Analysis and Synthesis of Structural Textures

Laurent Lefebvre Pierre Poulin

Département d'informatique et de recherche opérationnelle
Université de Montréal
{lefebvla, poulin}@iro.umontreal.ca

Abstract

With the advent of image based modeling techniques, it becomes easier to apply textures extracted from reality onto virtual worlds. Many repetitive patterns (structural textures) in human constructions can be parametrized with procedural textures. These textures offer a powerful alternative to traditional color textures, but they require the artist to program the desired effects. We present a system to automatically extract from photographs values for parameters of structural textures, giving the user the possibility to guide the algorithms. Two common classes of procedural textures are studied : rectangular tilings and wood. The results demonstrate that synthesizing textures similar to their real counterpart can be very interesting for computer-augmented reality applications.

Key words: procedural textures, texture mapping, image based modeling, feature extraction, wood texture, rectangular tiling, brick layout, computer-augmented reality.

1 Introduction

Repetitive patterns such as layouts of bricks and tiles are ubiquitous in human constructions. It is therefore not surprising to observe their frequent apparition as textures in computer graphics scenes. These textures are mainly produced by two methods : color textures created by artists or extracted from photographs, and procedural textures generated by algorithms.

1.1 Color Textures

Traditional texture mapping takes an image (often a photograph) of a pattern and applies it to 3D models [10]. We refer to these as *color textures* in this paper. While popular, the technique suffers from several shortcomings. First, the extracted texture presents perspective deformations that must be corrected. The texture must fit exactly the 3D model, otherwise extending or tiling the texture can be noticed even if adjacent tiles are seamless. The color for each texel results also from the real scene illumination, and it is very difficult to factor out the illumination effects from these colors. Finally, the texture remains of a fixed resolution.

With the advent of image based modeling for augmented reality applications, textures and models are both extracted from photographs of a real scene, leading to a closer relationship between the model and its texture. Several methods to exploit this relationship have been introduced [9, 3, 4, 20, 18, 19].

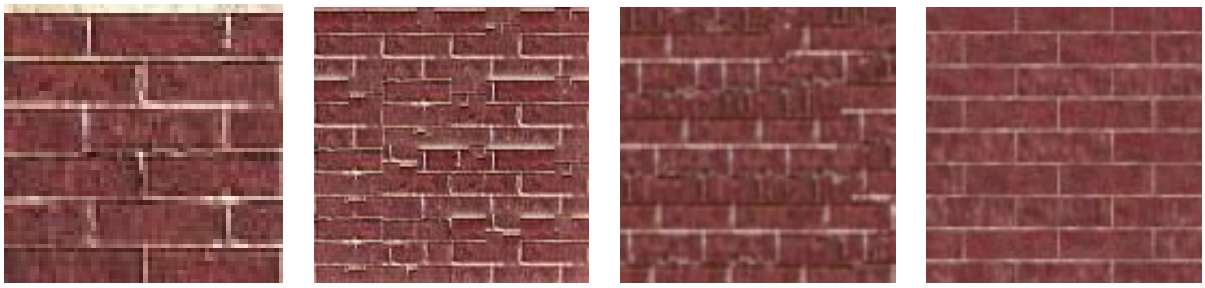
They give good results, but are not perfect. For example, visibility discontinuities on two adjacent faces would produce artifacts (see Figure 12 left). Consider also the situation where objects occlude at least partly the desired object from all angles. It is virtually impossible to remove the occluders from the extracted textures, or to determine what lies behind. Lighting artifacts due to camera flash, or shadows and reflections of the cameraman can occur in photographs. Extracting the texture augmented with its BRDF [24] can reduce the artifacts, unfortunately without removing them all. Moreover, extracting the BRDF is a difficult and lengthy operation that can currently be done only in highly controlled environments.

1.2 Procedural Textures

Procedural textures [16, 17] are generated by algorithms. As repetitive patterns are often easily defined by such algorithms, it falls under the responsibility of the artist to design the proper algorithm to reach his goal. Unfortunately, not all computer artists are necessarily efficient programmers.

Another direction consists in analyzing a texture to extract its parameters in order to synthesize a new texture with similar characteristics. All undesired features can thus be removed during the analysis phase, and a potentially smaller set of parameters is kept to generate the new texture at the desired resolution.

