Erschienen in: Computer Graphics Forum ; 29 (2010), 4. - S. 1431-1439
https://dx.doi.org/10.1111/j.1467-8659.2010.01740.x

Semi-Stochastic Tilings for Example-Based Texture Synthesis

Thomas Schlömer Oliver Deussen

University of Konstanz, Germany

Abstract

We investigate semi-stochastic tilings based on Wang or corner tiles for the real-time synthesis of example-based textures. In particular, we propose two new tiling approaches: (1) *to replace stochastic tilings with pseudo-random tilings based on the Halton low-discrepancy sequence, and* (2) *to allow the controllable generation of tilings based on a user-provided probability distribution. Our first method prevents local repetition of texture content as common with stochastic approaches and yields better results with smaller sets of utilized tiles. Our second method allows to directly influence the synthesis result which—in combination with an enhanced tile construction method that merges multiple source textures—extends synthesis tasks to globally-varying textures. We show that both methods can be implemented very efficiently in connection with tile-based texture mapping and also present a general rule that allows to significantly reduce resulting tile sets.*

1. Introduction

Creating rich and complex content is a major problem in computer graphics, especially in interactive applications where large amounts of content have to be produced very quickly and from limited data. Tile-based methods mitigate this problem by synthesizing large amounts of content out of a much smaller data set of tiles by generating a valid tiling. For example-based texture synthesis, tilings based on square *Wang tiles* with colored edges or, preferably, square *corner tiles* with colored corners have proven particularly useful. Once a set of carefully constructed tiles has been generated from a provided input texture, arbitrary amounts of this input texture can be produced at runtime in connection with tile-based texture mapping.

So far, research has only focused on tilings of stochastic nature which suffer from two unsolved problems. First, they are prone to local repetition artifacts as the random distribution of tile edge or corner colors often leads to noticeable clusters of tiles showing identical content (cf. Figure 5). And second, they are limited to homogeneous (stationary) textures as tiles are constructed only from a single input texture and are then distributed merely in a random i.e. in an uncontrolled way. In this paper, we propose two new tiling methods that solve these problems.

We show that the mentioned repetition artifacts can be minimized by our first method which replaces the stochastic distribution of tile colors with a pseudo-random distribution that is more uniform in the sense that it is less probable that neighboring tile edges or corners are of the same color (and hence represent the same content). Our method is based on the Halton low-discrepancy sequence which we utilize to pseudo-randomly enumerate the integer lattice and then assign colors to each edge or corner on the basis of these enumeration indices. At the same time, it allows random access to tiles which is important to maintain runtime synthesis in combination with tile-based texture mapping.

The limitation of stochastic tilings to stationary texture synthesis can be lifted by our second method which generalizes the tiling process by allowing the user to control the distribution of tiles. Tilings are derived from a user-specified color probability function which defines the probability for each edge or corner color at each point in the tiling space. We assign related but different textures to each color and construct tiles in a way that each texture content is represented equally in the resulting tile set. Since the controlled distribution of colors directly translates to the distribution of associated texture content, this strategy extends texture synthesis to globally-varying textures. Analyzing the underlying probability distribution with respect to occurring color combinations also allows us to significantly reduce the size of resulting tile sets.

2. Related Work

Tile-based methods have been applied to a variety of synthesis problems, among them the synthesis of textures [Sta97, CSHD03, NWT*05, LD06a], point distributions [KCODL06, Ost07], and volume data [LEQ*07, PGMG09]. A comprehensive overview can be found in [LKF*08] and a general introduction to tilings in [GS86]. For a good overview over the broad field of example-based texture synthesis we refer to [WLKT09].

Tile-based texture synthesis was first considered by Stam [Sta97] and later extended to example-based texture synthesis in [CSHD03, NWT*05, LD06a]. Cohen et al. [CSHD03] merged different patches of an input texture by constructing Wang tiles in correspondence to their edge colors and presented a first stochastic tiling algorithm which places tiles in scanline order. Wei [Wei04] and Lagae and Dutré [LD06a] improved this algorithm by allowing random access to tiles which is important for tile-based texture mapping [LN03, Lef08]. Fu and Leung [FL05] extended the tiling mechanism to arbitrary surfaces. Still, all of these approaches only generated stochastic tilings.

Cohen et al. [CSHD03] were also the first to consider tile construction from multiple input textures in order to generate non-stationary (globally-varying) results but still only in a stochastic way. Example-based texture synthesis of non-stationary characteristic was also considered by several non-tile-based approaches [Ash01, MZD05] which, however, do not allow runtime synthesis comparable to the performance of tile-based texture mapping. The idea of controlling tilings based on a probability distribution was considered for manually created textures or small patterns (textons) by [NC99, LN03] and for volume illustrations by Lu et al. [LEQ*07], but none of these techniques directly translates to example-based texture synthesis.

The problem of a more uniform distribution of colors also roughly parallels problems in vector error diffusion [SAFS99], color filter array design [Con09], or multi-class Poisson disk sampling [Wei09] but these solutions do not allow local evaluation as needed by our application scenario.

3. Our Tiling Methods

Before we introduce two new tiling methods, let us briefly recapitulate the necessary background on valid tilings based on Wang or corner tiles. We also formulate a basic function which captures such tiling methods in general.

3.1. Valid Tilings

Wang tiles are unit square tiles with colored edges [Wan61]. Since Wang tiles only enforce continuity with their horizontal and vertical but not their diagonal neighbors, continuity problems near tile corners may cause artifacts in synthesized signals, a problem commonly known as the corner problem.

