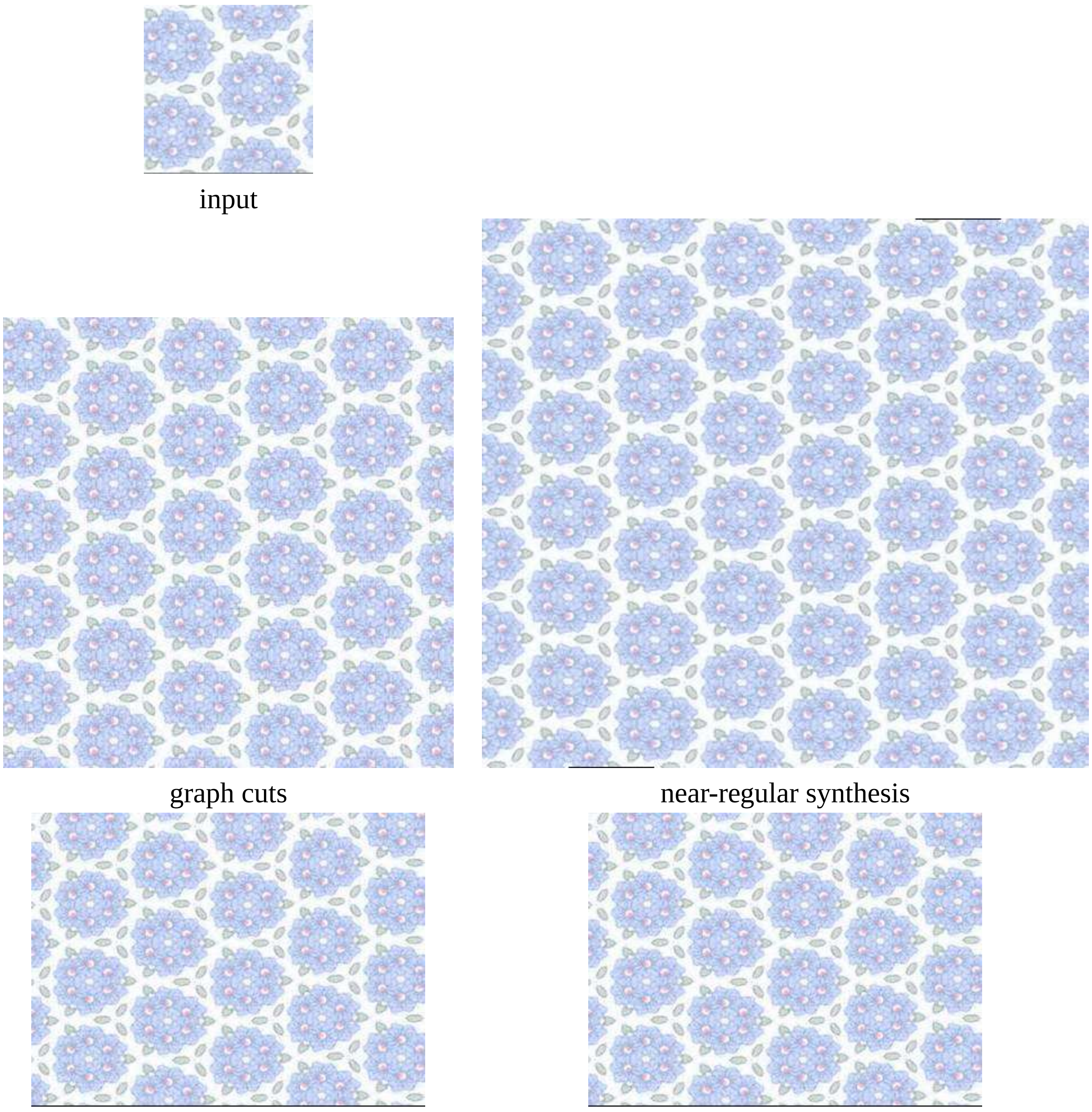


Figure 5: Synthesis results of a wallpaper texture.

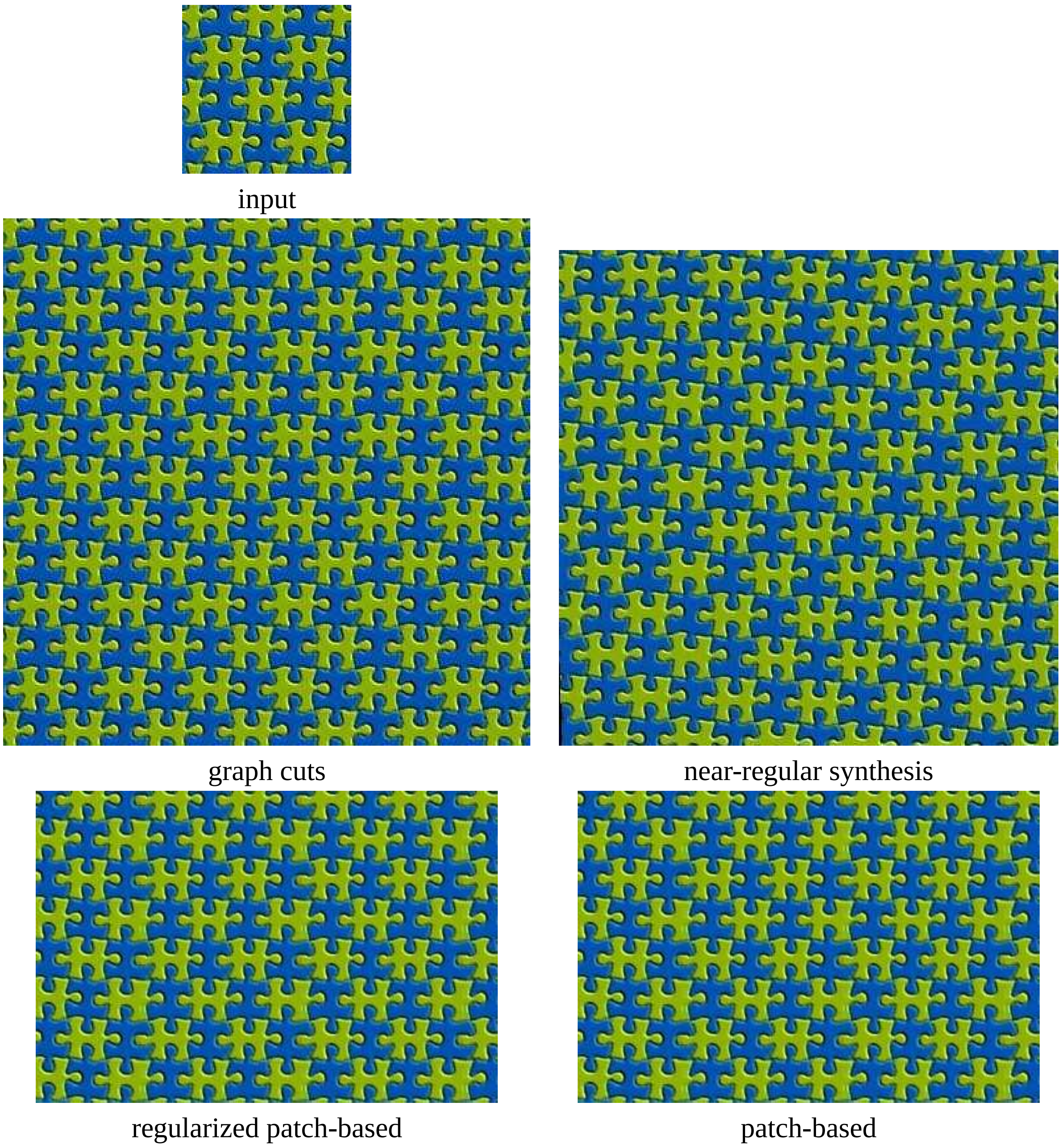
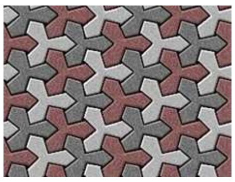
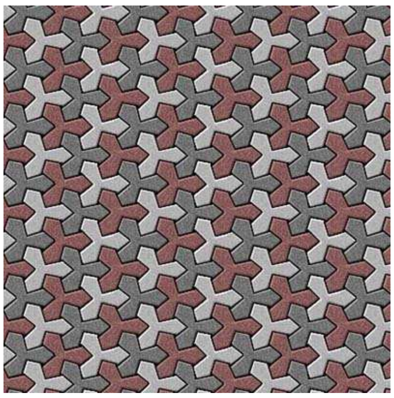
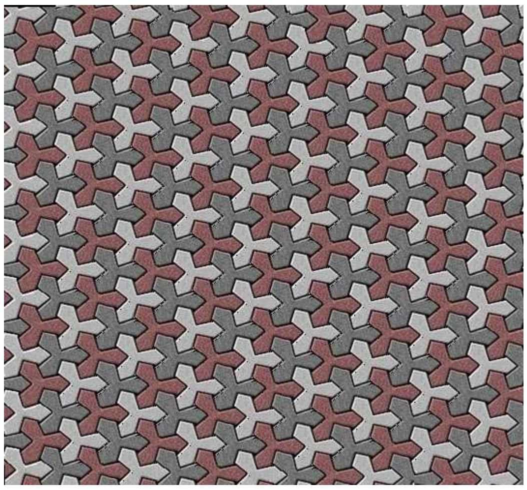
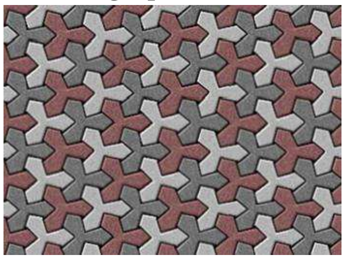


Figure 6: Synthesis results of a jigsaw puzzle texture.


input


graph cuts


near-regular synthesis


regularized patch-based

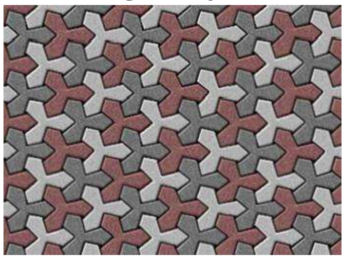
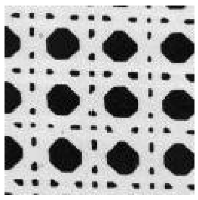
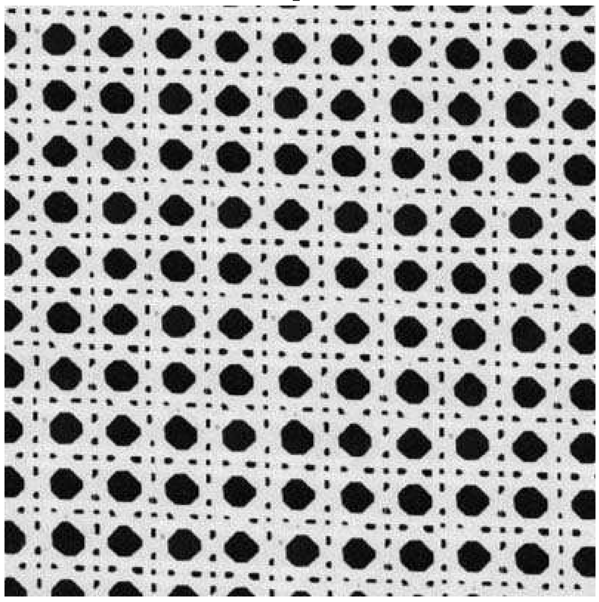

patch-based

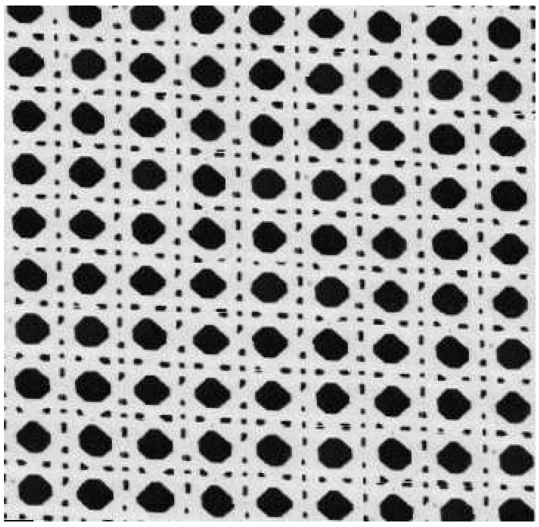
Figure 7: Synthesis results of a pavement tile texture.



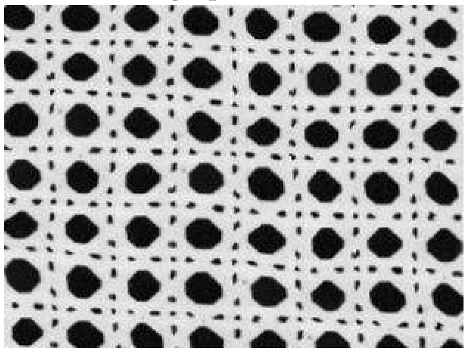
input



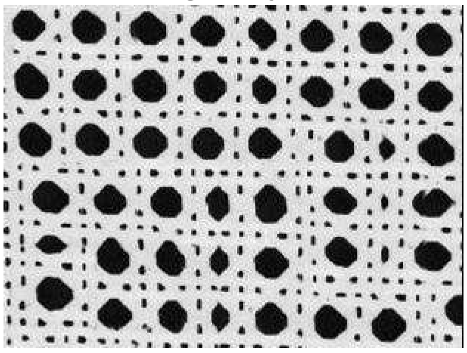
graph cuts



near-regular synthesis

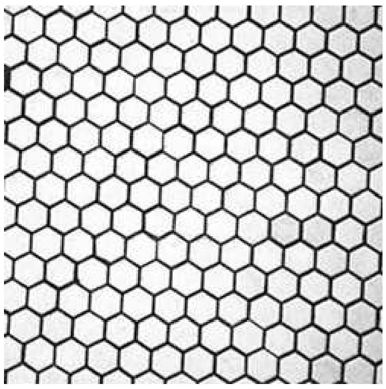


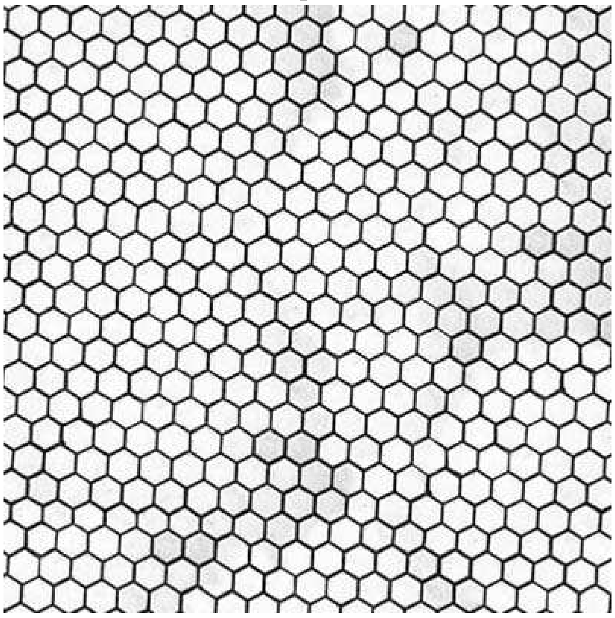
regularized patch-based

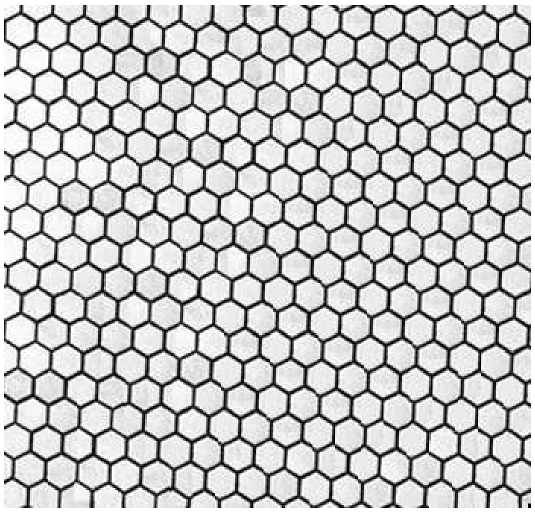


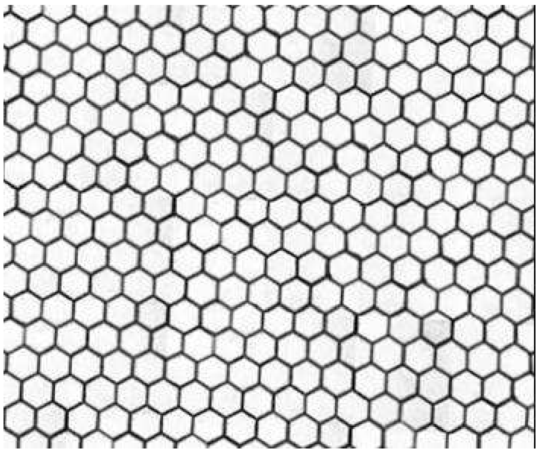
patch-based

Figure 8: Synthesis results of a punched card texture. Global regularity is not preserved in the patch-based result.


input


graph cuts


near-regular synthesis


regularized patch-based

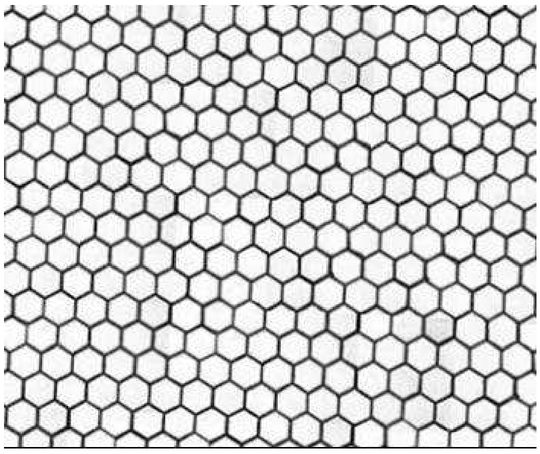

patch-based

Figure 9: Synthesis results of a hexagonal net texture.

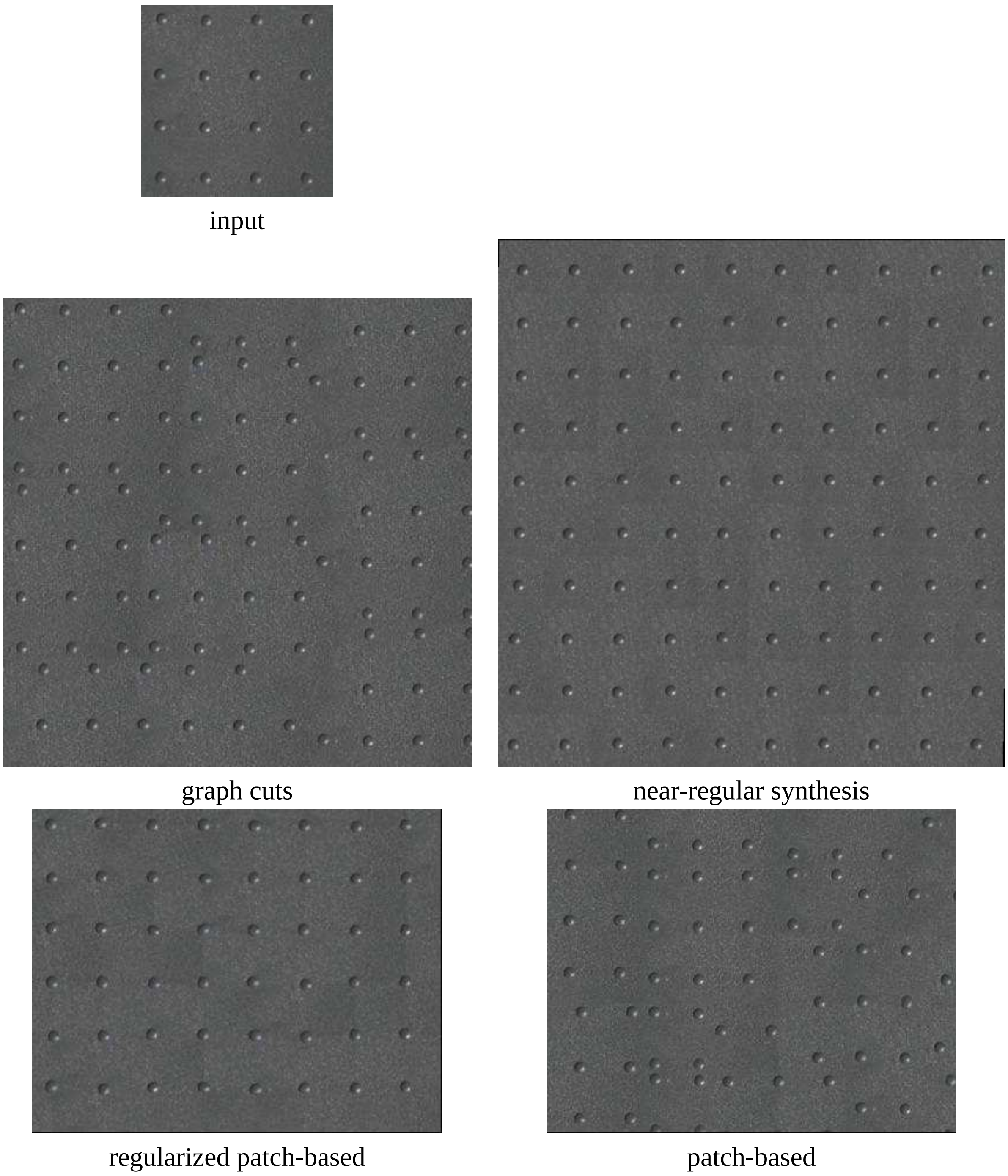


Figure 10: Synthesis results of a metal texture. Global regularity is not preserved in the graph cuts and patch-based results.