This approach is popular in image processing and texture recognition, and has been extended by multiresolution techniques or by non-parametric sampling [11, 12, 2, 5, 6]. These techniques give very good results on a large class of stochastic or semi-stochastic textures such as rocks, grass, stucco, etc. They have been applied to replace portions of textures in reconstructed 3D scenes [15]. Their generality frees the user from any intervention. However they usually fail in the case of a repetitive pattern such as bricks, where the semantics of the features



a b c d

Figure 1: Synthesizing a wall of bricks : (a) Original image; (b) Synthesis from De Bonet [2]; (c) Synthesis from Efros and Leung [6], implementation of Wei [22]; (d) Synthesis from our algorithm

(mortar lines, brick size and shape, regular tiling) is lost (see Figure 1).

Very little work has been devoted to synthesize structural textures with a priori information on the structure of the texture [23, 25]. The situation is similar in the feature extraction from structural textures, except on specialized applications such as road tracking by vehicles [8, 14].

1.3 Extraction and Synthesis of Structural Textures

We present in this paper our solution to the problem of generating textures adapted to a scene reconstructed from photographs. We are not attempting to generate a procedural texture identical to its corresponding color texture. However our textures should be sufficiently similar to them to adequately replace the color texture in order to remove its artifacts such as illumination from the real scene, occlusions, fixed texture resolution, etc. Because of their frequent use, we concentrate our efforts on algorithms for the analysis and synthesis of rectangular tilings and wood textures.

The process can be summarized by the following steps. A 3D model is created from photographs using our interactive photogrammetry reconstruction software [18], and a color texture is associated with each polygon. For each texture, the user selects the texture type and produces with various image processing tools [21] a binary mask representing the texture with black and white pixels. Given the color texture, the binary mask, and the texture class, the system analyzes the features and fits values to the parameters associated with this texture class. Finally a new procedural texture can generate adapted high resolution textures for the 3D models. The system allows user intervention at all steps, while automatically providing extracted values for all parameters.

Our system can create quality structural textures free from artifacts generally present in the photographs. Because of our knowledge of the structure of the texture, the resulting procedural texture can be further refined in several ways, including the addition of a full reflection model, a bump or displacement map, etc.

In the next two sections, we detail our procedural models for rectangular tilings and wood. Section 4 shows how these models can be augmented to create more sophisticated textures. Finally we conclude and discuss extensions for this work.

2 Rectangular Tiling

The first texture class we present is the rectangular tiling. Because of its ubiquity in most man-made constructions, many textures fall in this category : brick walls, ceramic tiles, skyscraper windows, hanging ceilings, etc.

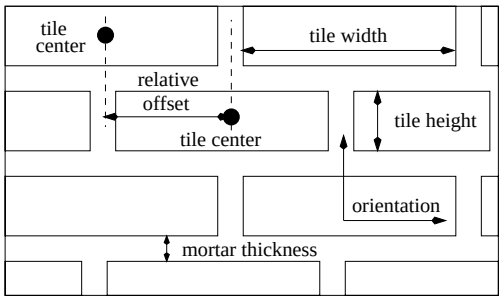


Figure 2: Our rectangular tiling model

We start by segmenting the color texture into a binary mask in order to facilitate parameter extraction. Several tools, including Sobel and pyramidal edge detectors, histogram segmentation thresholding, and morphological operators (opening, closing, eroding, etc.) are provided to the user to segment and clean up the color texture in order to create the binary mask. Most of these tools are interfaced from the C image processing library CVIP [21]. This segmentation step is fundamental because some of our parameter extraction algorithms can be sensitive to

noise. Fortunately, the user can observe the quality of his mask and refine it if necessary. Typically, an experienced user would not require more than a few minutes to produce a reasonable mask.

Our tiling model is simple enough to automatically extract all its parameters from a binary mask (created by the user) representing the texture. Our model consists of 6 scalar parameters (some illustrated in Figure 2) : width and height of the tiles, orientation of the pattern, mortar thickness, number of bricks along a wall, and relative offset between adjacent tiles on successive rows.

The Fourier transform (actually FFT) of the binary mask (see Figure 3) gives important clues in relation to the height of the tiles and their orientation.