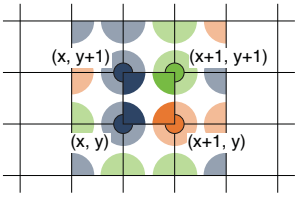


Figure 1: *Tilings based on corner tiles may be evaluated locally by assigning colors to an underlying integer lattice and then deriving the tile from the resulting color combination.*

For this reason, *corner tiles* were proposed as an alternative to Wang tiles [NWT*05, LD06a]. Corner tiles are unit square tiles with colored corners that enforce continuity with all their neighbors, and are stricter than Wang tiles in the sense that every set of corner tiles can be transformed into an equivalent Wang tile set, while the converse is not true. For these reasons, the remainder of this paper concentrates on corner tiles even though both of our tiling methods can be used with Wang tiles as well.

Let $\mathcal{T}$ be a finite set of corner tiles and let $\mathcal{C} = \{0, 1, \ldots, C-1\}$ be the set of $C \geq 2$ different colors in $\mathcal{T}$. As the tiles have four corners, $\mathcal{T}$ can contain at most C^4 different tiles. These tiles can be uniquely identified by their corner color combination or by a tile index i, i.e. they can be represented by C-ary numbers with 4 digits $(c_j)_{j=0}^{3}$ or by the decimal integers $0, 1, \ldots, C^4 - 1$. The two representations are connected by common radix conversion, i.e.

$$i = \sum_{j=0}^{3} c_j(i) C^j \quad \text{and} \quad c_j = (i/C^j) \bmod C \tag{1}$$

for $0 \leq j \leq 3$.

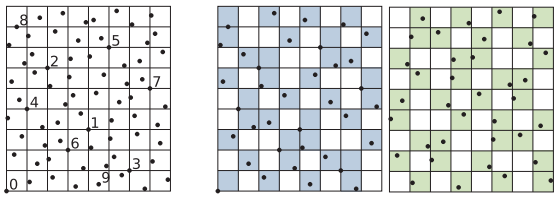
We now consider *tilings* of the plane in which tiles are placed on the integer lattice points with their edges axis-aligned, so that they partition the plane. The tiles may not be rotated. A given tiling is *valid* if tile corners have matching colors everywhere.

A straightforward way to generate a valid tiling is to place tiles in scanline order, ensuring that neighboring tiles have matching corner colors [CSHD03]. This way, however, a tiling has to be generated in its entirety in order to evaluate a single tile of interest. A better way is to align tile corners to the integer lattice points which have instead been assigned a color $c \in \mathcal{C}$ (cf. Figure 1). This way, tiles are implicitly defined by the resulting corner color combinations and can be evaluated locally [Wei04, LD06a]. This approach is captured by a function h that maps lattice points to colors, i.e.

$$h : \mathbb{Z}^2 \to \mathcal{C}. \tag{2}$$

We call h the *color distribution function*.

Existing research can be classified as choosing h as a stochastic hash function that returns random color values at each lattice point, hence producing stochastic tilings.



(a) (b)

Figure 2: (a) *The first 72 points of the scaled Halton sequence induce a stratification grid of size* 8×9*. While the sequence unfolds the grid cells get enumerated as indicated by the first 10 point indices.* (b) *Dividing the point set (and thus the set of grid cells) on the basis of these indices into two halves yields two evenly distributed corner color classes.*

3.2. Deterministic Tilings

However, distributing colors randomly may lead to large clusters of corners of the same color. As typically each color corresponds to specific texture content during tile-based synthesis, these clusters lead to local repetition artifacts in the synthesized results. We thus propose an alternative method for distributing corner colors that avoids such large clusters of the same corner color yet still maintains a pseudo-random appearance. In particular, we are interested in a more uniform distribution of corner colors in the sense that it is less probable that neighboring corners have identical color, and that each class of corner colors offers a distribution of comparable uniformity. At the same time we want to maintain the independent evaluation of corner colors to ensure random access to tiles just as existing direct stochastic tiling algorithms.

These requirements parallel the characteristic of low-discrepancy sequences from quasi-Monte Carlo theory which unfold incrementally in a way that a set of n points does not have to be discarded when generating the $(n+1)$-th point [Nie92]. In fact, every subset of such a sequence offers good uniformity properties with respect to axis-aligned boxes anchored at the origin. Of particular interest are radical-inverse based low-discrepancy sequences as they exhibit intrinsic stratification and as such pseudo-randomly enumerate voxels in any dimension [Kel04]. As we are only interested in the enumeration of the two-dimensional integer lattice (and not the points itself) we found the unscrambled Halton sequence sufficient for our purposes.

Tiling Method The Halton sequence [Hal60] is based on the van der Corput radical inverse function ϕ_b which maps integers to the unit interval by mirroring its b-adic expansion around the radix point [Nie92], i.e.

$$\phi_b(i) = \sum_{k=1}^{\infty} a_k(i) \cdot b^{-k}, \tag{3}$$

where $a_k(i)$ denotes the k-th digit of the integer $i \in \mathbb{N}_0$ in base b. The two-dimensional Halton sequence then constitutes as

$$x_i = \big(\phi_{b_1}(i), \phi_{b_2}(i)\big),$$

where the bases b_1 and b_2 have to be relatively prime and are typically picked as $b_1 = 2$ and $b_2 = 3$.

Multiplying the resulting point coordinates by powers of their respective bases reveals the aforementioned stratification property. An example is shown in Figure 2(a): here the first 72 points induce a stratification grid of size $2^3 \times 3^2$. In general, the scaled Halton sequence

$$x'_i = \big(2^{n_1}\phi_2(i), 3^{n_2}\phi_3(i)\big)$$

induces a stratification grid of size $2^{n_1} \times 3^{n_2}$ where the exponents $n_1, n_2 \in \mathbb{N}_0$ are chosen such that the intrinsic grid is large enough to cover a desired tiling resolution $T_x \times T_y$, i.e. $2^{n_1} \geq T_x + 1$ and $3^{n_2} \geq T_y + 1$.