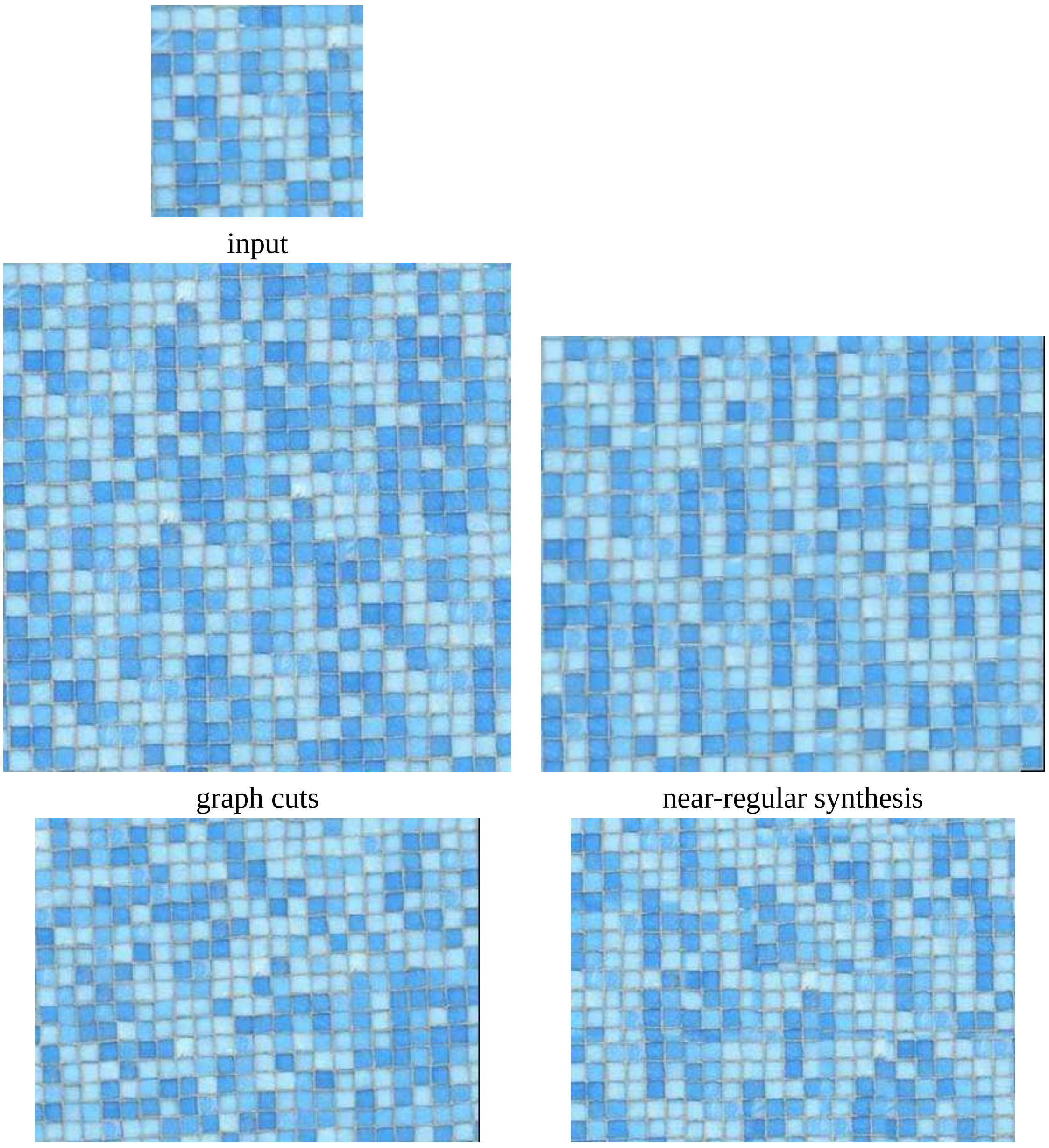


Figure 11: Synthesis results of a ceramic tile texture. The result by the patch-based approach does not preserve the regularity of the tile size in the middle part of the bottom rows.

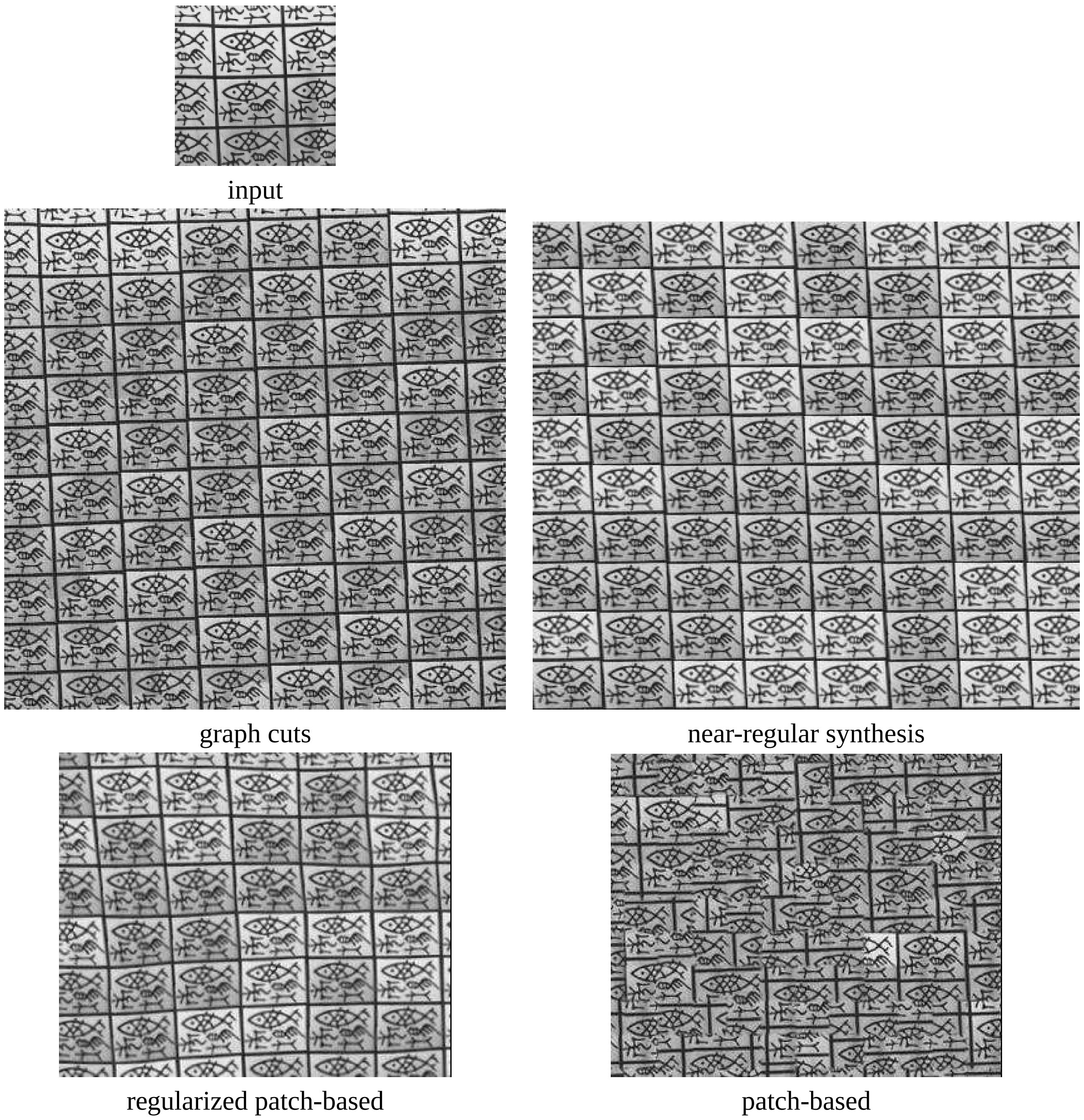


Figure 12: Synthesis results of a fish tile texture. The patch-based result does not preserve the global regularity.

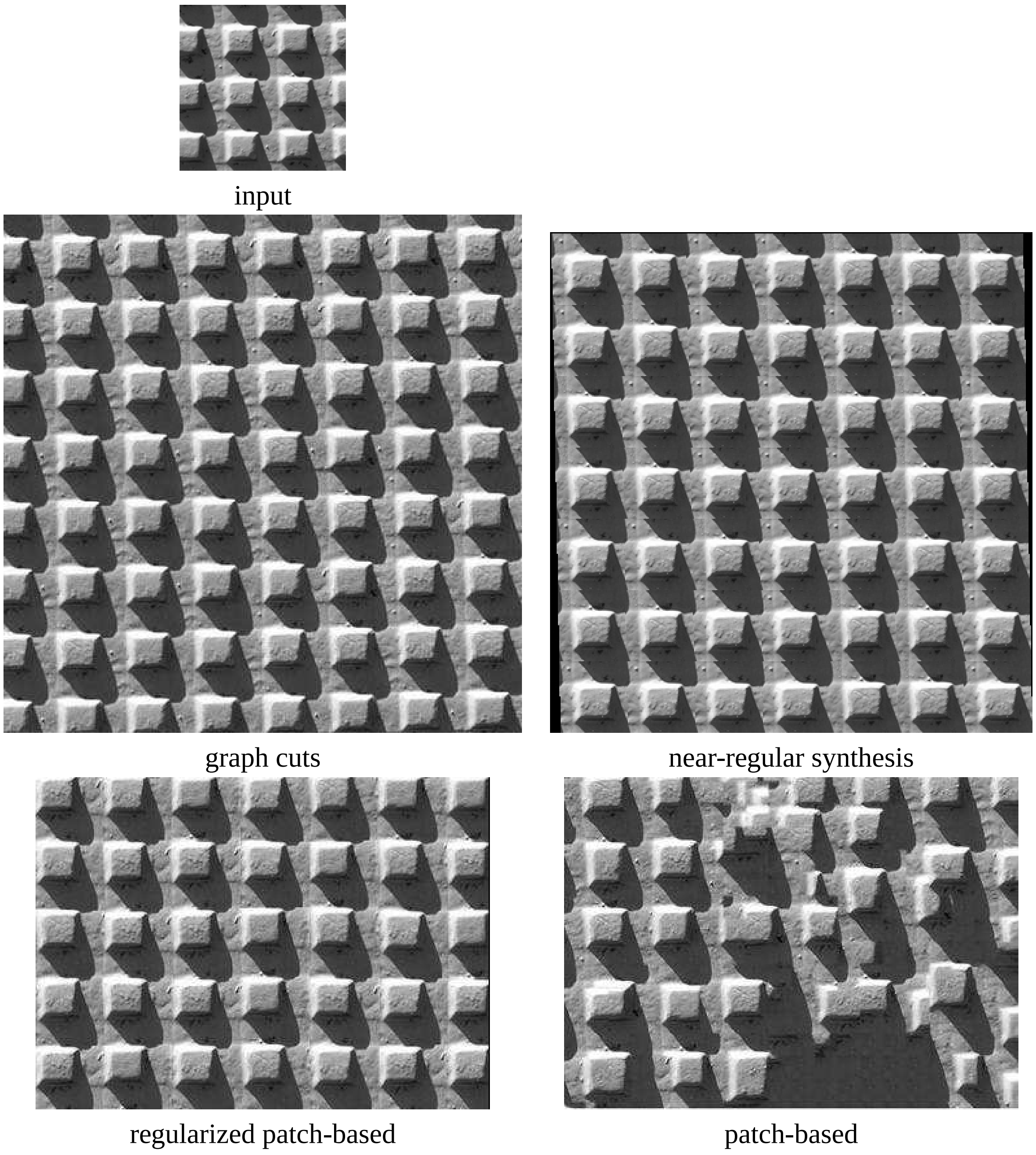


Figure 13: Synthesis results of a shaded wall texture. The patch-based result does not preserve the global regularity.

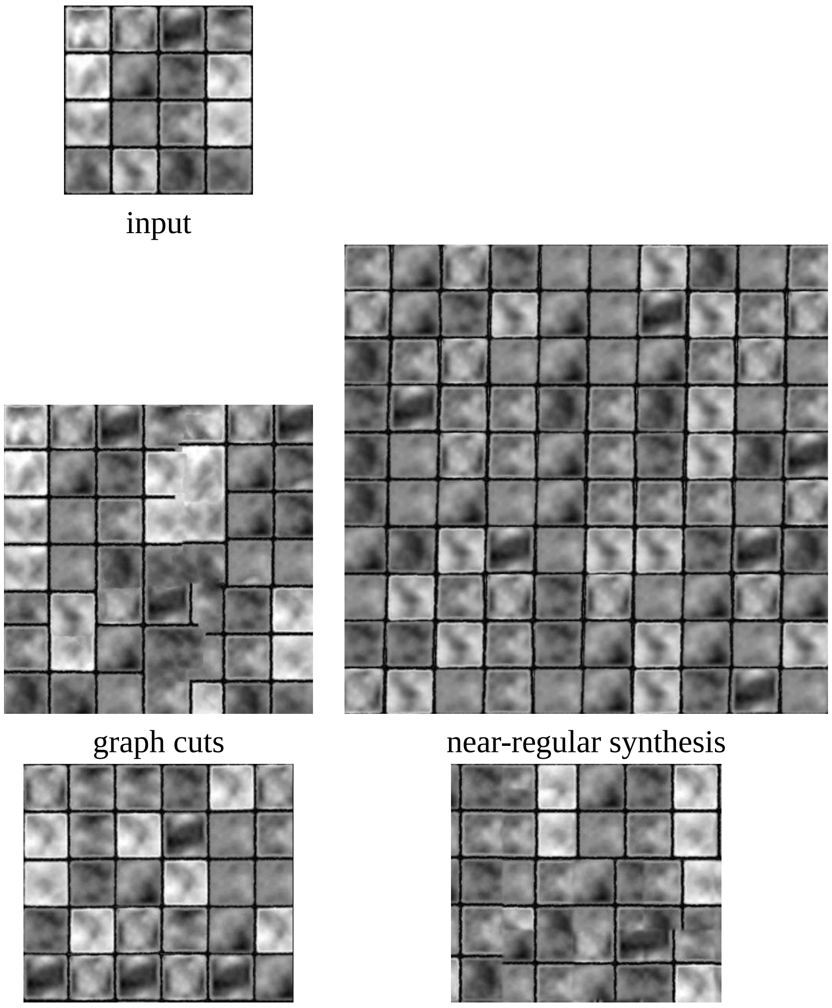


Figure 14: Synthesis results of a texture of squares. Regularity is not preserved in the graph cuts and patch-based results.

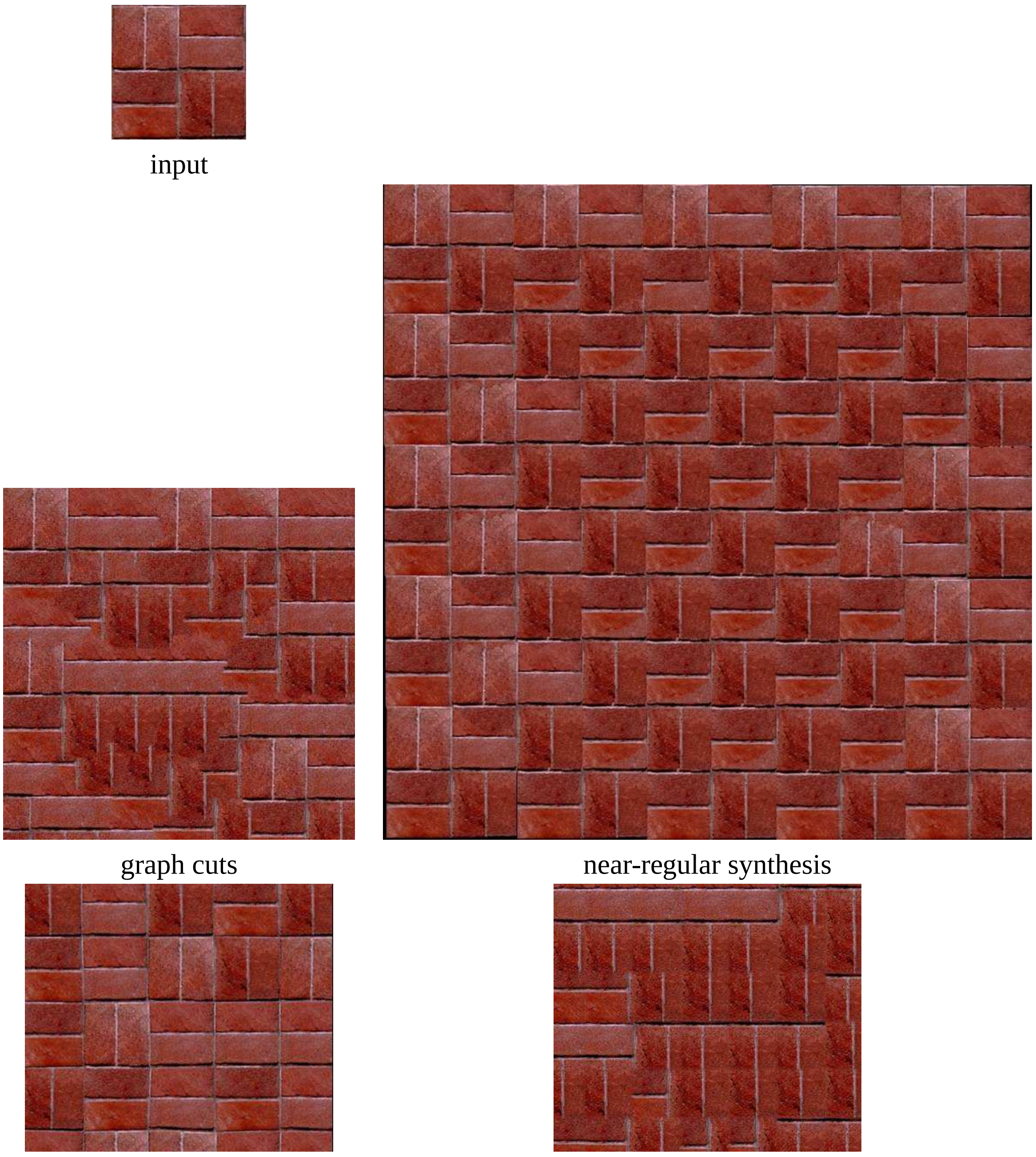
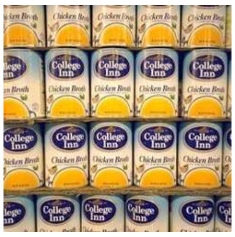


Figure 15: Synthesis results of a pavement tile texture. Only the near-regular synthesis result preserves the global regularity of the squares and vertical/horizontal line pattern.


input

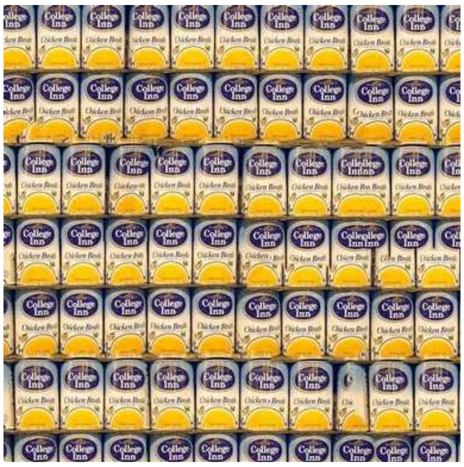
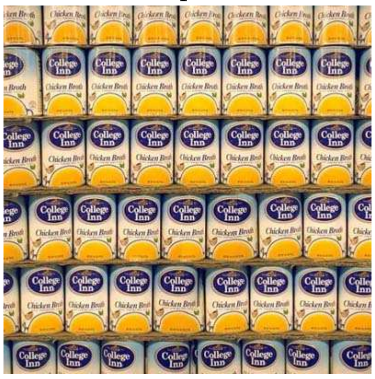
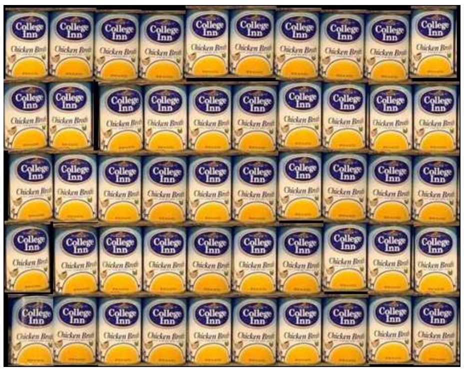
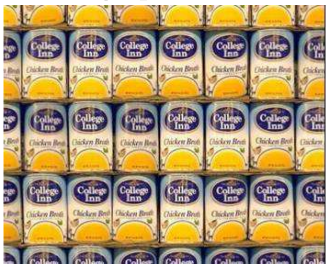


image quilting


graph cuts


near-regular synthesis


regularized patch-based

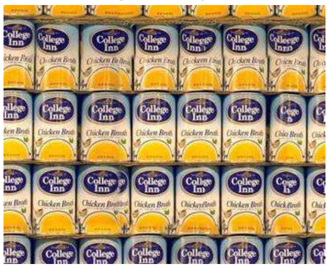

patch-based

Figure 16: Synthesis results of a texture of cans. Global regularity is not preserved in the patch-based and image quilting results. The cans in the third column of the patch-based result and the third column from the right of the image quilting approach are not synthesized correctly.

