M. Haindl and M. Hatka: BTF roller. In *Texture 2005: Proceedings of the 4th International Workshop on Texture Analysis and Synthesis*, pp. 89–94, 2005.

BTF Roller

Michal Haindl
Institute of Information Theory and Automation
Academy of Sciences of the Czech Republic
CZ18208, Prague, Czech Republic
haindl@utia.cas.cz

Martin Hatka
Faculty of Nuclear Sciences
and Physical Engineering
Czech Technical University
CZ11519 Prague, Czech Republic
hadis@email.cz

Abstract

This paper describes a simple method for seamless enlargement of natural bidirectional texture functions (BTF) that realistically represent appearance of given material surfaces. The novel texture synthesis method, which we call the BTF roller, is based on the overlapping tiling and subsequent minimum error boundary cut. One or several optimal double toroidal BTF patches are seamlessly repeated during the synthesis step. While the method allows only moderate texture compression it is extremely fast due to complete separation of the analytical step of the algorithm from the texture synthesis part. The method is universal and easily implementable in a graphical hardware for purpose of real-time rendering of any type of static textures.

1. Introduction

Physically correct virtual models require object surfaces covered with realistic nature-like colour textures to enhance realism in virtual scenes. To make virtual worlds realistic detailed scene models must be built. Satisfactory models require not only complex 3D shapes accorded with the captured scene, but also lifelike colour and texture. This will increase significantly the realism of the synthetic scene generated. Textures provide useful cues to a subject navigating in such a VR environment, and they also aid in the accurate detailed reconstruction of the environment. These textures can be either smooth or rough (also referred as the bidirectional texture function - BTF [5]). The rough textures which have rugged surfaces do not obey the Lambert law and their reflectance is illumination and view angle dependent. Both types of textures occur in virtual scenes models can be either digitized natural textures or textures synthesised from an appropriate mathematical model.

The purpose of a synthetic texture is to reproduce a given digitized texture image so that ideally both natural and synthetic texture will be indiscernible. However modelling of a natural texture is a very challenging and difficult task, due to unlimited variety of possible surfaces, illumination and viewing conditions simultaneously with the strong discriminative functionality of the human visual system. The related texture modelling approaches may be divided primarily into intelligent sampling and model-based-analysis and synthesis, but no ideal method for texture synthesis exists. Each of the existing approaches or texture models has its advantages and simultaneously limitations.

Model-based texture synthesis [2], [20],[1], [12], [17], [29],[10], [15] requires non-standard multi-dimensional (3D for static colour textures or even 7D for static BTFs) models. If such a texture space can be factorized then these data can be modelled using a set of less-dimensional random field models, but in any case such models are non trivial and they suffer with several unsolved problems which have to be circumvented (e.g. parameters estimation, synthesis, stability). Among such possible models the Gaussian Markov random fields are advantageous not only because they do not suffer with some problems of alternative options (see [13], [12], [14], [17] for details) but they are also relatively easy to synthesize and still flexible enough to imitate a large set of natural and artificial textures. Unfortunately real data space can be decorrelated only approximately, hence this approach suffers with some loss of image information. Alternative full nD models allow unrestricted spatial-spectral correlation modelling, but its main drawback is large amount of parameters to be estimated and in the case of Markov models also the necessity to estimate all these parameters simultaneously. Model-based methods are also mostly too difficult to be implemented in modern graphical card processors.

Intelligent sampling approaches [6], [9], [8], [19], [28] rely on sophisticated sampling from real texture measurements. Given a randomly selected starting block of texture in the image, they propagate out from it selecting new texture blocks. For each new block in the image, all neighboring blocks that have already been generated are checked and the example image (or images) is searched for similar textures. The k best such matches are found and then randomly chosen the corresponding new texture patch from among them. The methods [8],[9], [27] all vary in how the blocks are represented, how similarity is determined, and how the search is performed.

Intelligent sampling approaches, such as the presented method, are based on some sort of original small texture sampling and the best of them produce very realistic synthetic textures, usually better than model-based methods. However these methods require to store original texture sample, often produce visible seams, they are mostly computationally demanding, they cannot generate textures unseen by the algorithm, and they cannot reach the large compression ratio of model-based methods.

The rest of the paper is organized as follows. The following section describes a simple sampling approach based on the repetition of a double toroidal tile carved from the original texture measurement. The algorithm is summarized in the section 3. Results are reported in the section 4, followed by conclusions in the last section.