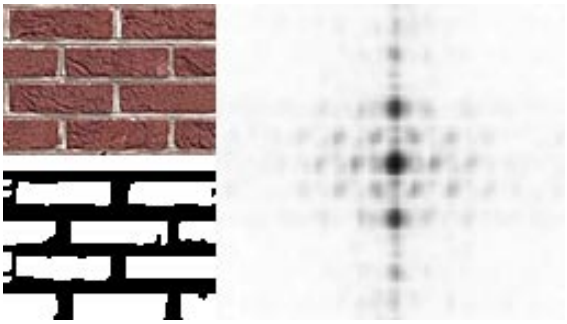


Figure 3: Color texture, segmented texture (binary mask), and scaled negative of the FFT of the mask

2.1 Height and Orientation

In a FFT image of resolution $2R \times 2R$, the center of the brightest group of points $(\hat{x}, \hat{y})$ other than the center of the image (DC frequency), corresponds to the major frequency in the texture. In a tiling pattern, this frequency is associated with the repetition of the base lines of the tile layers. The relation between the spectral and the frequency domains provides the height of the tiles measured as a proportion of the total image height as

$$\text{Height} = \frac{R}{2\sqrt{(\hat{x}-R)^2 + (\hat{y}-R)^2}}. \qquad (1)$$

Because the orientation of the base lines in the tiling pattern is invariant under the Fourier transform, the brightest point $(\hat{x}, \hat{y})$ in the FFT image also provides the orientation angle of the tiling pattern as

$$\text{Angle} = \arctan\left(\frac{\hat{x}-R}{\hat{y}-R}\right) \qquad \text{(in radians).} \qquad (2)$$

The sign of the angle is given by the sign of $(\hat{x} - R)$.

The height and orientation of the tiles are extracted with very good precision because the FFT is a global transformation which resists well to noise. These two values are the first extracted and serve as a basis for the extraction of the remaining parameters.

The pattern formed by the vertical mortar lines is usually more difficult to extract accurately from the FFT image, as illustrated by finding the second brightest cluster of points in Figure 3. The next section describes our spatial algorithms to extract the remaining information. While filters applied vertically or autocorrelation techniques could prove useful, these spatial algorithms have shown to be sufficiently reliable and efficient for our purpose.

2.2 Width and Relative Offset

The remaining parameters are extracted using local and more efficient spatial algorithms, even though they are more prone to the effects of noise. For simplicity in subsequent steps, we rotate the mask backward by the angle previously measured in order to align it with the axes of the image.

To estimate the width of the tiles, we trace a number of horizontal scanlines, counting the number of continuous white pixels (tiles) separated by black pixels (mortar). Scanlines within the mortar horizontal regions are discarded as well as tiles not bounded by mortar. Every tile width is sorted and the median is taken as the width of the tiles.

In presence of patterns formed with tiles of a small set of different lengths, this approach can find the length of each type of tile by separating them into categories. Although we did not investigate further this direction, one should be able to apply results in pattern detection [13] to determine more sophisticated tiling patterns.

The mortar thickness is similarly computed, but instead with vertical scanlines. This orientation is less affected by noise due to the segmentation. If horizontal and vertical thicknesses are different, this approach can easily be extended to satisfy a new model.

The relative offset between successive rows of tiles is also extracted with horizontal scanlines. A tile is selected randomly and its width is estimated as above with horizontal scanlines. An adjacent tile on a row above or under is found by tracing a vertical scanline crossing only one mortar region. The width of this second tile is also estimated like for the first tile. The relative offset is computed from the center of both tiles. The process is repeated for several tiles, discarding outliers, and averaging the median offsets over several selected tiles. If the relative offset divided by the tile width is close to zero, we assume we are in presence of an aligned tiling pattern often occurring in ceramic tiles.

2.3 The Number of Tiles

The number of tiles on the 3D polygon is not always automatically extracted. For instance, we often use a close-up view of the texture which includes more details than the extracted texture. In this case, the number of tiles in the close-up view is different from the actual number of tiles on the wall. Also, in many photographs taken to reconstruct a scene, the mortar regions are often too small to automatically and reliably determine the number of tiles.

The user therefore might want to specify how many tiles should appear in a row on the 3D polygon. Once this value is detected or entered, the system can lay tiles with the right width and the proper aspect ratio over the 3D polygon.

2.4 Filling the Interior

Interiors of tiles and mortar regions have often a more stochastic nature, and are therefore more suitable for automatic analysis and synthesis algorithms [11, 2, 6]. We use the C implementation by El-Maraghi [7] of the Heeger and Bergen algorithm [11].

The user selects one or more regions to represent tiles and one or more regions to represent mortar. He should avoid unsatisfactory regions with artifacts such as occlusions, highlights, shadows, poor resolution, aliasing, etc. The Heeger and Bergen algorithm then processes automatically the selected regions to generate similar high quality textures at the desired resolution.

