Available online at www.sciencedirect.com
ScienceDirect
Graphical Models 70 (2008) 43–56
Graphical Models
www.elsevier.com/locate/gmod

Efficient texture synthesis using strict Wang Tiles ☆

Xinyu Zhang, Young J. Kim *

Department of Computer Science and Engineering, Ewha Womans University, Seoul, Republic of Korea

Received 22 March 2006; received in revised form 16 October 2007; accepted 20 October 2007
Available online 12 November 2007

Abstract

Wang Tiles are constructed from four texture samples, arranged so they can always match a choice of other tiles at two edges. Because they are precomputed, Wang Tiles are a very efficient way to generate textures on the fly. But matching problems occur within tiles and at the corners of adjacent tiles. By replacing the edge-matching texture samples with a new sample in the center of the tile, and using the graph cut path-finding algorithm, we overcome these problems and introduce additional texture diversity. Our s-Wang Tiles are a stricter interpretation of the original Wang Tile design, and our tile set is also smaller than that required by ω-Tiles: only eight different tiles are required for a non-repetitive titling.

Keywords: Texture synthesis; Wang Tile; Tiling; Graph cut

1. Introduction

Example-based texture synthesis uses a given example image to create large images with similar visual characteristics. It is used in video games, flight simulators and scientific computations which require rapid high-resolution texturing of surfaces and at a less cost in texture memory in the graphics processors (GPUs). There are a number of algorithms for example-based texture synthesis. In general, they can be divided into three categories: pixel-based methods, patch-based methods and tiling-based methods. Pixel-based methods use neighborhood information for each pixel in the example image to identify the most likely value for neighboring pixels during synthesis. Patch-based methods look iteratively for optimized sub-images in the example image and create a synthesized image by minimizing the overlap error in overlapping regions. Tiling-based methods precompute a set of small tiles with boundary pixels colored in such a way that no seam is apparent between abutting tiles. Tiles allow textures to be synthesized very quickly.

In this paper, we present a new tiling-based method which is directly motivated by Wang Tiles [1] and ω-Tiles [2]. The first successful tiling-based algorithm for texture synthesis, Wang Tiles use a set of small precomputed tiles to synthesize a large texture on the fly. It has proved to be very efficient in memory-constrained applications which require real-time texture synthesis. However, there are some

☆ This work was supported by the Korea Research Foundation Grant funded by the Korean Government (MOEHRD, Basic Research Promotion Fund) (KRF-2007-331-D00400).
* Corresponding author. Fax: +82 2 3277 2306.
E-mail addresses: zhangxy@ewha.ac.kr (X. Zhang), kimy@ewha.ac.kr (Y.J. Kim).

1524-0703/$ - see front matter © 2007 Elsevier Inc. All rights reserved.
doi:10.1016/j.gmod.2007.10.002

intrinsic limitations in Wang Tiles: the corner problem, the joint problem and the sampling problem, which will all be analyzed in Section 2.2. They compromise the efficiency of the algorithm in some respects such as precomputed tile quality and tile set size. The recently introduced ω-Tiles are efficient, with a higher quality of precomputed tiles and smaller tile sets. Experimental results [2] show that ω-Tiles are able to generate synthesized textures of reasonable quality while using only 16 tiles, as against to the set of 64 tiles by Wang Tiles. However, to maintain a non-periodic synthesized pattern, the minimum size of a set of ω-Tiles should be 32. In contrast, the theoretical minimum size of a set of Wang Tiles is eight, although it has to be increased to 64 in practice because of intrinsic limitations. For further information how the number of tiles is determined in Wang Tiles and ω-Tiles, refer to Cohen et al. [1] and Ng et al. [2].

1.1. Main results

We have looked for insight into the intrinsic limitations of Wang Tiles. Based on our observations and analysis, we proposed strict Wang Tiles (or s-Wang Tiles). Our new algorithm is able to create a smaller set of tiles while improving their quality significantly. In brief, we combine four diamond-shaped sub-images sampled from the example texture, like Wang Tiles, but without overlapping region between neighboring tiles. We cut out the central portion to obtain an initial tile and then introduce a supplementary square-shaped sub-image which is also cut from the example texture. Using the graph cut technique, we construct a weighted graph relating the initial tile and the supplementary square-shaped sub-image and then find a minimum-cost path in the graph. The portion surrounded by all the paths is then replaced by the supplementary sub-image.

Our approach addresses the following three problems that arise in generating a set of Wang Tiles:

- *Tile quality:* Eliminating the corner problem and joint problem reduces the prominence of seams while generating Wang Tiles. A large search space between two overlapping sub-images guarantees more chances of finding a good cutting path. Our approach also uses graph cut, which is more efficient than dynamic programming.
- *Tile set size:* From a reappraisal of the corner problem, we reduce the size of tile set from 64 to eight not just in theory, but in practice. Even so, our approach manages to reduce the amount of repetition during texture synthesis.
- *Synthesis quality:* Our approach achieves a more varied pattern, because the set of tiles is not restricted to the four sub-images, which is the case with Wang Tiles, and because our synthesized textures embed more of the example texture. Overcoming the sampling problem that occurs with Wang Tiles also increases the texture diversity to some extent.

1.2. Organization

The rest of this paper is organized as follows: Section 2 addresses the intrinsic problems of Wang Tiles [1]: the *corner problem* (Section 2.2.1), the *joint problem* (Section 2.2.2) and the *sampling problem* (Section 2.2.3). Section 3 gives details of tile design in our strict Wang Tile scheme. Section 4 compares our approach with related work: Wang Tiles [1] and also ω-Tiles [2]. Section 5 presents our experimental results, and Section 6 concludes the paper, conceding limitations of our approach and suggesting future work.

2. Previous work and problems

2.1. Previous work

There have been many approaches to the synthesis of large textured areas from small example images.

2.1.1. Pixel-based algorithms

The synthesized texture is generated pixel by pixel. Each pixel is the result of a search of the given example, using the neighborhood already synthesized. The main advantage of pixel-based methods [3–13] is the ability to control textures at pixel level. However, these methods are sensitive to the size of the window during the search for pixels in the example images, and when deciding a pixel's neighbors in the synthesized texture. More recent work [14] provides a controllable parallel method at pixel level that can utilize modern GPUs.

2.1.2. Patch-based algorithms

The texture is generated patch by patch and cutting paths are sought which minimize overlap errors between patches. Specific algorithms include image

quilting using dynamic programming [15], graphcut texture [16], iterative optimization based on Markov Random Fields [17] and the earlier lapped textures technique [18]. Patch-based algorithms tend to preserve the global structure, but they sometimes introduce local artifacts in the regions of overlap between neighboring patches. Patch-based texture synthesis has been shown to produce high-quality textures faster than pixel-based approaches. The search for a good neighboring patch and a satisfactory cutting path is not simple and usually requires iterative optimization.