2. Double Toroidal Tile

The BTF data space for single material represents giga bytes of data for recent BTF measuring setups [23]. Even though any sampling based method cannot approach compression ratio of alternative probabilistic BTF methods, it is still important to select double toroidal tiles as small as possible to compress these huge original measurements. We assume mutually well registered data of the size $N \times M$ for fixed viewing angle and changing illumination angle hence the algorithm is repeated for every viewing angle but produces simultaneously identical tiles for all illumination angles.

2.1. Minimal Tile Size

The minimal rectangle to which the tile is inscribed is limited by the size of BTF measurements, the number of toroidal tiles we are looking for (n) and the sample spatial frequency content. From the Fourier transformation of a single monospectral perpendicularly illuminated texture component we detect the dominant low frequency f_r we want to preserve. The multiindex r has two components $r = [r_1, r_2]$, the first component is row and the second one column index, respectively. The rectangle vertical size is chosen to be $n_{row} \in \left\langle \frac{N}{f_{r_1}+0.5}; \frac{N}{f_{r_1}-0.5} \right\rangle$ and if we require $n > 1$ number of multiple tiles we add additional condition $n_{row} \leq \frac{N}{n}$. The horizontal tile size is found similarly.

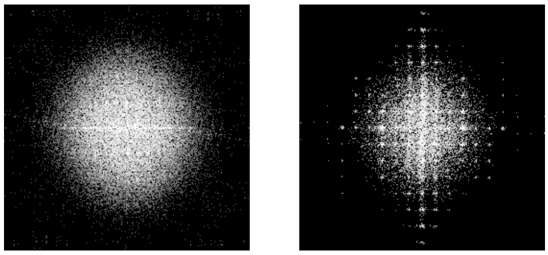


Figure 1. The amplitude Fourier spectra of the corduroy and proposte textures, respectively.

2.2. Overlapping

The double toroidal tile (see Fig.2) is limited by the selected minimal rectangle to be inscribed in from the original texture measurement. The texture tile is assumed to be indexed on the regular two-dimensional toroidal lattice. The optimal lattice searched by the algorithm allows for seamless repetition in both horizontal and vertical directions, respectively.

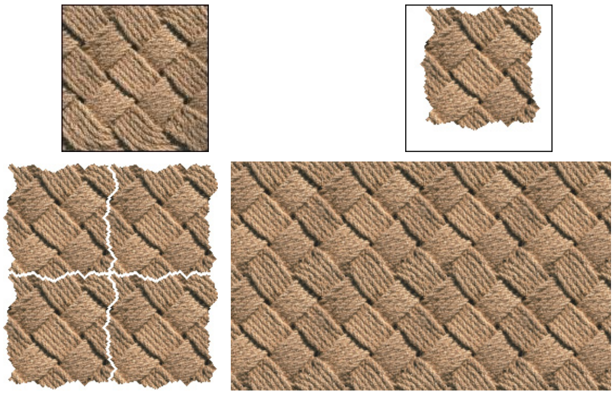


Figure 2. The roller principle - upper row input texture and toroidal tile, bottom row texture generation and the result, respectively.

Let us define the overlap error for a pixel r as follows:

$$\psi_r^h = \left(Y_r - Y_{r+[N-h,0]}\right)^2 \quad \forall r \in I_h \ , \qquad (1)$$

$$\psi_r^v = \left(Y_r - Y_{r+[0,M-v]}\right)^2 \quad \forall r \in I_v \ , \qquad (2)$$

where Y_r denotes a multispectral pixel indexed on the $N \times M$ underlying lattice. The index sets I_h, I_v are defined $I_h = (1, \ldots, h) \times (1, \ldots, M)$, $I_v = (1, \ldots, N) \times (1, \ldots, v)$. The horizontal and vertical overlaps are found from the following relations:

$$h = \frac{N}{\alpha f_{r_1}} \ , \qquad v = \frac{M}{\alpha f_{r_2}} \ , \qquad (3)$$

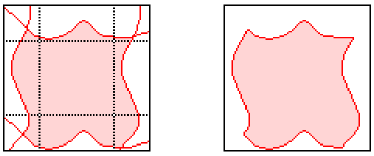


Figure 3. The optimal tile cuts in both directions.

and the constant α for the tile border cut is $\alpha = 2$.

