Journal of Imaging Science and Technology® 50(1): 93–102, 2006.

Multiresolution Texture Synthesis in Wavelet Transform Domain

D. S. Wickramanayake, H. E. Bez and E. A. Edirisinghe
Department of Computer Science, Loughborough University, Loughborough, Leicestershire LE11 3TU, United Kingdom

Abstract. *In this article we propose a multiresolution texture synthesis algorithm in which coefficient blocks of the spatiofrequency components of the input texture are efficiently stitched together to form the corresponding components of the synthesized output texture. We propose two algorithms to this effect. In the first, we use a constant block size throughout the algorithm. In the second, we adaptively split blocks so as to use the largest possible block size in order to preserve the global structure, while maintaining the mismatched error of the overlapped boundaries below a certain error tolerance. Throughout the algorithm designs, special consideration is given to minimizing the computational cost. We show that the adaptation of the multiresolution approach results in a fast, cost effective, flexible texture synthesis algorithm that is capable of being used in conjunction with modern, bandwidth-adaptive, Markov random field imaging applications. A collection of regular and stochastic test textures is used to prove the effectiveness of the proposed algorithm.* © *2006 Society for Imaging Science and Technology.* [DOI: 10.2352/J.ImagingSci.Technol.(2006)50:1(93)]

INTRODUCTION

Digital technology has already taken over much of the entertainment market with new developments in three-dimensional (3D) television, digital films, and modern video games. Television production is increasingly making use of 3D models, in applications including animation and virtual production. There are efforts to replace human actors by animated actors. Researchers are trying hard to make the animated figures to look more realistic. In adding realism to computer graphics applications, mapping natural textures into computer-generated images is vital. Most of the mapping algorithms used today either wrap presynthesized textures or directly synthesize textures, on surfaces. Texture synthesis has a variety of other applications in computer vision, graphics, and image processing, such as occlusion fill-in and image/video compression.

Textures have been traditionally classified as either regular or stochastic. Almost all the real world textures lie in between these two extremes. Natural examples of such textures include fur of animals, patterns of flowers, bark on a tree, etc., whereas fabric patterns and stone patterns on walls are examples of man-made textures. A good texture synthesis algorithm should have the ability to synthesize both these types successfully.

A texture synthesis method starts from a sample image and attempts to produce a texture with a visual appearance similar to that sample, by repeated placement of micropatterns of texture elements on a surface so that when perceived by a human observer, it appears to be generated by the same underlying stochastic process. Unfortunately, creating a robust and general texture synthesis algorithm has been proven difficult.

The problem of synthesizing textures has been studied extensively and numerous approaches have already been proposed. So far the most common approach to texture synthesis has been to develop a statistical model, which emulates the generative process of the texture, which it is intending to mimic. Markov random field (MRF) is a widely used texture model, which assumes the underline stochastic process is both local and stationary. It is accepted that MRF can model wide range of textures accurately. In our algorithm we use the properties of this model in frequency domain. Other successful but more specialized models include reaction-diffusion,[1,2] frequency domain,[3] and fractals.[4–6]

Algorithms using Markov random field models (or in a different mathematical form, Gibbs Sampling)[7–10] produces better perceptual results, which are good approximations to a broad range of textures. However the main drawback of these algorithms is their computational intensiveness that prevents them being used in real time texture synthesizing applications: even small texture patches can take hours or days to generate. Based on this MRF model Paget and Longstaff[11] proposed an algorithm with multiscale synthesis, incorporating local annealing. Results show that this model is able to produce a realistic texture. Although multiscale synthesis with local annealing improves the implementation of the MRF model, the speed of operation is below that required for Markov random field operation. Zhu, Wu, and Mumford[12] presented a statistical theory for texture modeling. Their algorithm combines filtering theory and Markov random field modeling through the maximum entropy principle, and interprets and clarifies many previous concepts and methods for texture analysis and synthesis from a unified point of view.

Another common approach is the physical simulation of the texture. In this method texture generation is done by directly simulating the physical generation process of specific textures such as corrosion, weathering, etc. Certain patterns such as fur, scale, and skin are modeled using reaction diffusion[13–16] and cellular texturing.[5,6] In addition some

Received Sep. 20, 2004; accepted for publication Feb. 11, 2005.
1062-3701/2006/50(1)/93/10/$20.00.

weathering and mineral phenomena can be reproduced by detailed simulations of texture.[17–19] The main disadvantage of these algorithms is that they cannot be applied to general categories.

Another commonly used approach is statistical feature matching. In this method certain features of the input texture are matched in constructing the resulting texture. These algorithms are more efficient than Markov random field algorithms. Heeger and Bergen[20] proposed a method for modeling textures by matching marginal histograms of image pyramids. This algorithm failed to give good results on structured textures. Simoncelli and Portilla[21] were able to improve the synthesis results on structures by using a complicated optimisation procedure. De Bonet[22] synthesized the textures from a wide variety of input images by shuffling the elements in the Laplacian pyramid representation. Although this method is better than Ref. 20, for structured textures it can produce boundary artefacts in some cases.

The inspiration for our work comes from the recent algorithms proposed by Efros and Freeman[23] and Ling and Liu.[24] Both these algorithms use patch based sampling and Ling and Liu[24] addresses the problem of constrained texture synthesis. They use *feathering* (Szeliski and Shum[25]) for edge smoothing while Efros and Freeman are using minimum error boundary cut. Further more Ling and Liu optimize the performance of their algorithm using an approximate nearest neighborhood (ANN) algorithm.[26] Their approaches are simple and work well with most textures. The proposed algorithm is able to perform multiresolution texture synthesis on a wider selection of textures with significant reduction in computational cost. In order to enable the comparison of our algorithm with that of others, we use the widely used benchmark texture synthesis algorithm of Ref. 23 as our benchmark, and perform our experiments on widely used test images.

PIXEL BASED IMAGE QUILTING

In Ref. 23 Efros and Freeman proposed a patch based texture synthesis algorithm in the pixel domain. The basic idea of the algorithm is to create a large *output texture* from a given small *sample texture*. The algorithm can be summarized as follows.

First, a random, $n \times n$ block [see Fig. 1(b)] is selected from the sample texture and is placed at the top left-hand corner of the output texture to be synthesized. Second, all possible (i.e., overlapping) $n \times n$ blocks in the sample texture are matched, using a certain overlap area [see Fig. 1(c)] to the above block. The block with the minimum matching error [see Eq. (1)] is found and blocks which fall within 10% of the minimum matching error are separated. A random block from this group is picked as the blocked to be patched to the block already synthesized in the output texture. This process is continued in raster scan order until the output texture is fully synthesized [see Fig. 1(a)].