2.1.3. Tiling-based algorithms

A set of tiles are precomputed and laid out seamlessly over the area to be textured. Synthesizing a large texture is simple and fast. These algorithms include aperiodic tiling on surfaces [19], triangular pattern-based tiling [20], hierarchical pattern mapping [21], Wang Tiles [1], parallel rendering of Wang Tiles in GPUs [22], procedural textures based on pattern [23], extracting tiles for near-regular textures [24,25] and, most recently, ω-Tiles [2]. Wang Tiles was the first successful tiling-based method, and has shown its advantages in real-time applications. The main drawback of Wang Tiles is its limited variety, due to the fixed number of sub-images (four sub-images for a set of eight tiles) captured from the given example texture, and the serious corner, joint and sampling problems. In ω-Tiles, each edge is identified by two colors, so a set of 32 tiles is needed to synthesize textures which are strictly non-periodic.

2.2. Wang Tiles and intrinsic problems

In Cohen et al.'s Wang Tiles [1], each tile has the structure shown in Fig. 1(b). The square central portion is cut out of a diamond-shaped image and becomes a Wang Tile. To construct each tile, four sub-images, identified by four colors—yellow, red, blue and green, are randomly cut out of the example texture, shown in Fig. 1(a), and arranged so that they overlap each other, following the pattern in Fig. 1(b). Four cutting paths are found to combine the four sub-images seamlessly, greatly reducing the visibility of the boundaries between them. An image quilting optimization algorithm [15] is used to find these cutting paths in overlapping regions. For convenience in later assembly, each edge is identified with the color of the sub-image to which it belongs. Because the same image sample is used to generate all the tiles that share an edge color, these tiles will always match each other. For example, if the east edge of a tile is colored green, a tile with a green west edge can be laid next to it.

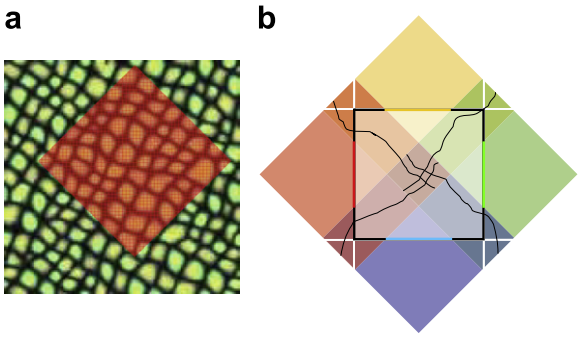

Fig. 1. Wang Tile. (a) The example texture out of which the four sub-images are cut and (b) Wang Tile structure presented by Cohen et al.

Wang Tiles represent a novel approach to real-time texturing applications and have outperformed most existing algorithms. But they have serious problems, which we address in this paper. We will describe these problems in the following sub-sections, and then present our solutions in Section 3.

2.2.1. Corner problem

We will analyze the corner problem by considering two Wang Tiles laid side by side. We will use r, g, b and y to identify four colors: red, green, blue and yellow. Then let T_{abcd} represents a Wang Tile with four colors $[a,b,c,d]$ where a, b, c, $d \in \{\mathrm{r},\mathrm{g},\mathrm{b},\mathrm{y}\}$. The four elements of a 4-tuple are used to enumerate the four edges, anticlockwise from the top. In Fig. 2(a), a Wang Tile T_{yrbg} with a right green edge can match a Wang Tile T_{bgyr} with a left green edge in the horizontal direction because of the way in which the tiles were created. The central segment of the green edge of T_{yrbg} is the former neighbor of the central segment of the green edge of T_{bgyr} in the original green sub-image. Therefore, the tiles fit—except the corners.

The corner problem is caused by the use of Efros' image quilting algorithm [15] during tile creation. Because it does not generate cutting paths that pass through the corners exactly, the edges near the corners may not be colored correctly, as shown in Fig. 2. A mismatch can be expected at almost every corner, because the probability that a cutting passes through the corner point is very low. Therefore most tile combinations will cause distortions at cor-

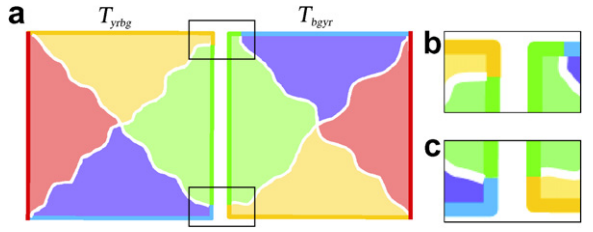

Fig. 2. Corner problem. (a) Two neighboring Wang Tiles with the corner problem highlighted by rectangle boxes; (b) the top corner, where the yellow segment of T_{yrbg} does not match the green segment of T_{bgyr}; and (c) the bottom corner, where the blue segment of T_{yrbg} does not match the yellow segment of T_{bgyr}. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this paper.)

ners. To overcome this corner problem, Cohen et al. suggested expanding the set of eight Wang Tiles to a set of $2^4 \times 8$ tiles. This large set of 128 tiles could then be condensed to a set of 64 Wang Tiles with matching corners. But this approach requires more texture memory during rendering. Furthermore, a set of 64 Wang Tiles (the base set has eight tiles) is still too small to synthesize a non-periodic texture, so a set of at least 144 tiles (eighteen tiles in the base set) was used by Cohen et al. [1].

The human visual system is sensitive to corners and lines [26]. Corners in particular are second-order features. So overcoming the corner problem should significantly reduce the number of visible artifacts.

2.2.2. Joint problem

In addition to the corner problem, Wang Tiles have a serious joint problem, which occurs at the center of each Wang Tile, and is also caused by Efros' image quilting algorithm [15] which is not really suited to the Wang Tile application. In each step of Efros' original application, the example texture is searched for the best matching sub-image. Thus the algorithm can always find a good cutting path in the overlapping region. However, Cohen et al. use a restrict number of sub-images (four for a set of eight tiles), selected at random, and overlap them. Using such a small selection of sub-images, it is difficult to obtain good paths for all overlapping regions.

Moreover, errors accumulate at the joints where all the sub-images overlap. Fig. 3 shows the generation of four cutting paths by means of image quilting during the creation of a Wang Tile. Usually, finding the first cutting path is relatively easy (Fig. 3(a) and (b)). But finding subsequent paths is more difficult because the overlap at the joint is distorted by the

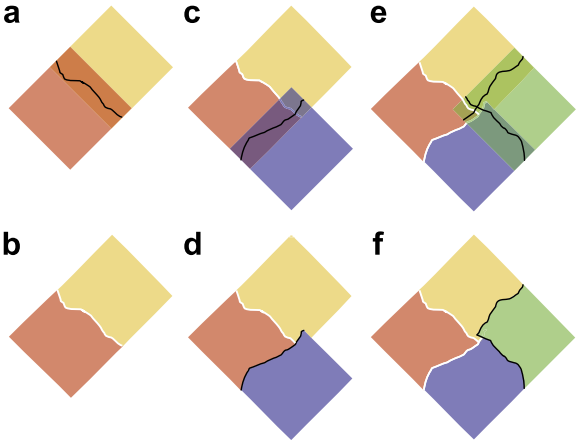

Fig. 3. Joint problem. (a and b) The first cutting path is found between the yellow sub-image and the red one, (c and d) the second cutting path between the image in (b) and the blue sub-image, (e and f) the third and the fourth cutting paths between the image in (d) and the green sub-image. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this paper.)

previous path. So the search for the second, the third and the fourth path is based on distorted sub-images. The distortion is very serious, particularly for the last cutting path (Fig. 3(e) and (f)).

