Vision, Modeling, and Visualization (2011)
Peter Eisert, Konrad Polthier, and Joachim Hornegger (Eds.)

Near-Regular Texture Synthesis by Random Sampling and Gap Filling

Diego Lopez Recas[1], Anna Hilsmann[1,2] and Peter Eisert[1,2]

[1]Fraunhofer Heinrich Hertz Institute, Berlin, Germany
[2]Humboldt University of Berlin, Germany

Abstract

This paper addresses the synthesis of near-regular textures, i.e. textures that consist of a regular global structure plus subtle yet very characteristic stochastic irregularities. Such textures are difficult to synthesize due to the complementary characteristics of these structures. In this paper, we propose a method which we call Random Sampling and Gap Filling *(RSGF) to synthesize near-regular textures. The synthesis approach is guided by a lattice of the global structure estimated from a generalized normalized autocorrelation of the sample image. This lattice constrains a random sampling process to maintain the global regular structure yet ensuring the characteristic randomness of the irregular structures. Results presented in this paper show that our method does not only produce convincing results for regular or near-regular textures but also for irregular textures.*

Categories and Subject Descriptors (according to ACM CCS): I.4.7 [Image Processing and Computer Vision]: Feature Measurement—Texture I.3.3 [Computer Graphics]: Picture, Image Generation—

1. Introduction

The objective of texture synthesis is to generate an arbitrarily sized image that reproduces the texture of a relatively small sample image. During the last years, many researchers in computer vision and computer graphics have proposed methods for texture synthesis and achieved impressive results for many kinds of textures. Sample-based methods compose the output only from extracted pieces of the input sample in contrast to methods that infer a statistical model for the input texture. *Near-regular textures*, i.e. textures that contain global regular structures and additional irregular stochastic structures (e.g. due to the yarn structure in cloth etc.) are still difficult to synthesize. This kind of texture is ubiquitous in the real world, such as cloth, bricks, tiles etc. A faithful reproduction of these textures should preserve both the regular pattern of the texture as well as the stochastic nature of the irregular structure. The latter might be subtle yet very important for the natural appearance of the result. Simple tiling would fail to reproduce the stochastic nature of the irregular structures and statistical methods would fail to reproduce the regular structure.

In this paper, we propose a method to synthesize near-regular textures in a constrained *random sampling* approach. In a first analysis step, we treat the texture as regular and analyze the global regular structure of the input sample texture to estimate two translation vectors defining the size and shape of a texture *tile*. In a subsequent synthesis step, this structure is exploited to *guide* or *constrain* a random sampling process so that random samples of the input are introduced into the output preserving the regular structure previously detected. This ensures the stochastic nature of the irregularities in the output yet preserving the regular pattern of the input texture. Although our method was developed for near-regular textures we observed that it produces also very good results for irregular textures.

The remainder of this paper is structured as follows. Section 2 gives an overview of related work. Section 3 describes the analysis step to estimate the regular structure (or lattice) of the texture sample. Section 4 explains the synthesis step of our approach. We present results in section 5 and conclude with a discussion and thoughts about future work in section 6.

2. Related Work

Sample-based texture synthesis techniques [EL99, WL00, EF01, WZ01, YLC02] generally reproduce a synthesized tex-

DOI: 10.2312/PE/VMV/VMV11/089-096

ture by copying pixels or patches from the input image into the output image in an intelligent manner.

Pixel-based methods [EL99, WL00] use the simple strategy of copying a single pixel at a time. However, resulting textures sometimes show visual artifacts, reported as *garbage accumulation*. Furthermore, sampling single pixel values from the input sample is time-consuming [LLX*01]. Patch-based methods [LLX*01, EF01, KSE*03] copy patches of pixels at a time to show real-time performances. Efros and Freeman [EF01] proposed an image quilting technique (IQ) where patches of a fixed size from the input image are stitched together along optimal seams. This work was extended by Kwatra et al. [KSE*03] who used a graph-cut approach to determine the optimal patch from the input. In general, patch-based methods avoid visual artifacts such as *garbage accumulation* as they copy spatially coherent patches from the input sample but often show other artifacts such as *texture discontinuity* and *repetition*. Both artifacts may be due to the lack of randomness in the copying strategy [SNS06]. Cohen et al. [CSHD03] generate Wang Tile sets to produce probably non-periodic tilings of non-repetitive texture. Wang Tiles are composed by assembling (similarly to IQ) the necessary 4-permutations of randomly selected diamond-shaped portions of input texture (one for each Wang Tile edge color). These techniques do not analyze the regular pattern of near-regular textures and are generally unable to ensure its faithful reproduction [LTL05].