The key observation now is that the scaled Halton points x'_i pseudo-randomly enumerate the induced grid such that each subset of points/grid cells is of nice uniformity. Thus, we divide the set of grid cells on the basis of the Halton point indices i' into C classes corresponding to the C corner colors of a desired tiling. Figure 2(b) shows an example for $C = 2$ colors where the first 36 cells are assigned to color class 0 (blue) and the second 36 cells to color class 1 (green).

In general, a corner color may be derived from a Halton point index i' by our color distribution function h with the mapping

$$h : (x, y) \mapsto \left\lfloor \frac{i'}{2^{n_1}3^{n_2}} C \right\rfloor. \tag{4}$$

Tile indices may then be derived via Equation (1).

Implementation Currently, the Halton indices i' are derived only in a "forward" manner from the original integers i and not directly from tiling grid coordinates (x, y). This is inefficient and prevents an implementation as a fragment shader for tile-based texture mapping because we have to compute every Halton point up to the one which falls into (x, y).

It was recently shown by Raab et al. [RGAK09, RGK10] that there is a direct way to compute the index of the first Halton point that falls into a specified voxel. For our two dimensional case their findings simplify to

$$i' = \big(l_1 p_2 m_1 + l_2 p_1 m_2\big) \bmod \big(p_1 p_2\big), \tag{5}$$

where an index i' has been decomposed into $i' = b_j^{n_j} h_j + l_j$ and is represented by its n_j least significant digits l_j and its remaining most significant digits h_j with respect to base b_j in dimension j. Furthermore, $p_1 = 2^{n_1}$ and $p_2 = 3^{n_2}$ with $m_1 = (p_2^{-1}) \bmod p_1$ and $m_2 = (p_1^{-1}) \bmod p_2$ their multiplicative

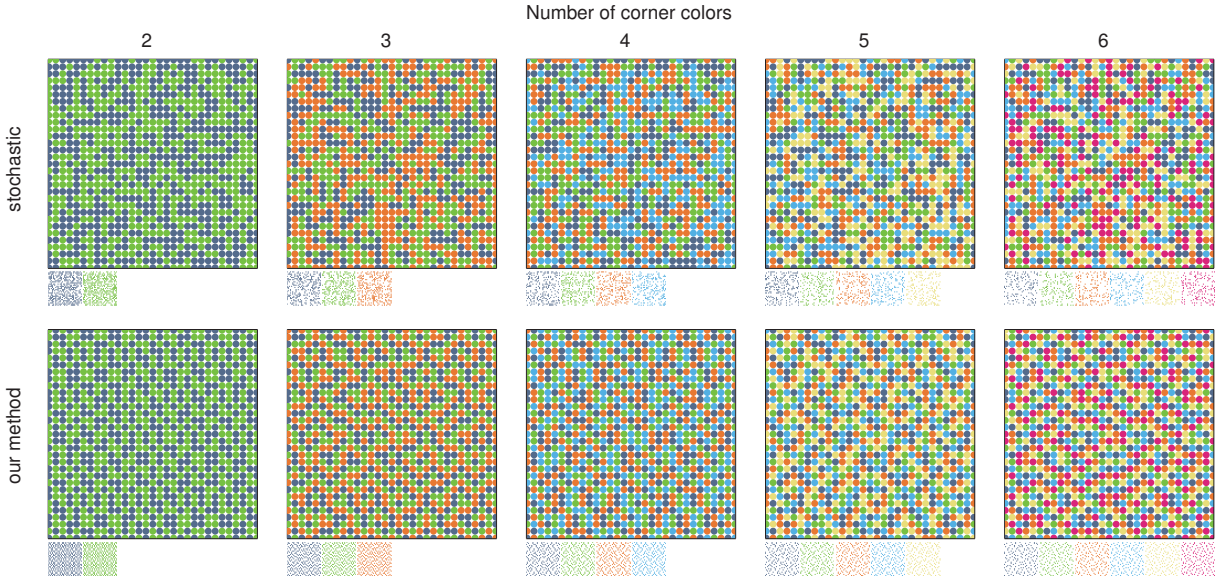


Figure 3: *Tilings of size* 30×30 *for various number of corner colors. Tile borders have been removed for a better view on the resulting color distributions. In contrast to the stochastic approach our method produces tilings without any large clusters of corners with identical color and an overall more "even" distribution of corner colors. This also becomes evident when considering the individual classes of colors which are depicted below each tiling.*

inverse, and

$$l_1 = \phi_2^{-1}\left(\frac{x}{2^{n_1}}\right) \text{ and } l_2 = \phi_3^{-1}\left(\frac{y}{3^{n_2}}\right),$$

where $\phi_{b_j}^{-1}$ is the inverse of (3) and reverses the digits before putting them to the left side of the radix point. For more details, we refer to [RGAK09].

Still, a remaining problem with (5) is that the products $l_1 p_2 m_1$ and $l_2 p_1 m_2$ quickly grow very large which is of special concern for the fragment shader implementation that typically only supports 32-bit integers. However, by using modular arithmetic we can rewrite (5) to yield

$$i' = \big(p_2(l_1 m_1 \bmod p_1) + p_1(l_2 m_2 \bmod p_2)\big) \bmod \big(p_1 p_2\big). \tag{6}$$

Since $l_1, m_1 \leq p_1$ and $l_2, m_2 \leq p_2$, the largest subtotal then is $\max\big(p_1^2, p_2^2, 2p_1p_2\big)$ which effectively pushes the limit for tilings based on (6) to e.g. square tilings of size $\approx 46{,}000^2$ before they would start to repeat. Also note that since the divisors in (6) are powers of the fixed primes 2 and 3, the modulo operations may be implemented as optimized versions using bitwise operators.