2.2.3. Sampling problem

The third problem with Wang Tiles is that the resulting synthetic texture is based on samples of a small fixed number of sub-images (for example, the four sub-images which are all that are theoretically required for a set of eight tiles) from the example texture. The problem has two factors, relating to the sampling position where each sub-image is sampled, and to the number of those sub-images.

- *Sampling position:* Since the creation of a Wang Tile requires the selection of a diamond-shaped sub-image from the example texture (see Fig. 1), the four corners of the example texture will never be sampled. That means the features lying in these regions have no chance to appear in the final synthesized texture, which may therefore deviate from the underlying global pattern of the example, especially if important features happen to be located near to the corners of the example texture.
- *Sample number:* In theory, Wang Tiles require only four sub-images to construct a set of eight tiles, so the resulting synthetic texture will only contain texture patterns from these four sub-

images. That may cause repetition in the resulting texture, particularly if these Wang Tiles are used to synthesize a large area.

3. Our strict Wang Tile scheme

Our tile design scheme strictly follows the definition of Wang Tiles. A Wang Tile set consists of square tiles with color-coded edges. The tiles can not be rotated. A valid tiling of the plane consists of any number of copies from the set laid down such that all contiguous edges have matching colors [1]. A stochastic tiling pattern can be laid out with theoretical minimum set of eight Wang Tiles. The tiling scheme guarantees that two tile choices available for each tiling step. However, Cohen et al.'s implementation of the Wang Tile design fails to incorporate this sound property, leading to the intrinsic problems described in the previous section. We now present a novel Wang Tile design scheme which strictly follows the definition of Wang Tiles and is represented by strict Wang Tile or s-Wang Tile for an abbreviation.

3.1. Tile design

From a given example texture, we randomly sample four diamond-shaped sub-images, like Cohen et al. We combine these four sub-images into a composite image, using the structure shown in Fig. 4(a). The four diamond-shaped sub-images are represented by four colors which correspond to the edge colors of a tile. For example, the structure in Fig. 4(a) corresponds to the tile T_{yrbg}.

The combination rule is also the one used by Cohen et al. [1], but without any of overlapping regions between neighboring tiles. Then the central square portion of the diamond-shaped composite image is cut out to obtain an initial Wang Tile

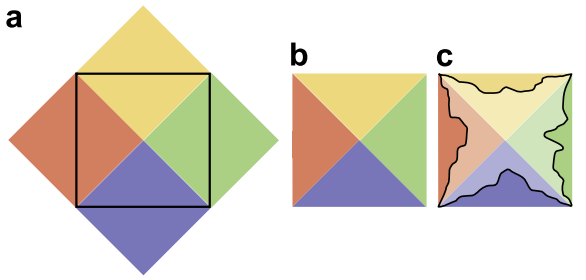

Fig. 4. The strict Wang Tile. (a) Wang Tile structure with four sub-images sampled from the example texture, (b) an initial Wang Tile cut out of (a), and (c) the final Wang Tile after graph cut and replacement.

(Fig. 4(b)), with the characteristic colored edges. But this Wang Tile can not be used to create a larger texture, because there are mismatches on the interior boundaries between sub-images. Fig. 6(a) shows a set of eight of these initial Wang Tiles. In order to eliminate the mismatches, a supplementary square-shaped sub-image is cut out of the example texture and used to cover these mismatches. Using the graph cut technique, we can now construct a weighted graph between the initial Wang Tile and the supplementary sub-image, and then find a closed minimum-cost path in that graph. The region inside the path is then replaced with the supplementary sub-image, while the region outside the path contains the pixels of the initial Wang Tile. The schematic of Fig. 4(c) shows the cutting path determined by graph cut, while the shadowed area shows the pixels replaced by pixels from the supplementary sub-image. In Section 3.2, we will illustrate how graph cutting works in detail.

Fig. 5(a) is a schematic description of the generation of a set of eight Wang Tiles using our tile design scheme. Each supplementary sub-image has its own color. In total there are $4 + 8$ sub-images sampled from the example texture. Because the construction of each Wang Tile now involves a sub-image based on the whole of the example texture, any pixel in the example texture may now be sampled. Fig. 6 shows a set of eight Wang Tiles, based on an example texture which is widely used in the graphics community.

Synthesizing a texture as a tiling, using a set of our s-Wang Tiles, is quite easy. Suppose the plane is to be tiled from West to East and from North to South. Each new tile must have N and W edges that match the S and E edges already placed. The set of eight tiles shown in Fig. 5(a) can be used to tile the plane non-periodically, since there are always two tile choices for each tilling step. Fig. 5(b) shows a small example tiling. In this case, when we get to the bottom-right tile, we have a choice of two tiles, T_{bgbg} and T_{bgyr}, that will match the existing edges both vertically and horizontally.

3.2. Graph cut

A minimum-cost path between the initial Wang Tile and the supplementary sub-image can be located as a cutting path between two overlapping images. We use the graph cut technique, which was originally used [16] to find a minimum cutting path and has since been extended to other applications in image and geometry processing, especially

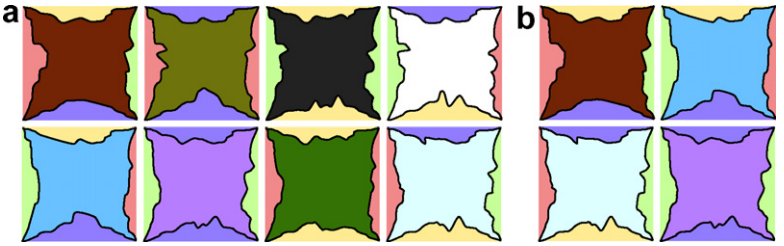

Fig. 5. Our s-Wang Tile pattern and tiling example. (a) A set of eight s-Wang Tiles enough to tile with a non-periodic pattern and (b) an example 2×2 tiling.

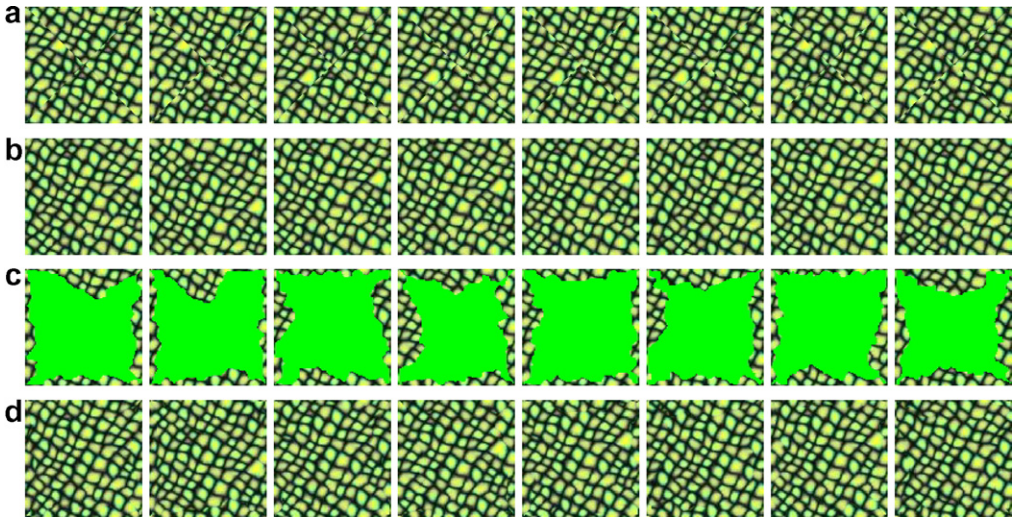

Fig. 6. A set of eight s-Wang Tiles generated by our approach. (a) Initial Wang Tiles generated from four sampled sub-images with overlap, (b) supplementary sub-images taken from the given example texture, (c) minimum-cost paths between the initial Wang Tiles (a) and the supplementary sub-images (b), and (d) final Wang Tiles, in which the portion inside the minimum cost paths are replaced by the supplementary sub-images.

