Reprint from
N.M. Thalmann · D. Thalmann (Eds.)

Communicating with Virtual Worlds

Springer-Verlag
Tokyo Berlin Heidelberg
New York London Paris
Hong Kong Barcelona
Budapest

A Context Sensitive Texture Nib

Tom Malzbender and Susan Spach

Abstract

Commonly, a "nib" refers to the point of a pen. When used with reference to painting packages, a nib refers to the pattern with which one draws. We introduce a technique for painting with *smart* nibs that draw and blend textures into an existing image. These nibs look in the neighborhood of pixels to be drawn and determine color values that create textures that blend in naturally with what is already in the image. Our technique is based on mimicking second order statistics of samples of scanned textures, typically taken from photographs of real textures occurring in nature. We use Color Co-Occurrence Matrices to capture and optimize these second order statistics. This technique is shown to provide interactive response and good performance over a broad range of stationary textures, from periodic to stochastic.

Keywords:

Texture Synthesis, Modelling, Paint System, Co-occurrence matrix, Image statistics.

1. Introduction

Several manufacturers of high end graphics workstations have recently included hardware support for texture mapping (Martin 90),(Segal 92). These products allow interactive rendering of detailed images by applying texture maps to simple geometric models. These texture maps are usually digitized with some sort of scanning hardware, or are procedurally defined by a function that generates a desired visual effect. One example of the latter approach is the approximation of wood grain by nested concentric cylinders perturbed by a noise function (Lewis 89). We are interested in an intermediate ground between these two approaches that uses parametric texture models measured directly from scanned texture imagery, just as one creates parametric or polygonal models that approximate an object's geometry. This eliminates the need for inventing a clever procedure to generate a texture. In this paper, we present a texture synthesis technique from such a parametric model and show one application of it to painting packages.

Numerous approaches have been applied to the problem of modelling texture. Even a cursory review of the various approaches is beyond the scope of this paper so we refer the reader to (Haralick 79), (Rao 90), (Garber 81). Much of this work concentrates on the analysis of textured regions in images for the purpose of texture classification. This capability is useful in scene understanding and computer vision. Some work has been

done on texture synthesis using models derived from acquired images. Unfortunately many of the techniques that have been developed are unsuitable for application in computer graphics, often because they are limited to a single color channel (grey scale) or binary images (black and white). Autoregressive models which allow us to control the autocorrelation function have been used, but we know that the autocorrelation function is insufficient to completely describe a texture (Pratt 78) (Pratt 81). Experimental work has been conducted by Julesz and others (Julesz 62), (Julesz 73), (Julesz 75) on what texture measures the human visual system is sensitive to. This has lead to what is commonly called the Julesz conjecture: the human visual system cannot effortlessly discriminate between textures that are similar in their first and second order statistics, but differ in their third or higher order statistics. Although counter-examples of this have been produced, the claim that much of the information used by the visual system in discriminating textures lies in the first and second order statistics is regarded as valid. For a quantized image with N color levels, $I[x,y] \in \{ L_1 \ldots L_N \}$, the order of statistics is given by:

1st Order: $p[\ I[x_1,y_1] = L_i\ ]$

2nd Order: $p[\ (\ I[x_1,y_1] = L_i\ ),\ (\ I[x_2,y_2] = L_j\ )\ ]$

3rd Order: $p[\ (\ I[x_1,y_1] = L_i\ ),\ (\ I[x_2,y_2] = L_j\ ),\ (\ I[x_3,y_3] = L_k\ )\ ]$

where *p* represents the probability of a pixel or set of pixels being in a certain state. The first order statistics are given by the histogram of the image, whereas the second order statistics are determined by the second order spatial averages, often measured by the Co-Occurrance Matrix defined in the next section.

We will describe the texture nib by first defining Co-Occurrance statistics, then show how to synthesize a field of texture (Gagalowicz 87), and lastly present techniques for extending the synthesis algorithm to interactive drawing with textures.