Evaluation Figure 3 compares various corner color distributions based on our deterministic method with distributions obtained from the stochastic tiling algorithm by Lagae and Dutré [LD06b]. In contrast to this stochastic approach our method produces tilings without large clusters or long streaks of corners of identical color and shows an overall distribution of corners colors that is more uniform, yet still pseudo-random. This becomes particularly evident when considering the individual classes of colors which are depicted below each tiling.

C	Neighbors mean		Neighbors std. dev.	
	stochastic	our	stochastic	our
2	3.9970	3.4967	1.4142	0.5857
3	2.6653	2.1548	1.3337	0.6322
4	1.9981	1.5566	1.2241	0.6620
5	1.5981	1.1536	1.1311	0.6965
6	1.3328	0.8032	1.0536	0.6720
7	1.1421	0.6968	0.9890	0.6516
8	0.9993	0.6058	0.9350	0.6236

Table 1: *Mean and standard deviation for the number of identically colored neighbors based on tilings of various sizes. The results for the stochastic approach are based on the stochastic hash function from Lagae and Dutré [LD06b].*

We also performed a quantitative analysis of the resulting corner color distributions where we were interested in identifying cluster of the same corner colors. For this purpose, we analyzed the local 8-neighborhood of each corner and counted the number of identically colored neighbors. Table 1 lists mean and standard (RMS) deviation for this measure based on 100 tilings of random resolutions up to 4096×4096. Compared to the stochastic approach our method consistently generates corners with fewer neighbors of identical color at smaller variance.

3.3. Controllable Tilings

While our deterministic method generates tilings with a more uniform distribution of corner colors, a natural extension to the tiling process is to allow the user to control the distribution of corner colors. As each color is associated with specific texture content during tile-based synthesis, this results in a direct way to influence the synthesized texture. This extends example-based synthesis to globally-varying textures as we will show in Section 4.

The key observation is that we can control resulting tilings by defining a random field that provides the probability for each corner color at each point in the tiling space. To determine a corner color at a given lattice point we then simply have to sample the discrete probability distribution at this lattice point.

Tiling Method Assume each point $\mathbf{x} \in [0,1]^2$ is assigned a discrete random variable $X_\mathbf{x}$ that can take values from our set of colors $\mathcal{C} = \{0,1,\ldots,C-1\}$. Let $p_c \equiv \Pr(\{X_\mathbf{x} = c\}), c \in \mathcal{C}$ be the probability that the color c is assigned to point $\mathbf{x}$. Then the table

$$P := \begin{pmatrix} 0 & 1 & \cdots & C-1 \\ p_0 & p_1 & \cdots & p_{C-1} \end{pmatrix}, \sum_{c=0}^{C-1} p_c = 1$$

is the discrete probability distribution (probability mass function) of the random variable $X_\mathbf{x}$.

With this definition, the user may provide a *color probability function* ρ with

$$\rho : [0,1]^2 \to \mathcal{P},$$

where $\mathcal{P} = \{P : \mathcal{C} \to [0,1] \,|\, \sum_{c=0}^{C-1} p_c = 1\}$ denotes the function space of all probability mass functions. Thus, ρ assigns each point $\mathbf{x} \in [0,1]^2$ an individual discrete distribution $P_\mathbf{x}$.

To derive a tiling from such a color probability function we simply sample ρ on the basis of our regular tiling grid and then realize the corresponding random variable. Hence, the corresponding color distribution function h is given by the mapping

$$h : (x,y) \mapsto X\left(\rho\left(\frac{x}{T_x}, \frac{y}{T_y}\right)\right), \tag{7}$$

where (x,y) are the coordinates of the corner of interest and $T_x \times T_y$ the desired tiling resolution. Again, tile indices may be derived via Equation (1).

This approach is a generalization of all tiling methods as ρ may be designed in ways that simulate either strictly deterministic or pure stochastic tilings. Note that it is nevertheless better to apply our deterministic tiling method from the previous section directly as it is independent of a probability distribution and extends naturally to arbitrarily large tilings. Also note that this direct tiling algorithm requires that corners shared by neighboring tiles obtain the same color despite being evaluated independently. One way to ensure this is to utilize the same long-period hash functions that are used for directly generating stochastic tilings [LD06b] as a (x,y)-dependent random number generator. This is what we did in our implementation.

$k \backslash C$	2	3	4	5	6	7	8	9
1	87.5%	61.7%	43.0%	31.0%	23.3%	18.1%	14.4%	11.7%
2	-	-	76.6%	58.2%	44.8%	35.2%	28.2%	23.1%
3	-	-	-	-	64.4%	51.2%	41.5%	34.1%
4	-	-	-	-	-	-	54.1%	44.7%

Table 2: *Savings both in terms of tile construction time and memory requirement if k disjunct pairs of corner colors can be excluded from a C-color tiling.*

Figure 4 shows various tilings using Equation (7). The corresponding color probability functions ρ are depicted below each tiling in the form of individual distribution layers for each corner color. Some corners share the same distributions such that their occurrence is equiprobable. Note that the probabilities sum up to 1 (white) everywhere.

Reduced Tile Sets The number of possible tiles for C corner colors is C^4, so the size of tile sets grows rapidly as C increases. This is of special concern for tile-based texture synthesis due to the amount of associated image content. For example, a 6-color tile set already contains $6^4 = 1296$ tiles.

But if we know that certain combinations of corner colors can never occur, we can significantly prune the tile set for a given color probability function ρ. For example, in Figure 4(c) ρ is designed in a way that the four corner colors in the upper left will never be adjacent to the two corner colors in the lower right. Likewise, in Figure 4(d), the two corner colors from the center will never share a tile with the two outer corner colors. Hence, all tiles containing such mutually exclusive pairs of corner colors can be safely omitted during generation and storage.