Sometimes, the distribution of tiles presents itself a special color pattern, or tiles in some regions might be more dirty than others. Once all parameters of the tiling model are extracted, it is easy to decide which selected region to generate from, according to the location of the current tile being treated.

2.5 Results of Rectangular Tiling

The results in this section use our method to analyze and synthesize tiling textures. Roughness, specularity, and reflectance are not currently extracted, but represent one of our topics for future work. At this moment, the user must enter values to set the corresponding parameters.

The left part of Figure 4 shows a bump mapped synthesized brick wall generated from the real wall of Figure 1a. All parameters were automatically extracted from the photograph. In the right part of the figure, some noise is added to displace the surface in order to increase the realism.

The same method is applied on a real ceramic wall on the left of Figure 5. The user selected analysis regions outside the specular highlight, thus avoiding mixing these colors in the synthesis. The presence of such highlights introduces difficulties to most analysis and synthesis algorithms [11, 2, 6], having a direct impact on the quality

Figure 4: Synthesized bricks with bump map (left) and synthesized bricks with displacement map (right)

of the resulting synthesized textures. A bump map function and a specular coefficient are applied on the generated tiles, which then behave properly under new illumination (Figure 5 right).

Figure 5: Real and synthesized ceramic walls

3 Wood

Wood textures, although not as frequent as rectangular tilings, represent nevertheless a great deal of textures in our environments. Wood is used for floors, doors, furniture, stairs, window frames, etc. It can be modeled from a simple growth pattern that can be extracted from photographs, and then generated onto 3D models. Another approach consists of creating an anisotropic 3D texture volume using a pair of orthogonal photographs [5] and mapping this volumetric texture onto 3D models.

While more sophisticated and complete wood models exist [1], we chose to keep our model simple enough to automatically extract values for its parameters from a single photograph.

The trunk of a tree oriented along the Z axis is modeled as a series of identical thickness concentric rings with alternating colors between *early* and *late* wood types. The distribution of colors follows a periodic sine function of frequency ω, starting from the central axis (x_c, y_c, z) of the trunk. The color of any 3D point (x, y, z) is computed by thresholding the following equation :

$$\sin\left(\omega\sqrt{(x-x_c)^2+(y-y_c)^2}\right).$$

A wood plank is modeled as a rectangular cut into this trunk. A tilt angle and a rotation angle are associated with

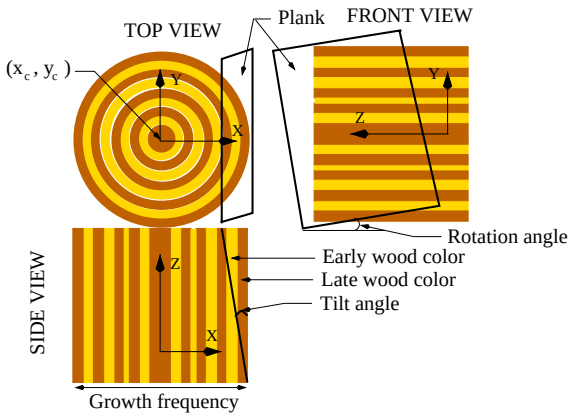


Figure 6: Our simple wood model

the plank, as illustrated in Figure 6.

The 10 scalar parameters of our wood model are sufficient to reproduce a fair range of wood types, notwithstanding knots. They are : trunk center (x_c, y_c), tilt and rotation angles, growth frequency, early/late wood ratio and colors, and turbulence intensity and frequency.

Once again, the user starts by converting the original texture into a binary mask image with thresholding and edge detection operators (see Figure 7).

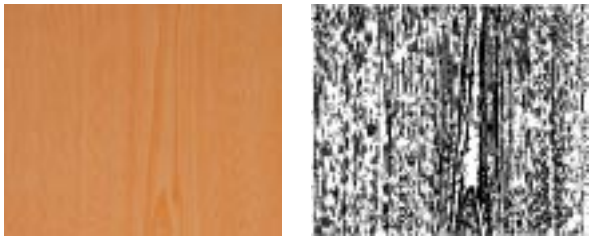


Figure 7: Real wood texture and segmented mask

3.1 Early and Late Wood Colors and Ratio

The binary mask is first traversed to determine the ratio of early wood pixels in the texture. In this paper, late wood is represented as black pixels and early wood as white pixels.

While traversing the corresponding pixels in the original texture, we sort the colors for each wood type. The 25th percentile of the early wood color and the 75th percentile of the late wood color provide satisfying colors for each wood type. This empirical solution avoids to wash out the colors resulting from the binary thresholding, and they are modifiable by the user if necessary.