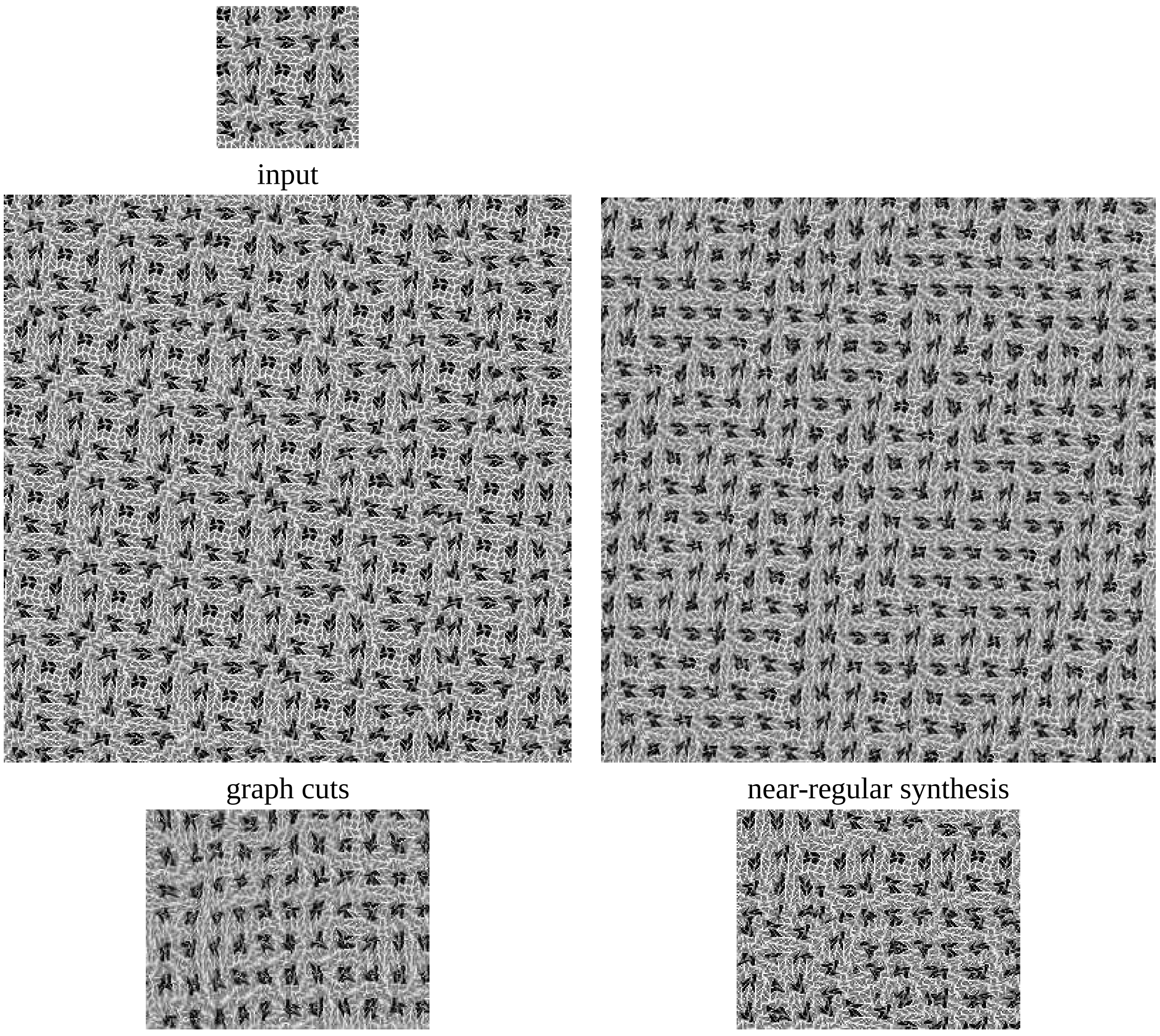


Figure 17: Synthesis results of a swirl texture. The regularity is not preserved in the bottom rows of the patch-based synthesis result.

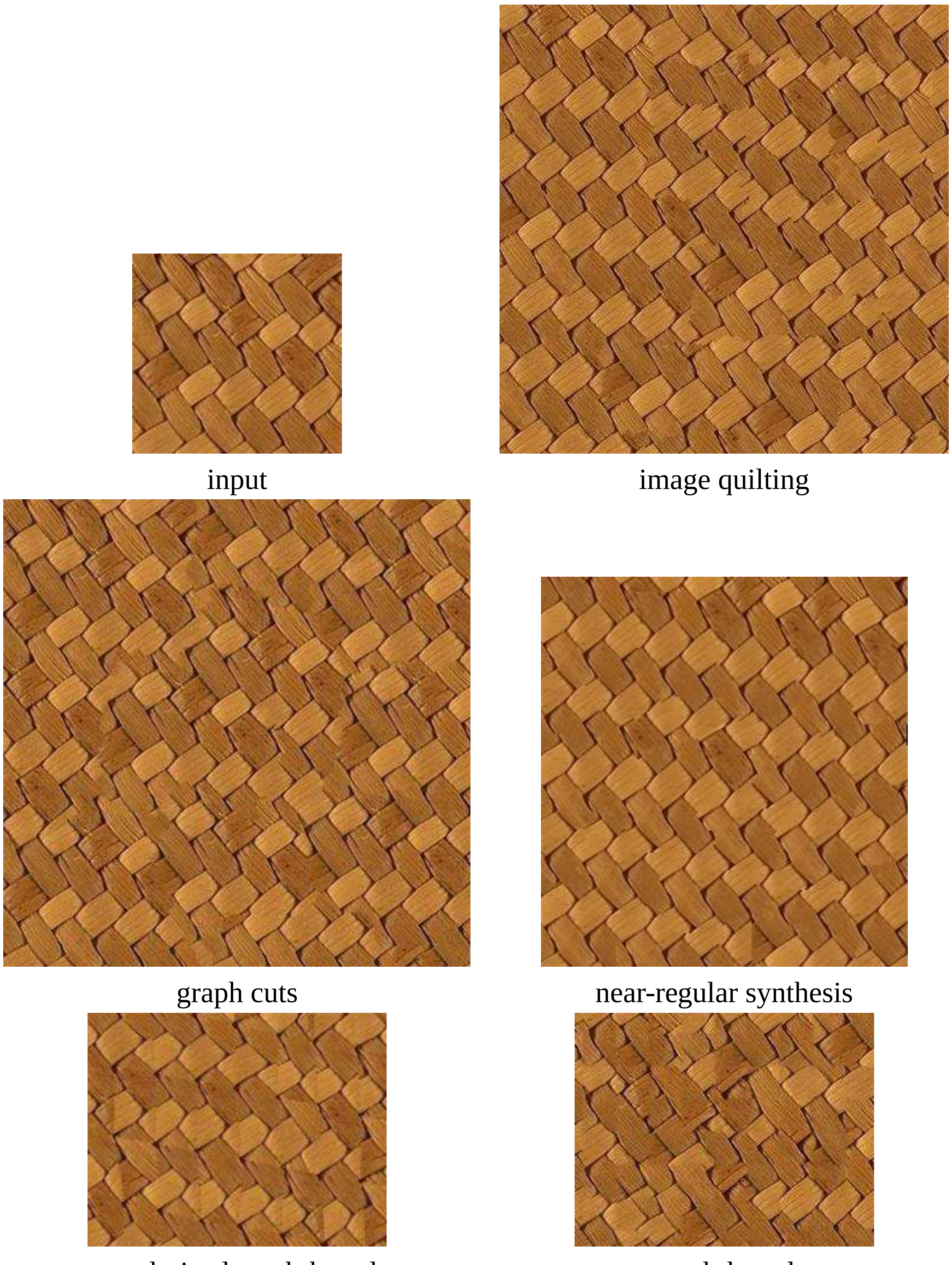
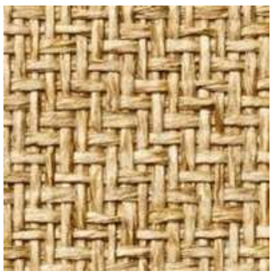
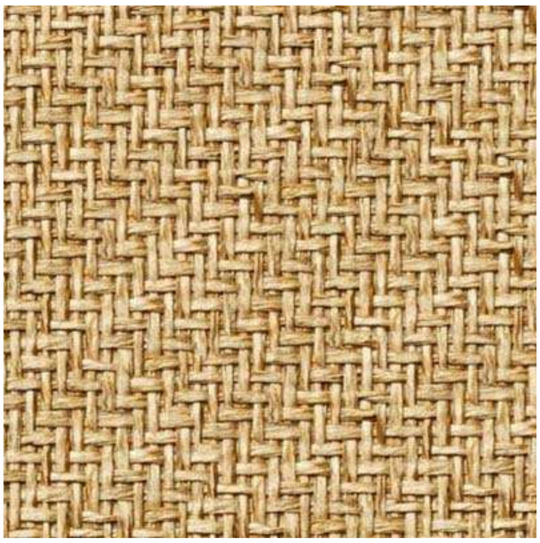
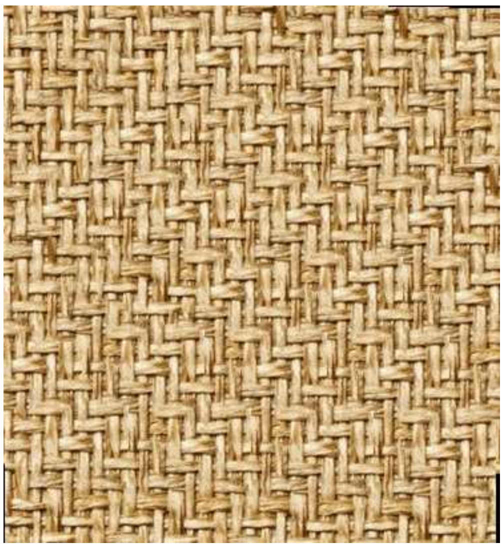
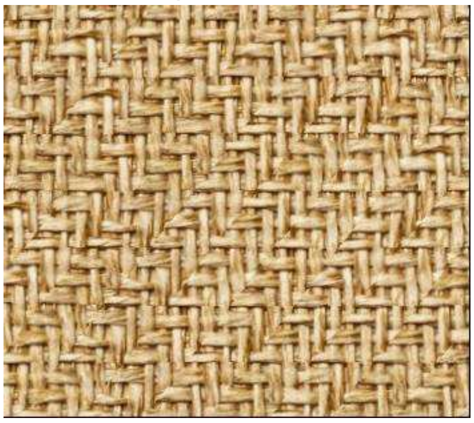


Figure 18: Synthesis results of a basket texture. Global regularity is not preserved in the graph cuts, patch-based, and image quilting approaches. The interwoven structure is not maintained in the lower-left portion of the graph cuts and the central bottom part of the image quilting approach.


input


graph cuts


near-regular synthesis


regularized patch-based

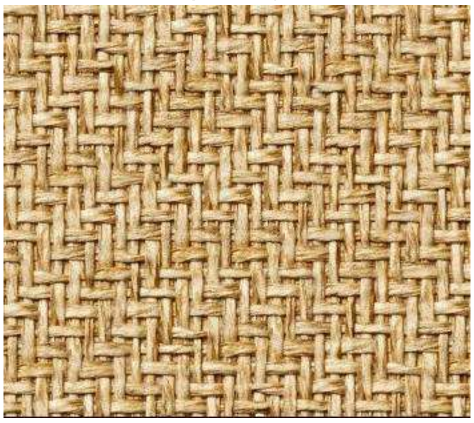
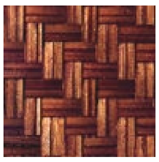
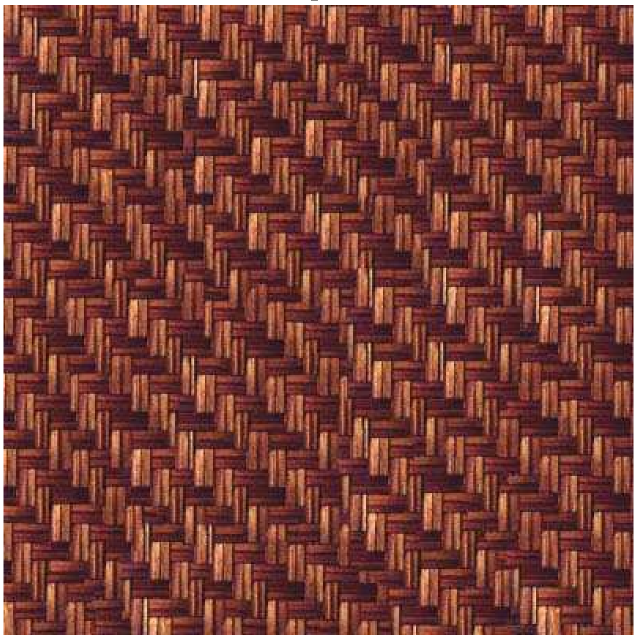
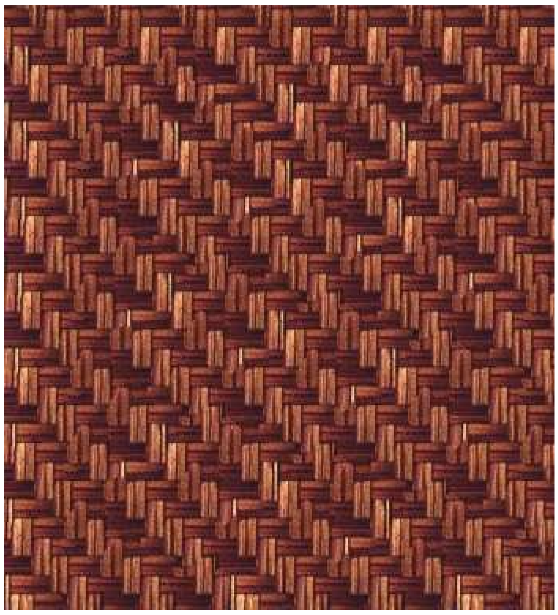
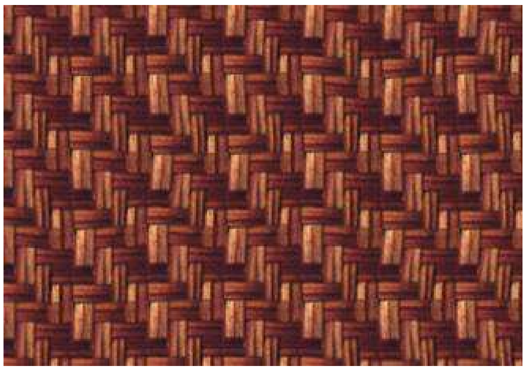

patch-based

Figure 19: Synthesis results of a fabric texture.


input


graph cuts


near-regular synthesis 1


regularized patch-based

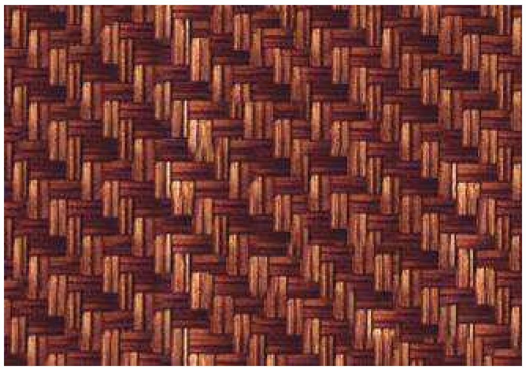
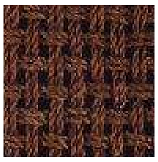
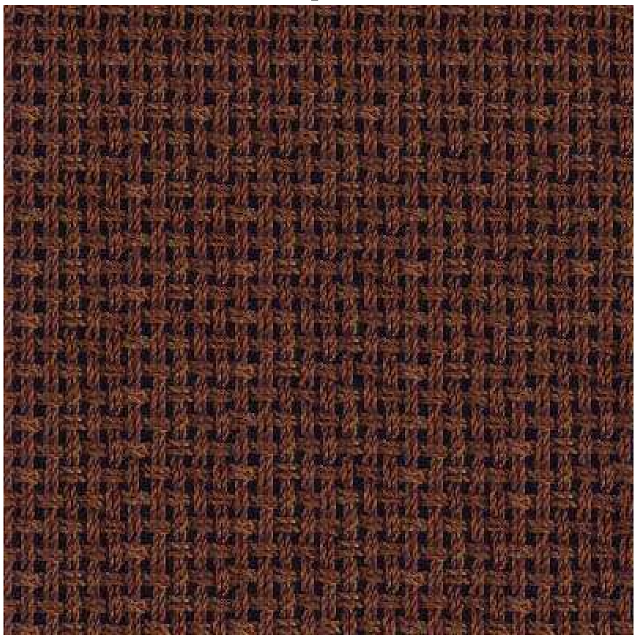

patch-based

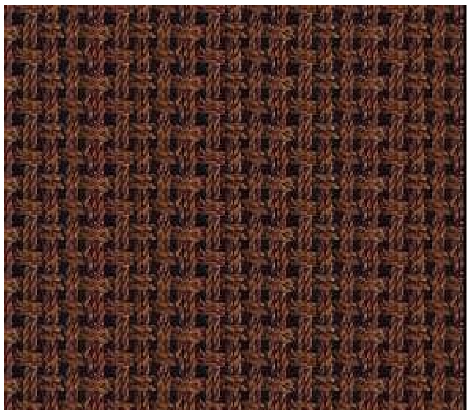
Figure 20: Synthesis results of a fabric texture.



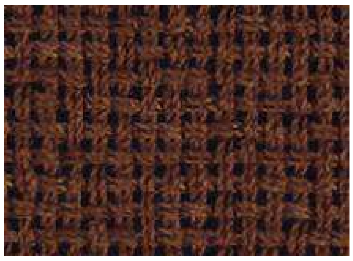
input



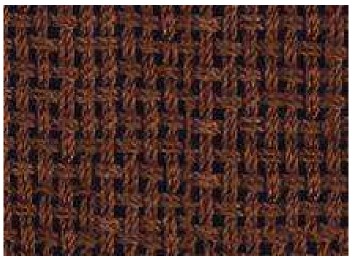
graph cuts



near-regular synthesis



regularized patch-based



patch-based

Figure 21: Synthesis results of a fabric texture.