2. Color Co-Occurrance Statistics and Measurements

A set of texture measures that captures the salient aspects of a large class of textures are the second order spatial averages of that texture. This measure has several names and slight variations, but the fundamental idea was introduced by Haralick in (Haralick 73) under the name of grey-tone spatial dependency matrix. Another name frequently given is the grey-scale co-occurrence matrix. Here we will call its extension from grey-scale into color (Gagalowicz 87) the color co-occurrence matrix, or CCM. Given an image of size (X,Y) consisting of a discrete set of color levels $I[x,y] \in \{ L_1 \ldots L_N \}$, the CCM is a four dimensional matrix given by:

$$CCM[\Delta x, \Delta y, L_h, L_r] = \frac{1}{K} \sum_{x=0}^{X} \sum_{y=0}^{Y} \delta(\ I[x,y] - L_h)\ \delta(\ I[x+\Delta x, y+\Delta y] - L_r)$$

L_h and L_r are color indices also taken from the set $L_h, L_r \in \{L_1 .. L_N\}$. Typically, these color levels $\{L_1 .. L_N\}$ are indices of a color lookup table where the actual (usually 24

bit) color values are stored. L_h, the home color level, is the color level of the pixel in question and L_r, the remote color level is the color level at offset (Δx, Δy), δ is the Kronecker delta function which takes a value of 1 if and only if its argument is 0. K is a normalization factor defined as:

$$K = (X - | \Delta x |)(Y - | \Delta y |)$$

which is simply the count of the number of times that a particular offset (Δx,Δy) appears in the image.

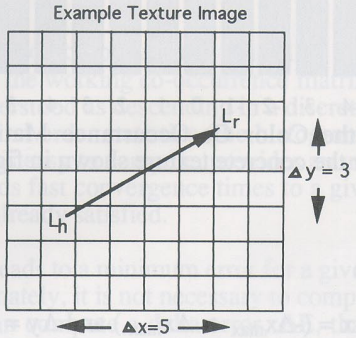


Fig. 1. Collecting Color Co-occurrence Statistics

Figure 1 provides a diagram of these relationships. An entry in the CCM is the probability that a given pixel at some location has value L_h and the pixel at offset (Δx,Δy) from this pixel has value L_r. This interpretation of a spatial average as a probability is only valid for a stationary texture, ie. a texture whose statistics don't vary across the texture sample and are not a function of position in the sample. Algorithmically, to measure the CCM of a particular image we first clear each entry of the CCM. Then we look at all the pairs of pixels for a given kernel size that occur in the image and increment the CCM entry corresponding to (Δx,Δy,L_h,L_r). Lastly, all entries of the CCM are normalized by K, the number of pixel pairs available in the image for a given (Δx,Δy). This procedure, as well as the synthesis procedure described in section 3 has been used in (Gagalowics 87). Figure 2 provides an example of what a typical slice of the co-occurrence matrix looks like. In this case we set $L_h = L_r = 6$, $\Delta x=0$ and vary Δy to produce a 1 dimensional slice of the 4 dimensional CCM for the digitized concrete image. Notice that the first order statistics are embedded in the CCM as well when $\Delta x=0$ and $\Delta y=0$.

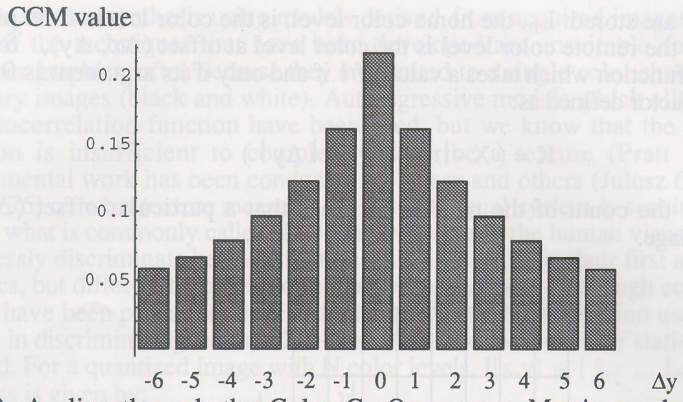


Fig. 2. A slice through the Color Co-Occurrance Matrix produced by holding $L_h = L_r = 6$ and $\Delta x = 0$ for the concrete texture shown in fig. 4.