2.3. Optimal Cut

The optimal cuts for both the horizontal and vertical edge is searched using the dynamic programming method. Alternatively we can use some other suboptimal search such as the A^* algorithm if necessary to speed up also the analytical part of the method. However for most applications the fast synthesis is prerequisite while the computation time for separately solved analytical part is of no importance. Both optimal cuts have to minimize the overall path error

$$\begin{aligned} \Psi^h_r &= \psi^h_r + \min\left\{\Psi^h_{r-[1,1]}, \Psi^h_{r-[0,1]}, \Psi^h_{r+[1,-1]}\right\} \ , \\ \Psi^v_r &= \psi^v_r + \min\left\{\Psi^v_{r-[1,1]}, \Psi^v_{r-[1,0]}, \Psi^v_{r+[-1,1]}\right\} \ . \end{aligned}$$

The combination of both optimal vertical and horizontal cuts creates the toroidal tile as is demonstrated on the Fig.3. However such a tile cannot be seamlessly repeated without small corner errors (Fig.4).

2.4. Corners Treatment

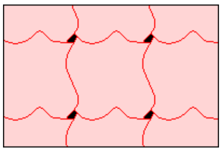


Figure 4. Repetition errors.

Seamless repetition for tile corners requires additional requirement that optimal cuts crossing points create a rect-angle vertices. This can be easily achieved for suboptimal cuts if the undesired starting pixels are initialized into infinite error ($\psi_r^{\cdot} = \infty$). For the optimal vertex we have to check all possible pixels from the index set $I_h \cap I_v$.

2.5. Rectangular Tile

Although we can directly use double toroidal tiles circumscribed by optimally cut borders, for more efficient storage and manipulation these tiles were converted into rectangular shapes. The upper and left tile parts are cut along the line and column

$$r_i^* = \frac{r_i^{*,\max} - r_i^{*,\min}}{2} \qquad i = 1, 2$$

where $r_i^{*,\max}, r_i^{*,\min}$ are corresponding maximal and minimal indices on the optimal cut. These cuttings are then combined with the corresponding opposite parts of the tile.

2.6. Multiple Tiles

Some textures with dominant irregular structures cannot be modelled by simple one tile repetition without clearly visible and visually disturbing regularly repeated effects. These textures are modelled by random changing of several tiles which have identical tile borders but different content. Using analogous approach to the optimal tile border selection, except the constant $\alpha = 4$ in (3), we find optimal cut between tile filling from another part of the input texture and the tile.

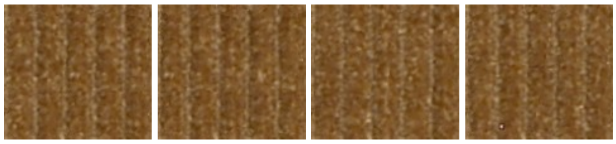


Figure 5. Multiple tiles for the BTF corduroy material.

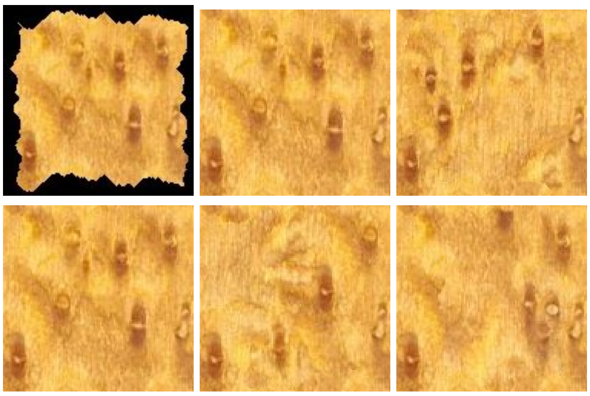


Figure 6. A tile and multiple rectangular tiles for the BTF wood material.

2.7. Synthesis

The synthesis of any required BTF texture size for a single tile case is simple repetition of the created double toroidal

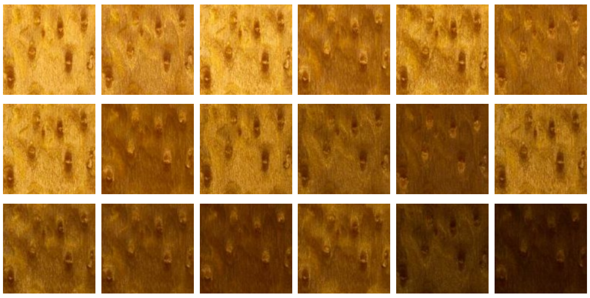


Figure 7. BTF wood rectangular tiles for different illumination angles.

tile in both directions until the required texture is generated. There is no computation involved in this step hence it can be easily implemented in real time or inside the graphical card processing unit (GPU). In the case of several mutually interchangeable tiles we need a uniform random generator to decide which tile will follow. This additional computation is very simple and can be realized inside the GPU as well.