Identifying non-adjacent pairs of corner colors is intricate in the case of a continuous color probability function but simple when it is provided in discrete form where a given probability map may be analyzed very quickly for possible corner color combinations, for example by linearly going through the map and keeping track of probabilities greater than 0 in the 8-neighborhood of each pixel. Analyzing the consequences when excluding k specific pairs of corner colors, however, is non-trivial and leads to an interesting combinatorial problem which we solve in Appendix A. In general, if k disjunct pairs of corner colors will never be adjacent in a tiling based on a provided probability function ρ, only

$$N_k = C^4 - 2k(6C^2 - 12C - 6k + 13)$$

tiles have to be generated and stored while still allowing to generate every possible tiling based on ρ. For a single pair of corner colors this already results in tile set reductions of 61.7%, 43.0%, 31.0% and 23.3% for a number of corner colors of 3, 4, 5, and 6 (cf. Table 2). For the examples in

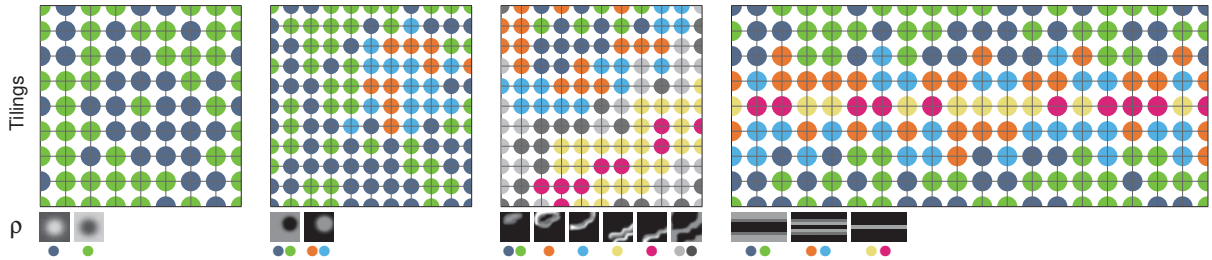


Figure 4: *User-controlled tilings based on the color probability functions depicted below each tiling. Some corner colors share the same probability distributions such that their occurrence is equiprobable.*

Figures 4(c) and 4(d) memory requirements even reduce by 82.0% and 61.7% since disjunct pairs of corner colors are complemented by non-disjunct pairs. Note that for interactive swapping of a probability function ρ, one may dynamically swap the corresponding tile sets (e.g. a combined texture in our application scenario) as well.

Efficiency Generating a controllable tiling via (7) involves the realization of a discrete random variable at each of a tile's four corners. As tilings may be arbitrary large, an efficient realization of such variables is critical for the fast generation of tilings using our approach. However, we can perform the necessary computations elegantly in $\mathcal{O}(1)$ based on Walker's method of "aliases" [Wal77].

This method transforms a given discrete distribution table P into two tables U and V, each the size of P, by precomputing the decision for each outcome such that the realization of the random variable reduces to a single comparison. For our application, this implies that we can determine each corner color in constant time, i.e. independent of the number of corner colors.

For discrete color probability functions, the two tables can be generated for every point in $\mathcal{O}(C)$ during preprocessing. It is even possible to combine them into a single table as each entry in the first table U actually denotes a fractional part in $[0,1]$ and each entry in V an integer in $\{0,\ldots,C-1\}$, our corner colors. Thus, each entry may be range-compressed into a single float, keeping the table size constant.

4. Example-Based Texture Synthesis

We now demonstrate the advantages of tilings based on our two methods in the domain of example-based texture synthesis where tile-based methods are of particular interest due to their performance in connection with tile-based texture mapping. In this context, individual tiles are constructed from a provided source texture and then get arranged by the tiling algorithm to produce the output texture.

Tile Construction Corner tiles are usually constructed by randomly choosing C patches from a single source texture and arranging them for every tile according to its corner color combination [NWT*05, LD06a] (also cf. Figure 6(a)). Hence, there is a direct connection between corner colors and the synthesized result. Resulting patch borders are then preferably covered by another unique center patch which is merged with the corner patches via an optimization or graph-cut technique [KSE*03, NWT*05, DZP07].

We extend this principle to multiple source textures where each corner color is assigned an input patch from a fixed corresponding source texture (cf. Figure 6(b)). Since we do not want to favor one of the source textures in the resulting tile set, the source texture for the gray center patch may be obtained by interpreting a tile's corner color combination as another discrete probability distribution. When sampled, this distribution yields the value of the predominant corner color and hence the source texture. In Figure 6(b), e.g., it is twice as probable that the center patch is chosen from the input corresponding to the orange corner than from the other two.

The important observation is that this approach leads to a balanced representation of each source texture in the resulting tile set. For controllable tilings, this means in particular: if a user increases the probability of a desired corner color for a specific tiling region, this leads to a proportional increase of the probability that the associated texture content will dominate this region after synthesis. Thus we maintain the direct connection between corner colors and synthesized content in case of multiple source textures.

Results We implemented both of our methods as fragment shaders in connection with tile-based texture mapping. Figure 5 shows texture synthesis results for our deterministic tiling method in comparison with current stochastic approaches. Here, all input textures (depicted below each result pair) are of resolution 128×128 while the displayed results are 640×640 cutouts (10×10 tilings) from near infinite tilings. All tiles were constructed using our variant from Figure 6(b) in case of multiple input textures.