The synthesis of near-regular textures is especially difficult as a global regular structure coexists with irregular and often stochastic variations that are subtle yet very characteristic of the texture and necessary for a natural appearance of the synthetic result. Many existing methods rely on the user to model the irregular structures of the texture [LLH04, LHW*04, LHW*06]. Nicoll et al. [NMMK05] use fractional Fourier analysis to separate the global regular from the irregular structures in the frequency domain and generate *Fractional Fourier Texture Masks* to guide the synthesis. However, the method suffers from degeneration problems if the frequencies are extracted inaccurately. Moreover, the frequencies are extracted with an intensity threshold that is dependent on each particular case and needs specific tunning. Liu et al. [LTL05] estimate the underlying regular lattice of the texture using the method in [LCT04]. Following, they proceed in the lattice direction to fill the image with rectangles containing a complete tile. They split the input in disjoint continuous tiles and select the alignment of the division manually. In this paper, we propose a patch-based synthesis method for near-regular textures which follows a similar strategy. We first estimate the global regular structure of the input image and exploit the result to finally *guide* a random sampling process which preserves both the global structure and the stochastic irregular structures of the input. We improve the lattice detection of [LCT04] by using a normalized crosscorrelation on all three RGB channels of the sample. Moreover, in the synthesis step the size of our sampling unit is not related to the tile of the texture. This allows the creation of *new* tiles not present in the input composed of pieces of different tiles if the synthesis blocks are smaller than the tile. This results in a richer output with more random (and thus more natural-looking) irregularities.

3. Regular Lattice Detection

A regular texture is akin to a 2D-periodic discrete signal. Similarly to the period of a 1D-periodic signal, the translation symmetry of a 2D-periodic signal f can be described by two independent *translation vectors* $\mathbf{v}_1, \mathbf{v}_2$ so that:

$$f(\mathbf{x}) = f(\mathbf{x} + a\mathbf{v}_1 + b\mathbf{v}_2) \tag{1}$$

with a, b integers and $\mathbf{x} = [x, y]^T$. To complete the analogy with the 1-dimensional space, we call the smallest parallelogram that has two independent translation vectors as its non-parallel sides a *tile* of a 2D-periodic signal, i.e. a tile is the 2D counterpart of a 1D period. To estimate the translation vectors of a given texture, we make use of the normalized crosscorrelation (NCC). NCC is known to be a good tool for template matching [BH01] especially in non-ideal lighting conditions as it is invariant to image brightness and contrast.

In the following sections, we describe how we calculate the autocorrelation of an RGB input texture sample (sections 3.1 and 3.2) and estimate the translation vectors describing the texture tile and periodicity from its local maxima distribution (section 3.3).

3.1. Generalized Normalized Crosscorrelation

The common formula of the normalized crosscorrelation NCC of a template t with an image f is [BH01]:

$$\gamma(\mathbf{x}') = \frac{\sum_{\mathbf{x}} (f(\mathbf{x}) - \bar{f}_{\mathbf{x}'})(t(\mathbf{x}-\mathbf{x}') - \bar{t})}{\sqrt{\sum_{\mathbf{x}} (f(\mathbf{x}) - \bar{f}_{\mathbf{x}'})^2} \sqrt{\sum_{\mathbf{x}} (t(\mathbf{x}-\mathbf{x}') - \bar{t})^2}} \tag{2}$$

where $\bar{t}$ and $\bar{f}$ denote the mean of the template and the covered image region. Both the mean of the image $\bar{f}_{\mathbf{x}'}$ and the sums are over all pixels $\mathbf{x}$ under the window containing the template positioned at $\mathbf{x}'$.

This categorization in *image* and *template* implies that it is assumed that the template is small compared to the image and thus the correlation is usually invalid or undefined where the template is not entirely contained in the image. Not allowing partial matches (or giving them invalid values) is especially inconvenient if we are to perform an autocorrelation or a crosscorrelation of two images of the same size, where every point but the origin is a partial match. We slightly modify the original definition to also consider partial matchings. The normalized crosscorrelation between an $N_f \times M_f$ image f and an $N_g \times M_g$ image g now yields:

$$\gamma(\mathbf{x}') = \frac{\sum_{\mathbf{x}} (f(\mathbf{x}) - \bar{f}_{\mathbf{x}'})(g(\mathbf{x}-\mathbf{x}') - \bar{g}_{\mathbf{x}'})}{\sqrt{\sum_{\mathbf{x}} (f(\mathbf{x}) - \bar{f}_{\mathbf{x}'})^2} \sqrt{\sum_{\mathbf{x}} (g(\mathbf{x}-\mathbf{x}') - \bar{g}_{\mathbf{x}'})^2}} \tag{3}$$

where the sums and the mean of both images $\bar{f}_{\mathbf{x}'}$, $\bar{g}_{\mathbf{x}'}$ are over all pixels $\mathbf{x}$ in the overlapping region between the images with g positioned at $\mathbf{x}'$. Note that the new formulation is equal to the original NCC for $g = t, N_g < N_f, M_g < M_f$ and $0 \leq x' < N_f - N_g$, $0 \leq y' < M_f - M_g$.