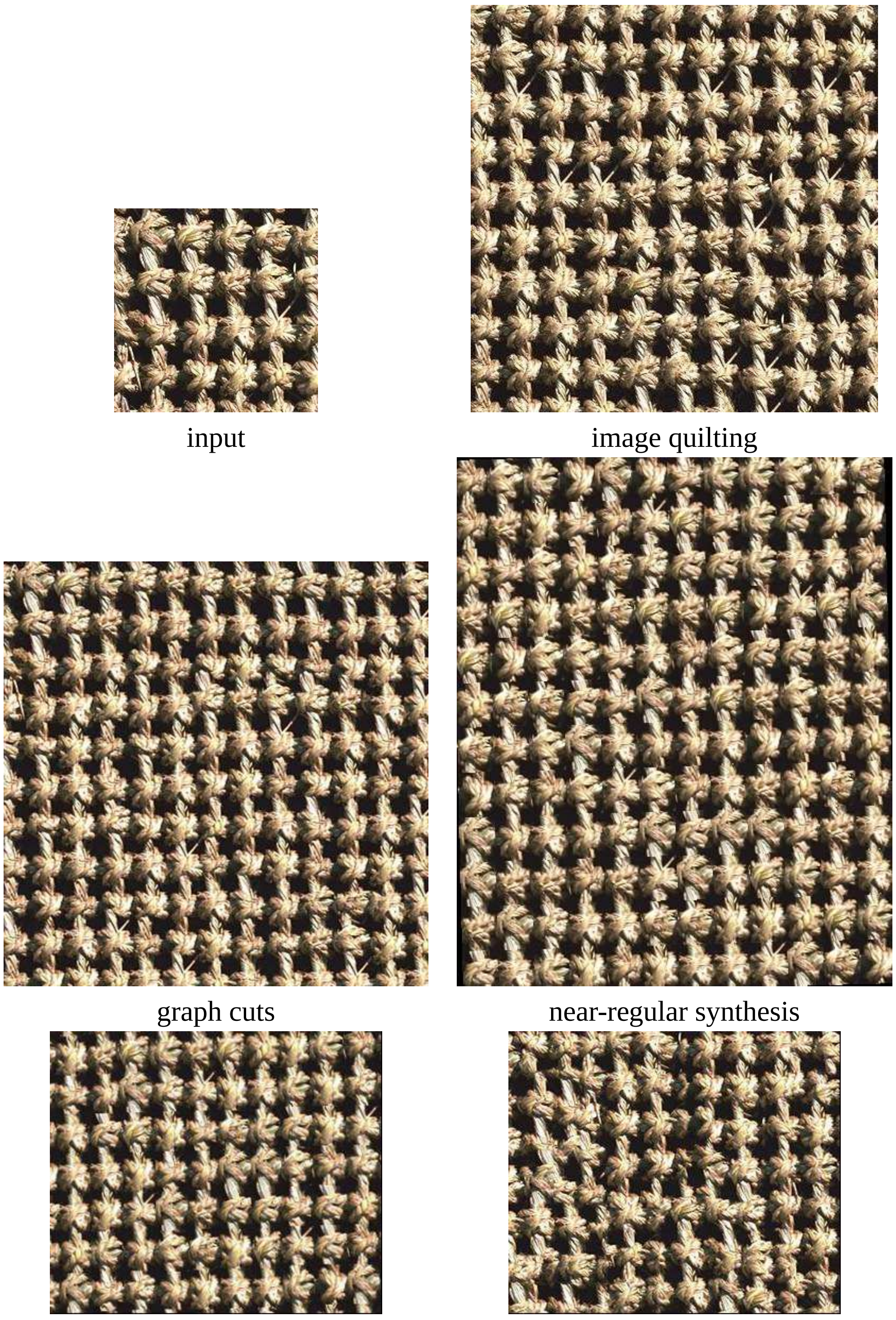


Figure 22: Synthesis results of a knotted mat texture. Global regularity is not preserved in the patch-based result.

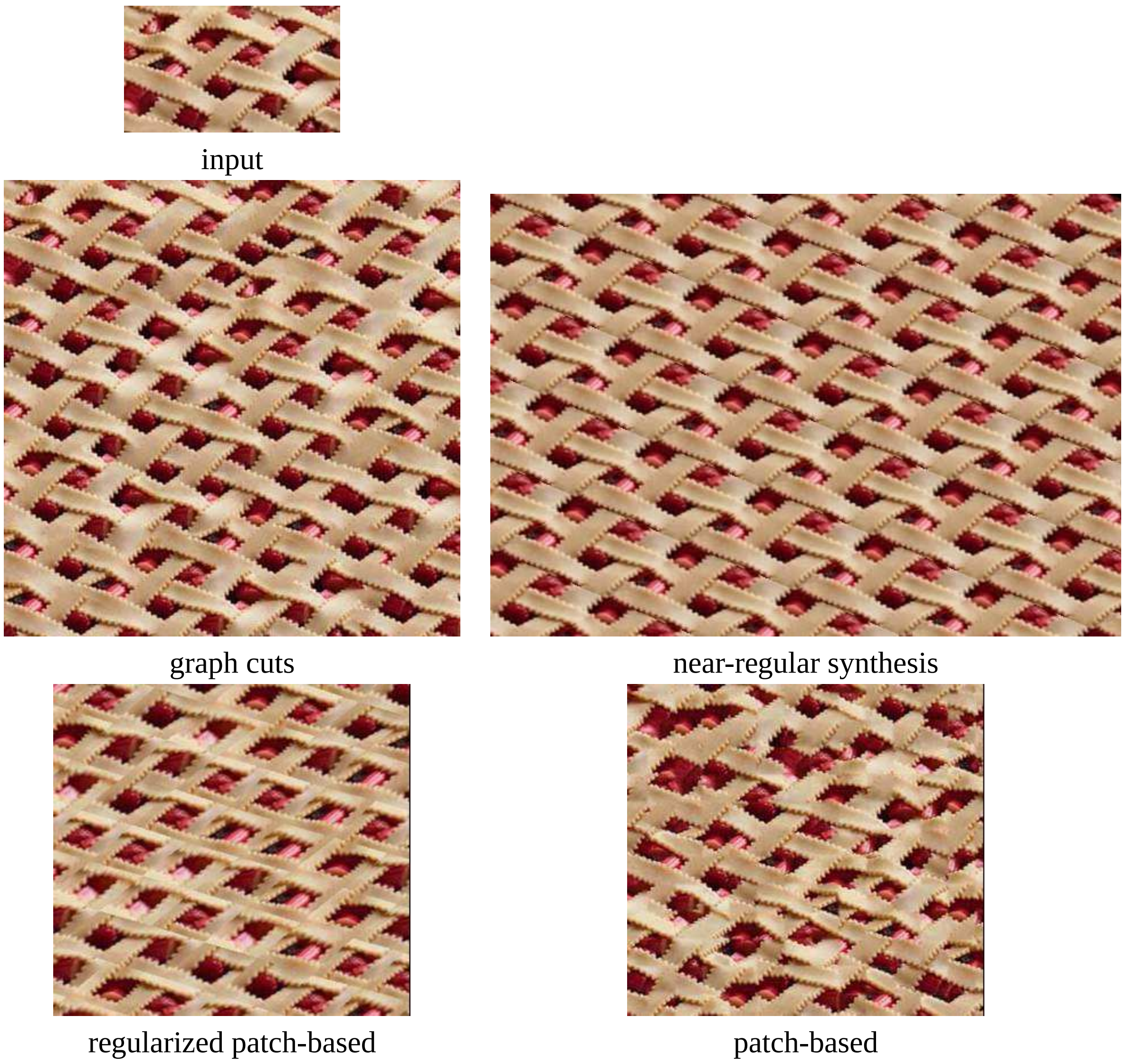


Figure 23: Synthesis results of a pie texture. The near-regular synthesis result almost preserves the interwoven structure, except there are some artifact edges in the intersections. Regularity is not preserved in the graph cuts, regularized patch-based, and patch-based results.

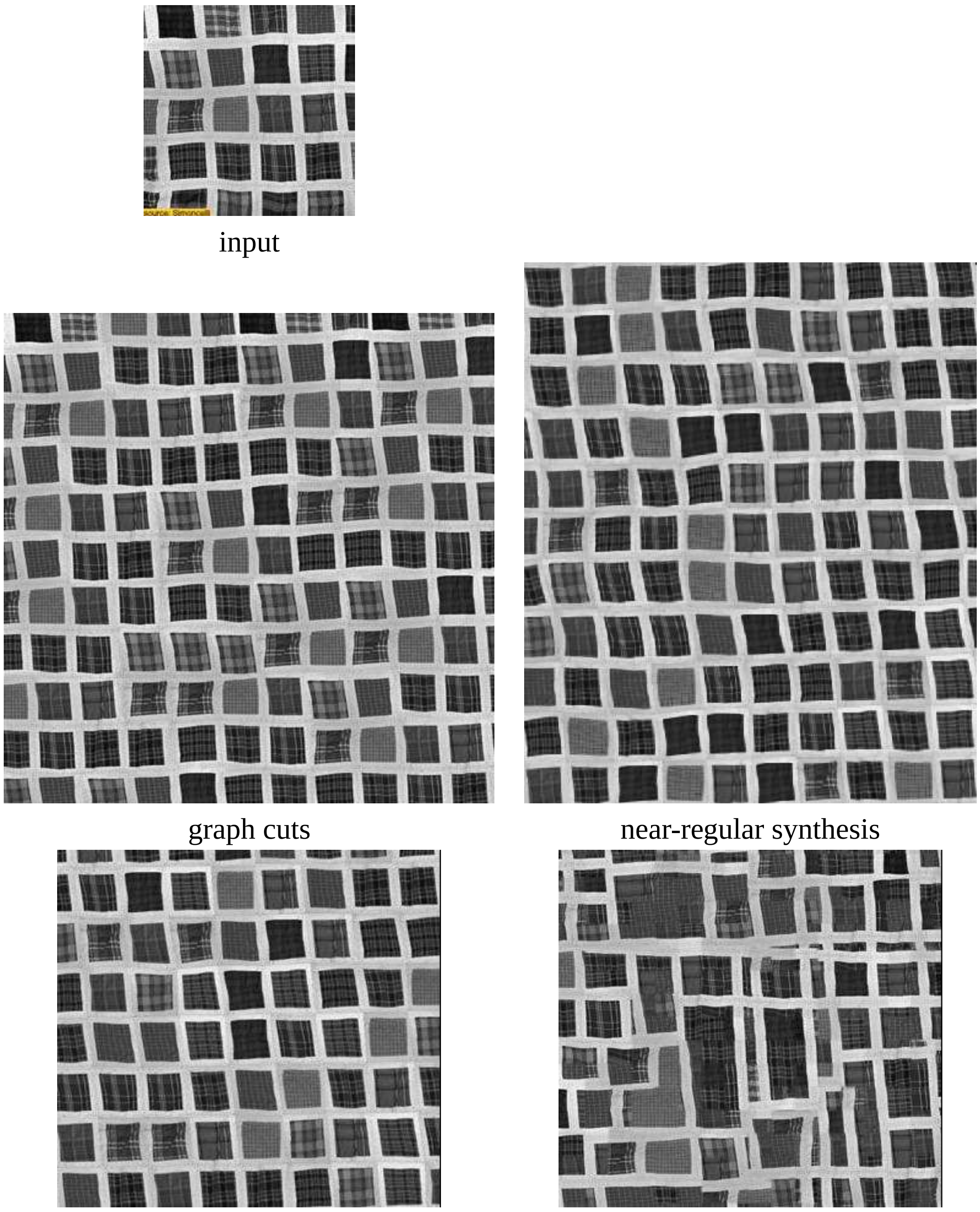


Figure 24: Synthesis results of a fabric texture. Global regularity is preserved in graph cuts, near-regular synthesis, and regularized patch-based results, but not the patch-based result

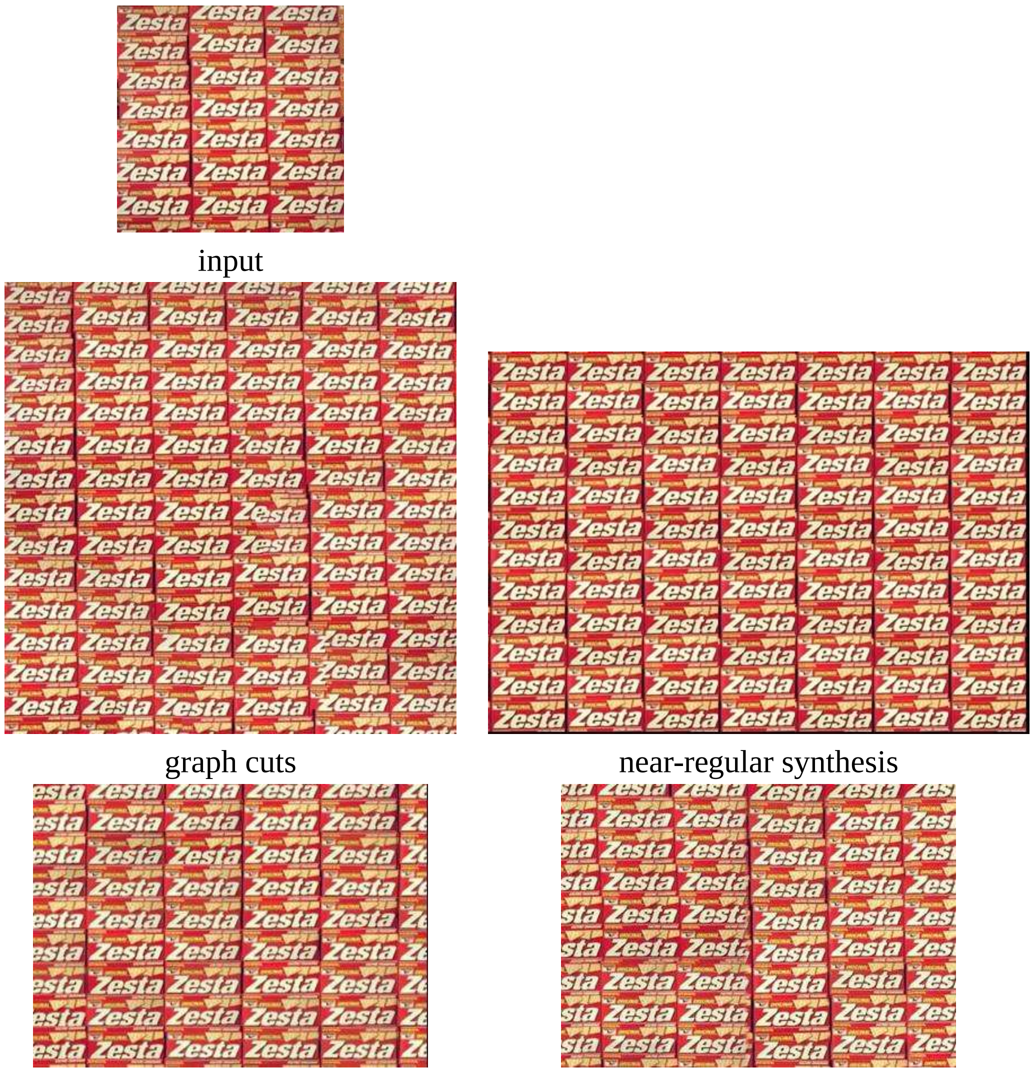
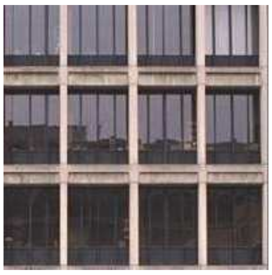
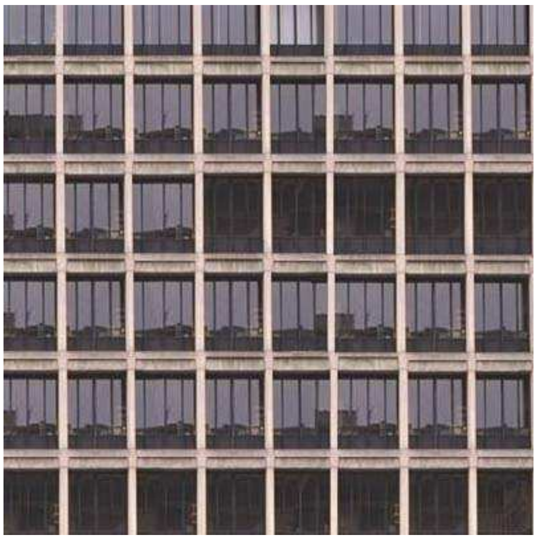
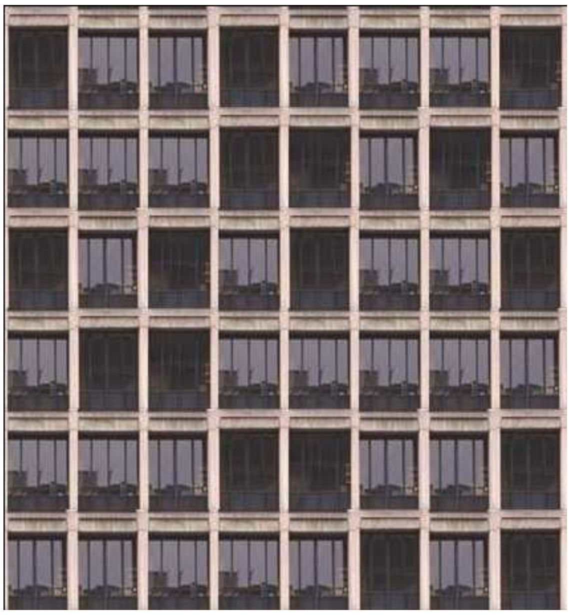
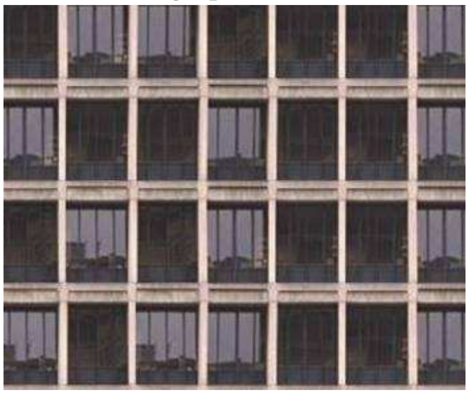


Figure 25: Synthesis results of a toothpaste texture.


input


graph cuts


near-regular synthesis


regularized patch-based

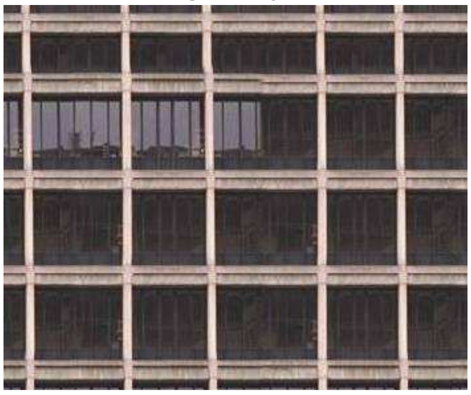

patch-based

Figure 26: Synthesis results of a window texture. The size of windows is varied in the patch-based result.