Finally, a dynamic programming based edge cutting technique along minimum cost paths[23] is used to smoothen the often visible straight line edge artefacts of the patched block boundaries. For a clearer understanding we

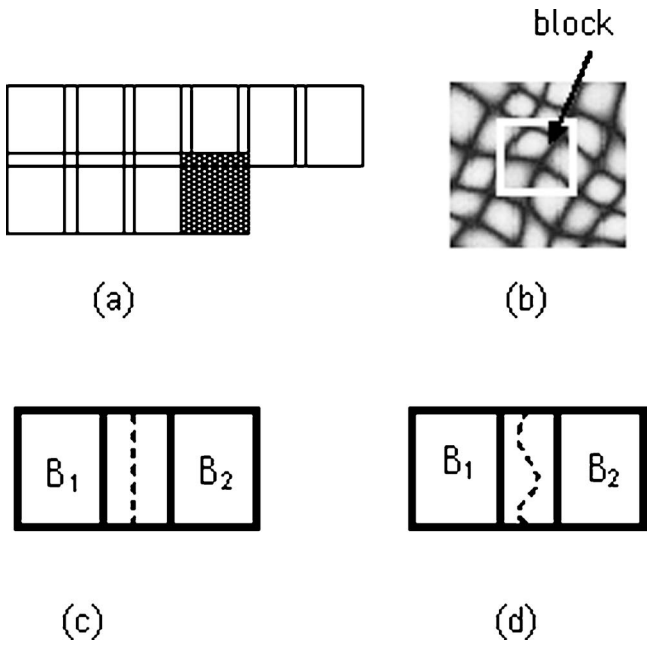

Figure 1. Pixel quilting algorithm: (a) pasting blocks in raster scan order, (b) randomly picking blocks from the sample, and (c) overlapping randomly picked block B_2 with already pasted block B_1. (d) Cutting through minimum error boundary.

provide a simplified pseudocode of the earlier algorithm as follows:

BEGIN
Pick a Random $n \times n$ *Block* from *Sample*,
Place in the top left corner of *OutputTexture*.
LastSynthesizedBlock=$n \times n$ *Block*
WHILE (*OutputTexture* is incomplete)
FOR all $n \times n$ *Blocks* of the *Sample*
Overlap with the *LastSynthesizedBlock* and/or *Block* directly above (depending on position)
Store *OverlapError* and corresponding *BlockNumber*.
ENDFOR
Find Minimum *OverlapError* and **Store** *Blocks* within 10% of it.
Pick a Random $n \times n$ Block amongst them
Patch block
LastSynthesizedBlock=above $n \times n$ *Block*
ENDWHILE
Perform Edge Cutting
END

Note that the quality of match between the overlapping areas of two blocks is calculated in terms of the sum of squared error (i.e., the L^2 norm), as follows:

$$\mathrm{SSE} = \sum_{p \in O} [I_1(p) - I_2(p)]^2, \tag{1}$$

where $I_x(p)$ is the intensity value of the pixel p, O is the set representing all pixels belonging to the overlap area of overlapping blocks, I_1 and I_2.

The typical overlap used is 1/6th the width of the block. The block having the best matching overlap, and all other blocks whose matching error is within 10% of that of the best's is selected as the subset from which subsequently a block is picked up randomly. This block will be used to patch the output image at the location to the right of the previously patched block (in this instance, the top left-hand corner block). This process is continued until the whole output image is formed. In selecting nonboundary type blocks (and the last column of blocks), the overlap considered includes both the overlap with the block in front (as discussed above) and the block above. The output image formed following the above procedure is then subjected to a second stage in which each overlapping set of blocks is combined together along a line of best fit, i.e., by performing a minimum error boundary cut, rather than the more obvious straight edge cut.

Unfortunately, the above algorithm cannot be used for real time texture synthesis. This is due to its complexity resulting from two key approaches adopted. First, the use of exhaustive searching in choosing the best match from the sample texture (note: all possible overlapping blocks are considered) causes computational power to be wasted. Second, the use of a random picking technique (described above) in selecting the final block to be patched with the preceding block, often results in the seam between the two adjacent (i.e., patched) blocks to be visible. Even though a minimum error boundary cutting technique is used to smoothen off these sudden changes in texture, it involves computationally extensive methodologies such as dynamic programming and thus would not be suitable for real time applications.

In order to resolve the problems discussed above, we propose the use of a multiresolution framework, which is capable of faster texture synthesis.

MULTIRESOLUTION IMAGE QUILTING ALGORITHM

Generally, the process of texture synthesis can be mathematically represented by Eq. (2), where F is the texture synthesis function, which takes the input texture sample I_{sample} as the input and synthesizes a texture, I_{output}:

$$I_{\text{output}} = F(I_{\text{sample}}). \tag{2}$$

The proposed multiresolution approach to texture synthesis starts by applying an n level ($n=3$ in our experiments) two-dimensional (2D) discrete wavelet transform (e.g., Haar transform) to the sample image, I_{sample}. The application of single level 2D wavelet transform will result in decomposing I_{sample} into a set of component images (subbands)

$$I_{\text{sample}} = f(C_{sa0}, C_{sh0}, C_{sv0}, C_{sd0}), \tag{3}$$

where $C_{sa0}, C_{sh0}, C_{sv0}, C_{sd0}$ are the image subbands (coefficient matrices) corresponding, respectively, to low-resolution approximation, horizontal detail, vertical detail, and diagonal detail of the sample input texture. Similarly two-level and three-level decompositions are obtained by applying 2D wavelet transform to the low-resolution subband of the previous decomposition level. This could be mathematically presented as follows:

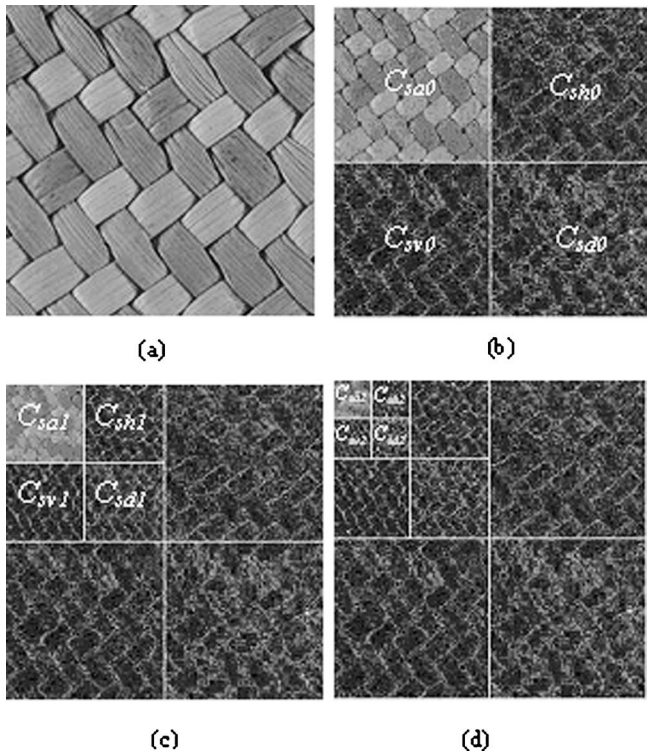

Figure 2. Transforming the sample texture into a multiresolution image representation. (a) Sample texture, (b) single level decomposition, (c) two level decomposition, and (d) three level decomposition.

$$C_{sa0} = f(C_{sa1}, C_{sh1}, C_{sv1}, C_{sd1}), \tag{4}$$

$$C_{sa1} = f(C_{sa2}, C_{sh2}, C_{sv2}, C_{sd2}). \tag{5}$$

For the purpose of clarity, a three-level wavelet decomposition of a sample texture is illustrated in Fig. 2.