This generalization makes the computation potentially much costlier, but its computation in the frequency domain allows us to keep it acceptably efficient. A reformulation of equation (3) makes clear what operations need to be done:

$$\gamma(\mathbf{x}') = \frac{\mathcal{N}\mathcal{S}_{fg} - \mathcal{S}_f \mathcal{S}_g}{\sqrt{\mathcal{N}\mathcal{S}_{f^2} - \mathcal{S}_f^2}\sqrt{\mathcal{N}\mathcal{S}_{g^2} - \mathcal{S}_g^2}} \tag{4}$$

with

$$\begin{aligned}
\mathcal{S}_{fg}(\mathbf{x}') &= \sum_{\mathbf{x}} f(\mathbf{x}) g(\mathbf{x} - \mathbf{x}') \\
\mathcal{S}_f(\mathbf{x}') &= \sum_{\mathbf{x}} f(\mathbf{x}) \\
\mathcal{S}_g(\mathbf{x}') &= \sum_{\mathbf{x}} g(\mathbf{x} - \mathbf{x}') \\
\mathcal{S}_{f^2}(\mathbf{x}') &= \sum_{\mathbf{x}} f(\mathbf{x})^2 \\
\mathcal{S}_{g^2}(\mathbf{x}') &= \sum_{\mathbf{x}} g(\mathbf{x} - \mathbf{x}')^2 \\
\bar{f}_{\mathbf{x}'} &= \frac{\mathcal{S}_f(\mathbf{x}')}{\mathcal{N}(\mathbf{x}')} \\
\bar{g}_{\mathbf{x}'} &= \frac{\mathcal{S}_g(\mathbf{x}')}{\mathcal{N}(\mathbf{x}')}
\end{aligned} \tag{5}$$

and $\mathcal{N}(\mathbf{x}')$ is the number of pixels in the overlapping region between the images when g is at position $\mathbf{x}'$. Equation (4) is equivalent to equation (3) and shows that the computation can be obtained from a few integrations and the (unnormalized) crosscorrelation between the two images.

The unnormalized crosscorrelation $\mathcal{S}_{fg}$ is the most expensive operation and is equivalent to the convolution $f(\mathbf{x}) * g(-\mathbf{x})$ and can be computed as $\mathcal{F}^{-1}\{\mathcal{F}(f)\mathcal{F}^*(g)\}$, where $\mathcal{F}$, $\mathcal{F}^{-1}$ stand for the Direct and Inverse Fourier Transforms respectively. Precomputed integrations (integral images) make the other computations $\mathcal{S}_f$, $\mathcal{S}_g$, $\mathcal{S}_{f^2}$, $\mathcal{S}_{g^2}$ and $\mathcal{N}$ computable in linear time [Lew95].

3.2. Multi-Channel NCC

The normalized crosscorrelation is only applicable to a pair of single-channel images (e.g. grayscale images) and not directly to RGB images. A simple multiplication of correlation coefficients or a blind mean would not pay attention to the actual amount of information contained in each channel and might considerably distort the combined result. We propose here a weighted sum of the correlation coefficient of each separate channel as a robust way to get a unified measure for RGB images.

The denominator of equation (3) is the product of the standard deviations of both images in the overlapping and can be thought of as a measure of the magnitude of structure information they carry. This measure specially emphasizes sharp edge-like changes, as they are more energetic than smooth transitions. This makes it appropriate for weighting in our combined measure, considering that human perception is specially sensitive to edges when recognizing objects and patterns. Let γ^i be the generalized normalized crosscorrelation of the RGB channels $f^i, g^i, i = R, G, B$ of two multi-channel images f, g. Then, the combined correlation coefficient γ is:

$$\gamma(\mathbf{x}') = \frac{\sum_{i=R,G,B} \gamma^i(\mathbf{x}')\sigma_{f^i}\sigma_{g^i}}{\sum_{i=R,G,B} \sigma_{f^i}\sigma_{g^i}} \tag{6}$$

where

$$\sigma_{f^i} = \sqrt{\sum_{\mathbf{x}} (f^i(\mathbf{x}) - \bar{f}^i{}_{\mathbf{x}'})^2} \tag{7}$$

$$\sigma_{g^i} = \sqrt{\sum_{\mathbf{x}} [g^i(\mathbf{x} - \mathbf{x}') - \bar{g}^i{}_{\mathbf{x}'})^2} \tag{8}$$

3.3. Translation Vectors Estimation

The NCC of a 2D regular texture sample image with itself has absolute maxima at linear combinations of the translation vectors $\mathbf{x}' = a\mathbf{v}_1 + b\mathbf{v}_2$ with a, b integers. Although we want to analyze near-regular texture samples, we expect their normalized autocorrelation to still have high local maxima at multiples of the translation vectors. Detecting which peaks are related to the regular structure of the input texture is not a trivial task. Deviations from regularity present at the texture sample can cause the ideally absolute maxima to decrease and become only local maxima that may be corrupted with other spurious local maxima. Figure 1 shows that simple thresholding may be inconvenient in some cases. No simple thresholding method can get correct smallest translation vectors in both cases without user intervention. Our fully automatic method for the tile estimation is based on the following observations:

i. The autocorrelation has the symmetry $\gamma(\mathbf{x}) = \gamma(-\mathbf{x})$.
ii. If two vectors are translation vectors for the near-regular texture under analysis, the normalized autocorrelation of the input sample has high local maxima not only at those displacements but also at every of their integer multiples.
iii. A translation vector $\mathbf{v}$ is equivalent to its opposite $-\mathbf{v}$ as they have the same multiples.
iv. A pair of translation vectors $\{\mathbf{v}_1, \mathbf{v}_2\}$ is equivalent to $\{\mathbf{v}_2, \mathbf{v}_1\}$, $\{\mathbf{v}_1 + k_2\mathbf{v}_2, \mathbf{v}_1\}$ and to $\{\mathbf{v}_1, \mathbf{v}_2 + k_1\mathbf{v}_1\}$ for any $k_1, k_2 \in \mathbb{Z}$ as they all have the same set of multiples and thus define the same periodicity.