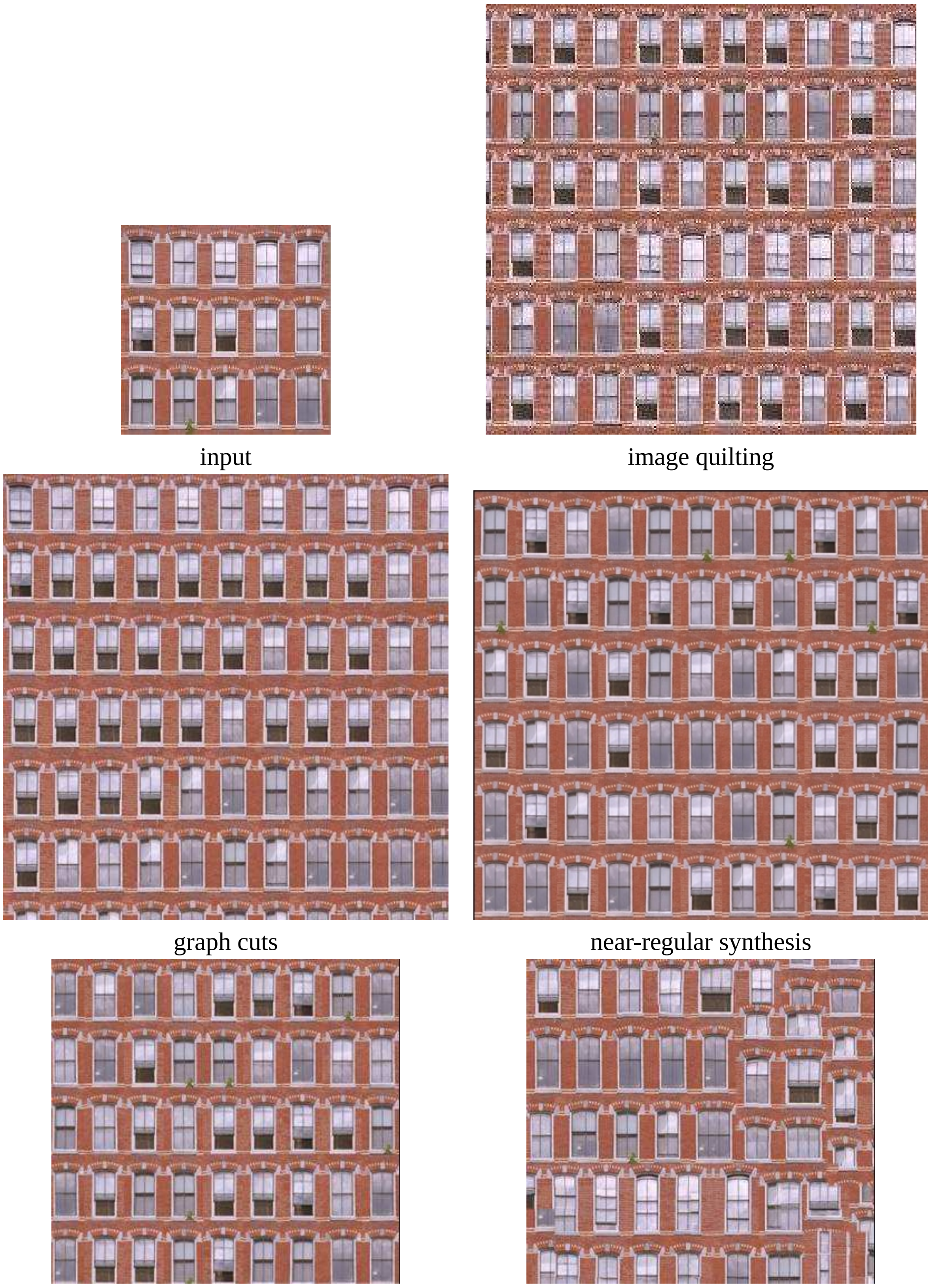


Figure 27: Synthesis results of a window texture. Global regularity is violated in the patch-based result.

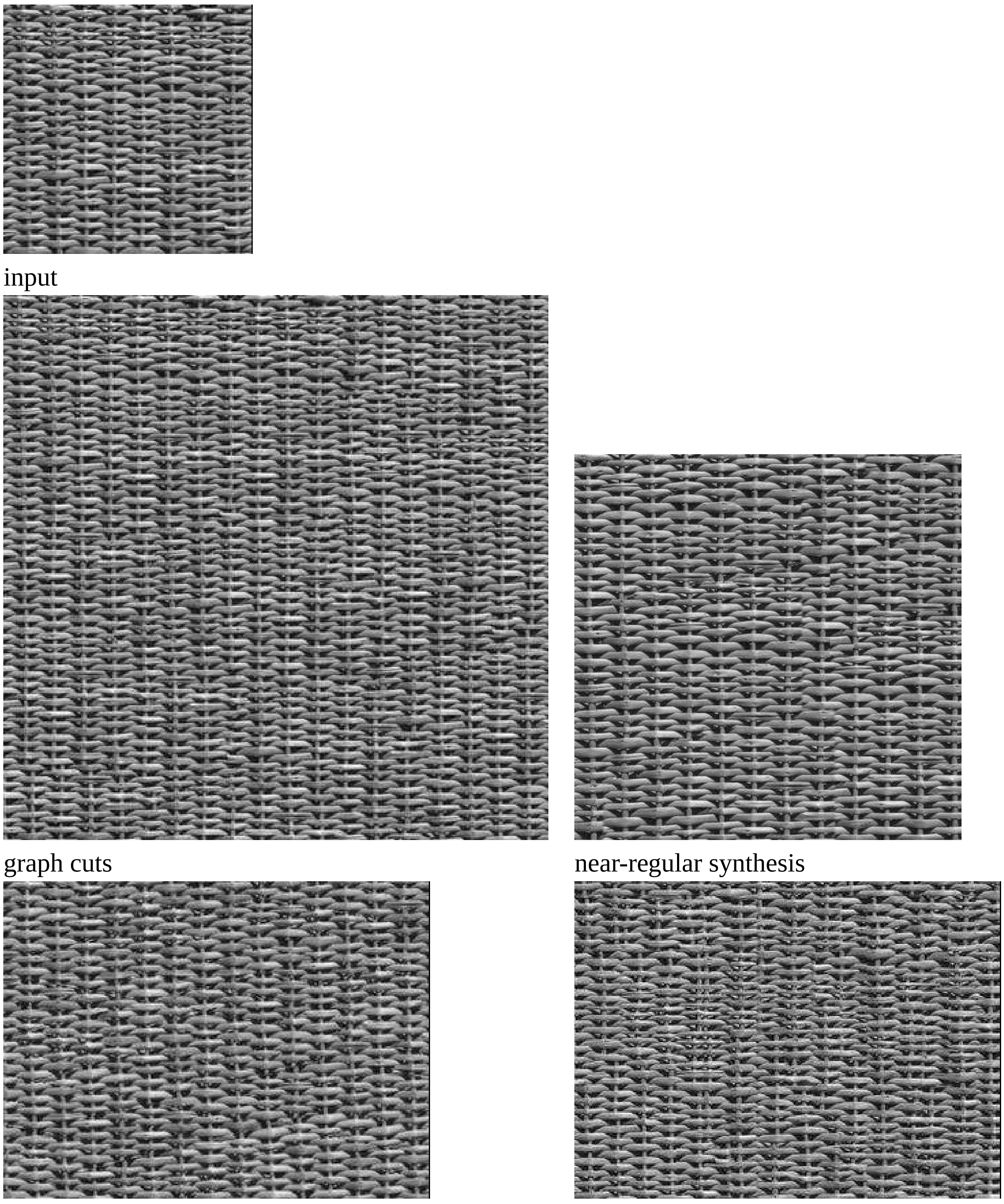


Figure 28: Synthesis results of a fabric texture. In the graph cuts result and patch-based result, there are some vertical straws are disconnected.

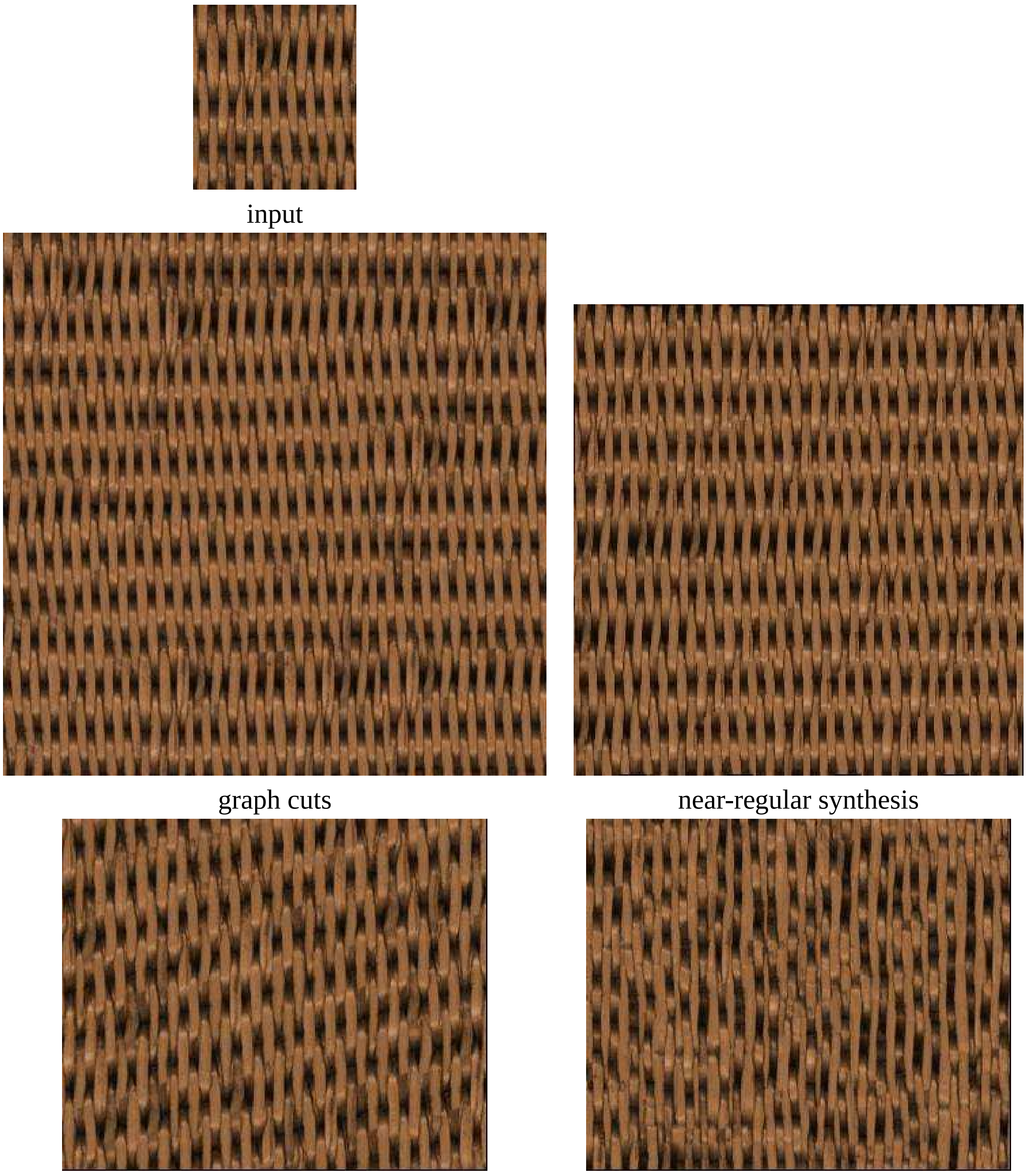


Figure 29: Synthesis results of a basket texture. Regularity is preserved in the graph cuts and near-regular synthesis results, but not the regularized patch-based and patch-based results. Note that the orientation of texture in the regularized patch-based result is not correctly synthesized.

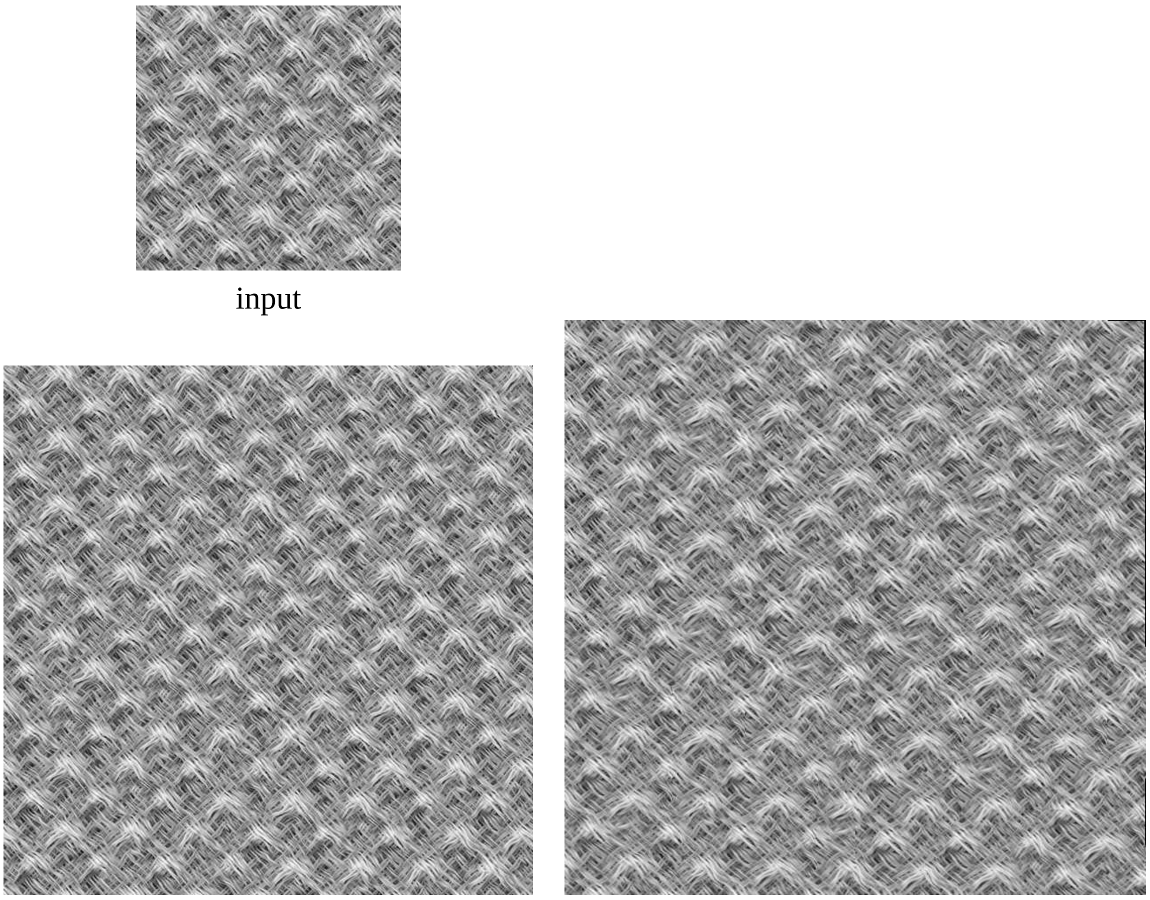
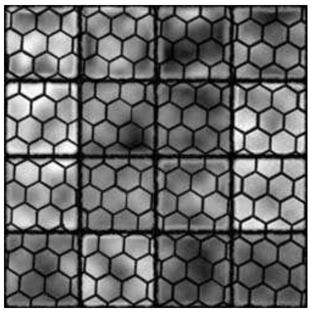
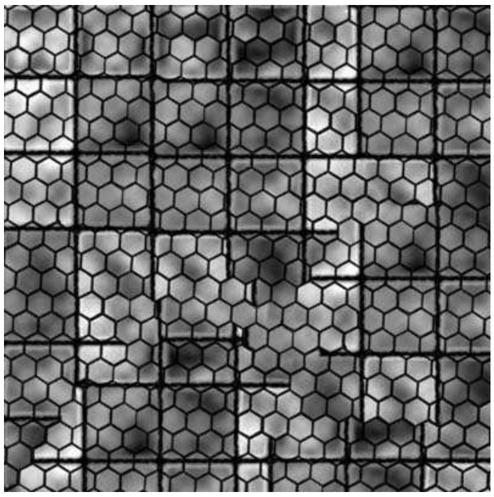


Figure 30: Synthesis results of a fabric texture.


input


graph cuts

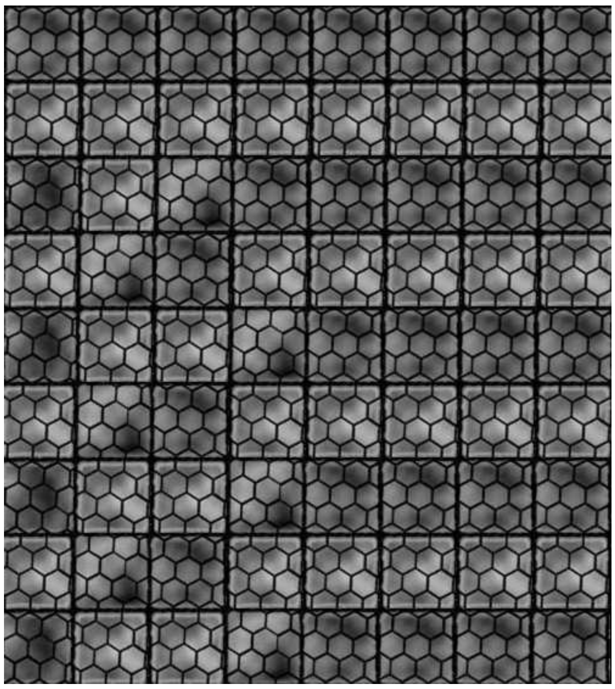

near-regular synthesis

Figure 31: The near-regular texture synthesis result preserves the regularity of the squares, but the hexagon net structure is not maintained. Therefore, both algorithms failed in this texture.

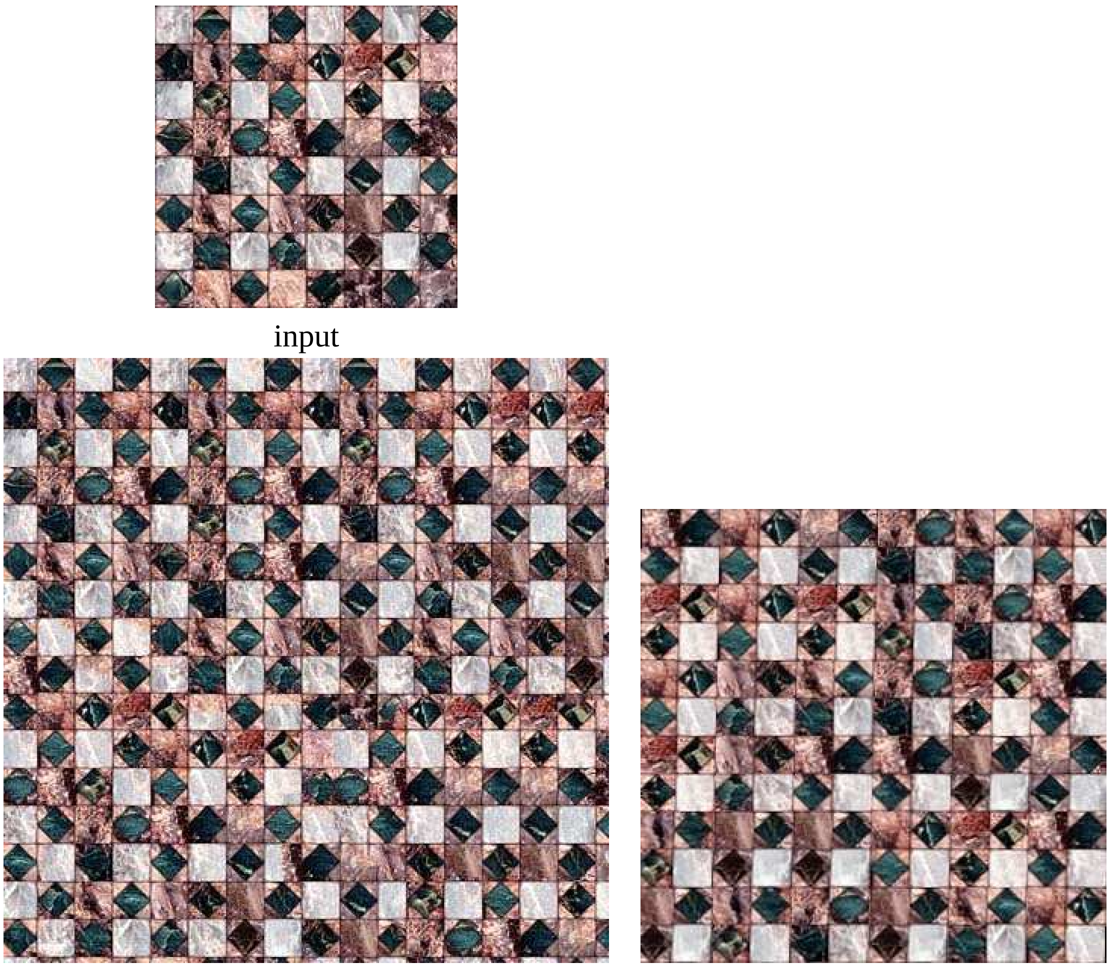
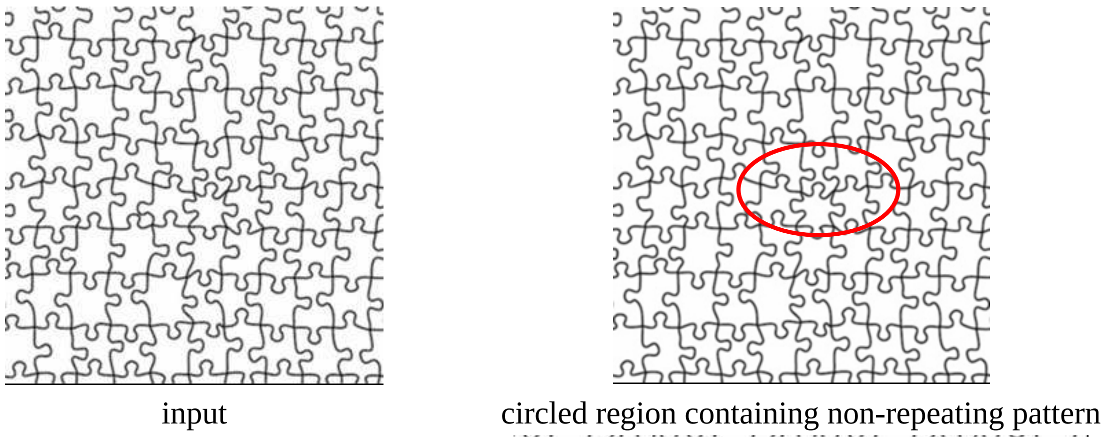
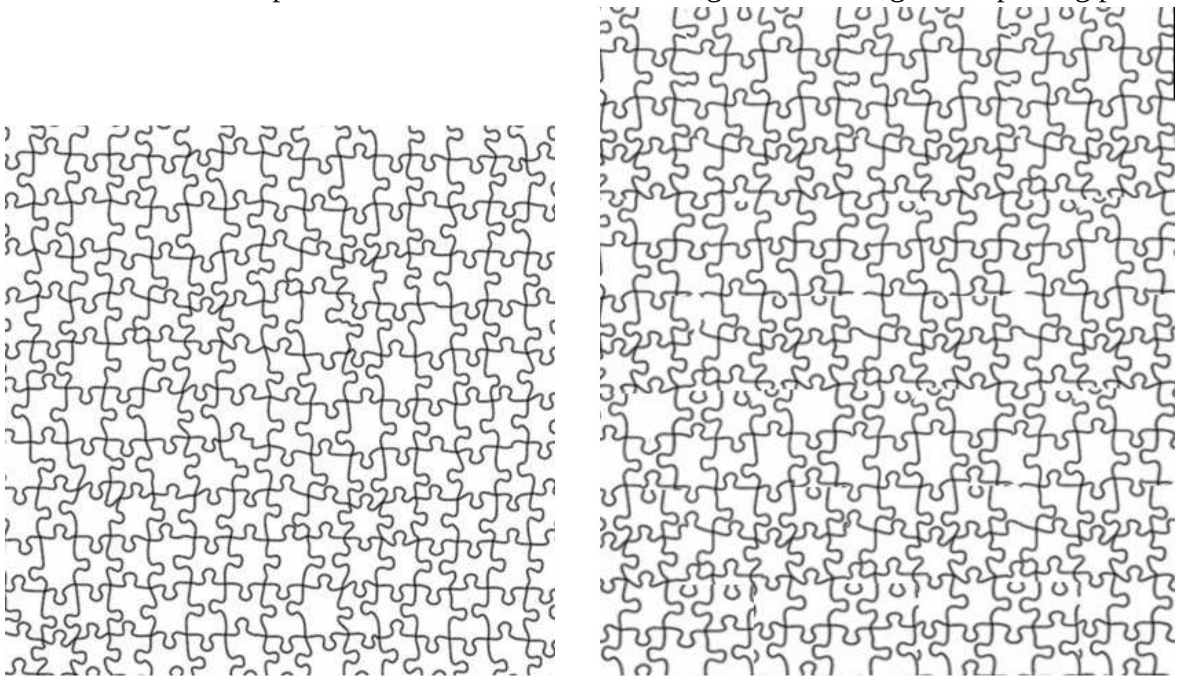