3. The Roller Algorithm

The completely automatic roller algorithm is as follows:

- Analysis
  1. Find the minimal inscribed rectangle.
  2. Find the optimal vertex r^*.
  3. Search for optimal horizontal and vertical cuts starting from r^* with final points in the corresponding rectangle vertices.
  4. Create the double toroidal texture tile.
  5. Modify the tile shape to be rectangular.
  6. Create multiple tiles:
     (a) tile interior specification,
     (b) search for the optimal upper / lower horizontal and left / right vertical cuts,
     (c) interior replacement.
- Synthesis

The analytical steps 2) to 5) are executed only for the first double toroidal tile, while the optional 6) step is performed for each additional required tile. The number of tiles is the only parameter specified by the user. The analytical part is completely separated from the synthesis. The most time consuming part of the analysis is the minimal tile specification together with its position in the input texture sample. In the worst case ($f_{r_1} = f_{r_2} = 2$) it is proportional to $T \propto MN^2h(N-h) + NM^2v(M-h)$. The optimal cuts search time requirement is proportional to $T \propto v^2hN - 2v^2h^2 + Mvh^2$ and interior replacement needs: $T \propto (M - n_{col})(N - n_{row})(n_{col}h - n_{row}v)$. The optimal tile size evaluation time is in most graphical applications not important while the important synthesis step contains either no computations at all or only uniform number generation.

4. Results

We have tested the algorithm not only on BTF textures but also on several hundred colour and grayscale textures from the VisTex database [24], Noctua database, Brodatz textures [3] and mainly from our extensive texture database, which currently contains over 1000 colour textures. Tested textures were either natural such as bark, wood, plants, water, etc., or man-made knitwear, upholstery, brick wall, textiles, food products and many others. Several of these results (brick wall, rattan, leafs, jeans cloth, sugar, sky and text) are demonstrated in the following images. Such unusually extensive testing was possible due to simplicity and efficiency of the algorithm and it allowed us to get insight into the algorithm properties. BTF data we use are from the University of Bonn [23]. We have tested the algorithm on BTF colour measurements such as upholstery, lacquered wood, knitwear or leather textures. Each BTF material sample comprised in the Bonn database is measured in 81 illumination and viewing angles, respectively and has resolution 800×800. Figs.8,9,10 demonstrates synthesised results for three different materials: fabric, wool and leather. Fig. 9 illustrates the lacquered wood smooth BTF synthesis.

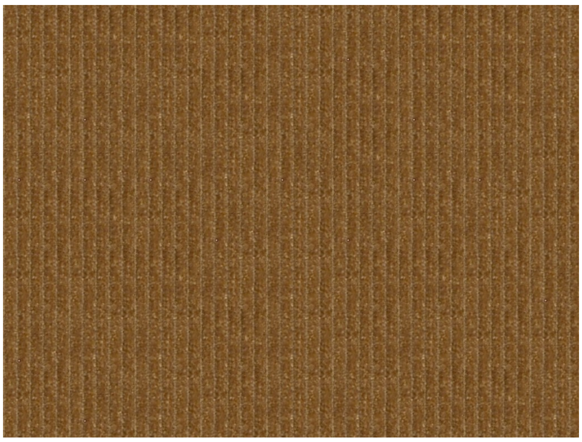


Figure 8. Enlarged BTF corduroy material.

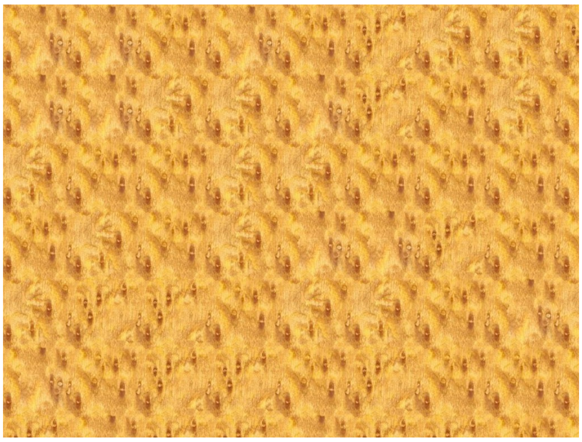


Figure 9. Enlarged BTF wood material.