If a wavelet decomposition strategy similar to the above is performed on the output texture, I_{output} (which is yet to be generated) the following set of equations can be used for its mathematical representation:

$$I_{\text{output}} = f(C_{oa0}, C_{oh0}, C_{ov0}, C_{od0}), \tag{6}$$

$$C_{oa0} = f(C_{oa1}, C_{oh1}, C_{ov1}, C_{od1}), \tag{7}$$

$$C_{oa1} = f(C_{oa2}, C_{oh2}, C_{ov2}, C_{od2}). \tag{8}$$

Therefore, the three-level wavelet decomposition of the sample and output textures consist of ten subbands each [see Fig. 2(d)].

As the proposed algorithm involves the representation/processing of the sample and synthesized textures in multilevel discrete wavelet transform (DWT) decomposed states described earlier, we generalize the notations used in Eqs. (3)–(8) as follows: in general we represent an image subband as C_{kpl} where $k \in \{s, o\}$, $p \in \{a, h, v, d\}$, and $l \in \{0, 1, 2\}$. In

the above notation k represents decomposed image type ($k=s$ for sample image, $k=o$ for output/synthesized image), l represents the decomposition level (0–1st level, 1–2nd level, 2–3rd level), and p represents the subbands within each decomposition level (a—low resolution, h—horizontal, v—vertical, d—diagonal). Note that f represents the inverse and f^{-1} represents the forward, discrete wavelet transform function.

The basic idea of proposed multiresolution texture synthesis algorithm is to synthesize each subband of the output texture by the corresponding subband of the input, sample texture. This could be mathematically represented as $C_{opl}=F_{pl}(C_{spl})$, where F_{pl} represents the synthesis function for subband under consideration, i.e., (p,l). This texture synthesis procedure can be described in more detail as follows.

Let $B_{spl(x,y)}$ represent a general square block of the decomposed sample image (i.e., $k=s$), located at position (x,y) relative to the subband (p,l)'s origin. Note that in our experiments we have set the size of the above block to be $2^{5-l}\times 2^{5-l}$. In the first phase of the proposed texture synthesis procedure we only consider the $p=a$ subband of the 3rd decomposition level of the output image, i.e., the subband C_{oa2} is synthesized from samples of C_{sa2}. We first pick a block, $B_{sa2(x1,y1)}$, randomly from C_{sa2} and place it in the top left-hand corner of the output coefficient image, C_{oa2}. Simultaneously blocks on all remaining subbands of the sample texture, which corresponds in location to the random block above (i.e., $B_{spl(x1,y1)}$) are transferred to the top left-hand corners of the associated subbands, C_{opl}. [See Fig. 3(a) dotted blocks.] Note that at the end of this step the synthesis of top left-hand corner blocks of all subbands (size $2^{5-l}\times 2^{5-l}$) of the output texture will be completed. Subsequently, all possible (i.e., overlapping) blocks, $B_{sa2(x,y)}$ of similar size (i.e., $2^3\times 2^3$) from the input sample image's C_{sa2} component are picked and matched, for a good overlap with the first block, i.e., the previously synthesized block. The matching criteria used is as follows.

In general, if $B_{opl(x1,y1)}$ and $B_{spl(x2,y2)}$ are two blocks to be matched, we say $B_{spl(x2,y2)}$ is the best match for $B_{opl(x1,y1)}$ if $d(B_{opl(x1,y1)},B_{spl(x2,y2)})$ is minimum for all possible B_{spl} blocks where,

$$d(B_{spl(x2,y2)},B_{opl(x1,y1)}) = \sum_{i\in\partial B}\{[\partial B_{opl(x1,y1)}(i) - \partial B_{spl(x2,y2)}(i)]^2\}, \quad (9)$$

where ∂B_{xpl} is the edge zone of block $B_{xpl(x,y)}$ [see Fig. 3(c)] and i is an element (coefficient) within the edge zone.

In the proposed scheme, the matching described earlier for the best neighbor is done only considering two subbands of the lowest level of decomposition. The two subbands include the low resolution subband and one out of the detail subbands. In our experiments due to the use of a three-level DWT decomposition, we use the combined matching error of the blocks in subbands $p=a$ and $p=d$ of decomposition level $l=2$ to compute the total matching error. This is illustrated in Fig. 3(c). In general, the decision to select the most appropriate detail subband in the above matching process can be based on the total energy contained in the relevant detail component. This is justified as a higher energy detail component of a particular type; say C_{sh2} component, would mean that the horizontal details of the original image would be more significant and visually important than the vertical or diagonal detail.

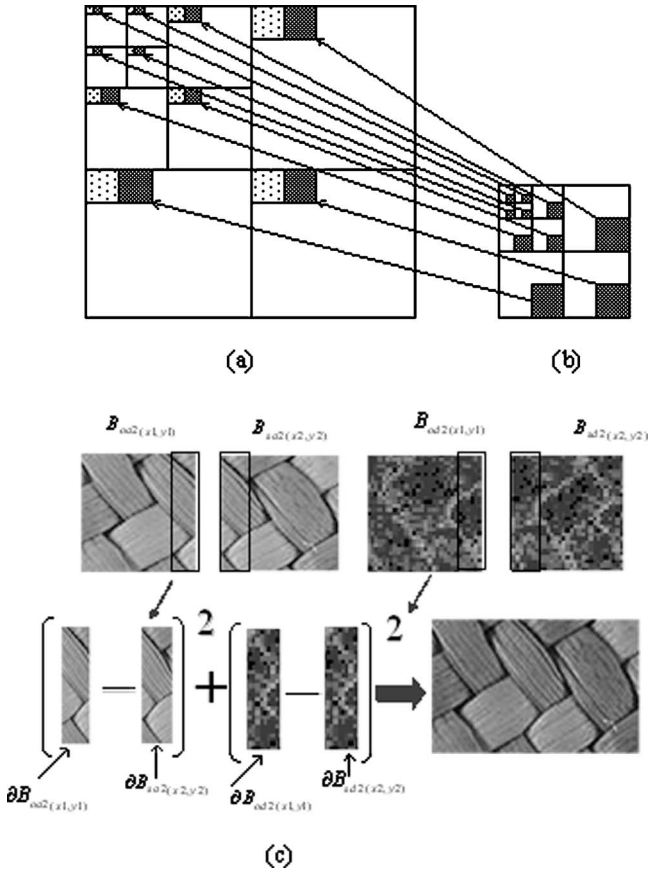

Figure 3. Construction of the output texture. (a) First random block (dotted) and its best match (shaded) (second block) placed on top left hand corner of the output texture together with the corresponding coefficient blocks of detail components in all three levels. (b) Selected best match for the first block with its corresponding coefficient blocks from the input texture. (c) The matching criterion.