In the case of stochastic tilings, large clusters of the same corner color translate to local repetition of texture content, and increasing the number of corner colors from 2 to 3 or 4

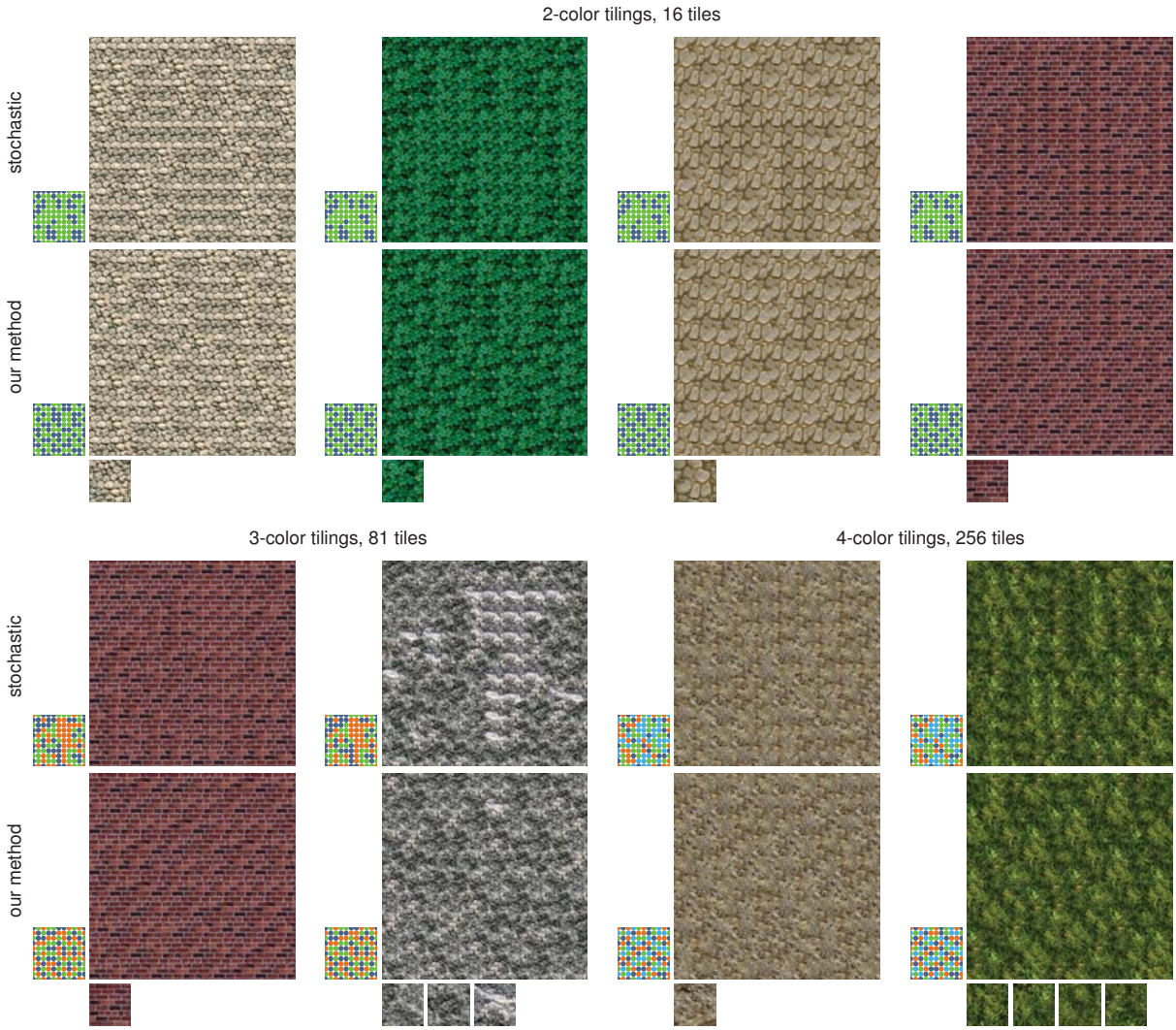


Figure 5: *Texture synthesis results based on stochastic tilings and our deterministic approach. Large clusters of corners with identical color may become harmful when synthesizing textures with salient features as in these examples. Increasing the number of corner colors and/or related input textures does not necessarily improve on this behavior as indicated by the bottom examples. Our method avoids large clusters of the same corner color such that undesirable repetition artifacts become much less detectable. Note that the results for each method are based on the same set of synthesized tiles.*

(and thus the number of tiles from 16 to 81 or 256) does not necessarily improve results as indicated by the bottom examples. In contrast, the tilings produced by our method show a more even distribution of corner colors that is free of color clusters and makes repetition artifacts muss less detectable, even for $C = 2$ colors with only 16 tiles. We want to emphasize that the results for both tiling methods are based on the same set of synthesized tiles, i.e. the improvements stem solely from the better arrangement of tiles using our method.

Figure 7 shows results based on user-controlled tilings which were utilized to synthesize globally-varying textures from source textures related in a non-stationary way. Synthesizing a tile set from such input textures yields a rich set of tiles which can be used to generate large amounts of the same globally-varying texture, akin to stationary textures.

Both of our tiling methods are as fast as existing direct stochastic tiling algorithms and run at several hundreds frames per second on current graphics hardware. In the case of controllable tilings we could also reduce memory requirements by a typical 40% to 70% (depending on the probability map), as described in Section 3.3. In addition, tiles can be interactively rearranged in correspondence to an underlying probability distribution, and due to Walker's alias method these increased degree of freedoms come at negligible costs.

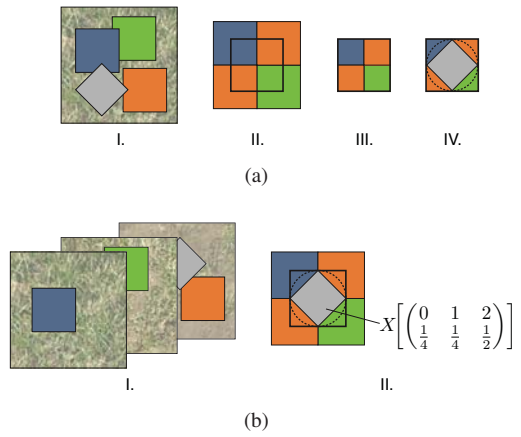


Figure 6: (a) *Common approach to tile construction for example-based texture synthesis: (I.) Input patches are chosen randomly from a single source texture, (II.) arranged on a tile according to its corner color combination, (III.) cut out to fit the tile size, and (IV.) get their seams covered by a unique center patch.* (b) *(I.) Our approach extends this principle to multiple source textures, and (II.) chooses the center patch by interpreting the tile's corner color combination as a discrete probability distribution.*

5. Conclusion

We introduced two new methods for generating tilings based on Wang or corner tiles and demonstrated their advantages in the domain of example-based texture synthesis. Our deterministic tiling method improves upon the results of current stochastic approaches by preventing local repetition of texture content. As a consequence, synthesized textures are of better quality even when using smaller sets of tiles. Our second method generalizes these approaches and allows the user-controllable generation of tilings while maintaining the speed of the others.