in texturing and video synthesis [16,2,27,28], foreground removal [29], image blending [30] and geometric processing [31,32].

The schematic diagram in Fig. 7 shows a region of overlap between two image patches A and B. Each pixel in the overlap is represented by a node in graph. Each edge of the graph is assigned a weight W which is the sum of the transition overlap error of the two nodes, divided by the sum of their gradients:

$$W(s,t,A,B) = \frac{\|A(s)-B(s)\| + \|A(t)-B(t)\|}{\|G_A^d(s)\| + \|G_A^d(t)\| + \|G_B^d(s)\| + \|G_B^d(t)\|},$$

where s, t are neighboring pixels in the overlapping patches A and B, $G_A^d(s)$ denotes the gradient of pixel s in patch A along d (gradient direction from s to t), and $G_A^d(t)$, $G_B^d(s)$ and $G_B^d(t)$ are similarly defined. This formulation is also used by Ng et al. [2].

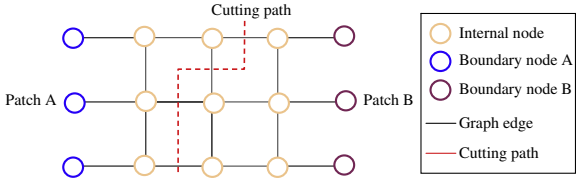

Fig. 7. Schematic graph formulation of the region of overlap between two patches: the minimum-cost graph cut is shown as a red line. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this paper.)

To find a cutting path using the graph cut technique, the value of all the pixels in the border surrounding the overlapping region are constrained to come from existing pixels [16]. In Fig. 7, we select the three leftmost boundary nodes (highlighted in blue) to come from patch A, and the three rightmost boundary nodes (highlighted in dark red) to come

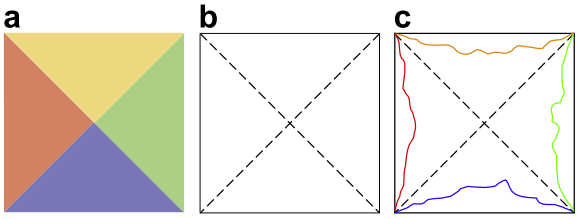

Fig. 8. Construction of a weighted graph. (a) An initial Wang Tile, (b) boundaries of a graph cut: the four edges of the square are taken from the initial Wang Tile and the two dotted diagonals are taken from the supplementary sub-image, and (c) a typical graph cut after searching.

from patch B. The search for a cutting path will be bounded by these two constraints. A boundary of any shape can be used to guide a graph cut.

We use the constraints shown in Fig. 8. For an initial Wang Tile (Fig. 8(a)), and a square supplementary sub-image, we construct a weighted graph as described above. To bound the central portion to be replaced by the supplementary sub-image, we constrain the nodes along the dotted line (Fig. 8(b)) to come from the supplementary sub-image, and the boundary nodes on the four edges of the square to come from the initial Wang Tile. Using this weighted graph and the constraints represented by the boundary nodes, a continuous closed graph cut can be found that visits each corner of the square in turn (Fig. 8(c)). This graph cut is the path through the graph edges that corresponds to the minimum sum of weights. We use the supplementary sub-image to cover the area delineated by this cut. The mismatches between sub-images in the initial Wang Tile disappear. In Fig. 6, we show how a set of eight Wang Tiles is created using this method.

4. Comparisons

For the sake of completeness, here we briefly describe the concept of ω-Tiles. ω-Tiles start with randomly obtaining a set F of four small square patches from the input texture. With these, it forms a square block to construct eventually an ω-Tile. Each block is a non-overlapping arrangement of four (not necessarily distinct) patches of F (Fig. 9(a)). Then an intermediate tile is cut from the center of the arrangement (Fig. 9(b)). The seams in the intermediate tile are removed by replacing the interior of the tile with other pattern from the input texture to generate an ω-Tile (Fig. 9(c)). We refer readers to see [2] for more details.

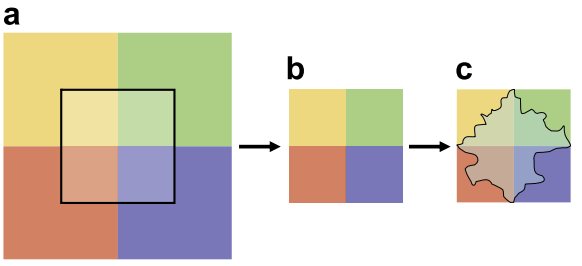

Fig. 9. Construction of ω-Tiles.

Fig. 10 and Table 1 summarize and contrast the way in which Wang Tiles, ω-Tiles and our s-Wang Tiles are designed.

All these methods use a fixed number of sub-images sampled from the example texture to make a tile which can be assembled seamlessly into a large texture. The percentage of the final tile occupied by these four sub-images determines the amount of repetition that occurs in the final synthesized texture. For example, the original Wang Tiles [1] used only four sub-images to generate a set of tiles, and these four sub-images extend over the whole of each tile (see Fig. 10(a)). With a set of only eight Wang Tiles, some repetition is unavoidable. Cohen et al. use a minimum set of eighteen tiles and in practice they used 144 to overcome the corner problem and synthesize a reasonable texture. In ω-Tiles [2], the four sub-images extend over nearly 50% of each tile (see Fig. 10(b)). Therefore, compared with original Wang

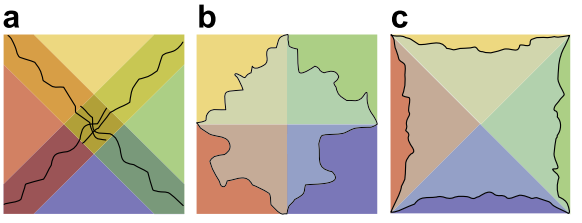

Fig. 10. Design schemes of three types of tiles. (a) Wang Tiles, (b) ω-Tiles, and (c) our s-Wang Tiles.