In combining the two overlap errors, i.e., the overlap error between the blocks in the low-resolution component and overlap error between the corresponding blocks (i.e., some spatial location) in the selected detail component ["diagonal" in the illustration of Fig. 3(c)]; we use the sum of the squares of the square-errors [Eq. (1)] of the two components, as the matching criteria. For example, if diagonal details are prominent, Eq. (9) can be modified as follows:

$$d(B_{(x2,y2)},B_{(x1,y1)}) = \sum_{i\in\partial B}(\{[\partial B_{oa2(x1,y1)}(i) - \partial B_{sa2(x2,y2)}(i)]^2 + [\partial B_{od2(x1,y1)}(i) - \partial B_{sd2(x2,y2)}(i)]^2\}). \quad (10)$$

Once the best matching block to the first block of the output

image is found and located in the output images C_{oa2} component using the above procedure, the corresponding coefficient blocks of size $2^{5-l}\times 2^{5-l}$ from all detail images of the sample texture image are transferred to the appropriate components of the output texture image. [See Figs. 3(a) and 3(b).] This process is continued until the whole output texture image is constructed in a three-level decomposed format. Note that when constructing blocks other than those belonging to the first row and first column of blocks, the two overlaps, first corresponding to the overlap with the block in front and second corresponding to the overlap with the block above, needs to be considered. Finally, an inverse DWT is performed on the output image decomposition to create the output texture image, I_{output}.

Therefore, the proposed texture synthesis process described in detail above, can be summarized as follows:

(1) Apply an N-level DWT decomposition to the sample image and assume an empty N-level decomposed structure of the texture to be synthesized.
(2) Use a modified block quilting based texture synthesis approach to synthesize the lowest resolution subband of the texture to be synthesized from the lowest resolution subband of the sample texture (note: one detailed subband of the lowest resolution level is used in the above block matching to improve the quality of this synthesis).
(3) For each best matching block position at the lowest resolution level in step-2, transfer the corresponding blocks of corresponding size of all other subbands [see Fig. 3(a)] to the decomposed structure of the empty output texture.
(4) Repeat step 3 until the decomposed structure of the output texture is completely filled.
(5) Apply an N-level inverse DWT to obtain the synthesized texture in pixel domain.

In order to maintain the global structure of the overall texture it is important to select the block size to be sufficiently large. This also accounts for increased efficiency of the algorithm as the choice of blocks available for filling the output texture becomes less, making the process fast (see Block size). Unfortunately the selection of large block sizes makes it increasingly difficult to find overlapping areas providing a good match, which in turn may effect the quality of the resulting texture. On the other hand, the use of small block sizes will increase the synthesis time and may have a negative effect on the global structure of the synthesized texture (see Block size). It has been shown that the selection of the optimum size of the block is dependent on the repeating pattern contained in the texture to be synthesized.[23] Thus, in an effective implementation of the proposed algorithm we need to have a trade off between the image quality and efficiency in selecting the block size. We use the following procedure to achieve optimum trade off. For blocks in the decomposition level l, we start with a block size of $2^{5-l}\times 2^{5-l}$ coefficients. Subsequently, a maximum allowed overlap error (mean-squared error) threshold δ_1, beyond which the visual quality is poor, is defined. If overlap error is greater than δ_1, the block is split into four smaller blocks. (See Fig. 4.) Finally, we repeat the above procedure for these four blocks. However, the process is not repeated iteratively to smaller block sizes due to the need of preserving the global structure.

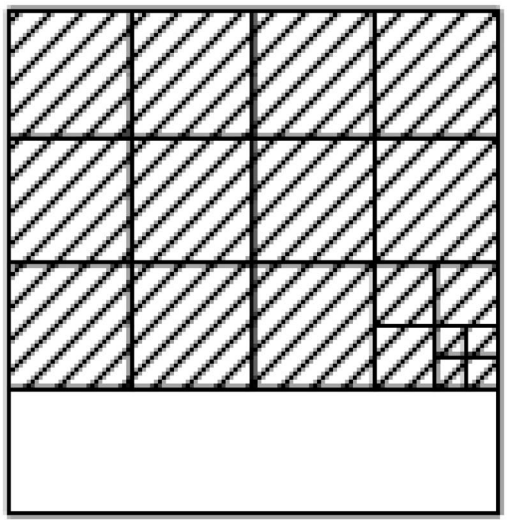

Figure 4. Changing the block size adaptively.

The following section provides detailed experimental results to justify the design and implementation of the proposed algorithm and to critically analyze its performance.

EXPERIMENTAL RESULTS AND ANALYSIS

In order to analyze the performance of the proposed algorithm, experiments were performed on a widely used set of popular, test, texture images (see Fig. 5), consisting of textures of both regular [Figs. 5(b), 5(c), and 5(e)] and stochastic nature [Figs. 5(a), 5(d), and 5(f)]. A typical sample size of 184×184 pixels was used to synthesize textures of size 512×512 pixels. The selection of publicly available texture images for our experiments should enable readers to compare the performance of our algorithm with that of others (see Fig. 10 for a comparison with Ref. 23).

In Design Considerations we further investigate the performance of the proposed algorithm under three varying design considerations and in Comparison with benchmark we compare it directly with that of Ref. 23. A close visual inspection of Fig. 5, illustrates a very good synthesized texture quality with no visibility of straight line edges (i.e., mismatches) between patched blocks. To further support the subjective investigations and to allow a direct comparison with textures synthesized in Ref. 23 we propose the use of the objective quality measure, mean squared difference of slopes (MSDS),[27] which gives a measure of continuity of texture across block boundaries. MSDS is defined as the mean of the square of the difference between the slope across two adjacent blocks and the average between the slopes of each of the two blocks close to the boundary. In

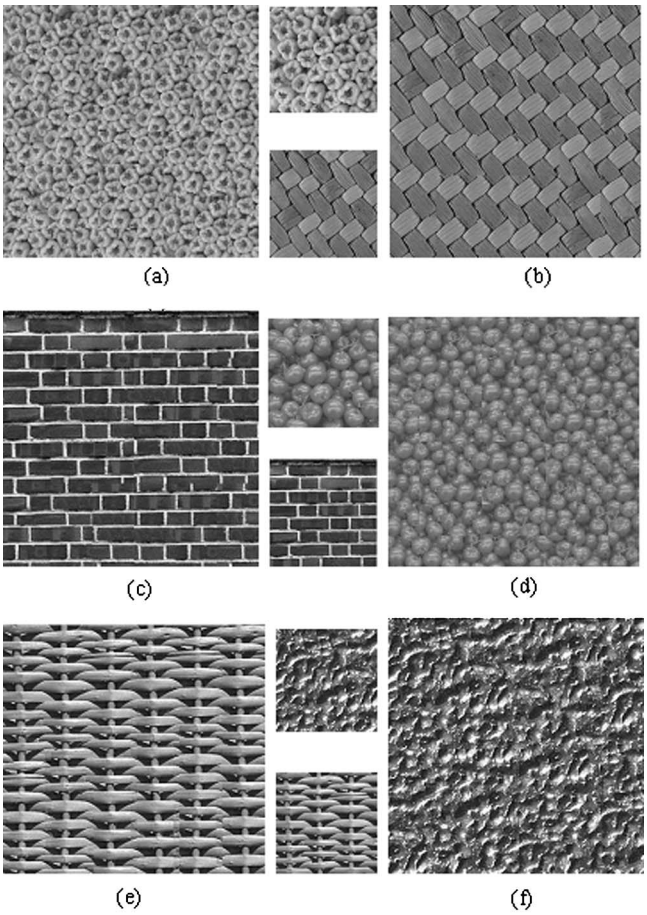

Figure 5. Multiresolution image quilting results. Sample texture (smaller images) and synthesized texture (larger images) using proposed algorithm.