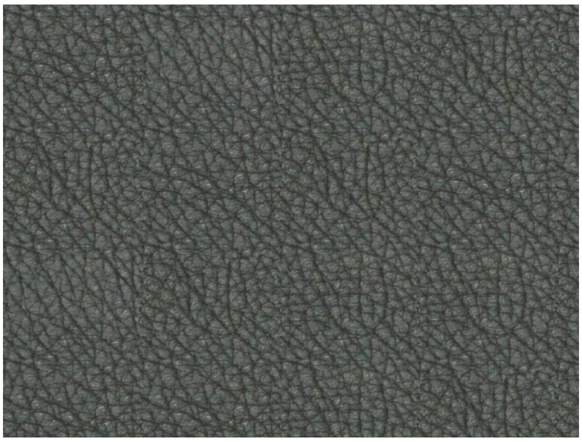
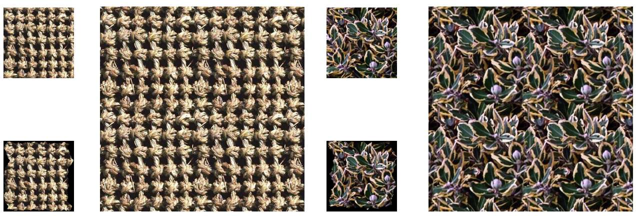


Figure 10. Enlarged BTF foil texture.



Resulting textures are mostly surprisingly good for such a very simple algorithm. For example our results on the text texture (the second from the bottom) are indistinguishable (see [8]) from results on the same texture using much more complicated and slower image quilting algorithm [8]. Obviously there is no optimal texture modelling method and also the presented method fails on some textures as can be seen on the last metal sheet example. This texture can be successfully modeled using the alternative probabilistic model [17]. However on most of our failure examples also

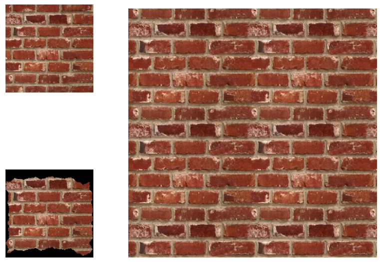
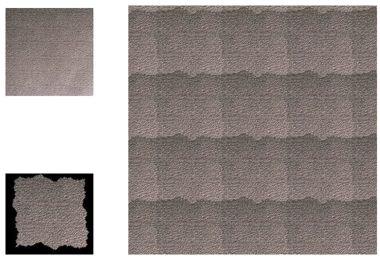


Figure 11. Synthesis examples (upper left - original sample, lower left - tiles, right - resulting texture.

some alternative intelligent sampling methods failed (e.g., [8] failed on the same metal sheet example as well).



5. Conclusions

The test results of our algorithm on available BTF data are visually indiscernible for all our BTF data and also for most of tested colour textures. The test results of our algorithm on our extensive natural texture collection are encouraging. The presented method is extremely fast, fully automatic, very simple and easily implementable even in the graphical processing unit. The method offers only moderate compression ratio ($\approx 1:4$) for transmission or storing texture information while it has negligible computation complexity. Another drawback of the method is that it does not allow a BTF data space restoration or modelling of unseen (unmeasured) BTF space data unlike some probabilistic BTF models.

The roller method can be used for easy and fast seamless synthesis of any required texture size for many natural or man made BTF textures. The method's extension for alternative texture types or some other spatial data such as the reflectance models parametric spaces is straightforward.

5.1. Acknowledgments

This research was supported by the EC projects no. IST-2001-34744 RealReflect, FP6-507752 MUSCLE, the GAAV grants No.A2075302, 1ET400750407 and partially by the MŠMT grant 1M6798555601 DAR.

References

[1] J. Bennett and A. Khotanzad. Multispectral random field models for synthesis and analysis of color images. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 20(3):327–332, March 1998.

[2] J. Besag. Spatial interaction and the statistical analysis of lattice systems. *Journal of the Royal Statistical Society, Series B*, B-36(2):192–236, February 1974.

[3] P. Brodatz. *Textures: A Photographic Album for Artists and Designers*. Dover Publications, 1966.

[4] R. Chellappa. Two-dimensional discrete gaussian markov random field models for image processing. In L. Kanal and A. Rosenfeld, editors, *Progress in Pattern Recognition 2*, pages 79–112, North-Holland, 1985. Elsevier.

[5] K. Dana, B. van Ginneken, S. Nayar, and J. Koenderink. Reflectance and texture of real-world surfaces. *ACM Transactions on Graphics*, 18(1):1–34, January 1999.