Figure 32: Synthesis results of a mosaic texture. The regularity of the squares is not preserved in the graph cuts result.


input

circled region containing non-repeating pattern


graph cuts

near-regular synthesis

Figure 33: Synthesis results of a jigsaw puzzle texture. The global regularity of this texture is not easy to observe. Near-regular texture synthesis can not preserve the regularity of this texture. The upper-right image shows that the input texture does not contain a complete tile as the pattern in the circled region only appears once in the texture.

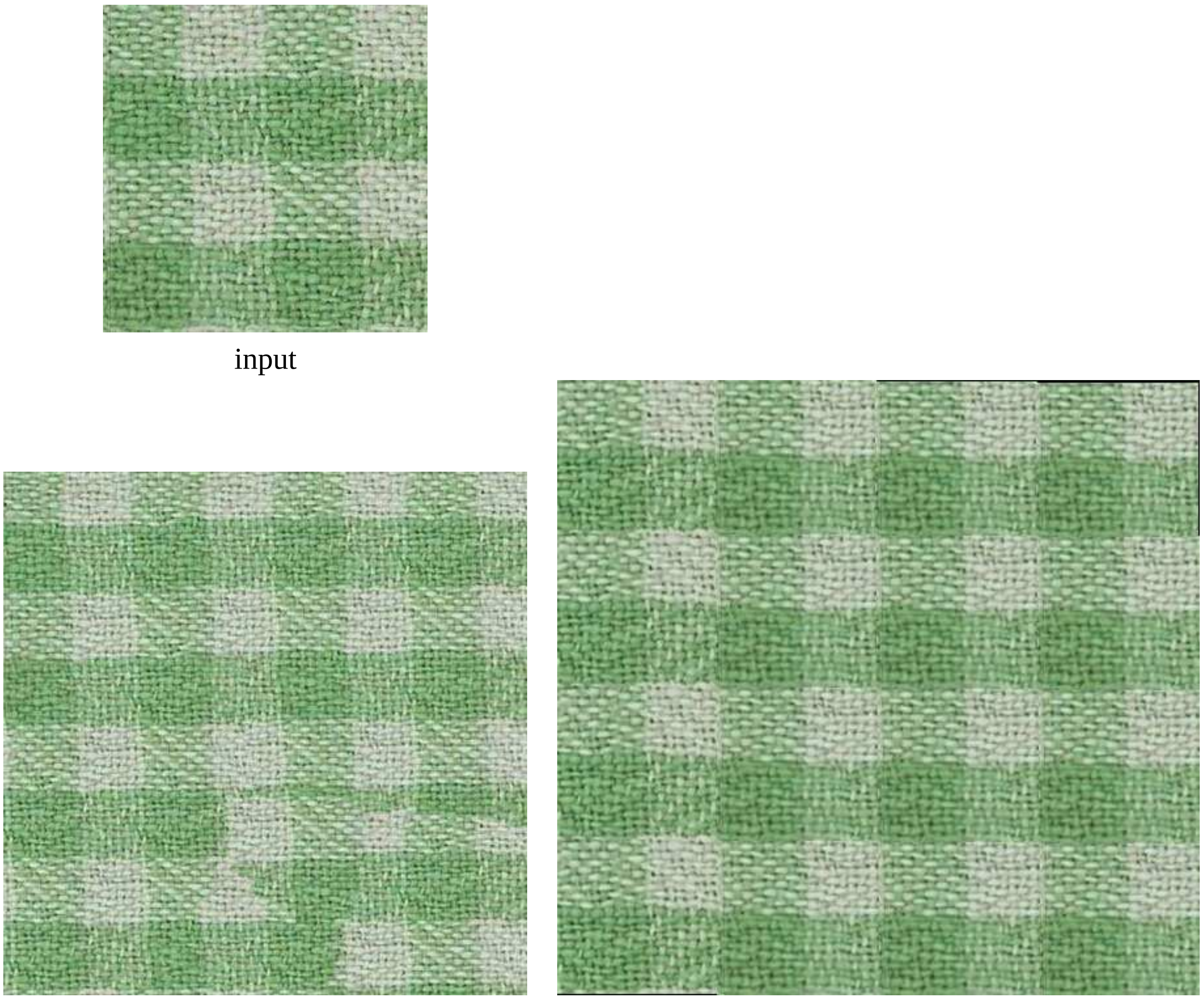


Figure 34: Synthesis results of a fabric texture. Global regularity is not preserved in the graph cuts result.

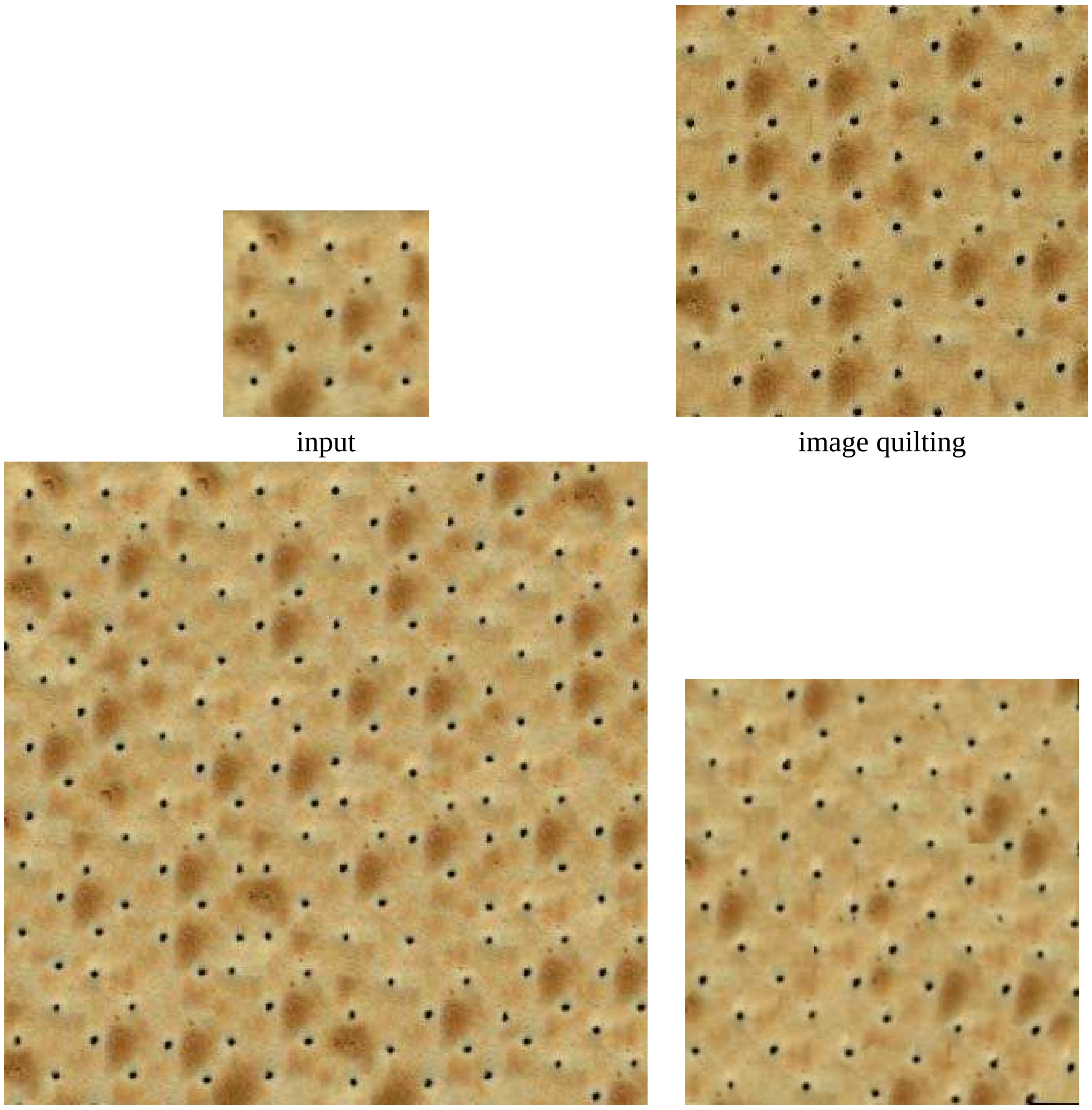


Figure 35: Synthesis results of a cracker texture. There are some misplaced small holes in the image quilting and the graph cuts results.

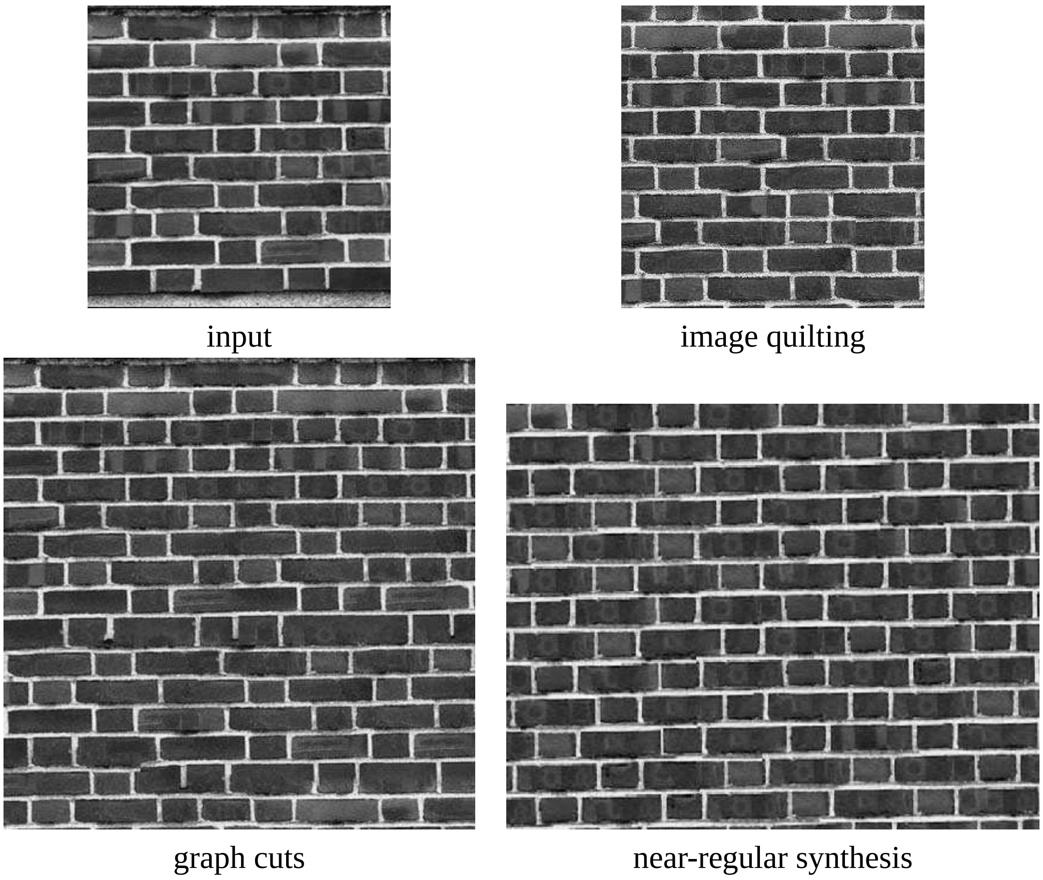
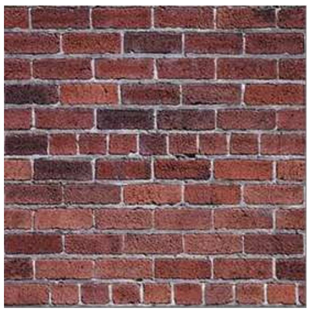


Figure 36: Synthesis results of a brick wall texture. The regularity of the grout structure is not maintained in the image quilting and graph cuts results.


input

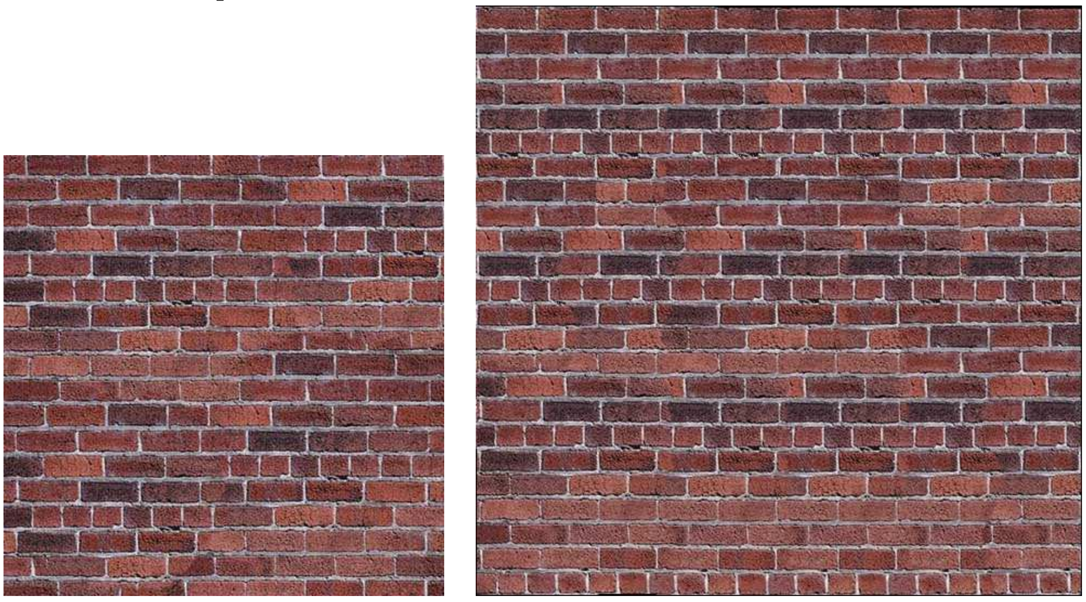
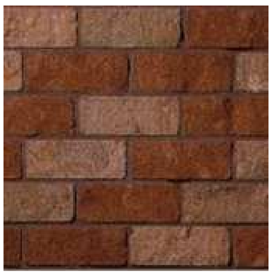
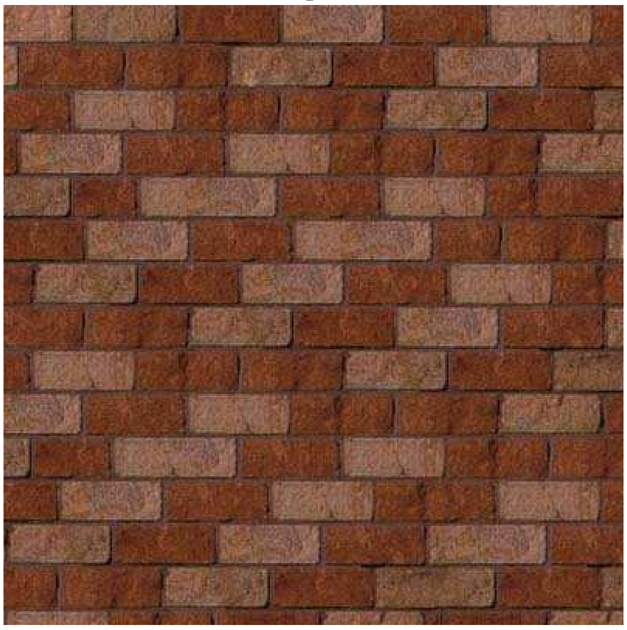

graph cuts near-regular synthesis

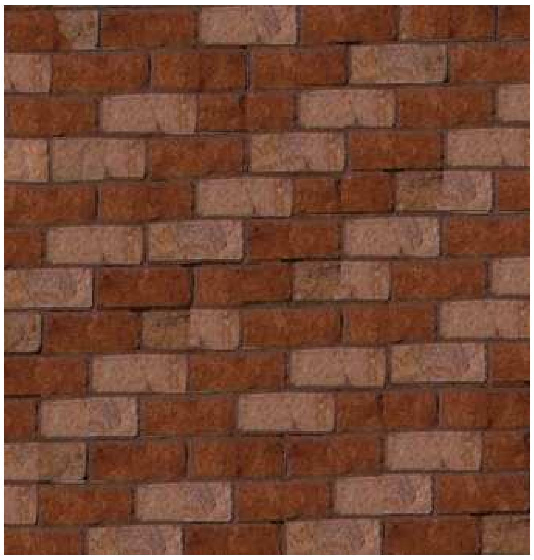
Figure 37: Synthesis results of a brick wall texture, where every sixth row consists of smaller bricks. The graph cuts result does not preserve this regularity.



input



graph cuts



near-regular synthesis

Figure 38: Synthesis results of a brick wall texture. In the graph cuts result, some bricks in the 5th and 6th row are 50% larger than the others.