2.1 Color Quantization

If the CCM offsets span $\Delta x = (-\Delta x_{max} \rightarrow \Delta x_{max},)$ and $\Delta y = (-\Delta y_{max} \rightarrow \Delta y_{max})$ then the CCM will be a four dimensional matrix of size:

$$(2 * \Delta x_{max} + 1) * (2 * \Delta y_{max} + 1) * N * N$$

It is important to keep both the number of color levels N and the size of the offset kernel small to keep the CCM size tractable. For this reason we quantize the original image produced by scanning photographs of real textures from a 24 bit representation down to a low number of color levels, typically 8 in our case. Although at first this may seem a severe truncation of information, one can typically retain excellent texture image quality with this small number of levels. This could be a result of the fact that textures often have high spatial frequency content and that the color sensitivity of the visual system drops with spatial frequency (Granger 73), (Green 68). There are numerous methods in the literature for color quantization (Gervautz 90), (Heckbert 82), (Pratt 78). Many of these, including our method, perform clustering in color space. We choose to use the K-means clustering method described in (Lim 90). It iteratively finds a set of N colors that approximate the centroids of clusters in color space that the texture sample contains. Although we perform this clustering in the (R,G,B) color space for convenience, a perceptually uniform color space such as CIE LUV (Foley 90) might prove more appropriate.

3. Synthesizing a Texture from White Noise

Because the CCM attempts to capture the significant features of a texture that our visual system is sensitive to, we expect that two images with similar CCMs will be visually similar as well. This suggests a texture synthesis procedure first proposed by (Gagalowics 85) and discussed further in (Gagalowics 87). First we measure the CCM of some texture of interest, here referred to as the destination CCM. Also we define a

working image, initialize it, and measure its CCM. Each pixel in the working image is then iteratively visited and is changed to the color value that minimizes the euclidean distance between the two CCMs in question. This amounts to minimizing the error function:

$$E = \sum_{\Delta x=-\Delta x_{max}}^{\Delta x_{max}} \sum_{\Delta y=-\Delta y_{max}}^{\Delta y_{max}} \sum_{L_h=1}^{N} \sum_{L_r=1}^{N} (CCM_w [\Delta x, \Delta y, L_h, L_r] - CCM_d [\Delta x, \Delta y, L_h, L_r])^2$$

where CCM_d is the destination co-occurrence matrix of the texture one would like to approximate and CCM_w is the working co-occurrence matrix of the image that is being modified. This can be understood as descending in a discrete error space. Typically we use noise with the same first order statistics (the histogram) to initialize the working image. This starting point is easy to compute, provides an unbiased initial condition in our search space, and yields fast convergence times to a given image quality since the first order constraints are already satisfied.

To find which color level leads to a minimum error for a given pixel, we must test each possible color level. Fortunately, it is not necessary to compute the error, E, for each of these tests. Instead one can compute a delta error, ΔE, that would be introduced by changing the pixel's value from its current level, L_c, to the level being tested, L_t. We then accept the test level yielding the smallest ΔE and proceed to the next pixel. We define CCM_t to be the CCM of our working image with the pixel in question having level L_t instead of L_c. It will differ only slightly from CCM_w. For each offset, $\Delta=[\Delta x,\Delta y]$, there are 4 changes to CCM_w that are caused by changing a pixel from L_c to L_t. If we define

$$L_c = I(x,y)$$

$$L_r^+ = I(x + \Delta x, y + \Delta y)$$

$$L_r^- = I(x - \Delta x, y - \Delta y)$$

then these 4 changes are:

$$1)\ CCM_t[\Delta x, \Delta y, L_c, L_r^+] = CCM_w[\Delta x, \Delta y, L_c, L_r^+] - \frac{1}{K}$$

$$2)\ CCM_t[\Delta x, \Delta y, L_t, L_r^+] = CCM_w[\Delta x, \Delta y, L_t, L_r^+] + \frac{1}{K}$$

$$3)\ CCM_t[\Delta x, \Delta y, L_r^-, L_c] = CCM_w[\Delta x, \Delta y, L_r^-, L_c] - \frac{1}{K}$$

$$4)\ CCM_t[\Delta x, \Delta y, L_r^-, L_t] = CCM_w[\Delta x, \Delta y, L_r^-, L_t] + \frac{1}{K}.$$