Table 1
Comparison between Wang Tiles, ω-Tiles and s-Wang Tiles

	Wang Tiles	ω-Tiles	Ours
Corner problem	Yes	No	No
Joint problem	Yes	No	No
Sampling problem	Yes	No	No
Tile quality	Fair	Good	Good
Number of tiles	64	32	8
Percentage by four tiles	100%	50%	25–30%
Synthesis quality	Fair	Good	Good
Combination rule	Simple	Complex	Simple

Tiles, fewer tiles are needed to achieve the same quality of texture. In theory, a set of at least 32 ω-Tiles is required to maintain non-periodicity, although 16 tiles seem sufficient in practice [2]. In our s-Wang Tiles, the four sub-images only extend over 25–30% of each tile, as illustrated schematically in Fig. 10(c). The effect of the four original sub-images is localized to the edges of the tile, and their contribution is to avoid seams in the final texture. The supplementary sub-images increase the diversity of texture in the tiles. Thus our s-Wang Tiles can synthesize good textures with a smaller set of tiles.

In Fig. 11, we give some samples generated using these three types of tiles. We refer readers to see http://www.heroicsalmonleap.net/mle/wang/ for an implementation of Wang Tiles and http://www.comp.nus.edu.sg/tants/w-tile/ for more results on ω-Tiles. As shown in Table 1, our approach has the following advantages compared with the original Wang Tiles and with ω-Tiles.

- *No corner problem*. Unlike the original Wang Tiles, our s-Wang Tiles match correctly along their edges, and at corners. As a result, we do not need to increase the tile set by encoding for corner type.
- *No joint problem*. Our scheme introduces a supplementary sub-image sampled from the example texture to replace the central portion of the initial Tiles. This makes the final pixel information more diverse. We also use graph cut, instead of dynamic programming, to eliminate the boundary mismatches between sub-images. Because the graph cut takes place in a large area, the quality of the resulting path is good.
- *No sampling problem*. Our scheme is not restricted to four sub-images unlike the original Wang Tiles. And the supplementary sub-images sampled from the example texture are not diamond-shaped, but square, which means that any portion of the example texture may be sampled.
- Compared with ω-Tiles, our approach can synthesize less repetitive texture pattern, because the four sub-images needed to avoid seams between tiles occupy a smaller area. In addition, our tile combination rule is simper than that for ω-Tiles [2].
- Our approach can reduce the number of tiles from 144 Wang Tiles or 32 ω-Tiles to eight, without compromising the quality of the final texture.
- In addition, our approach has the advantages compared with the non-tiling methods. The tile set can be generated at a precomputation stage. Tiling can be performed on the fly, and can be supported by modern GPUs [22,14].

5. Experimental results and discussion

5.1. Experimental results

Figs. 12–14 demonstrate texturing examples synthesized by using our approach. The first column of each figure shows the input textures including sto-

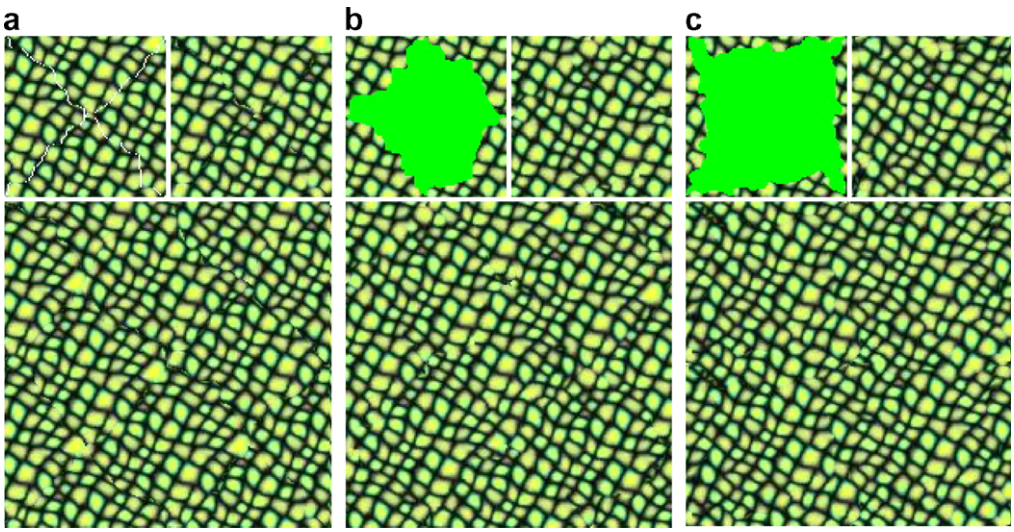

Fig. 11. Tile construction and synthesized texture using three types of tiles: (a) Wang Tiles, (b) ω-Tiles, and (c) Our s-Wang Tiles. In each picture, the top two images show the cutting path searched in a tile (left) and the resulting tile (right). The lower image is a 2×2 synthesized texture.