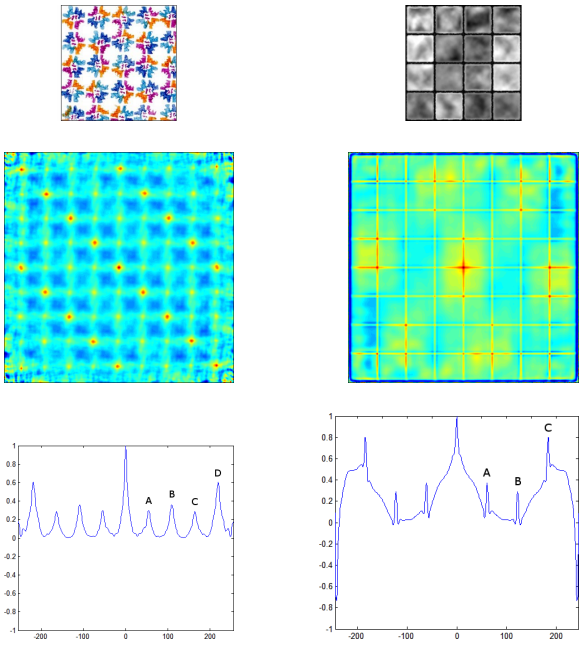

Figure 1: *The rows show the sample image, the normalized autocorrelation matrix and its center row respectively. On the left, only peak D is related to the regular structure of the input texture (peaks A, B and C are spurious local maxima). On the right, all peaks A, B and C are due to the regular structure of the input texture.*

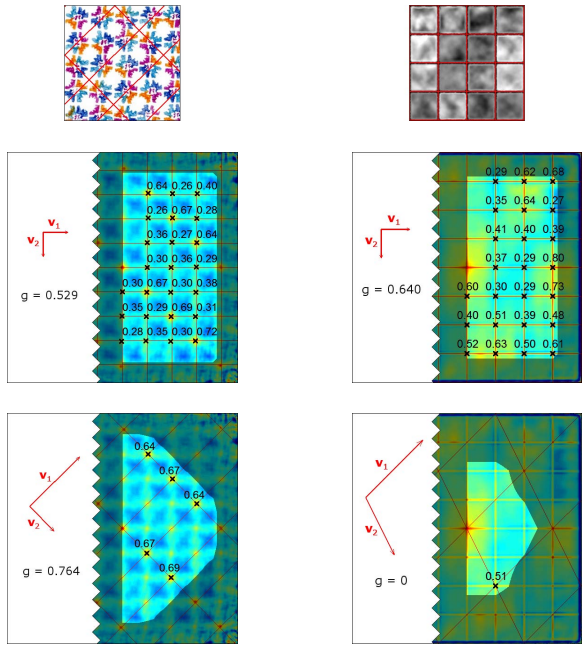

Figure 2: *Texture samples with their detected lattice and examples of* goodness *evaluation. Black crosses mark the values of the autocorrelation that compose the sum for the* goodness *evaluation.*

v. Some spurious maxima are likely to appear at the borders of the autocorrelation matrix, where the values are obtained from the comparison of a smaller pixel region.
vi. Other spurious maxima not at the borders of the autocorrelation are generally lower than the peaks caused by the periodicity of the texture.

Given these observations, we proceed as follows to get the translation vectors:

1. Mark every local maximum of the right half of the autocorrelation matrix (the left half is redundant given the symmetry of γ).
2. Form a candidate set c_1 of displacement vectors from the origin (i.e. the center of the matrix) to each marked local maximum.
3. Sort the vectors in c_1 in ascending length.
4. For each vector $\mathbf{v}_1$ in c_1 form c_2 with the vectors that are sorted after $\mathbf{v}_1$ in c_1 (these have at least the same length as $\mathbf{v}_1$).
5. Reduce every $\mathbf{v}_2$ in c_2 to $\mathbf{v}_2 = \mathbf{v}_2 - \left[\frac{\mathbf{v}_2 \cdot \mathbf{v}_1}{|\mathbf{v}_1|^2}\right] \mathbf{v}_1$ (and take $\mathbf{v}_2 = -\mathbf{v}_2$ if it lies on the left semi-plane) and eliminate duplicates and vectors shorter than $\mathbf{v}_1$. The operator $[.]$ is the round-to-nearest-integer operation and the dot is the scalar dot product.
6. Evaluate the *goodness*:

$$g(\{\mathbf{v}_1, \mathbf{v}_2\}) = \frac{\left(\sum_{a,b \in \mathbb{Z}} \gamma(a\mathbf{v}_1 + b\mathbf{v}_2)\right)^{\alpha}}{n} \tag{9}$$