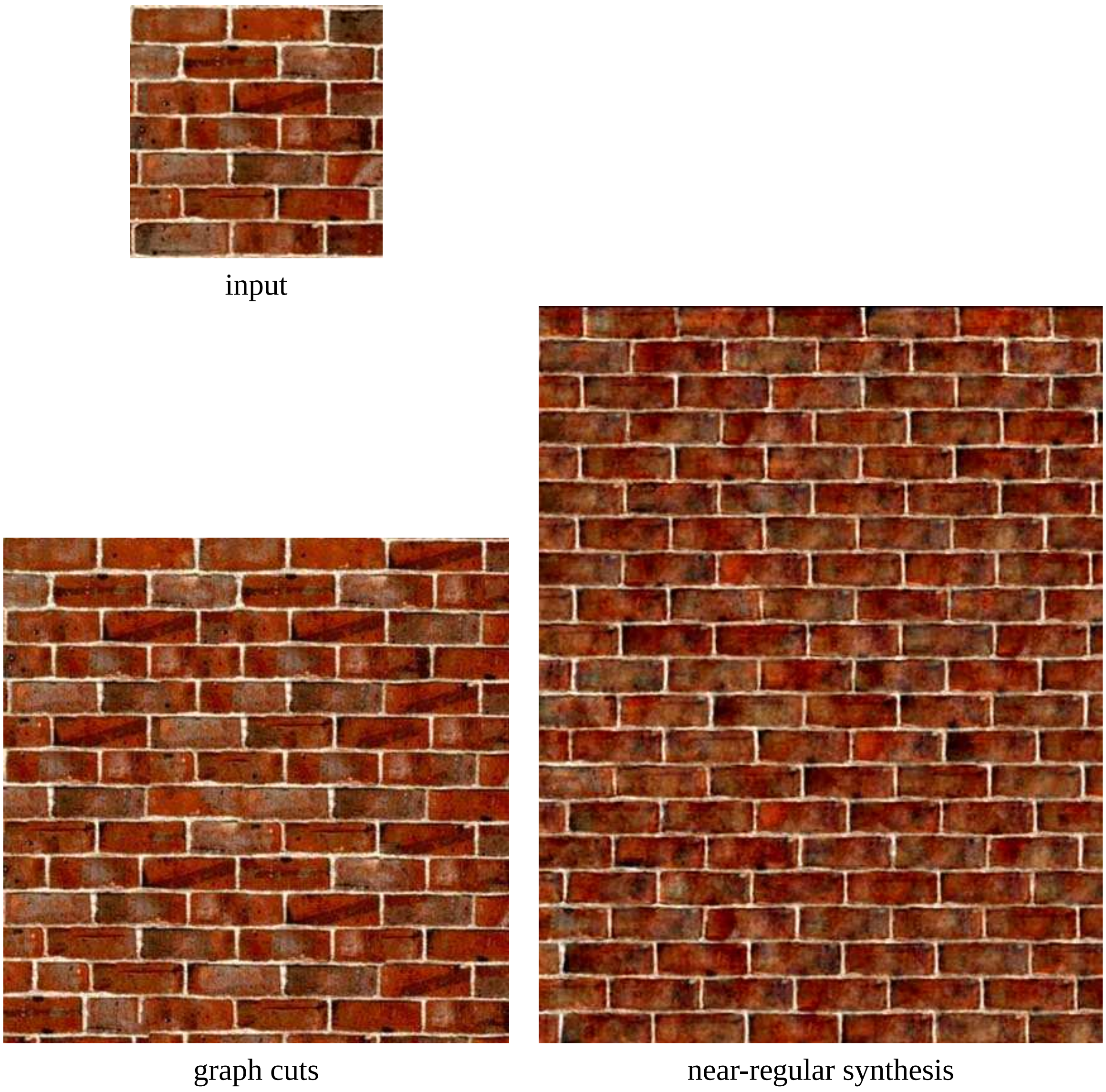
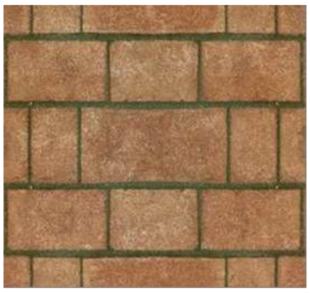
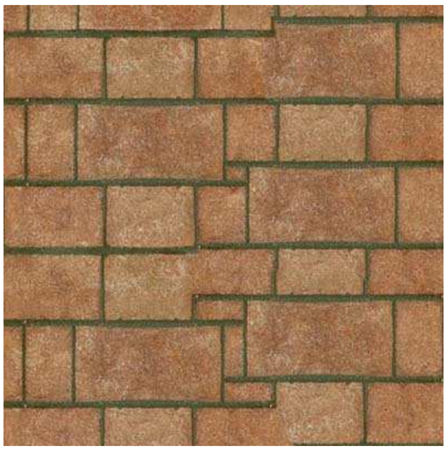


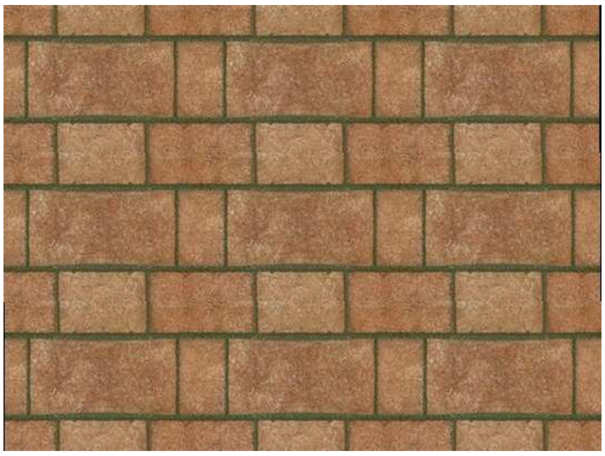
Figure 39: Synthesis results of a brick wall texture. Global regularity is preserved in both results.



input



graph cuts



near-regular synthesis

Figure 40: Synthesis results of a brick wall texture. The graph cuts result does not preserve the global regularity.

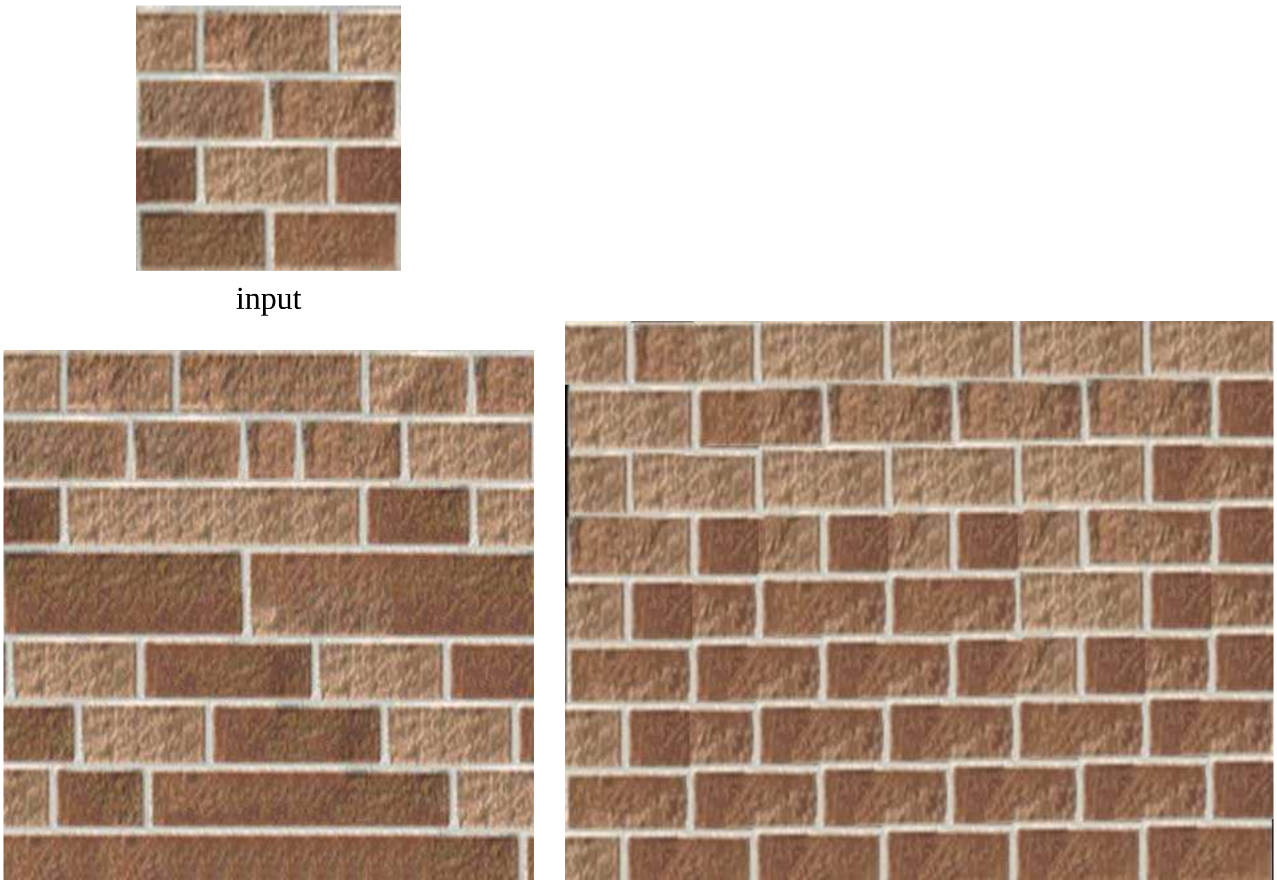
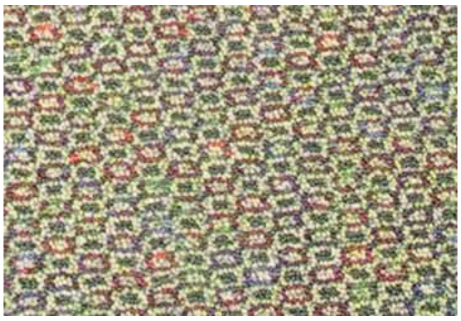
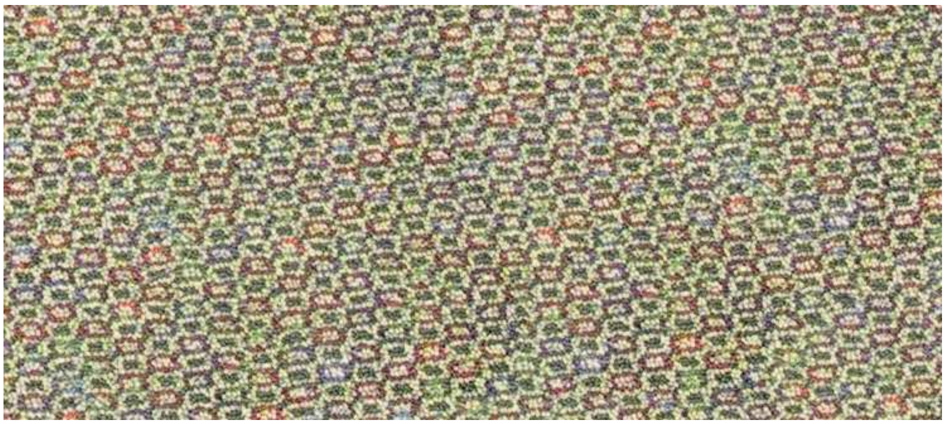


Figure 41: Synthesis results of a brick wall texture. The grout structure is not preserved in the graph cuts result.


input


graph cuts

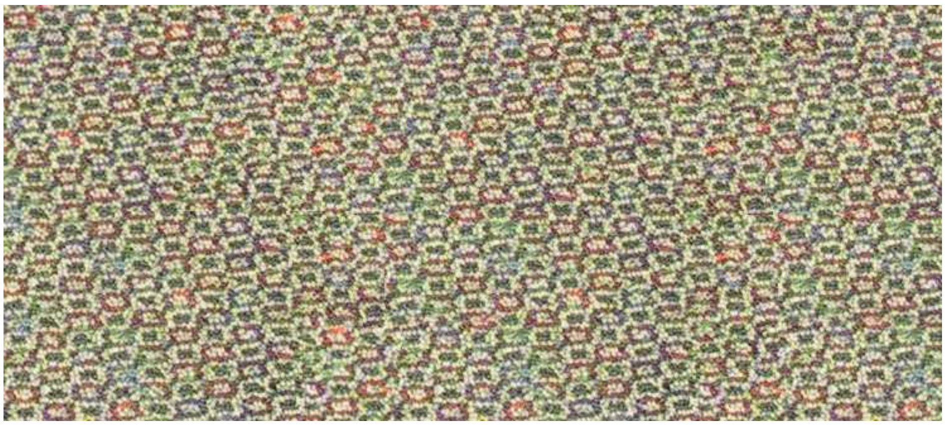
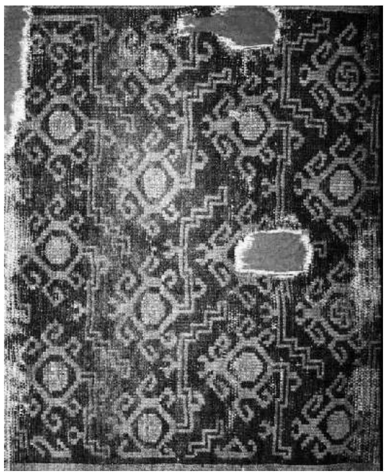
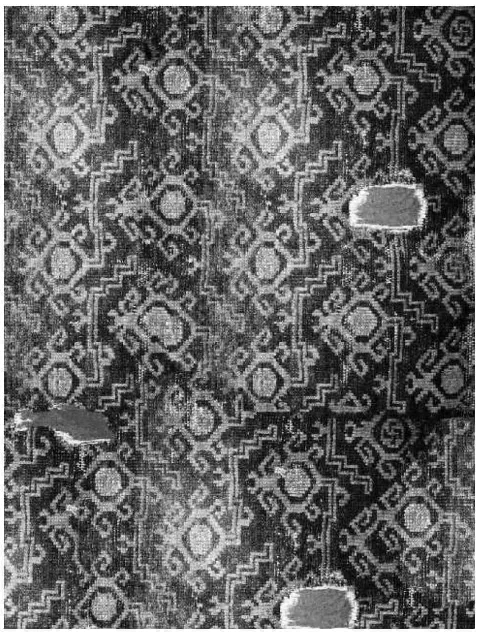

near-regular synthesis

Figure 42: Synthesis results of a carpet texture.


input


graph cuts

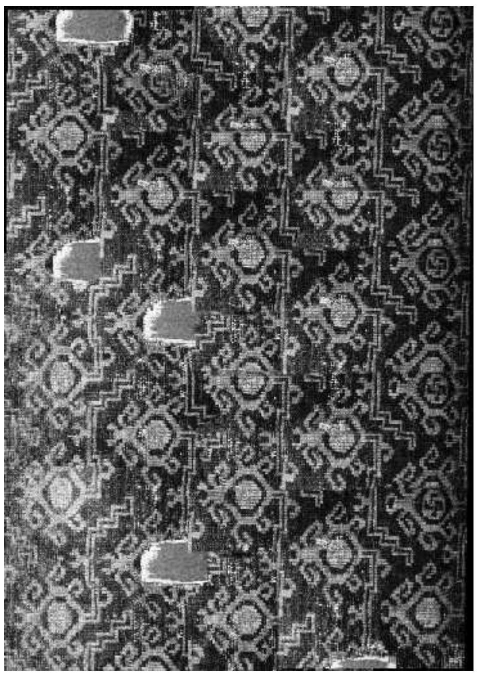
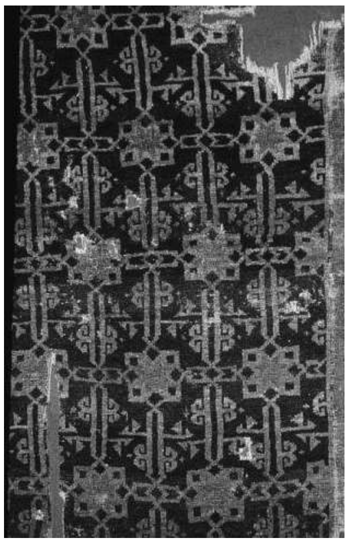
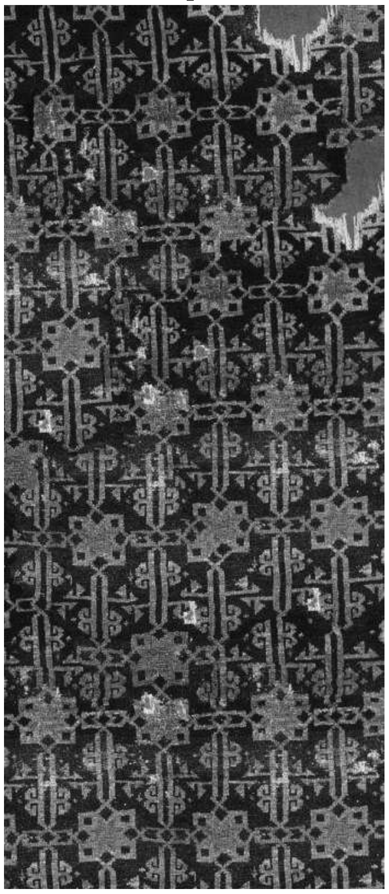

near-regular synthesis

Figure 43: Synthesis results of a damaged rug texture. Global regularity is not preserved in the bottom portion of the graph cuts result.


input


graph cuts

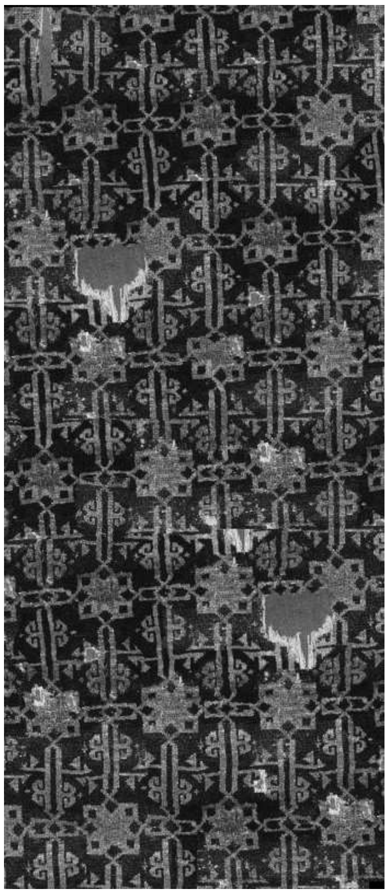
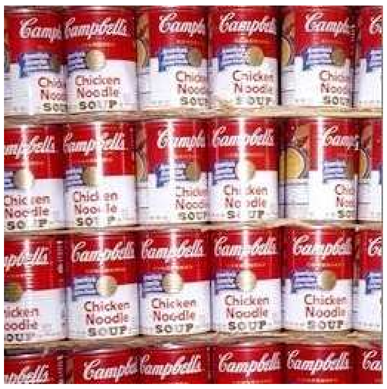
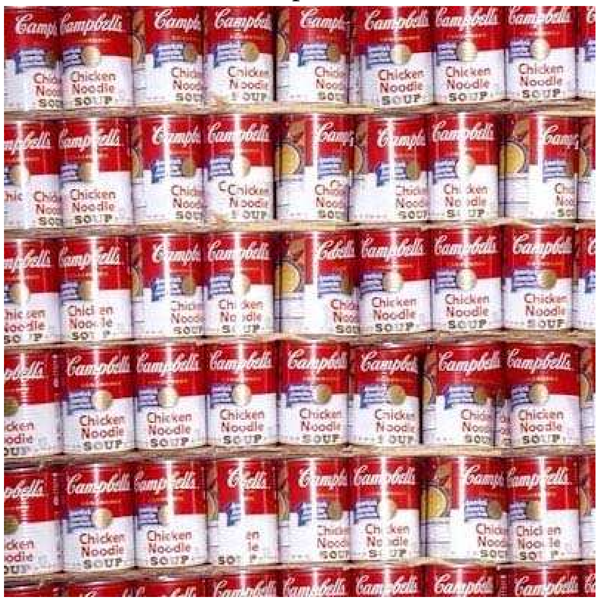

near-regular synthesis

Figure 44: Synthesis results of a damaged rug texture. Vertical patterns are not aligned at the middle part of the graph cuts result.