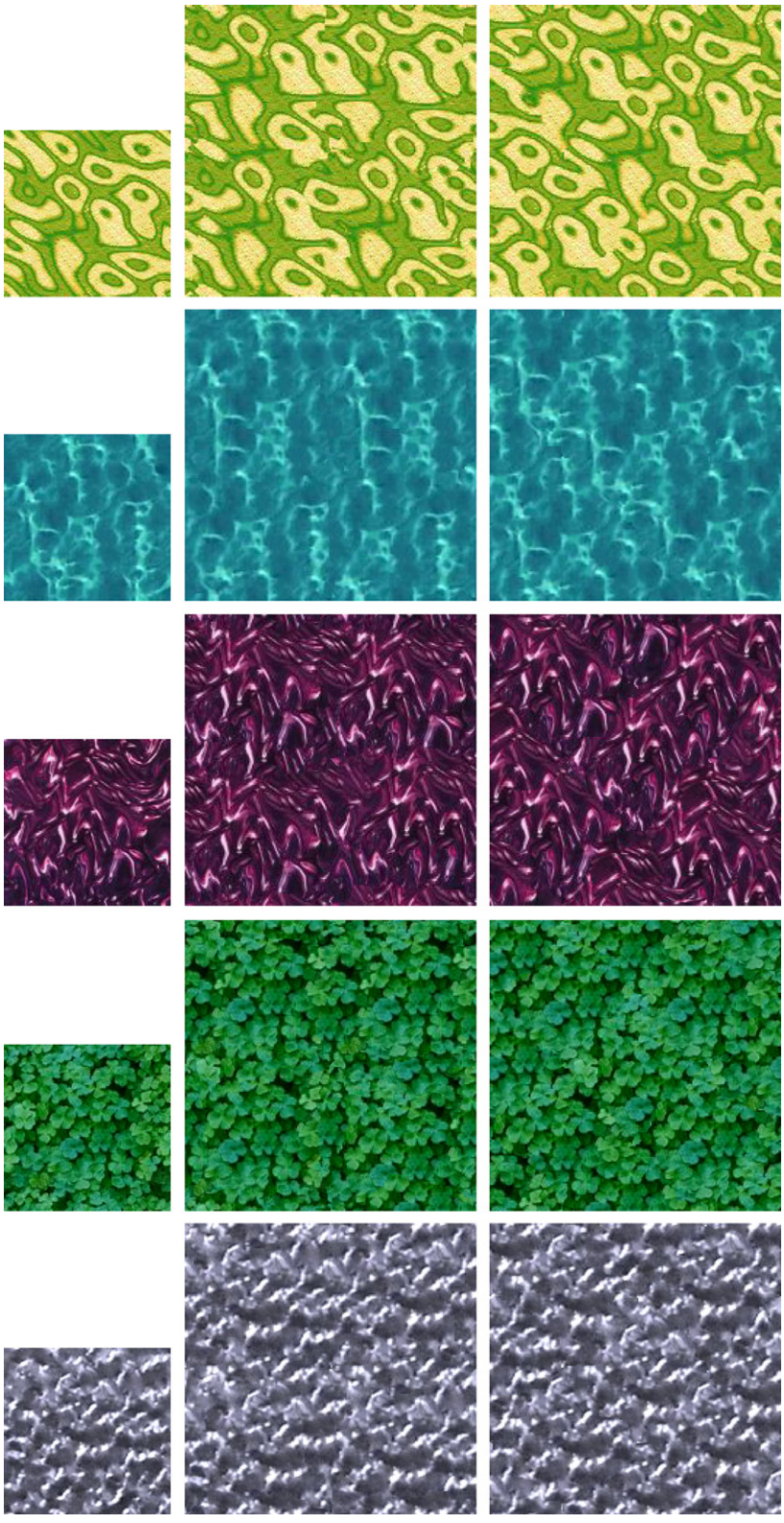

Fig. 12. Stochastic textures. From left to right: input textures, our s-Wang Tiles, and ω-Tiles.

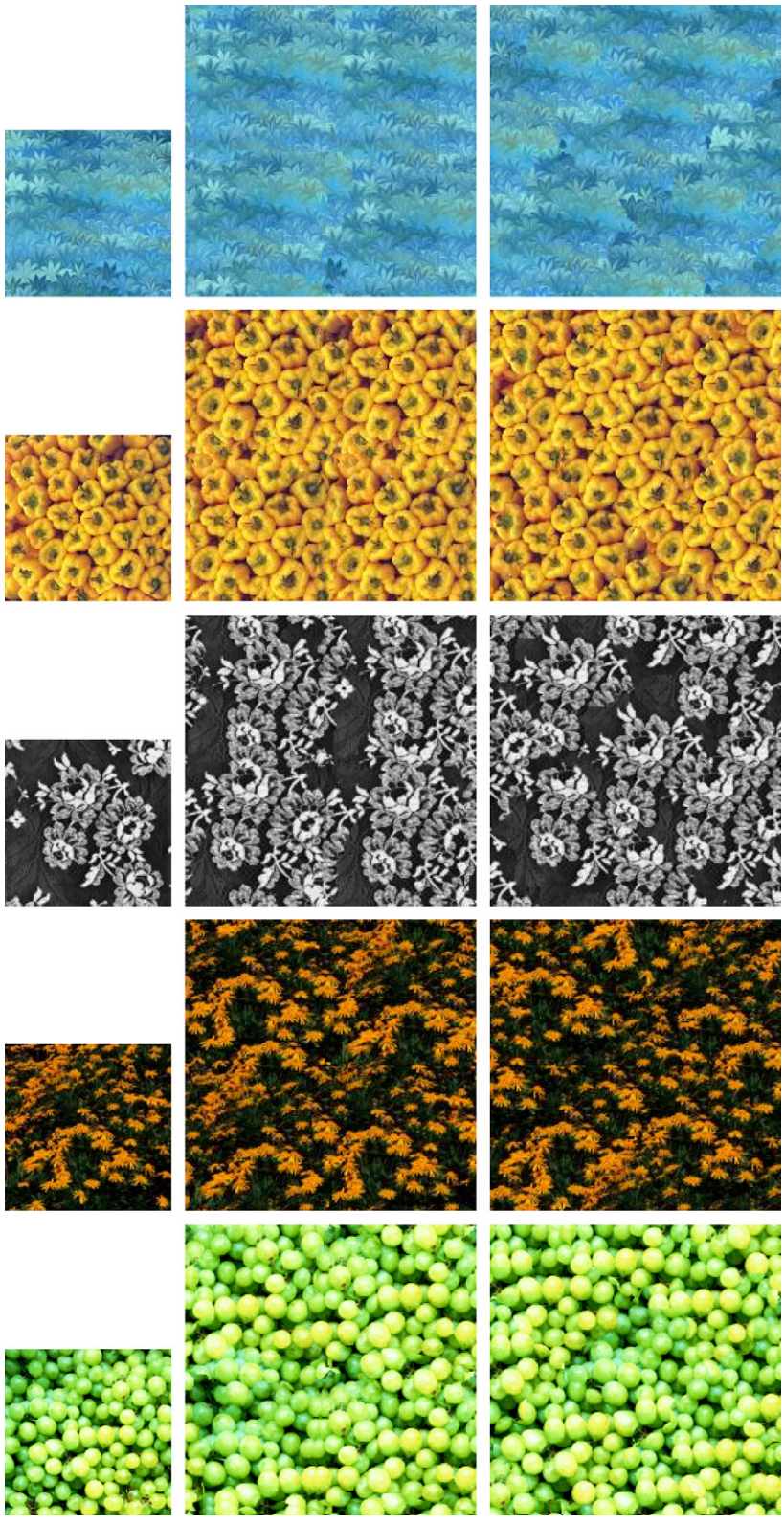

Fig. 13. Semi-structured textures. From left to right: input textures, our s-Wang Tiles, and ω-Tiles.