relation to Fig. 6, where $w(i,j)$ and $f(i,j)$, $(0 \leqslant i, j \leqslant N-1)$ are two adjacent (patched) $N \times N$ blocks, the MSDS along an edge (boundary) can be mathematically defined as

$$\mathrm{MSDS}_{\mathrm{edge}} = \sum_{k=0}^{N-1} \left([f(k,0) - w(k,N-1)] - \left\{ \frac{[w(k,N-1) - w(k,N-2)] + [f(k,1) - f(k,0)]}{2} \right\} \right)^2 \Big/ N. \tag{11}$$

The overall MSDS of the synthesized texture is calculated by totaling the squared difference of slopes over all such edges (boundaries) in the synthesized texture and calculating the mean.

The quality metric MSDS has been used in the literature to evaluate the performance of blocking artifact reduction algorithms in DCT based image coding. It is capable of checking local smoothness across straight edge, block boundaries. Thus, it is not ideal in quantifying the maintenance of global structure of synthesized textures. However, it provides a reasonable objective metric to compare the performance of texture synthesis algorithms. This is particularly true for regular textures.

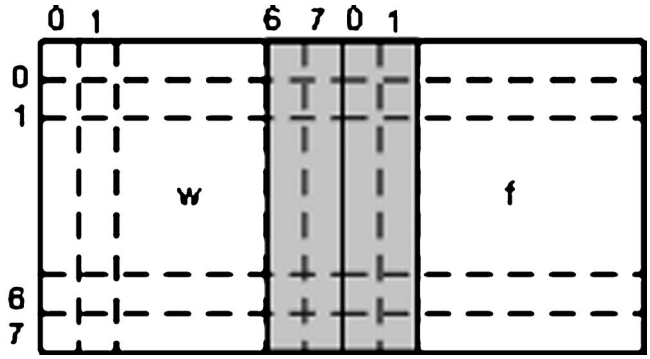

Figure 6. Pixels used in MSDS calculation.

Design Considerations

Closer subjective analysis of the synthesized textures in Fig. 5 and further experimental analysis carried out by us have revealed that there are important factors that are crucial in deciding the quality of the final texture synthesized by the proposed algorithm. They are:

(1) Level and type of detail used in selecting the matching block.
(2) Size of the unit of construction used.
(3) Overlapping area used in matching.

Level and type of detail

In contrast to the method proposed by Efros and Freeman, we have adapted a multiresolution matching strategy in selecting the adjacent blocks of the output texture. The use of pixel level detail in Ref. 23 would not only make the texture synthesis inefficient, but also unsuitable for real time texture synthesis capabilities expected by modern imaging applications. This is because full-resolution level details (pixel domain) have to be considered within the synthesis algorithm.

In the experiments performed we have assumed a typical three-level decomposition resulting in the lowest resolution component image to be 1/64 times the size of the original image. The decisions about matching adjacent blocks are taken only considering the above component and one other detail component (see Multiresolution Image Quilting Algorithm) from the third level of decomposition. Thus the computational complexity is considerably reduced as compared to the method proposed in Ref. 23 which performs block matching in the pixel domain, i.e., full resolution level. The exclusive use of the low resolution component is justified due to the fact that it possesses the highest energy of all four components at a given level of decomposition. Further experimental results indicated that the use of the additional detail component (see Multiresolution Image Quilting Algorithm) resulted in better matching as compared to using only the lowest resolution component, justifying its use. Further experiments also showed that using detail images of higher resolution levels does not significantly improve the quality of the final output texture.

Instead of randomly picking a block from a set of blocks, which matches within 10% of the error of the best

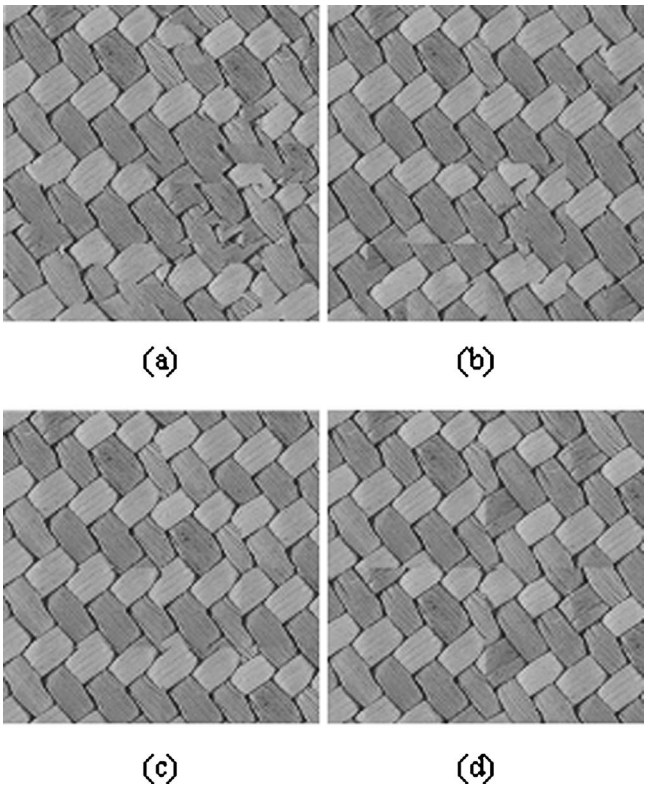

Figure 7. Block size vs synthesized texture quality (a) 16×16, (b) 32×32, (c) 64×64, and (d) 128×128.

matching block,[23] the proposed texture synthesis method selects the best possible match in terms of the overlap criteria discussed above. The experiments carried out by us revealed that the random block picking technique proposed in Ref. 23 is ineffective when used in conjunction with the proposed multiresolution-based approach. This is due to two reasons. The first is the fact that the proposed algorithm performs block quilting (matching) using the $p=a$ and $p=d$ (say) subbands of the lowest resolution level ($l=2$) and a randomly selected block within 10% of the error of the best matching block could be vastly different from the best matching block. Second, in order to keep the computational cost down we do not carry out the minimum error boundary cut proposed in Ref. 23 or the feathering technique proposed in Ref. 24. This simplification also means that the proposed technique needs less memory capacity to carry out this task.