[6] J. De Bonet. Multiresolution sampling procedure for analysis and synthesis of textured images. In *ACM SIGGRAPH 97*, pages 361–368. ACM Press, 1997.

[7] J. Dong and M. Chantler. Capture and synthesis of 3d surface texture. In *Texture 2002*, volume 1, pages 41–45. Heriot-Watt University, 2002.

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

[9] A. A. Efros and T. K. Leung. Texture synthesis by non-parametric sampling. In *Proc. Int. Conf. on Computer Vision (2)*, pages 1033–1038, Corfu, Greece, 1999.

[10] J. Grim and M. Haindl. Texture modelling by discrete distribution mixtures. *Computational Statistics Data Analysis*, 41(3-4):603–615, January 2003.

[11] J. Haindl, M. Filip. Fast btf texture modelling. In M. Chandler, editor, *Texture 2003*. IEEE Computer Society, 2003.

[12] M. Haindl. Texture synthesis. *CWI Quarterly*, 4(4):305–331, December 1991.

[13] M. Haindl. Texture synthesis. Technical Report CS-R9139, Centrum voor Wiskunde en Informatica, Amsterdam, The Netherlands, 1991.

[14] M. Haindl. Texture modelling. In B. Sanchez, J. M. Pineda, and F. Ferri, editors, *Proceedings of the World Multiconference on Systemics, Cybernetics and Informatics*, volume VII, pages 634–639, Orlando, USA, July 2000. IIIS.

[15] M. Haindl, J. Grim, P. Somol, P. Pudil, and M. Kudo. A gaussian mixture-based colour texture model. *Proc. 17th ICPR*, vol.3, pages 177–180, 2004. IEEE Press.

[16] M. Haindl and V. Havlíček. Multiresolution colour texture synthesis. In K. Dobrovodský, editor, *Proceedings of the 7th International Workshop on Robotics in Alpe-Adria-Danube Region*, pages 297–302, Bratislava, June 1998. ASCO Art.

[17] M. Haindl and V. Havlíček. A multiresolution causal colour texture model. In F. J. Ferri, J. M. Inesta, A. Amin, and P. Pudil, editors, *Advances in Pattern Recognition, Lecture Notes in Computer Science 1876*, chapter 1, pages 114 –122. Springer-Verlag, Berlin, August 2000.

[18] M. Haindl and V. Havlíček. A multiscale colour texture model. In R. Kasturi, D. Laurendeau, and C. Suen, editors, *Proceedings of the 16th IAPR International Conference on Pattern Recognition*, volume I, pages 255 – 258, Quebec City, 2002. IEEE Press.

[19] D. Heeger and J. Bergen. Pyramid based texture analysis/synthesis. In *ACM SIGGRAPH 95*, pages 229–238. ACM Press, 1995.

[20] R. Kashyap. Analysis and synthesis of image patterns by spatial interaction models. In L. Kanal and A.Rosenfeld, editors, *Progress in Pattern Recognition 1*, North-Holland, 1981. Elsevier.

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

[22] X. Liu, Y. Yu, and H. Shum. Synthesizing bidirectional texture functions. *ACM SIGGRAPH 2001*, 2001.

[23] J. Meseth, G. Müller, and R. Klein. Preserving realism in real-time rendering. In *OpenGL Symposium*, pages 89–96. Eurographics Association, Switzerland, April 2003.

[24] R. Pickard, C. Graszyk, S. Mann, J. Wachman, L. Pickard, and L. Campbell. Vistex database. Technical report, MIT Media Laboratory, Cambridge, 1995.

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

[26] X. Tong, J. Zhang, L. Liu, X. Wang, B. Guo, and H.-Y. Shum. Synthesis of bidirectional texture functions on arbitrary surfaces. *ACM Transactions on Graphics (TOG)*, 21(3):665–672, 2002.

[27] L. Wei and M. Levoy. Texture synthesis over arbitrary manifold surfaces. In *ACM SIGGRAPH 2001*. ACM Press / Addison Wesley Longman, 2001.

[28] Y. Xu, B. Guo, and H. Shum. Chaos mosaic: Fast and memory efficient texture synthesis. Technical Report MSR-TR-2000-32, Redmont, 2000.

[29] S. Zhu, X. Liu, and Y. Wu. Exploring texture ensembles by efficient markov chain monte carlo - toward a "trichromacy" theory of texture. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 22(6):554–569, June 2000.