We introduced a novel variant for tile construction that allows a balanced merging of multiple input textures such that texture synthesis may be extended to the synthesis of non-stationary textures at runtime. Both of our methods can be efficiently employed in connection with tile-based texture mapping where significant amounts of memory may be saved by excluding tiles corresponding to non-adjacent pairs of corner colors.

In future work, we would like to investigate the applicability of both methods for the synthesis of other signals such as point distributions or volumetric content where tile-based approaches are of particular interest.

Acknowledgements We thank mental images GmbH for supporting this research by granting early access to their research results. Additional thanks go to Holger Dammertz for fruitful early discussions, and Daniel Heck and Thorsten Rieß for their help with the combinatorics.

References

[Ash01] ASHIKHMIN M.: Synthesizing natural textures. In *Proceedings of the 2001 Symposium on Interactive 3D graphics* (2001), ACM, pp. 217–226. 2

[Con09] CONDAT L.: Color filter array design using random patterns with blue noise chromatic spectra. *Image and Vision Computing* (2009). 2

[CSHD03] COHEN M. F., SHADE J., HILLER S., DEUSSEN O.: Wang tiles for image and texture generation. In *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* (2003), vol. 22, ACM, pp. 287–294. 2

[DZP07] DONG W., ZHOU N., PAUL J.-C.: Optimized tile-based texture synthesis. In *Graphics Interface* (2007), pp. 249–256. 6

[FL05] FU C.-W., LEUNG M.-K.: Texture tiling on arbitrary topological surfaces using Wang tiles. In *Rendering Techniques* (2005), Eurographics Association, pp. 99–104. 2

[GKP94] GRAHAM R. L., KNUTH D. E., PATASHNIK O.: *Concrete Mathematics*. Addison-Wesley, 1994. 9

[GS86] GRÜNBAUM B., SHEPHARD G. C.: *Tilings and patterns*. W. H. Freeman & Co., 1986. 2

[Hal60] HALTON J. H.: On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals. *Numerische Mathematik 2* (1960), 84–90. 3

[KCODL06] KOPF J., COHEN-OR D., DEUSSEN O., LISCHINSKI D.: Recursive Wang tiles for real-time blue noise. In *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* (2006), vol. 25, ACM, pp. 509–518. 2

[Kel04] KELLER A.: Myths of computer graphics. *Monte Carlo and Quasi-Monte Carlo Methods* (2004), 217–243. 3

[KSE*03] KWATRA V., SCHÖDL A., ESSA I., TURK G., BOBICK A.: Graphcut textures: image and video synthesis using graph cuts. In *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* (2003), ACM. 6

[LD06a] LAGAE A., DUTRÉ P.: An alternative for Wang tiles: Colored edges versus colored corners. *ACM Transactions on Graphics 25*, 4 (2006), 1442–1459. 2, 6

[LD06b] LAGAE A., DUTRÉ P.: Long-period hash functions for procedural texturing. In *Vision, Modeling, and Visualization* (2006), Aka GmbH, pp. 225–228. 4, 5

[Lef08] LEFEBVRE S.: *Filtered Tilemaps*. Shader X6. Charles River Media, 2008. 2

[LEQ*07] LU A., EBERT D. S., QIAO W., KRAUS M., MORA B.: Volume illustration using Wang cubes. *ACM Transactions on Graphics 26*, 2 (2007). 2

[LKF*08] LAGAE A., KAPLAN C. S., FU C.-W., OSTROMOUKHOV V., DEUSSEN O.: Tile-based methods for interactive applications. ACM SIGGRAPH 2008 Classes, 2008. 2

[LN03] LEFEBVRE S., NEYRET F.: Pattern based procedural textures. In *Proceedings of the 2003 Symposium on Interactive 3D Graphics* (2003), ACM, pp. 203–212. 2

[MZD05] MATUSIK W., ZWICKER M., DURAND F.: Texture design using a simplicial complex of morphable textures. *ACM Transactions on Graphics (Proceedings of SIGGRAPH) 24*, 3 (2005), 787–794. 2

[NC99] NEYRET F., CANI M.-P.: Pattern-based texturing revisited. In *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* (1999), pp. 235–242. 2

[Nie92] NIEDERREITER H.: *Random number generation and quasi-Monte Carlo methods*. SIAM, 1992. 3

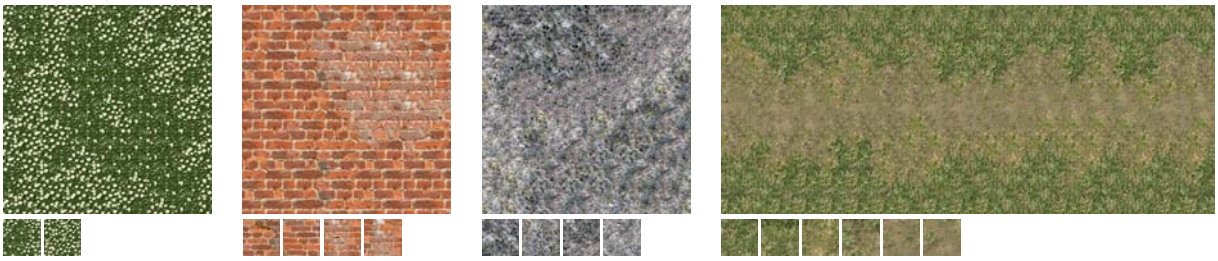


Figure 7: *Synthesis results for globally-varying textures based on the user-controlled tilings from Figure 4. The two examples to the right are based on pruned tile sets (reduced by 82.0% and 61.7% resp.) as described in Section 3.3.*