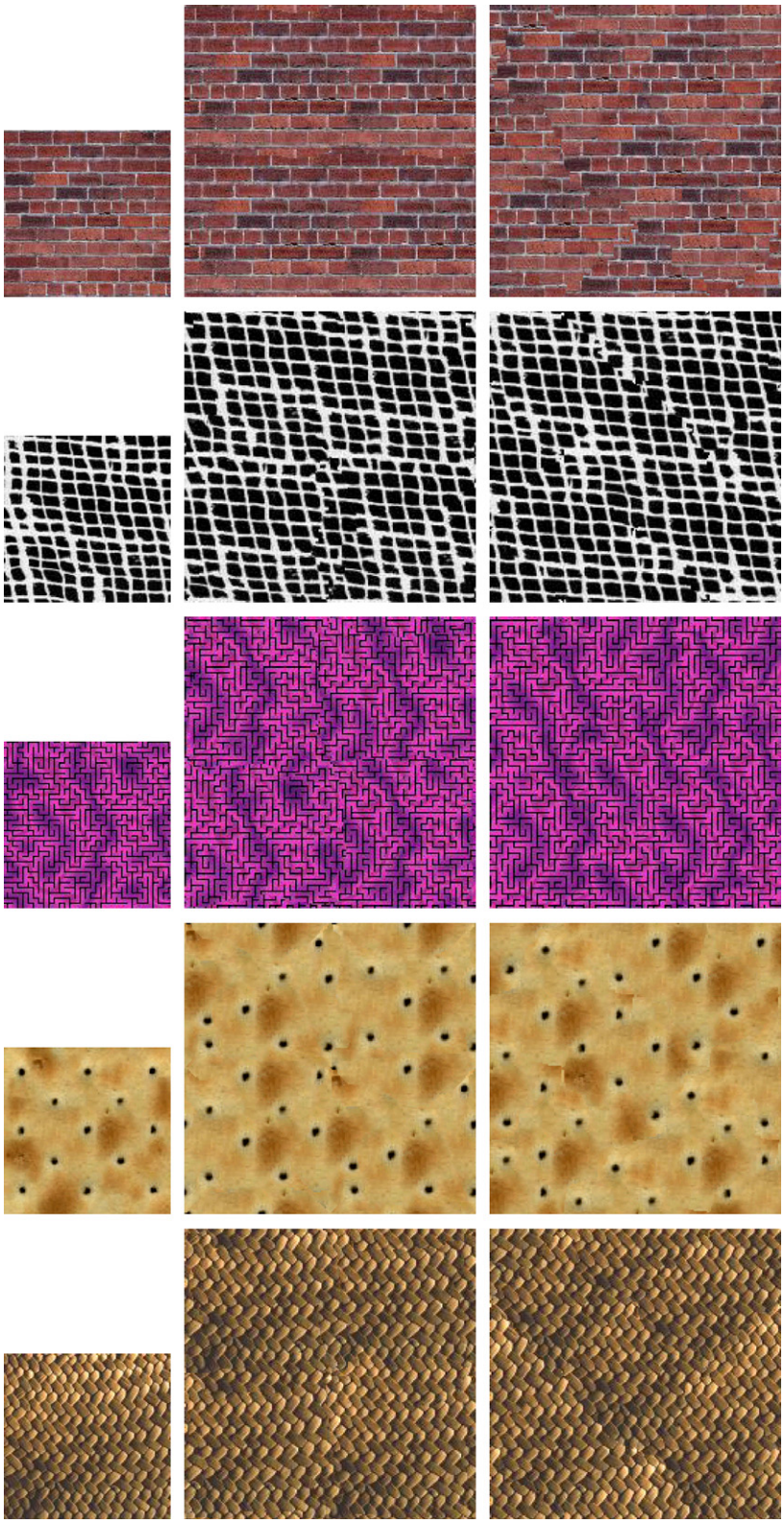

Fig. 14. Structured textures. From left to right: input textures, our s-Wang Tiles, and ω-Tiles.

chastic (Fig. 12), semi-structured (Fig. 13) and structured textures (Fig. 14). The second and third columns show the resulting textures synthesized using our approach and ω-Tiles, respectively. Finally, Fig. 15 demonstrates the examples of large textures synthesized by our approach for stochastic, semi-structured and structured input textures. All the example textures have a resolution of 128×128 and all the s-Wang Tiles have a resolution of 100×100. All the tiled textures are synthesized with a set of eight s-Wang Tiles. Precomputing a set of tiles takes 6–7 s and synthesizing any large texture can be done on the fly. Our algorithm was implemented using C++ under MS Windows. All results are obtained on an Intel Xeon 2.8 GHz PC.

5.2. Discussion

5.2.1. Smoothing

A few postprocessing techniques have been reported to reduce the seam along a cutting path. [16] utilizes blending operations to make the cutting path smooth. Our preliminary work on ω-Tiles [2] uses the poisson approach [33] to smooth the prominent seams in ω-Tiles. We have noticed that these techniques can remove seams to a certain extent, but new artifacts, for example, blur effect, can be introduced. This phenomenon may result in destroying the global visual appearance. For example, refer to Fig. 10b-vi in [2]. As a result, we did not employ smoothing techniques in our s-Wang Tiles.

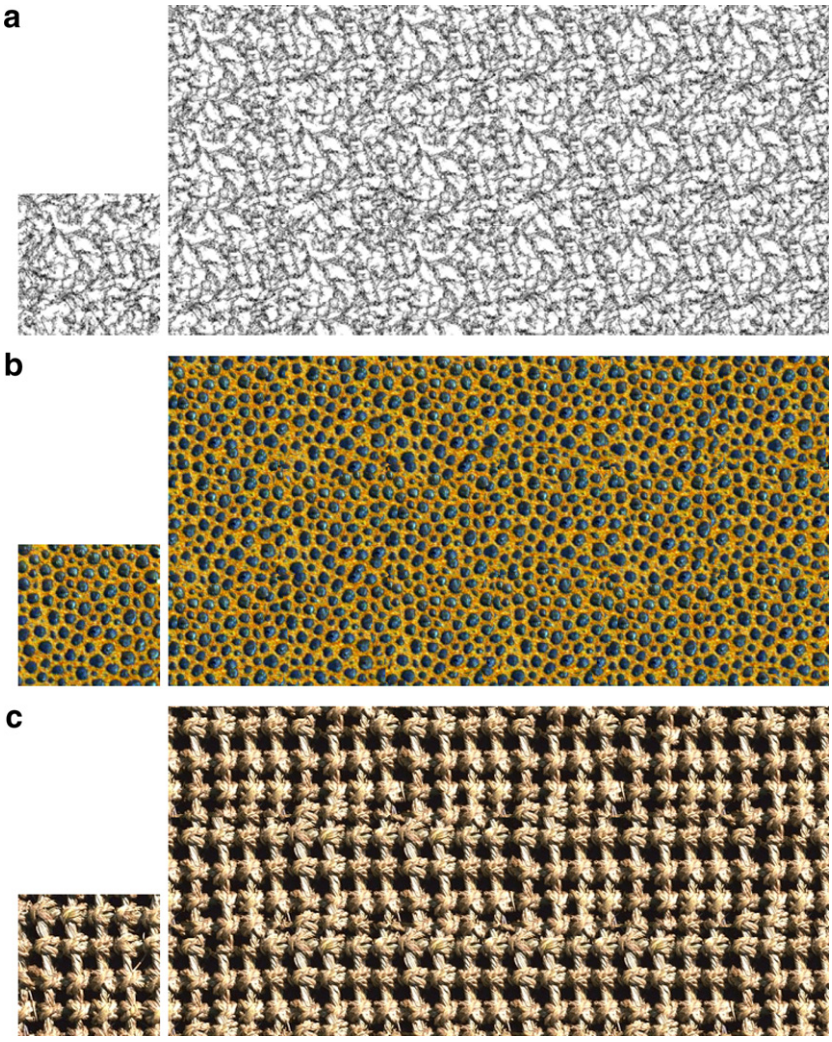

Fig. 15. Large textures (3×6) synthesized using s-Wang Tiles. (a) Stochastic texture, (b) semi-structured texture, and (c) structured texture.

5.2.2. Tile resolution

The maximum resolution of each tile can be determined by the resolution of input textures. In general, the higher the tile resolution becomes, the better global synthesis quality the resulting textures can generate, because the low tile resolution induces a small search region for finding a cutting path, i.e., less opportunities for finding a good cutting path. On the other hand, the low tile resolution can generate more texture diversity and requires less precomputation time to generate tiles. In order to compensate for these requirements, in our case, we choose 100×100 as a tile resolution for 128×128 input textures throughout our experiments. As a result, we can preserve the global synthesis quality while maintaining the texturing diversity.