Block size

In Multiresolution Image Quilting Algorithm it was mentioned that the size of the element used in building the overall texture accounts for the overall quality of the texture and the efficiency of the algorithm. In Fig. 7 we compare the synthesized texture quality obtained when using block sizes 16×16, 32×32, 64×64, and 128×128 (note: these are the equivalent full resolution block sizes). A close examination of Fig. 7 reveals that at block sizes 16×16 and 128×128, the block boundaries are more visible than at intermediate block sizes. Lower block sizes often result in the algorithm being unable to capture the global repetitive pattern that may be present in the sample texture. This may lead to a degradation of synthesized texture quality near boundaries. On the other hand, if the block size is too large, the number of matching candidates becomes less. This may affect the synthesized texture quality. Within our experiments we found out that an equivalent block size of 64×64 at full resolution level suits most textures. Note that this means at level-three decomposition the actual block size considered is 8×8. Further experiments revealed that for irregular textures, the effects of block size variation is minimal as compared to that of regular textures.

Table I. The effect of varying the block size on synthesis time and quality.

Block size	Synthesis time of a 512×512 texture (S)	MSDS
16×16	33.25	0.0129
32×32	9.92	0.0136
64×64	1.94	0.0140
128×128	0.24	0.0187

Table I tabulates the synthesis time and MSDS values of the texture of Fig. 7, when using different block sizes. Note that as expected the doubling of the block dimensions result in a more than four fold decrement in synthesis time. However, the MSDS values increases in increasingly larger steps giving rise to a need in selecting a compromised block size. Thus, the best approach could be to use a fully adaptive, variable block size algorithm rather than the fixed block size algorithm.

In the latter algorithm proposed in this article, we adaptively change the block size. Three different block sizes, 8×8, 4×4, and 2×2 are used in the resolution level $l=2$. Figure 8 shows the results synthesized using the proposed method when the adaptive block size algorithm is used. Figure 9 compares the performance of the proposed algorithm using fixed sized blocks with that of using variable size blocks. It clearly shows the improvement of the quilting quality obtainable with the adaptive block size selection algorithm.

In Fig. 7 it is illustrated that when a block size of 128×128 is used, the subjective quality of synthesized texture is superior when compared to that obtained when using a block size of 16×16. However, the MSDS values of Table I contradicts this as it gives a markedly smaller MSDS value when a block size of 16×16 is used. A close look at the synthesized textures show that even though the global smoothness of synthesized texture in Fig. 7(a) is low (highly visible artifacts at the bottom right corner), local smoothness (luminance gradients) in the texture across block boundaries is relatively better as compared to that of Fig. 7(d). This concludes that MSDS is not an ideal metric to measure and compare synthesized texture qualities under varying block

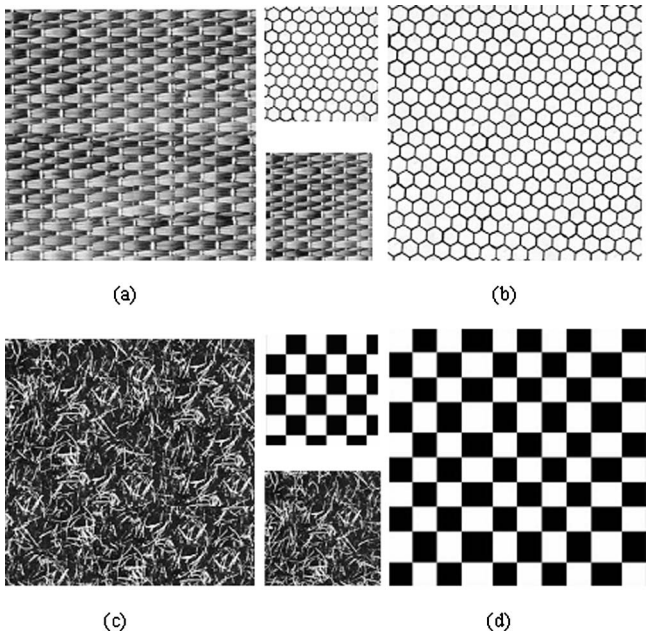

Figure 8. Texture synthesis results with adaptive block sizes.

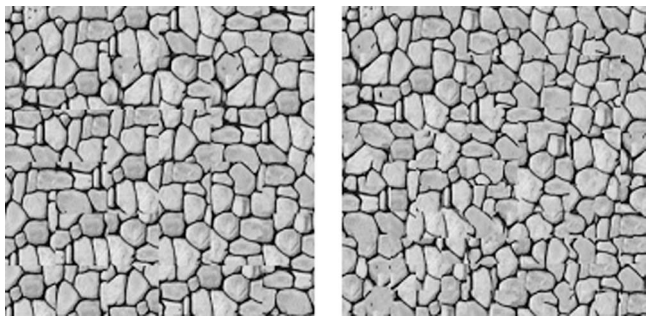

Figure 9. Performance comparison: The use of fixed (left) vs adaptive (right) block size.

Table II. The effect of varying the overlap size on synthesis time and quality.

Overlap	Synthesis time of a 512 × 512 texture (*S*)	MSDS
8	12.14	0.0186
16	21.76	0.0191
24	28.92	0.0194
32	33.45	0.0202
40	37.95	0.0215

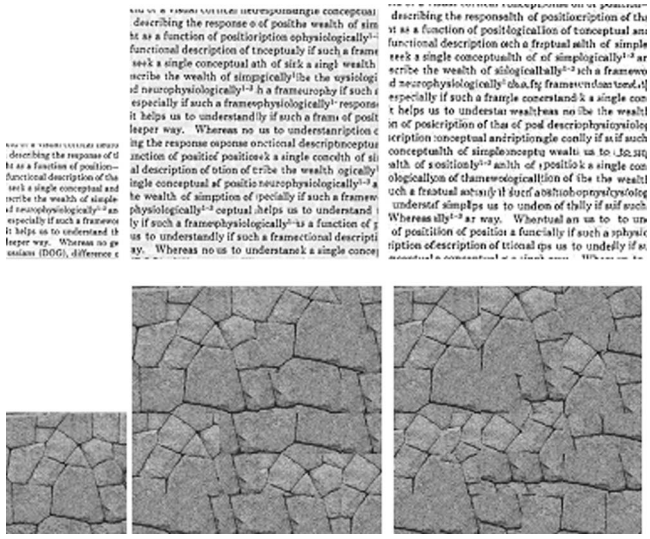

Figure 10. Comparison of performance of the purposed algorithm with Efros' and Freeman's (see Ref. 23) algorithm. Left: Sample input textures. Middle: Output textures generated using proposed algorithm. Right: Output texture generated using Efros' and Freeman's algorithm.

sizes. Note, however, that given the same block size, it is a good metric to compare and contrast synthesized textures, because global texture is maintained at an equivalent level when identical block sizes are used by two algorithms.