[NWT*05] NG T.-Y., WEN C., TAN T.-S., ZHANG X., KIM Y. J.: Generating an ω-tile set for texture synthesis. In *Computer Graphics International* (2005), pp. 177–184. 2, 6

[Ost07] OSTROMOUKHOV V.: Sampling with polyominoes. In *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* (2007), vol. 26, ACM, p. 78. 2

[PGMG09] PEYTAVIE A., GALIN E., MERILLOU S., GROSJEAN J.: Arches: a framework for modeling complex terrains. In *Proceedings of Eurographics* (2009), vol. 28, pp. 457–467. 2

[RGAK09] RAAB M., GRÜNSCHLOSS L., ABRAMOV J., KELLER A.: Computer graphics with enumerating QMC sequences in voxels. US Patent Application No. 2009/0141026 A1, June 2009. 3, 4

[RGK10] RAAB M., GRÜNSCHLOSS L., KELLER A.: Enumerating certain quasi-monte carlo sequences in voxels. To appear, 2010. 3

[SAFS99] SHAKED D., ARAD N., FITZHUGH A., SOBEL I.: *Color Diffusion: Error-Diffusion for Color Halftones*. Tech. rep., HP Laboratories Israel, 1999. 2

[Sta97] STAM J.: *Aperiodic Texture Mapping*. Tech. rep., ERCIM, 1997. 2

[Wal77] WALKER A. J.: An efficient method for generating discrete random variables with general distributions. *ACM Transactions on Mathematical Software 3*, 3 (1977), 253–256. 6

[Wan61] WANG H.: Proving theorems by pattern recognition II. *Bell Systems Technical Journal 40* (1961), 1–42. 2

[Wei04] WEI L.-Y.: Tile-based texture mapping on graphics hardware. In *Graphics Hardware* (2004), pp. 55–63. 2

[Wei09] WEI L.-Y.: *Multi-Class Poisson Disk Sampling*. Tech. rep., Microsoft Research, 2009. 2

[WLKT09] WEI L.-Y., LEFEBVRE S., KWATRA V., TURK G.: State of the art in example-based texture synthesis. In *Eurographics 2009, State of the Art Report* (2009). 2

Appendix A: Excluding Tiles

In this appendix, we derive the number of affected tiles N_k when excluding tiles with k disjunct pairs of corner colors. A derivation including non-disjunct pairs works similar but is a bit more involved.

As the total number of tiles in a complete corner tile set $\mathcal{T}$ equals C^4, the number of tiles without $i \le C$ specific corner colors equals $M_i = (C-i)^4$. We are now interested in the number of tiles T_s without an s-element set $S \subseteq \mathcal{C}$ of corner colors, $\mathcal{C} = \{0,\dots,C-1\}$, i.e. those tiles where all s colors of S do not appear at the same time. Using Iverson notation [GKP94] we observe for $s = 2$ colors $\{a,b\}$

$$\begin{aligned} T_2 &= \sum_{\mathcal{T}} [\neg(a \wedge b)] = \sum_{\mathcal{T}} [\neg a \vee \neg b] \\ &= \sum_{\mathcal{T}} [\neg a] + [\neg b] - [\neg a \wedge \neg b] \\ &= \sum_{\mathcal{T}} [\neg a] + \sum_{\mathcal{T}} [\neg b] - \sum_{\mathcal{T}} [\neg a] \cdot [\neg b] \\ &= 2M_1 - M_2 = 2(C-1)^4 - (C-2)^4. \end{aligned} \tag{8}$$

It can be shown that for any s-element set of corner colors this observation generalizes to

$$\begin{aligned} T_s &= \sum_{i=1}^{s} \binom{s}{i} (-1)^{i+1} M_i \\ &= C^4 - \underbrace{\sum_{i=0}^{s} \binom{s}{i} (-1)^i (C-i)^4}_{=0 \text{ if } s>4}. \end{aligned} \tag{9}$$

The second summand disappears for $s > 4$ since M_i is a polynomial of degree $n = 4$ [GKP94]. This reflects the fact that there can be no tile with an $(s > 4)$-element set of corner colors when we have tiles with just four corners.

Now let N_k denote the total number of tiles without k disjunct *pairs* of corner colors. Similar to (8) it can be observed that for $k = 2$ disjunct pairs of corner colors $\{a,b\}$ and $\{c,d\}$

$$\begin{aligned} N_2 &= \sum_{\mathcal{T}} [\neg(a \wedge b) \wedge \neg(c \wedge d)] \\ &= \sum_{\mathcal{T}} [\neg(a \wedge b) + \neg(c \wedge d)] - [\neg(a \wedge b \wedge c \wedge d)] \\ &= 2T_2 - T_4, \end{aligned}$$

which, analogous to (8), can be shown to generalize to

$$\begin{aligned} N_k &= \sum_{p=1}^{k} \binom{k}{p} (-1)^{p+1} T_{2p} \\ &\overset{(9)}{=} \binom{k}{1} T_2 - \binom{k}{2} T_4 + \sum_{p=3}^{k} \binom{k}{p} (-1)^{p+1} C^4 \\ &= C^4 - 2k(6C^2 - 12C - 6k + 13), \end{aligned}$$

where $0 \le k \le \lfloor C/2 \rfloor$.