All other entries in CCM_t are identical to those in CCM_w. Unfortunately, these 4 changes need not refer to independent CCM entries, so in computing ΔE we must keep track of which CCM entries are changed by accepting the test level L_t. We will call this set of changed entries Ω. The differential error ΔE is then:

$$\Delta E = \sum_{\Omega} (CCM_t[\,] - CCM_d[\,])^2 - (CCM_w[\,] - CCM_d[\,])^2$$

An example of this technique applied iteratively to all pixels in an image starting with white noise is given in Fig. 4. Plots of the error, E, and number of pixels changed in each iteration are given in Fig. 3.

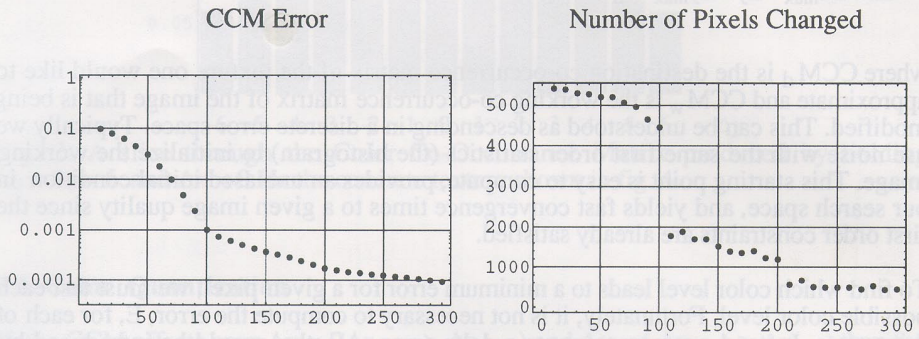
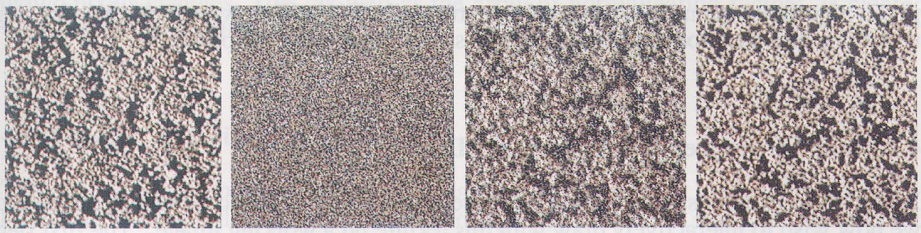


Fig. 3 - Results from a texture synthesis experiment. The left image shows the CCM Error as a function of the percent of pixels visited. The right image shows the number of image pixels changed as we iterate.

4. The Texture Nib

The texture synthesis procedure described above is quite powerful in that contextual information is being used for choosing pixel colors. This suggests application of the technique to illustration, graphics or painting systems ubiquitous on personal computers today. Current advanced painting packages allow areas of texture to be block-copied from one region to another or regions to be filled with particular textures. However, these techniques are not context dependent and don't blend the textures in naturally with surrounding details. The ideas discussed above can be harnessed to provide interactive drawing with predefined textures that use contextual information and provide smooth texture blending. In addition, users may select textured regions of images to define new texture nibs that have these same blending properties.

Figure 5 shows XNib, the texture drawing program. In this example, the working base texture image contains a crushed soda can on a gravel texture. The user selects which nib texture to draw with and is free to vary the size of the nib being drawn with as well as the extent of the CCM, (Δx_{max},Δy_{max}). The nib cursor (shown in red on the working base texture) is 12x12 pixels, the gravel texture is selected and the CCM extent is $\Delta x_{max} = \Delta y_{max} = 3$. The dotted red line under the working base texture shows the user the CCM extent. As the user draws with the texture nib, the pixels under the nib cursor are changed to the color of the nib texture that minimizes the difference between the measured CCM of the nib texture and the current CCM of the



original noise partial synthesis synthesized texture

Fig. 4. Full field texture synthesis example.

Fig 5. Xnib texture drawing program

pixels that have been drawn with the texture nib. We will now outline our drawing technique and describe the associated data structures used in the process.