3.2 Wood Growth Frequency and Rotation Angle

The center of the brightest group of points (other than the DC frequency) in the Fourier transform of the mask corresponds to the major frequency in the texture. It provides the growth frequency. The orientation of this frequency determines the rotation angle of the plank in the trunk. These computations are given by Equations (1) (actually its inverse) and (2) of Section 2 to determine the principal frequency and orientation of the wood.

Once the orientation is determined, the mask and its texture are rotated backward to be aligned with the image axes for the extraction of the tilt angle and other parameters.

3.3 Tilt Angle

The appearance of the wood texture is highly influenced by the tilt angle of the plank in the trunk. The tilt angle controls mainly the shape of ellipses in the texture, and therefore these ellipses can be used to determine this angle.

Our wood model does not handle the presence of knots in the texture. Detecting and synthesizing knots is part of our future work. Unfortunately, large knots can considerably bias most automatic algorithms because they behave similarly to the trunk itself, but at a smaller scale.

Our current solution is approximative, but simple and efficient. It is based on the observation that as the ellipses curve, less and less late wood (or respectively of early wood) appear in successive horizontal scanlines. If the tilt angle is null, the texture shows only straight lines with an identical number of early wood pixels. We therefore proceed by counting the number of late wood pixels in successive horizontal scanlines, and associate an empirically measured factor with this variation.

Obviously, this method is sensitive to noise in the mask and suffers in presence of a large number of knots. However on our test set of wood textures, the results were surprisingly visually satisfying.

We are currently investigating another direction, where the wood grain is traced in the mask. If an elliptic path is identified, we compute its center and eccentricity. By randomly selecting paths in the mask, we expect that statistics on the resulting ellipses should more reliably estimate the tilt angle.

3.4 Turbulence Intensity and Frequency

We observe that turbulence in wood patterns has usually a fairly low frequency. We set a typical value by default, that the user can modify if necessary.

The intensity of this turbulence is estimated in regions outside high curvature ellipses. We compute the variation of wood type between vertical scanlines of pixels. If there is no variations, we are in presence of parallel ver-

tical wood grain, thus the intensity of the turbulence is set to null. Otherwise, it is set according to the degree of variation.

3.5 Trunk Center

The coordinates of the trunk center (x_c, y_c) relative to the plank are also estimated statistically.

Assume one tangent of the plank is oriented along Y. If the plank is tilted, ellipses will appear in the image. The center of these ellipses corresponds to the Y coordinate (y_c) of the center of the trunk. To find this center, we compute the color variance of each column of pixels in the image. The column with the largest variance (*i.e.*, intersecting more ellipses) is associated with the trunk center. If the variance is smaller than a certain threshold, a default value outside the image is assigned to the Y coordinate.

The X coordinate is more difficult to extract accurately. For a trunk located at $x_c \neq 0$, the width of successive layers of wood type varies, as illustrated in Figure 8. Once again, statistics on rows of pixels in the image are used to indicate the location of the X coordinate (x_c) of the trunk center. As a general rule of thumb, except for the central layer which can be thinner, the larger the widths become, the closer to the center we are.

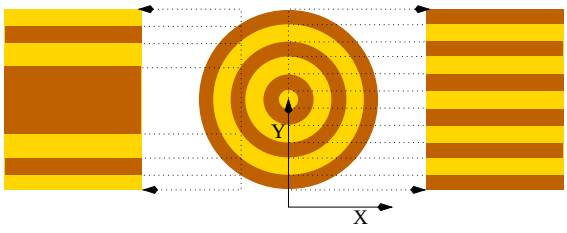


Figure 8: Setting the X coordinate of the trunk center

3.6 Results of Wood Textures

Although our extraction algorithms are only approximative for a number of parameters, and that many of these parameters are closely interrelated, they usually perform very well when visually comparing real wood textures and the synthesized textures. Figure 9 shows a typical result of our method.

An attractive aspect of this model is that the values we extract from a single 2D image of wood, are producing a complete 3D procedural texture, which has an infinite resolution and can be applied on any surface, even curved surfaces.

In order to evaluate the precision of our parameter extraction algorithms, we generated a number of textures with our procedural wood model, and compared the values extracted with the real values. We generated ten different wood patterns for which we modified each parameter, one at a time. After each session, we kept the difference between the estimated value and the real one. The results appear in Table 1 as percentages of the difference with the real value. Note that these percentages are computed on the maximum scale values for most of the parameters. For all our generated textures, we found that the error was proportional to the real value for the tilt angle and the frequency. For these two parameters, the percentages are computed with respect to the real value instead of the maximum scale value.