Area of overlap

In selecting the matching block, the area of overlap will also account for the overall quality of the synthesized texture and speed. Use of a smaller number of overlapping elements (coefficients) results in an increased speed but more visible artifacts at block boundaries. On the other hand, an increase in the number of overlapping elements results in fewer artefacts but an increased synthesis time. Table II tabulates the total synthesis time and MSDS values of the synthesized textures when different overlap sizes are used. Note that the block size used is 64 × 64 and the overlaps quoted are measured in pixels, both at full resolution (i.e., in pixel domain) level. The results in Table II confirm the conclusions drawn earlier. Note that, a too extensive increase in overlapping area (overlap = 40) will result in noticeable artifacts as it makes it more difficult for the algorithm to make the correct decision on the perceptually best matching block. For our experiments we have adapted an overlap of a single coefficient row (or column) at the third level of decomposition. This amounts to an overlap of eight pixel rows (or columns) in the pixel domain. Note that in Table I the best speed and quality performance is obtained when using the earlier overlap.

Comparison with Benchmark

Figure 10 compares the subjective performance of the proposed algorithm with that of Efro's and Freeman's. The results illustrate that the subjective quality of the proposed technique is marginally better than that of Ref. 23. Note that for a fair comparison with Ref. 23, we have used the fixed block size version of our algorithm for visual comparison in Fig. 10. In addition, Table III tabulates the MSDS values of the synthesized textures obtained with the proposed and benchmark algorithms for all test images considered in Figs. 5 and 10. These results show that the objective quality performance of the proposed technique is marginally better

Table III. Performance comparison: Efros' (Ref. 23) versus proposed.

	MSDS	
	Efros'	Proposed
Fig. 5(a)	0.0204	0.0201
Fig. 5(b)	0.0152	0.0146
Fig. 5(c)	0.0314	0.0306
Fig. 5(d)	0.0082	0.0081
Fig. 5(e)	0.0296	0.0287
Fig. 5(f)	0.0835	0.0776
Text image in Fig. 10	0.0744	0.0661
Granite image in Fig. 10	0.0243	0.0155

than that of Ref. 23. In particular, for regular textures the proposed algorithm performs relatively better in comparison to the algorithm of Ref. 23, because the benchmark algorithm picks up a random block among a set of best matching candidate blocks (within 10% of the best match) in patching two adjacent blocks. Although when using random textures this process may not create significant visual artifacts along block boundaries, when using regular textures, noticeable artifacts could arise and a randomly selected as a randomly selected block would not be the best choice.

The earlier results show that both the subjective and objective performance of the proposed multiresolution texture synthesis algorithm is slightly better than that of Ref. 23. However, due to the use of the multiresolution approach the best contribution of the novel technique is on the algorithmic simplification achieved. Given the fact that in the proposed technique the output texture is synthesized based only on a search on the lowest resolution level of the decomposed sample texture image, a comprehensive computational complexity investigation revealed that theoretically the proposed algorithm is approximately 64 times faster than the benchmark algorithm. This theoretical value has been calculated based on several assumptions. First, we have assumed that the computational cost of additions, subtractions, and modulus operations are equal and that only the diagonal detail subband (out of the three detail subbands) of the lowest resolution level is combined with the lowest resolution subband in calculating the selective area diffraction (SAD) of the overlapped areas. Second, in order to enable an equivalent single pixel shifted block to be selected in the full pixel resolution level, searching is actually performed on the lowest resolution subbands of 64 different wavelet decompositions of the sample image, each offset by one pixel. When the synthesized texture size is considerably large as compared to the sample texture, the earlier additional cost of needing 64, single pixel shifted DWTs of the relatively small sample image, can be neglected. However, when using a personal computer with a Pentium III 700 MHz processor, nonoptimized *MATLAB* code, software based wavelet transforms, no hardware acceleration, and synthesizing a 512×512 texture from a 184×184 sample, the effective speedup finally achieved was approximately tenfold. In particular when synthesizing all textures considered in Figs. 5 and 10, Efros' algorithm took an average of 98.26 s per texture whereas the proposed method took only 9.63 s. Optimization of the software based DWT implementations could lead to further speeding up of the proposed algorithm. Further investigations revealed that use of graphics hardware capable of doing fast wavelet transformations would improve the speed up obtainable to approximately 30-fold. However, when the algorithm is used within a image/video coding systems that require the multiresolution decomposition of texture detail, the requirement for dedicated wavelet transform for the purpose of texture quilting becomes nonessential. Therefore, the effective speedup that can be achieved becomes much higher than the tenfold value stated above.

APPLICATIONS OF THE PROPOSED ALGORITHM

The multiresolution texture synthesis technique adopted within the proposed scheme enables its use in modern interactive media applications that require progressive/interactive image/video transmission. Of specific interest is the video gaming industry that requires graphical objects to possess varying surface texture resolutions (thus appearance of reality) depending on the visual significance of the object within the viewed scene. In addition, the specific use of DWT in obtaining the multiresolution decomposition makes the proposed technique adoptable to texture synthesis applications that are used in conjunction with the JPEG2000 image coding standard and the Animation Framework Extension (AFX) of the MPEG-4 video coding standard, which are based on DWT based coding. In addition it can be used in relation to MIP-mapping[28] which is used for antialiasing in texture mapping which is used in majority of current graphics applications.

In Fig. 11, we illustrate the synthesis of the output texture using a multiresolution approach. The synthesis of the texture in Fig. 11(b) is done using only the C_{sa2} subband of the sample texture in Fig. 11(a). The synthesis of texture in Fig. 11(c) is done using all subbands at resolution level $l=2$. In synthesizing the texture in Figs. 11(d) and 11(e), the detailed subbands ($p=v$, $p=h$ and $p=d$) in the next higher resolution level are respectively considered in addition. Figures 11(f)–11(i) illustrate the wrapping of the above textures [i.e., those of Figs. 11(b)–11(e)] onto a sphere, which is a more frequently encountered scenario within modern applications.

The ability of multiresolution texture synthesis is a key advantage provided by the proposed algorithm. This was not possible by the adaptation of the pixel based approach to texture quilting as used in Refs. 23 and 24.