6. Conclusions and future work

We have presented an improved technique to create a smaller set of more efficient and stricter Wang Tiles for synthesizing large textures without the problems associated with the original Wang Tiles. Using graph cut and sampling more than four sub-images from the given example texture, both tile and texture quality are improved, while the size of the tile set is reduced. We have demonstrated our approach on a wide range of texture examples.

One limitation of our approach is the requirement for more precomputation time, because it is necessary to traverse the whole example texture to find the most suitable sub-images using graph cut which is a time-consuming algorithm. However, once a set of tiles are generated as precomputation, the synthesis procedure now becomes quite fast. Moreover, mipmapping can be easily dealt with by generating a series of pre-synthesized textures at different resolutions.

The experimental results have shown that s-Wang Tiles are very suitable for synthesizing stochastic and semi-structured textures. We also have demonstrated synthesizing structured textures and their final results show reasonable quality. But for those textures with obvious structural features or near-regular patterns, the global pattern may not be well preserved. For example, some noticeable artifacts, as shown in the second row of Fig. 13 and second and fourth rows of Fig. 14, can be recognized. Thus, for future work, we plan to take into account pattern structures existing in the given example textures. We would also like to integrate pattern detection into the search for finding good cutting paths using graph cut. Another area for future research direction is incorporation of user control mechanism into tiling-based texturing methods.

References

[1] M.F. Cohen, J. Shade, S. Hiller, O. Deussen, Wang tiles for image and texture generation, ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH) 22 (3) (2003) 287–294.

[2] T.-Y. Ng, C.H. Wen, T.-S. Tan, X.Y. Zhang, Y.J. Kim, Generating an ω-Tile set for texture synthesis, in: Proceedings of the 23rd Computer Graphics International 2005 (CGI'05), 2005, pp. 177–184.

[3] D. Garber, Computational models for texture analysis and texture synthesis, Ph.D. Dissertation, University of Southern California, 1981.

[4] A.A. Efros, T.K. Leung, Texture synthesis by non-parametric sampling, in: Proceedings of ACM SIGGRAPH, 1999, pp. 1033–1038.

[5] K. Popat, R. Picard, Novel cluster-based probability model for texture synthesis, classification, and compression, in: Proceedings of Visual Communication and Image Processing, 1993, pp. 756–768.

[6] L. Wei, M. Levoy, Fast texture synthesis using tree-structured vector quantization, in: Proceedings of SIGGRAPH, 2000, pp. 479–488.

[7] M. Ashikhmin, Synthesizing natural texture, in: Proceedings of the Symposium on Interactive 3D Graphics, 2001, pp. 217–226.

[8] X. Tong, J. Zhang, L. Liu, X. Wang, B. Guo, H.-Y. Shum, Synthesis of bi-directional texture functions on arbitrary surfaces, ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH) 21 (3) (2002) 665–672.

[9] S. Zelinka, M. Garland, Towards real-time texture synthesis with the jump map, in: Proceedings of the 13th Eurographics workshop on Rendering, 2002, pp. 99–104.

[10] J.S. DeBonet, Multiresolution sampling procedure for analysis and synthesis of texture image, in: Proceedings of ACM SIGGRAPH, 1997, pp. 361–368.

[11] A. Hertzmann, C.E. Jacobs, N. Oliver, B. Curless, D.H. Salesin, Image analogies, in: Proceedings of ACM SIGGRAPH, 2001, pp. 327–340.

[12] J. Zhang, K. Zhou, L. Velho, B. Guo, H.-Y. Shum, Synthesis of progressively-variant textures on arbitrary surfaces, ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH) 22 (3) (2003) 295–302.

[13] L. Tonietto, M. Walter, Towards local control for image-based texture synthesis, in: Proceedings of the 15th Brazilian Symposium on Computer Graphics and Image Processing, 2002, p. 252.

[14] S. Lefebvre, H. Hoppe, Parallel controllable texture synthesis, ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH) 24 (3) (2005) 777–786.

[15] A.A. Efros, W.T. Freeman, Image quilting for texture synthesis and transfer, in: Proceedings of ACM SIGGRAPH, 2001, pp. 341–346.

[16] V. Kwatra, A. Schodl, I. Essa, G. Turk, A. Bobick, Graphcut texture: image and video synthesis using graph cuts, ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH) 22 (3) (2003) 277–286.

[17] V. Kwatra, I. Essa, A. Bobick, N. Kwatra, Texture optimization for example-based synthesis, ACM Transactions on

Graphics (Proceedings of ACM SIGGRAPH) 24 (3) (2005) 795–802.
[18] E. Praun, A. Finkelstein, H. Hoppe, Lapped textures, in: Proceedings of ACM SIGGRAPH, 2000, pp. 465–470.
[19] J. Stam, Aperiodic texture mapping, in: Technical Report, European Research Consortium for Informatics and Mathematics, 1997.
[20] F. Neyret, M. Cani, Pattern-based texturing revisited, in: Proceedings of SIGGRAPH, 1999, pp. 235–242.
[21] C. Soler, M.P. Cani, A. Angelidis, Hierarchical pattern mapping, ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH) 21 (3) (2002) 673–680.
[22] L.-Y. Wei, Tile-based texture mapping on graphics hardware, in: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS Conference on Graphics Hardware, 2004, pp. 55–63.
[23] S. Lefebvre, F. Neyret, Pattern based procedural texture, in: Proceedings of the Symposium on Interactive 3D Graphics, 2003, pp. 203–212.
[24] Y. Liu, W.-C. Lin, J.H. Hays, Near regular texture analysis and manipulation, ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH) 23 (3) (2004) 368–376.
[25] Y. Liu, Y. Tsin, W.-C. Lin, The promise and perils of near-regular texture, International Journal of Computer Vision 62 (1–2) (2005) 145–159.
[26] I.A. Shevelev, V.M. Kamenkovich, G.A. Sharaev, The role of lines and corners of geometric figures in recognition performance, Acta Neurobiologiae Experimentalis 63 (4) (2003) 361–368.
[27] J. Wang, P. Bhat, R.A. Colburn, M. Agrawala, M.F. Cohen, Interactive video cutout, ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH) 24 (3) (2005) 585–594.
[28] V. Kolmogorov, R. Zabih, What energy functions can be minimized via graph cuts? in: Proceedings of the Seventh European Conference on Computer Vision-Part III, Springer-Verlag, Berlin, 2002, pp. 65–81.
[29] C. Rother, V. Kolmogorov, A. Blake, grabcut: Interactive foreground extraction using iterated graph cuts, ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH) 23 (3) (2004) 309–314.
[30] A. Agarwala, M. Dontcheva, M. Agrawala, S. Drucker, A. Colburn, B. Curless, D. Salesin, M. Cohen, Interactive digital photomontage, ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH) 23 (3) (2004) 294–302.
[31] A. Sharf, M. Alexa, D. Cohen-Or, Context-based surface completion, ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH) 23 (3) (2004) 878–887.
[32] Y. Boykov, V. Kolmogorov, Computing geodesics and minimal surfaces via graph cuts, in: Proceedings of the IEEE International Conference on Computer Vision, 2003, pp. 26–33.
[33] P. Pérez, M. Gangnet, Blake, Poisson image editing, ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH) 22 (3) (2003) 313–318.