Figure 9: Real wood and synthesized wood

The parameter with the worst estimated value is the intensity of the turbulence. While some perturbation appears in wood, the exact value of this perturbation is less important. We observed that our extracted values were usually quite visually satisfying. Since the early/late wood ratio, the early wood color, and the late wood color are all extracted using the mask, errors are caused mainly by differences between the masks created by the user for each texture.

Real wood planks are usually cut either almost parallel or almost perpendicular to the wood grain. Considering this fact in our generated textures, we only made small modifications to the tilt angle around 0° and 90° to evaluate the tilt angle extraction algorithm. Our extraction algorithm is less precise when the cut is made at angles not near 0° or 90°. For example, we generated a texture in which the tilt angle was 30° and the extracted value had an error of 36% (this result is not included in Table 1). In such cases, since the shape of the ellipses is defined by the tilt angle, the user can modify the tilt value to achieve

Wood Model Parameter	Maximum Scale Value	Statistics on Differences (%)		
		Average	STD	Max
Tilt	$*$	8.09	4.56	14.29
Orientation	180°	0.35	0.24	0.6
Frequency	$*$	2.47	4.81	12.34
Ratio Early/Late	8	5.5	3.5	10.5
Turbulence Intensity	30	10.3	7.16	20.63
x_c	512	2.71	2.46	6.56
y_c	512	2.27	2.21	6.50
Early Intensity	256	3.18	0.76	4.94
Late Intensity	256	2.79	0.94	3.38

$*$ Proportional to the real value.

Table 1: Extracting values for simulated wood

the desired effet.

For the remaining parameters, our algorithm is usually sufficiently precise with average errors around 2-5% (see Table 1). In the case of an unusual error, the user can correct the problem in a few steps. For example, for the parameter y_c, the user can easily specify the center of the ellipses in the original image.

4 Improving the Procedural Models

Modifying the appearance of a digitized texture requires much work by the artist. Once we have extracted values for our simple models, adding new features specific to the resulting textures is much simpler because we have a structural definition of the texture. It is therefore simple to add a mirror reflection only on the interiors of tiles, to displace the mortar lines below the tiles depth, to raise the roughness of early wood type, etc.

4.1 Augmenting the Realism with Various Maps

Bump or displacement maps can be applied to the structural models according to the nature of each structure. For instance, one can apply a bump map on individual bricks, while forcing a deeper displacement map on the mortar regions. Figures 5 and 11 (right) show bump maps applied to a ceramic wall and a hardwood floor. Figure 10 shows a displacement map on brick walls.

Similarly, different specularity coefficients and roughnesses can be applied on specific elements of a structural model to create highlights on tiles but not on mortar. Mirror reflections on hardwood floors can provide a clean varnished appearance (Figure 13).

4.2 Matching Textures on Adjacent Polygons

Another advantage of a procedural texture is that one can set constraints to satisfy the structural nature of the texture, or parameters that must be shared by several instances of the texture.

Figure 10 shows a brick layout texture ensuring all mortar lines match, as well as setting the relative offset of the bricks to match the width of a brick.

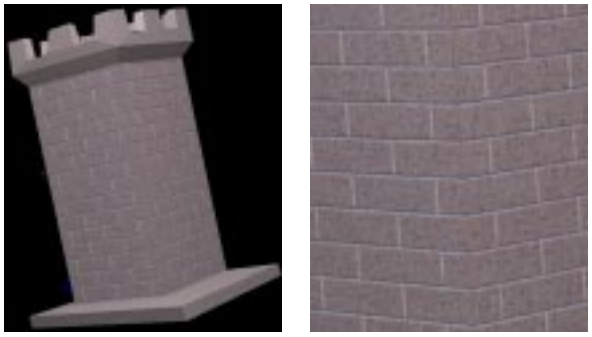


Figure 10: Brick walls on a tower and close-up view

4.3 Procedural Textures and Color Textures

While procedural textures offer a powerful method to generate structural textures, many textures such as paintings or text cannot be modeled by procedures.

When this happens, we give the user the possibility to choose in the texture space which part should be generated procedurally, and which part should be extracted as a traditional color texture. This is achieved by letting the user paint a mask to indicate the regions associated with each texture. The picture frame in Figure 12 and the metal screen of the fireplace in Figure 13 are color textures. Tiles, bricks, and the hardwood floor in these figures are procedural.