7. The pair of vectors with the highest *goodness* is the final estimation of the translation vectors.

The sum in equation (9) is over the multiples of $\{\mathbf{v}_1, \mathbf{v}_2\}$ that are on the right half of the autocorrelation excluding the origin and overlappings of less than an 85% of the candidate tile. The *goodness* is zero if either $\mathbf{v}_1$ or $\mathbf{v}_2$ has no valid multiple. The basic idea is to rate a pair of candidate vectors with the mean γ at their integer multiples. α is chosen slightly greater than 1 to give priority to smaller tiles in case of similar mean γ (a tile composed of two tiles is still a valid tile but we seek to find the smallest valid tile). All our results were achieved with $\alpha = 1.12$.

4. Constrained Random Sampling and Gap Filling

In this section, we present a near-regular texture synthesis technique that we call *random sampling and gap filling*. The intention is to synthesize a near-regular texture of an arbitrary size from a previously analyzed (as explained in section 3) sample image. The previous analysis step is exploited to preserve the repetition pattern of the near-regular texture sample. At the same time the output should maintain the

characteristic random irregularities of the input texture for a natural appearance of the result. This is ensured by our *random sampling* method guided by the translation vectors from the previous section.

Our synthesis approach splits the output space in square-shaped blocks of a constant size. Note the difference between a *tile* which defines the regular structure, i.e. the periodicity of the texture and a *patch* used for synthesis. While a tile defines the regular structure of the sample texture, represented by the estimated translation vectors, a patch is here a square-shaped block of an arbitrary size (usually smaller than a tile) in the input and output images. It is the unit we use for filling the output. In a regular texture, the tile defines the periodicity of patches in the input. Each of the blocks in the output image is filled with a texture patch from the input sample in two substeps. We usually call *patch* any piece of texture extracted from the input sample during the synthesis process.

The *random sampling* step ensures that stochastic deviations from regularity are introduced in the output by filling half the output blocks with randomly selected patches from the input. This random selection process is constrained by the estimated regular structure represented by the translation vectors. The *gap filling* step fills the remaining blocks with patches that make transitions between already synthesized blocks smooth, thus reducing visible seams that might have appeared if the procedure was completely random. Both substeps are fully described in sections 4.1 and 4.2 respectively.

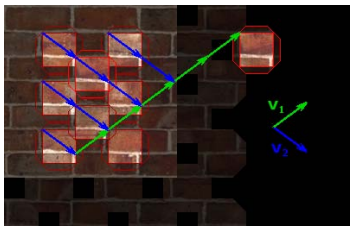

Figure 3: *Example of* random sampling *in progress with the input image superimposed. The distance between the candidate patches (i.e.* similar *to the current block) and the current location is a multiple of the translation vectors* $\mathbf{v}_1, \mathbf{v}_2$.

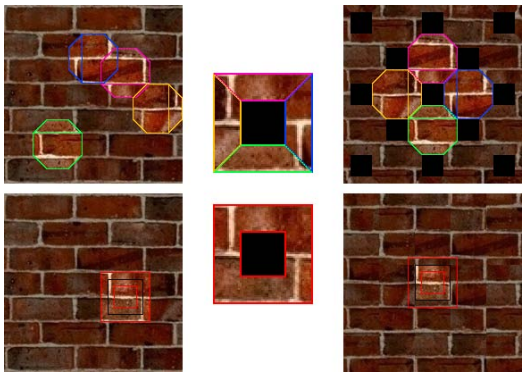

Figure 4: *Example of* gap filling. *Left: location of the neighbors of the current gap (up) and the finally selected patch (down) within the input texture sample. Center: detail of full neighboring border (up) and full border of the selected best-matching patch (down). Right: output detail after* random sampling *(up) and after* gap filling *(down).*

4.1. Random Sampling

The first step in our synthesis process fills every second block of the output (in a chess-like fashion) with a randomly selected patch from the input. To preserve the regular structure of the input texture, the random selection of the patch to fill each block is constrained or guided by the regular structure, described by two independent translation vectors $\mathbf{v}_1, \mathbf{v}_2$ estimated in the analysis step. In near-regular textures, this regular structure appears diffused with stochastic deviations from regularity, but still the estimated translation vectors tell us where to find *similar* texture patches. Texture patches that appear at integer multiples of the translation vectors in the input image are *similar*, i.e. the regular texture part is equal but the slight yet characteristic irregularities of the texture may differentiate them. We say that a texture patch has *similar* counterparts at every integer multiple of the translation vectors. Therefore, for each second block in the output, the *similar* patches at integer multiples of the translation vectors from the current block are preselected from the input. Next, one of the candidate patches is chosen randomly to fill the current block. We proceed sequentially until every second block has been filled with a source patch. See figure 3.