input


graph cuts

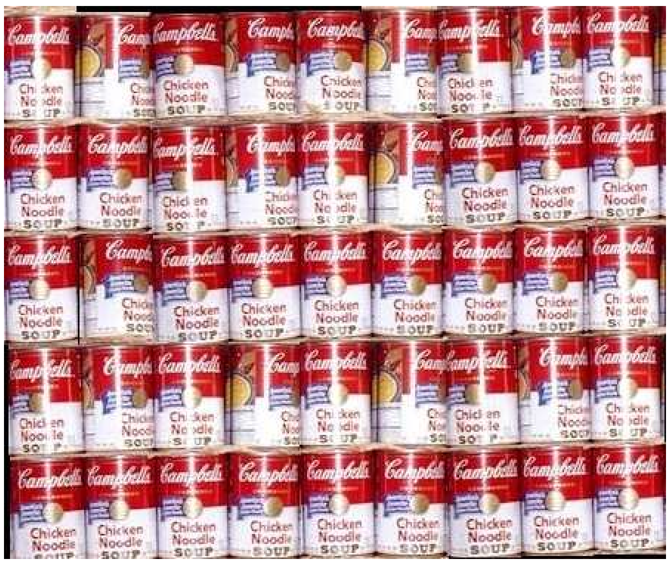
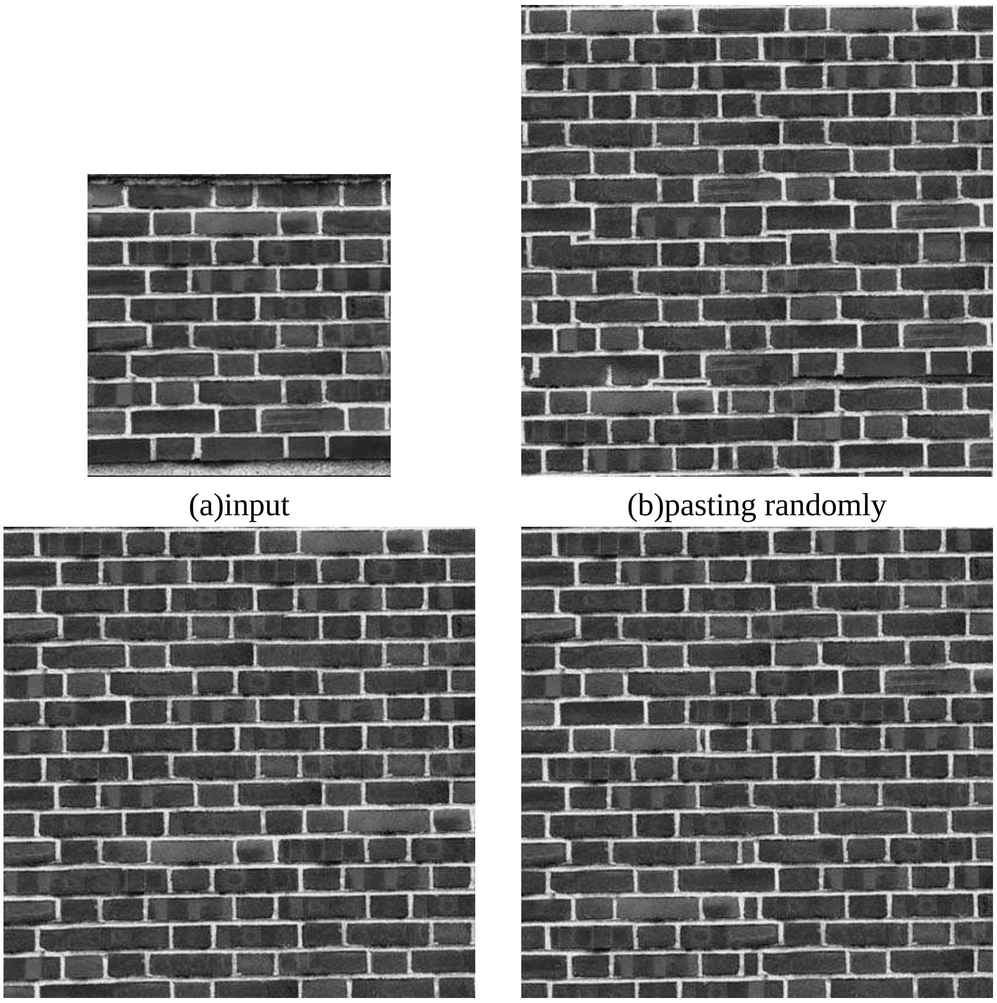

near-regular synthesis

Figure 45: Synthesis results of a texture of cans. Some cans in the 5th and 6th rows of the graph cuts result are not synthesized correctly (bottom-right area).


(a)input (b)pasting randomly

(c)pasting at local minima (d)pasting at local minima and maxima

Figure 46: This figure shows the results due to different settings for patch placement in the graph cuts approach. In the synthesis process, an error map is computed using seam cost around each pixel and then the location around which we want to paste the new patch is picked by sampling from this error map. The results shown are obtained respectively by (b)picking the pasting location randomly, (c)computing a local minima for these error values and picking a location from one of these, (d)computing both local minima and maxima and picking a location from one of these. The synthesis result of the same texture by the near-regular synthesis approach is shown in Figure 36.

References

[1] M. Ashikhmin. Synthesizing natural textures. In *Symposium on Interactive 3D Graphics*, pages 217–226, 2001.

[2] Z. Bar-Joseph, R. El-Yaniv, D. Lischinski, and M. Werman. Texture mixing and texture movie synthesis using statistical learning. *IEEE Transactions on Visualization and Computer Graphics*, 7(2):120–135, 2001.

[3] J. S. De Bonet. Multiresolution sampling procedure for analysis and synthesis of texture images. In *ACM SIGGRAPH*, pages 361–368, 1997.

[4] A. A. Efros and W. T. Freeman. Image quilting for texture synthesis and transfer. In *ACM SIGGRAPH*, pages 341–346, 2001.

[5] A. A. Efros and T. K. Leung. Texture synthesis by non-parametric sampling. In *ICCV (2)*, pages 1033–1038, 1999.

[6] D. J. Heeger and J. R. Bergen. Pyramid-based texture analysis/synthesis. In *ACM SIGGRAPH*, pages 229–238, 1995.

[7] V. Kwatra, A. Schödl, I. Essa, G. Turk, and A. Bobick. Graphcut textures: Image and video synthesis using graph cuts. In *ACM SIGGRAPH*, pages 277–286, 2003.

[8] S.-Y. Lee, K.-Y. Chwa, and S. Y. Shin. Image metamorphosis using snakes and free-form deformations. In *ACM SIGGRAPH*, pages 439–448, 1995.

[9] L. Liang, C Liu, Y. Xu, B. Guo, and H.Y. Shum. Real-time texture synthesis by patch-based sampling. *ACM Transactions on Graphics*, 20(3):127–150, 2001.

[10] Y. Liu and R. T. Collins. A computational model for repeated pattern perception using frieze and wallpaper groups. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 537–544, June 2000.

[11] Y. Liu, R. T. Collins, and Y. Tsin. A computational model for periodic pattern perception based on frieze and wallpaper groups. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(3):354–371, March 2004.

[12] Y. Liu and Y. Tsin. The promise and perils of near-regular texture. In *The 2nd International Workshop on Texture Analysis and Synthesis*, 2002.

[13] Y. Liu, Y. Tsin, and W.-C. Lin. The promise and perils of near-regular texture. *International Journal of Computer Vision*, 2004. To appear.

[14] J. Portilla and E. P. Simoncelli. A parametric texture model based on joint statistics of complex wavelet coefficients. *International Journal of Computer Vision*, 40(1):49–71, 2000.

[15] L.-Y. Wei and M. Levoy. Fast texture synthesis using tree-structured vector quantization. In *ACM SIGGRAPH*, pages 479–488, 2000.

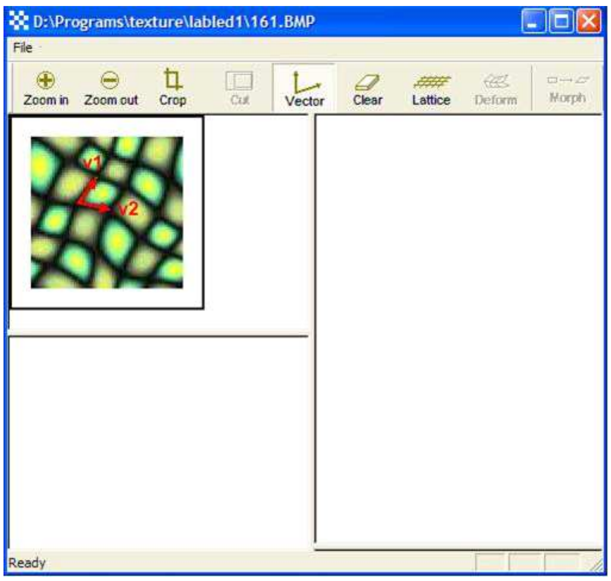
[16] L.-Y. Wei and M. Levoy. Texture synthesis over arbitrary manifold surfaces. In *ACM SIGGRAPH*, pages 355–360, 2001.

[17] J. Zhang, K. Zhou, L. Velho, B. Guo, and H.-Y. Shum. Synthesis of progressively-variant textures on arbitrary surfaces. In *ACM SIGGRAPH*, pages 295–302, 2003.

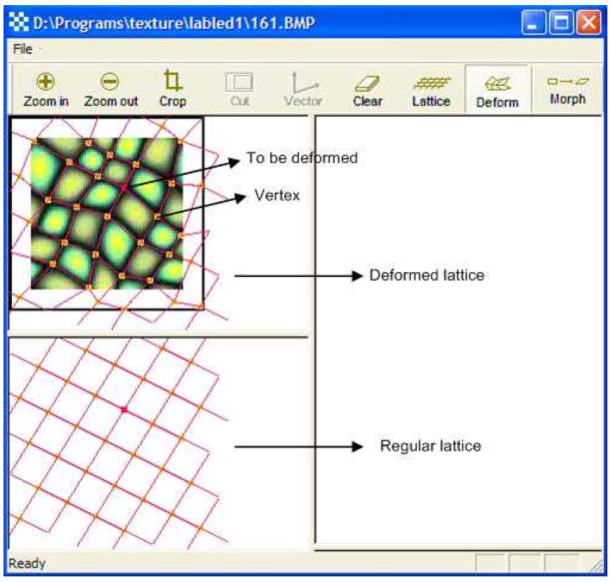
A Regularized patch-based approach

In near-regular texture synthesis, it is desired to extract the ”texton”, which is the basic element in the texture image. We consider the near-regular texture as an array of arbitrarily deformed textons. We build a regular lattice and warp the original deformed textons to the regular shapes. Then we can synthesize the regular texture. Finally, we compute a deformation field and warp the regular texture to a near-regular texture.

The whole system includes two parts. One is for analysis and the other is for synthesis. The analysis process is semi-automatic. We design an interface for users to select textons, build and deform the regular lattice. Then we can automatically synthesize the regular texture by using the patch based method, and the deformed near-regular texture by warping. Figure 47 illustrates how to select a texton, which corresponds to a quadrilateral or hexagon. We use two vectors to specify a quadrilateral. Then we can automatically build a regular lattice.



(a)



(b)

Figure 47: This figure illustrates how to select a texton, which corresponds to a quadrilateral or hexagon. (a) The user draws two vectors to specify a quadrilateral. (b) a regular lattice is then automatically constructed based on the two vectors.

In order to get the regular textons, we need to compute the warp field between the deformed lattice and regular lattice, as shown in Figure 47(b). In our method, we manually deform the lattice to match the deformed textons, and then we use the vertices in the deformed lattice and regular lattice as correspondence points. We use the multilevel free-form deformations approach (MFFD)[8] to calculate a warp field based on the correspondence between the regular and deformed lattice.

We modify the patch-based approach to synthesize the regular texture based on the regular textons. Figure 48 illustrates the synthesized regular texture. The regularized patch-based approach is very similar to the original patch-based method, but with two differences. One is in the generating order. In the standard method, the synthesis follows scan-line order. Here we follow the order of regular lattice generation. The earlier the vertices of a texton are found, the earlier the texton is ”patch-pasted”. The other difference deals with patches. In the standard method, a patch is searched for the whole image. Here, the patch can only be selected from the set of textons in the warped image. Figure 48 shows a generated regular texture based on the patch-based method.

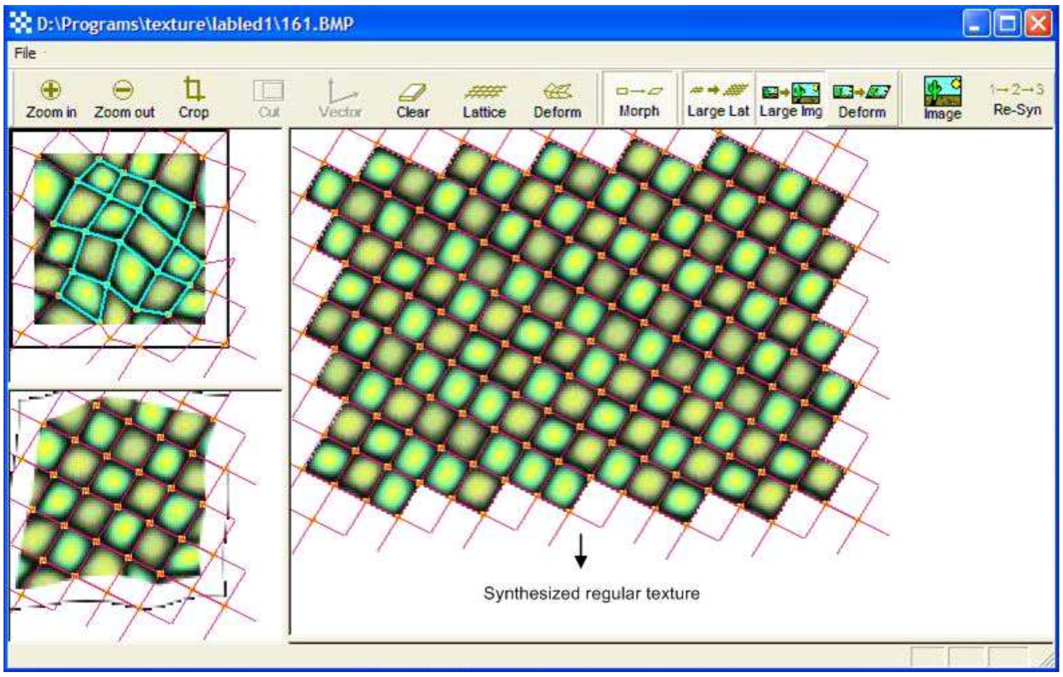


Figure 48: After a texture is regularized, the patch-based approach is used to synthesis this texture. The right window shows the result.

In order to generate the near-regular texture (deformed texture), we need to compute the deformation field. Inspired by the theory of Markov Random Field, for a quadrilateral texton, each vertex in the deformed lattice is affected by its eight neighbors. For example, in Figure 49, the yellow knot represents the vertex M, and eight orange knots represent the neighbors of M. Each neighbor can be discriminated by its relative position to the center vertex, such as left-top, top, right-top, right, right-bottom, bottom, and left-bottom. We can build a distance density map $p_i(x-x_0, y-y_0)$ for each neighbor. For each center vertex $M(x_0, y_0)$, if its neighbor $N_i(x, y)$ exists,

$$p_i(x - x_0, y - y_0) = \frac{1}{Z}\exp(-\frac{dist(M(x_0, y_0), N_i(x, y))}{2\sigma^2}) \quad (1)$$

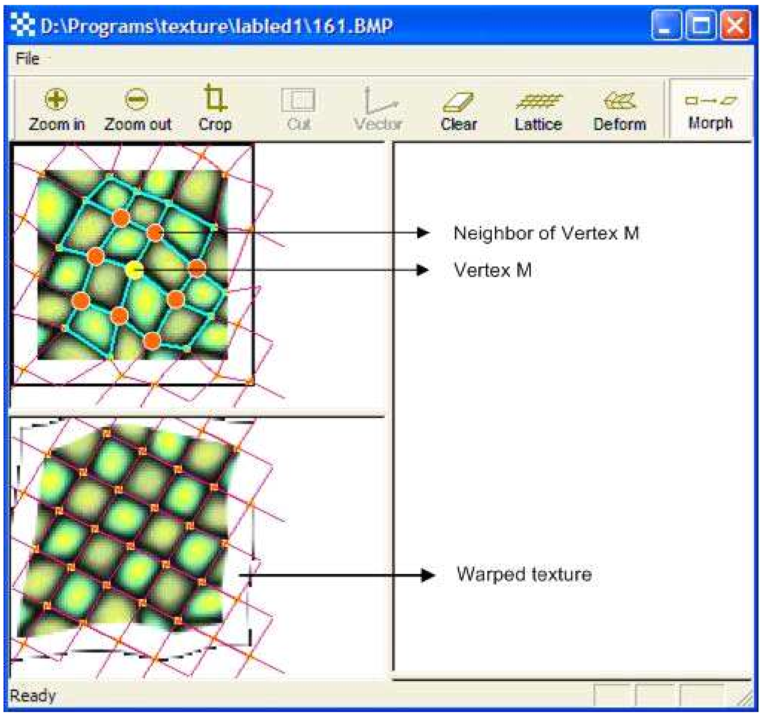


Figure 49: The upper-left window shows the definition of a vertex (yellow knot) and its neighbors (orange knots) on the lattice.

Then we can generate a deformation field as follows.

1. Initialize the deformed lattice by randomly moving each vertex in the regular lattice within a small window.
2. Deform the lattice iteratively:
   In each round, given a vertex $M(x_0, y_0)$ and a small window $[x_0-\sigma_x, x_0+\sigma_x, y_0-\sigma_y, y_0-\sigma_y]$, its neighbors are $N_i(x_i, y_i), i = 1, 2, ..., 8.$ we can compute the likelihood as follows:

$$p(x0 + dx, y0 + dy) = \prod_i p_i(x_i - (x0 + dx), y_i - (y0 + dy)) \tag{2}$$

   where $dx \in (-\sigma_x, \sigma_x)$ and $dy \in (-\sigma_y, \sigma_y)$ are the displacement of the vertex.

   The optimal position of M can be obtained by

$$(dx^*, dy^*) = arg \max_{dx,dy} p(x0 + dx, y0 + dy) \tag{3}$$

   The method is sensitive to the parameters such as σ_x and σ_y. In other words, the parameters are relative to the size of textons

After we get the regular and deformed lattice, we can warp the synthesized regular texture in the same way mentioned before. Figure 50 shows the final result of the synthesized texture.

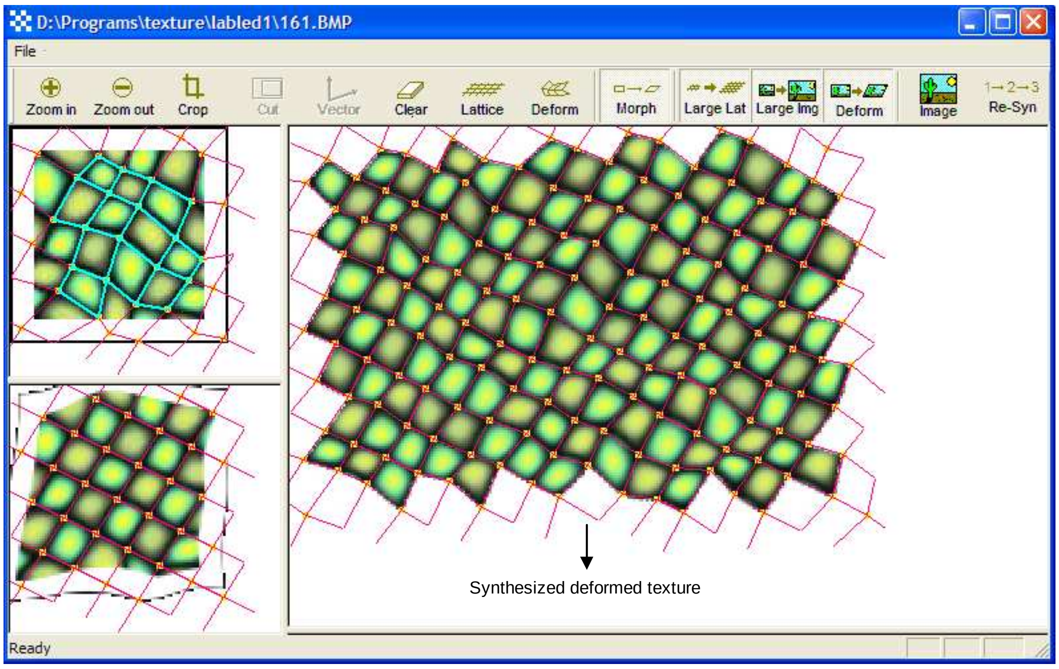


Figure 50: Final result of the synthesized texture.