4.4 Multiple Structural Textures

On appropriate models, multiple structural textures can be applied on the same 3D surface. Hardwood floors exploit the two texture types described in this paper.

The rectangular tiling method extracts fairly reliably with the FFT the width of the planks, and the wood

modeling method fills in the interior of each plank. Because less information can be extracted from a plank than from a large wood panel, the similarity of the generated wood texture might not be as precise. However we found that the nature of wood leads to satisfying results even with narrow planks. Figure 11 gives an example of this method.

When the photographs do not provide enough information to generate detailed wood textures, one can extract the necessary information from different photographs, and apply the result on the floor layout.

5 Conclusion

Color textures extracted from photographs can suffer from several artifacts resulting from different resolutions, misalignments, undesired occluding objects, missing portions in the texture, shadows, highlights, etc.

We have presented a technique to extract structural textures from photographs, and synthesize similar new textures from the corresponding procedural textures. Unlike random textures which can be generated automatically, the features in structural textures are significant and must be modeled accurately.

We focused our efforts on tiling patterns and wood patterns, both frequent in man-made constructions. For both patterns, we presented techniques to determine automatically values for their parameters, while giving the user the possibility to guide the algorithms or modify any value that was extracted. Structural textures inspired by photographs were generated as examples. Other results can be accessed from the web site associated with this paper from `www.iro.umontreal.ca/labs/infographie/papers`.

Our simple textural models lead to good results where no satisfying color textures could have been extracted. The procedural generation of these textures has many advantages. It can be adapted to the desired resolution, compressed more efficiently than color textures of similar quality, manipulated to generate different results, and augmented by other familiar rendering techniques to increase their realism.

While our parameter extraction techniques were developed specifically for our structural models, many features of these techniques could remain appropriate for parameter extraction in other structural models. For instance, the FFT is frequently used to find orientations and frequencies in image processing. We have tried our algorithms on several different tile patterns and wood styles, and we are confident that they should apply well to other such patterns.

Procedural textures cannot replace all color textures in computer graphics applications. However when they can be used, they offer great advantages that should be exploited. Although advanced synthesis algorithms such as the one proposed by Efros and Leung [6] are more general, they often take hours to synthesize structural textures and the results are not always semantically correct. Our algorithm running on an SGI Onyx R4400 processor takes only a few minutes for both tuning the parameters and synthesizing new textures. All the textures contained in this paper were created with at most three iterations for extracting the parameters (in the case of the hardwood floor of Figure 11). The longest time for texture synthesis was one hour for the fireplace, mainly because many bricks had to be synthesized using the Heeger and Bergen algorithm [11].

6 Future Work

While the current results are encouraging, we feel there is still much information in photographs that could be exploited to improve our structural textures.

The geometry recovered by our image based modeling tool can easily indicate regions in umbra, potential highlights, reflections, etc. These phenomena can significantly alter the colors of the extracted procedural textures. Rather than just neglecting these regions, one could develop techniques to extract values for these phenomena, thus obtaining a more complete structural model from the photographs.

Acknowledgements

We acknowledge financial support from NSERC, and an equipment donation from the DiTER of the Université de Montréal. Martin Blais developed an important portion of our current reconstruction system. We thank Jeremy De Bonet for providing the original and the synthesized textures in Figure 1. We also thank Li-Yi Wei [22] and T. El-Maraghi [7] for making available their respective implementations of Efros [6] and Heeger and Bergen [11] algorithms.

Figure 11: Real (left) and synthesized (center and right) hardwood floor. A small bump map is applied to curve the edges of each linear piece in the right figure.

Projected color textures | Synthesized view | Substituting the floor texture

Figure 12: Reconstruction of structural textures (synthesized tiles) from three photographs of a showcase bathroom. The system lets the user try out other combination of tiling patterns onto the image-based modeled 3D scene.

Real fireplace | Synthesized views

Figure 13: Reconstruction of structural textures (bricks and varnished hardwood floor) synthesized from a single photograph of a fireplace. The metal screen in front of the fireplace is a color texture extracted from the photograph. Small objects on the chimney are properly removed from the photograph without causing any holes.

References

[1] J.W. Buchanan. Simulating wood using a voxel approach. *Eurographics '98*, 17(3):C105–C112, 1998.

[2] J.S. De Bonet. Multiresolution sampling procedure for analysis and synthesis of texture images. In *SIGGRAPH 97 Conference Proceedings*, Annual Conference Series, pages 361–368, August 1997.