4.2. Gap Filling

If the previous step had synthesized all the blocks of the output with the described random selection of patches, the irregularities of the texture might have created visible discontinuities. This step reduces that risk. The remaining not-yet-synthesized blocks (gaps) are surrounded by already synthesized blocks. Each of them is to be filled with a patch from the input sample that best agrees with its surrounding neighbors.

To evaluate how good a patch agrees with the surrounding neighbors of the current block, a full neighboring border is composed from wedge-shaped borders of each of the neighboring blocks (see figure 4). We prefer this to an alternative blending between rectangular overlapping borders, as the latter would cause blurring and detail loss. The full neighboring border is matched to every possible patch in the input. We propose here two ways of measuring the match between the neighboring border and the border around a patch.

The first alternative is to make use of the generalized NCC with the neighboring border as the template to correlate with the input texture image. In this case, the template has a zero-

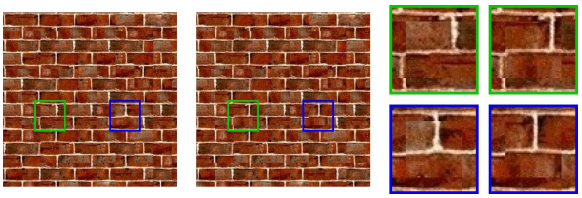

Figure 5: *Differences between NCC border matching and color differences matching. NCC matching (on the left) can sometimes give slightly more blocky results, while color differences matching (on the right) may adjust sharp edges slightly worse.*

padded hole in the middle, but we can compute the NCC by subtracting the corresponding integration of the hole to $\mathcal{S}_f$, $\mathcal{S}_g$, $\mathcal{S}_{f^2}$, $\mathcal{S}_{g^2}$ and $\mathcal{N}$ in equation 4. The patch with the highest correlation is then chosen as the best matching patch.

The second alternative is to evaluate the squared color differences for each position within the input:

$$\chi(\mathbf{x}') = \sum_{\mathbf{x}} \sum_{i=R,G,B} (f_i(\mathbf{x}) - g_i(\mathbf{x}-\mathbf{x}'))^2 \qquad (10)$$

where f is the input sample, g is the composed border and the outer sum is over the overlapping region. Note that equation (10) can also be computed in the frequency domain (see equation (4)). In this case, the best match is that with the smallest χ.

Both alternatives yield similarly good results (see figure 5). In general, the normalized crosscorrelation may adjust sharp edges more accurately in some cases (e.g. lines between bricks in a brick wall) while the color differences also pays attention to the overall brightness and contrast. This may sometimes make the NCC result look slightly more blocky, especially if the input sample was taken with non-uniform illumination. Moreover, the color differences approach has less computation complexity.

Although in most cases it is enough to look for the best border matching piece of texture from the input without further restrictions, in some cases this does not ensure that the global structure of the texture is preserved. To avoid a structure misreproduction, we limit the search space as in the previous section, but with a looser restriction. In section 4.1, we considered only those patches as candidates whose distance from the current position is a multiple of the translation vectors $\mathbf{v}_1, \mathbf{v}_2$. Now, we allow a certain tolerance around those candidates, i.e. patches that are a certain configurable number of pixels apart from the former candidates are also taken into account. A tolerance of ± 3 pixels has been enough in all studied cases. This ensures structure preservation and allows little adaptive adjustments that may be required in some cases (e.g. when the translation vectors are not completely accurate). Once all output blocks have been filled in, linear alpha-blending is applied between adjacent blocks to make transitions even smoother and generate the final result.

5. Results

We tested our fully automatic regular structure detector with several texture samples. The main advantage of our regular structure detector is that it does not need any user assistance to estimate the tile and the regular structure of the texture. Other existing methods generally require user intervention [LLH04, LHW*04, LHW*06]. The tile estimation was correct in most of the tested cases (see figure 7), but the proposed method has difficulties in some specific cases, mainly:

- The texture is a combination of non-aligned (near-)regular textures of different lattices.
- The texture is highly geometrically distorted or the irregularities are similarly energetic to the regular structure.

Synthesis tests showed that a trade-off exists in the election of an appropriate block size. Larger block sizes make the synthesis faster and create less transitions thus producing nicer results. However, larger blocks also reduce the randomness of the irregularities and may cause noticeable repetition, something that may also be caused by a sample containing too few tiles. All 256×256 results in figures 6 and 7 and the additional material were obtained from 128×128 input samples, with 40×40 blocks and $10px$-width borders (gap filling). The process successfully preserves the regular pattern of the input texture as long as the lattice was detected correctly. Compared to direct tiling of a representative tile, the ensured randomness of the output makes the result look far more natural even if the irregularities create slightly visible seams. We also observe that some discontinuities at block junctions are created because the actual translation vectors of the texture have non-integer coordinates (subpixel level) but we estimate them as the pixelwise position of the corresponding peaks in the autocorrelation making the location of *similar* patches not completely accurate.

Our method does neither suffer from *garbage accumulation* that pixel-based methods report [EL99, WL00] nor from any kind of accumulated misalignment that other patch-based techniques [LLX*01, EF01, KSE*03] may cause, as each second block is filled with a "fresh" aligned patch in the constrained *random sampling* substep. Furthermore, the maximum disagreement between neighboring blocks is limited regardless of the output size in our technique.