CONCLUSION AND FUTURE WORK

In this article we have introduced a novel approach to synthesizing textures under a multiresolution framework. We have provided experimental results and an in-depth analysis,

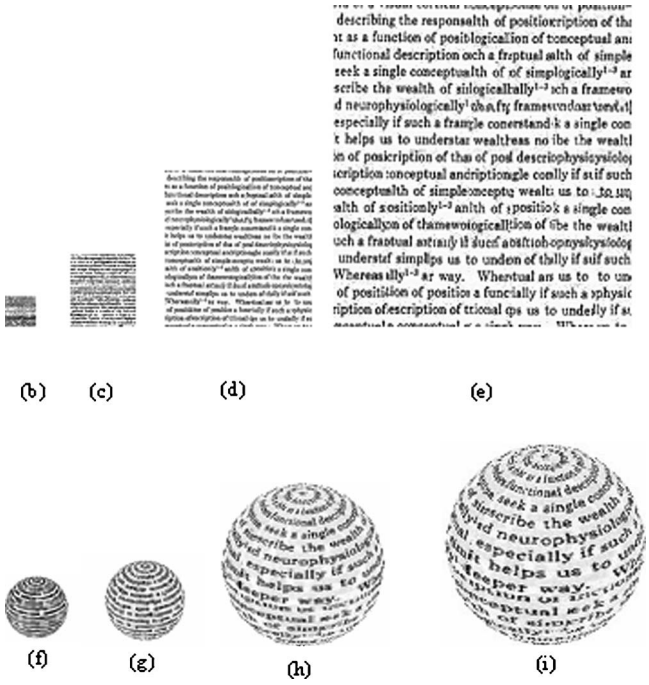

Figure 11. Multiresolution texture synthesis: (a) Input texture. (b)–(e) Synthesized textures onto a 2D surface (f)–(i) Synthesized textures onto a sphere, at progressively higher resolutions.

proving that the proposed method works remarkably well, producing equivalent (or better) output texture quality as compared to the method proposed in Ref. 23, at approximately sixfold decrement of computational cost. The multiresolution nature of the proposed framework also makes it easily applicable to modern imaging applications needing progressive transmission capabilities.

In designing the earlier multiresolution texture synthesis algorithm we have made a compromise between the synthesized texture quality and the algorithmic complexity by not performing seamless edge construction algorithms as in Refs. 23 and 24. However, due to the multiresolution approach and the novel matching criteria adopted, we have managed to obtain perceptually equivalent (or better) synthesized texture quality to that of Refs. 23 and 24 with much less computational complexity. We are currently looking at the implementation optimization of the algorithms and the use of fast, simple, seamless edge cutting/construction algorithms. We are also in the process of applying the idea to handle the texture synthesis part omitted from consideration in the fast *MESHGRID* coding algorithm of Ref. 29, which has been a key contribution to the MPEG-4 AFX coding standard. This is expected to extend the applicability of the *MESHGRID* algorithm to full, fast, multiscalable 3D object/surface coding.

REFERENCES

[1] A. Witkin and M. Kass, "Reaction-diffusion textures", *Proc. SIGGRAPH '91* (July 1991).
[2] G. Turk, "Generating textures on arbitrary surfaces using reaction-diffusion", *Proc. SIGGRAPH '91*, 289–298 (1991).
[3] J. P. Lewis, "Texture synthesis for digital painting", *Proc. SIGGRAPH '84*, 245–252 (1984).
[4] A. Fournier, D. Fussel, and L. Carpenter, "Computer rendering of stochastic models", *Comm. ACM*, 371–384 (1982).
[5] K Perlin, "An image synthesizer", *Proc. SIGGRAPH '85*, 287–296 (1985).
[6] S. P. Worly, "A cellular texture basis function", *Proc. SIGGRAPH '96*, 291–294 (1996).
[7] A. Efros and T. Leung, "Texture synthesis by non-parametric sampling", *International Conference for Computer Vision*, **2**, 1033–1038 (1999).
[8] K. Popat and R. Picard, "Novel cluster based probability model for texture synthesis", *Visual Communication and Image Processing*, 756–768 (1993).
[9] R. Paget and I. Longstaff, "Texture synthesis via noncasual nonparametric multi-scale Markov random field", IEEE Trans. Image Process. **7**(6), 925 (1998).
[10] K. Popat and R. W. Picard, "Novel cluster-based probability model for texture synthesis, classification, and compression", *Visual Communications and Image Processing*, 756–768 (1993).
[11] R. Paget and I. D. Longstaff, "Texture synthesis via a noncausal nonparametric multiscale Markov random field", IEEE Trans. Image Process. **7**(6), 925 (1998).
[12] S. Zhu, Y. Wu, and D. Mumford, "Filters, random fields and maximun entropy (FRAME)-towards a unified theory for texture modeling", Int. J. Comput. Vis. **27**(2), 107 (1998).
[13] A. Turing, "The chemical basis of morphogenesis", Philos. Trans. R. Soc. London, Ser. B **237**, 37 (1952).
[14] J. B. L. Bard, "A model for generating aspects of zebra and other mammalian coat patterns", J. Theor. Biol. **93**(2), 363 (1981).
[15] J. D. Murray, "On pattern formation mechanisms for lepidopteran wing patterns and mammalian coat markings", Philos. Trans. R. Soc. London, Ser. B **295**, 473 (1981).
[16] H. Meinhardt, "*Models of Biological Pattern Formation*" (Academic, London, 1982).
[17] J. Dorsey, E. Edelman, J. Legakis, H. W. Jensen, and H. K. Pedersan, "Modelling and rendering of weathered stone", *Proc. SIGGRAPH '99*, 225–234 (1999).
[18] W. Becket, and N. I. Badler, "Imperfection for realistic image synthesis", *J. Visualization and Computer Animation* **1**(1), 26–32 (1990).
[19] S. Hsu, and T. Wong, "Simulating dust accumulation", IEEE Comput. Graphics Appl. **15**(1), 18–22 (1995).
[20] D. J. Heeger and J. R. Bergen, "Pyramid-based texture analysis/synthesis", *Proc. SIGGRAPH '95*, 229–238 (1995).
[21] E. Simonceli and J. Portilla, "Texture characterization via joint statistics of wavelet coefficient magnitudes", *Fifth International Conference on Image Processing*, **1**, 62–66 (1998).
[22] J. S. De Bonet, "Multiresolution sampling procedure for analysis and synthesis of texture images", *Proc. SIGGRAPH 97*, 361–368 (1997).
[23] A. Efros and W. T. Freeman, "Image quilting for texture synthesis and transfer", *Proc. SIGGRAPH '01*, 341–346 (2001).
[24] L. Ling, C. E. Liu, Y. Xing, B. Guo, and H.-Y. Shum, "Real time texture synthesis by patch based Sampling", ACM Trans. Graphics **20**(3), 127 (2001).
[25] R. Szeliski, and H.-Y. Shum, "Creating full view panoramic mosaics and environment maps", *Proc. SIGGRAPH*, 251–258 (1997).
[26] D. M. Mount, *ANN Programming Manual* (Department of Computer Science, University of Maryland, College Park, MD).
[27] S. Minami and A. Zakhor, "An optimization approach for removing blocking effects in transform coding", IEEE Trans. Circuits Syst. Video Technol. **5**, 64 (1995).
[28] L. Williams, "Pyramidal parametrics", Comput. Graphics **17**(3), 1 (1983).
[29] I. A. Salomie, A. Munteanu, A. Gavrilescu, G. Lafruit, P. Schelkens, R. Deklerck, and J. Cornelis, "MESHGRID—A compact, multiscalable and animation-friendly surface representation", IEEE Trans. Circuits Syst. Video Technol. **14**(7), 950 (2004).