[3] P.E. Debevec, C.J. Taylor, and J. Malik. Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach. In *SIGGRAPH 96 Conference Proceedings*, Annual Conference Series, pages 11–20, August 1996.

[4] P.E. Debevec, Y. Yu, and G.D. Borshukov. Efficient view-dependent image-based rendering with projective texture-mapping. In *Ninth Eurographics Workshop on Rendering*, pages 105–116, Vienna, Austria, June 1998.

[5] J.M. Dischler, D. Ghazanfarpour, and R. Freydier. Anisotropic solid texture synthesis using orthogonal 2d views. *Eurographics '98*, 17(3):C87–C95, 1998.

[6] A.A. Efros and T. K. Leung. Texture synthesis by non-parametric sampling. In *IEEE International Conference on Computer Vision (ICCV'99)*, sep 1999.

[7] T. El-Maraghi. An implementation of Heeger and Bergen's texture analysis/synthesis algorithm. *University of Toronto, www.cs.utoronto.ca/∼tem/2522/texture.html*, September 1997.

[8] D. Geman and B. Jedynak. An active testing model for tracking roads in satellite images. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 18(1):1–14, January 1996.

[9] P. Havaldar, M.-S. Lee, and G. Medioni. View synthesis from unregistered 2-D images. In *Graphics Interface '96*, pages 61–69, May 1996.

[10] P.S. Heckbert. Survey of texture mapping. *IEEE Computer Graphics and Applications*, 6(11):56–67, November 1986.

[11] D.J. Heeger and J.R. Bergen. Pyramid-based texture analysis/synthesis. In *SIGGRAPH 95 Conference Proceedings*, Annual Conference Series, pages 229–238, August 1995.

[12] A.N. Hirani and T. Totsuka. Combining frequency and spatial domain information for fast interactive image noise removal. In *SIGGRAPH 96 Conference Proceedings*, Annual Conference Series, pages 269–276, August 1996.

[13] R. Karp, R. Miller, and A. Rosenberg. Rapid identification of repeated patterns in strings, trees, and arrays. In *Proc. of the ACM Symposium on the Theory of Computing*, pages 125–136, 1972.

[14] W. Kasrpzak and H. Niemann. Adaptive road recognition and ego–state tracking in the presence of obstacles. *International Journal of Computer Vision*, 28(1):5–26, 1998.

[15] C. Loscos, M.-C. Frasson, G. Drettakis, B. Walter, X. Granier, and P. Poulin. Interactive virtual relighting and remodeling of real scenes. In *Tenth Eurographics Workshop on Rendering*, pages 329–340, Granada, Spain, June 1999.

[16] D.R. Peachey. Solid texturing of complex surfaces. *Computer Graphics (SIGGRAPH '85 Proceedings)*, 19(3):279–286, July 1985.

[17] K. Perlin. An image synthesizer. In *Computer Graphics (SIGGRAPH '85 Proceedings)*, volume 19, pages 287–296, July 1985.

[18] P. Poulin, M. Ouimet, and M.-C. Frasson. Interactively modeling with photogrammetry. In *Ninth Eurographics Workshop on Rendering*, pages 93–104, Vienna, Austria, June 1998.

[19] C. Rocchini, P. Cignoni, and C. Montani. Multiple textures stitching and blending on 3D objects. In *Tenth Eurographics Workshop on Rendering*, pages 119–130, Granada, Spain, June 1999.

[20] Y. Sato, M.D. Wheeler, and K. Ikeuchi. Object shape and reflectance modeling from observation. In *SIGGRAPH 97 Conference Proceedings*, Annual Conference Series, pages 379–388, August 1997.

[21] S.E. Umbaugh. *Computer Vision and Image Processing*. Prentice-Hall inc., 1998.

[22] L. Wei. An implementation of Alyosha Efros' texture synthesis algorithm. *Stanford University, www.graphics.stanford.edu/∼liyiwei/project/texture/efros*, January 2000.

[23] R. Yokoyama and R.M. Haralick. Texture synthesis using a growth model. *Computer Graphics and Image Processing*, 8(3):369–381, December 1978.

[24] Y. Yu, P. Debevec, J. Malik, and T. Hawkins. Inverse global illumination: recovering reflectance models of real scenes from photographs. In *SIGGRAPH 99 Conference Proceedings*, Annual Conference Series, pages 215–224, 1999.

[25] S.W. Zucker. Toward a model of texture. *Computer Graphics and Image Processing*, 5(2):190–202, June 1976.