We have also tested our synthesis approach with some far from regular and some completely stochastic textures skipping the analysis step and taking $\mathbf{v}_1 = (1,0)^T$, $\mathbf{v}_2 = (0,1)^T$, thus unconstraining the selection of patches. The results are as good or better than with existing patch-based methods in most cases, but the sequential procedure may be more appropriate for some non-completely stochastic kind of textures. Results are shown in figure 6. We also refer the reviewers to the additional material for further results and a comparison between our results and IQ [EF01].

Figure 6: *Results achieved with our method for irregular textures.*

6. Conclusions and Future Work

In this paper, a new approach for near-regular texture synthesis has been proposed. There are two key observation behind this method. First, a generalized version of the normalized crosscorrelation paves the way to a simpler estimation of the two independent vectors defining the translational symmetry of the texture under analysis than in [LCT04]. Second, once the two translation vectors have been estimated, we know the 2-dimensional periodicity of the texture. Hence, it is not necessary to take a whole tile as the unit of sampling. We use the latter observation to form sets of *similar* texture patches from which we can randomly chose patches to sparsely fill the output image, thus ensuring the randomness of the characteristic irregularities of the texture. Finally, the gaps left by the previous sparse synthesis are filled with suitable best agreeing patches from the input. Examples shown in this paper demonstrate the high quality of the results that our method can achieve.

Future work will concentrate on improvement of the lattice estimation procedure as well as subpixel level estimation of the translation vectors. Subpixel level estimation of the translation vectors could be used to refine the location of the *similar* patches when composing the output. Moreover, simple tiling of a representative tile could be used to perform a prior correction of geometric distortions in the input. Alternatively, adaptive deformations could be applied to each gap in the gap filling step to better agree with its neighbors (e.g. make lines between bricks in a brick wall coincide). Other ways to finally smooth transitions between neighboring blocks, as image quilting [EF01] or graph cuts [KSE*03] instead of linear blending, could as well be explored.

References

[BH01] BRIECHLE K., HANEBECK U. D.: Template Matching Using Fast Normalized Cross Correlation. In *Proc. of SPIE: Optical Pattern Recognition XII* (2001), vol. 4387, pp. 95–102. 2

[CSHD03] COHEN M. F., SHADE J., HILLER S., DEUSSEN O.: Wang Tiles for Image and Texture Generation. In *ACM SIGGRAPH 2003 Papers* (New York, NY, USA, 2003), SIGGRAPH '03, ACM, pp. 287–294. 2

[EF01] EFROS A. A., FREEMAN W. T.: Image quilting for texture synthesis and transfer. In *Proc. SIGGRAPH 2001* (2001), pp. 341–346. 1, 2, 6, 7

[EL99] EFROS A. A., LEUNG T. K.: Texture synthesis by non-parametric sampling. In *Proc. Int. Conf. on Computer Vision (ICCV 1999)* (Corfu, Greece, 1999). 1, 2, 6

[KSE*03] KWATRA V., SCHODL A., ESSA I., TURK G., BOBICK A.: Graphcut textures: Image and video synthesis using graph cuts. *ACM Transactions on Graphics, SIGGRAPH 2003 22*, 3 (July 2003), 277–286. 2, 6, 7

[LCT04] LIU Y., COLLINS R. T., TSIN Y.: A computational model for periodic pattern perception based on frieze and wallpaper groups. *IEEE Trans. Pattern Anal. Mach. Intell. 26* (March 2004), 354–371. 2, 7

[Lew95] LEWIS J. P.: Fast normalized cross-correlation. In *Vision Interface* (1995), Canadian Image Processing and Pattern Recognition Society, pp. 120–123. 3

[LHW*04] LIN W.-C., HAYS J. H., WU C., KWATRA V., LIU Y.: *A Comparison Study of Four Texture Synthesis Algorithms on Regular and Near-regular Textures*. Tech. rep., School of Coumpter Science Carnegie Mellon University, 2004. 2, 6

[LHW*06] LIN W.-C., HAYS J. H., WU C., KWATRA V., LIU Y.: Quantitative evaluation on near regular texture synthesis. In *Proc. Int. Conf. on Computer Vision and Pattern Recognition (CVPR 2006)* (New York, NY, USA, June 2006), vol. 1, pp. 427 – 434. 2, 6

[LLH04] LIU Y., LIN W.-C., HAYS J.: Near-regular texture analysis and manipulation. vol. 23, ACM, pp. 368–376. 2, 6

[LLX*01] LIANG L., LIU C., XU Y., GUO B., SHUM H.: Real-time texture synthesis by patch-based sampling. *ACM Transactions on Graphics 20* (2001), 127–150. 2, 6

[LTL05] LIU Y., TSIN Y., LIN W.-C.: The promise and perils of near-regular texture. *Int. J. Comput. Vision 62* (April 2005), 145–159. 2

[NMMK05] NICOLL A., MESETH J., MÜLLER G., KLEIN R.: Fractional fourier texture masks: Guiding near-regular texture synthesis. *Computer Graphics Forum 24*, 3 (2005), 569–579. 2

[SNS06] SHIN S., NISHITA T., SHIN S. Y.: On pixel-based texture synthesis by non-parametric sampling. *Computers & Graphics 30*, 5 (2006), 767–778. 2

[WL00] WEI L.-Y., LEVOY M.: Fast texture synthesis using tree-structured vector quantization. In *Proc. SIGGRAPH 2000* (2000), pp. 479–488. 2

[WZ01] WILLIAM W. L. L., ZENG B.: Fast texture synthesis by feature matching. In *Proc. Int. Conf on Image Processing (ICIP 2001)* (2001), pp. 614–617. 1

[YLC02] YU Y., LUO J., CHEN C. W.: Multiresolution block sampling-based method for texture synthesis. In *Proc. Int. Conf on Pattern Recognition (ICPR 2002)* (2002), pp. 239–242. 1

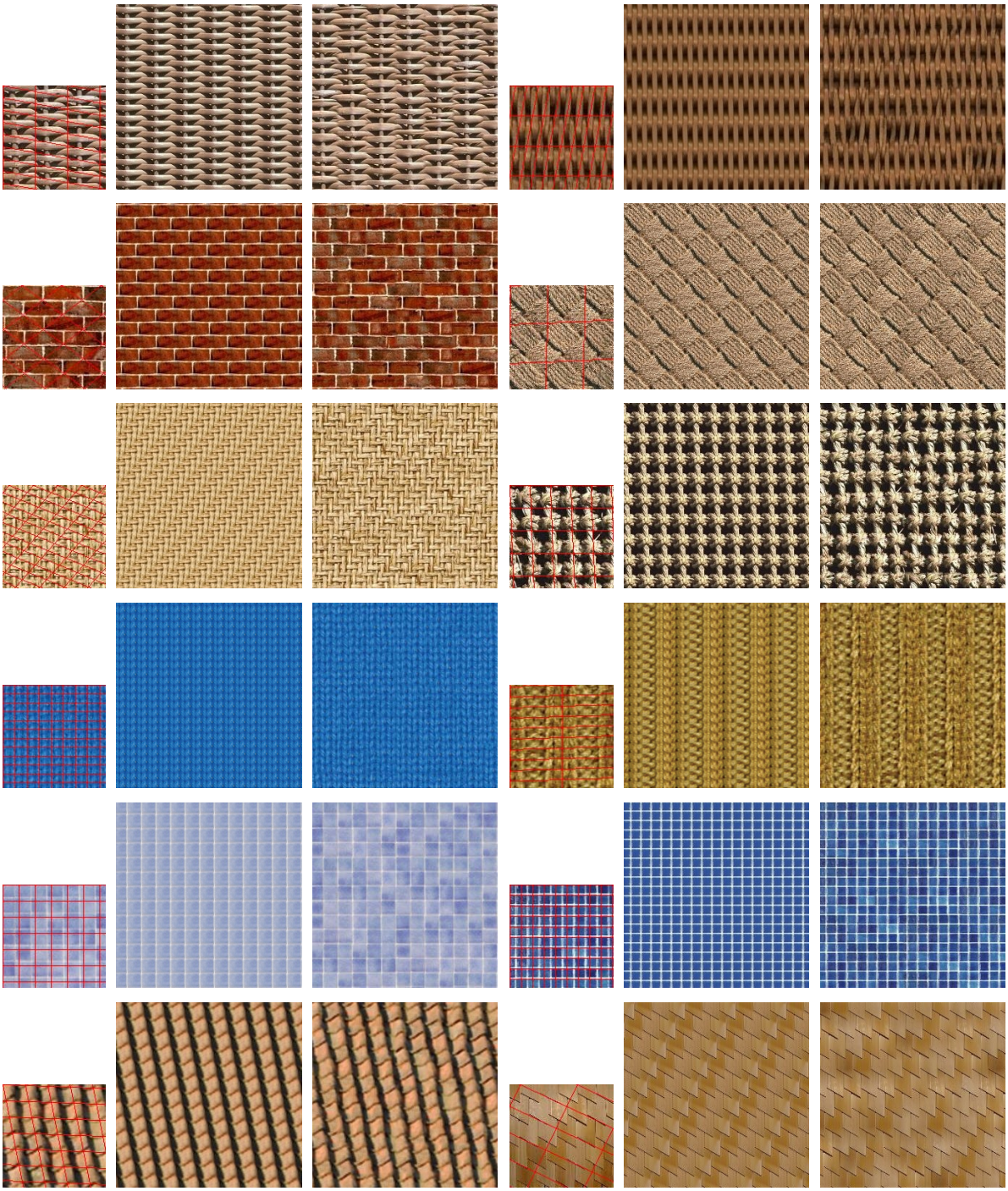

Figure 7: *Results of our texture synthesis approach. The small images depict the sample image with the detected lattice structure. The center images show a result achieved by simple tiling with one texture tile and the right images show the results achieved with our method. Note how the preservation of the small stochastic structures enhances the natural appearance of the synthesized result.*