4.1 Data Structures for Interactive Display

As a starting point, assume that we have measured the CCM of some textured area and we now wish to interactively draw with this texture on a working image. The first step is to encode the working image, which is usually a 24 bit color image,

$$\vec{W}[x,y] = (R[x,y], G[x,y], B[x,y])$$

relative to the colors that are used in the texture nib. We call this image *base relative*, B_r, and it is recomputed every time a new texture nib is selected. If the euclidean distance, d, between the working image pixel color and the closest nib color is less than some threshold, φ, we assign that nib color to base relative. Otherwise, a flag is set for the pixel indicating that the color that appears at this location is not relevant to the nib texture statistics collected. More formally:

$$d^2[L_i] = (R[x,y] - C_R[L_i])^2 + (G[x,y] - C_G[L_i])^2 + (B[x,y] - C_B[L_i])^2$$

$$B_r[x,y] = \begin{cases} L_{min} & \text{if } \underset{1<i<N}{\text{Min}}(d^2[L_i]) \le \varphi^2 \\ flag & \text{otherwise} \end{cases}$$

where L_{min} is defined as $d^2[L_{min}] = \underset{1<i<N}{\text{Min}}(d^2[L_i])$ and $(C_R[L_i],C_G[L_i],C_B[L_i])$ are the color values associated with the nib at color index L_i. In the synthesis procedure outlined in section 3 we minimize the difference between our destination and working CCMs. In our interactive drawing program, the destination CCM is that of the nib which is computed either as a preprocess or on the fly. Our working CCM is less straightforward in the interactive case. We build a working CCM from the statistics of the pixels that have been drawn by the current texture nib. Since this set of pixels can be of arbitrary shape, computation of the CCM normalization factors, K_d, is done with the aid of a bitmap called the *occupancy map*, O[x,y], which takes a value of 1 if a pixel has been visited by the nib since the last nib selection, and 0 otherwise. As each pixel is drawn, the map is incremented and an array of dynamic normalization factors, K_d is updated by referencing the map.

$$K_d[\Delta x,\Delta y] = \sum_{x=0}^{X} \sum_{y=0}^{Y} \delta(O[x,y]) \delta(O[x-\Delta x,y-\Delta y])$$

An unnormalized working CCM is also updated as each pixel is drawn. We maintain an unnormalized working CCM to avoid having to "unnormalize" repeatedly as the set of

drawn pixels changes. This, as well as the array of normalization factors, is cleared when the nib is selected.

4.2 Texture Painting Dynamics

Typically, painting packages allow the user to vary the size of the drawing nib. Providing this capability for our texture painting is more difficult since the computation time for determining an optimal pixel color can be quite long for large (Δx_{max},Δy_{max}) values. To allow the user to draw with larger nibs and retain smooth mouse performance, we compute how many pixels, η, can be computed in a given time step (eg. say .1 seconds) that ensures interactive response. We then sequence through η pixels of the nib area in a deterministic pseudorandom sequence. The subsequent pseudorandom sequence will start where the previous sequence ended and is designed in such a way that all pixels in the nib are visited once before any pixels are visited twice. In this way, under high computational loads a spray paint pattern of textured pixels is deposited at each cycle of the event loop and a solid area is filled in if the user keeps the nib cursor stationary. The pseudorandom sequence is implemented using an exclusive-or shift-register feedback scheme described in (Press 88).

Our texture nib prototype also allows one to define new texture nibs by outlining a textured region of the current image. The CCM for that region is then measured and can be used to draw with. Depending on the size of the region selected and the color quantization involved, this can typically be done in a matter of a few seconds. We also provide the capability of zooming into arbitrary regions of the image and drawing while pixels are enlarged. In addition, various data structures such as base relative and map can be interactively displayed at any point.

5. Results

We have implemented the texture nib program in C and Motif on a HP 730 workstation, a 76 MIPS machine. On this platform we measure a speed of 450 pixel updates per second in the texture drawing, which allows interactive use. The conditions for this benchmark are N=8 and $\Delta x_{max}= \Delta y_{max} = 2$ pixels, yielding a 5x5 CCM kernel size. Pixel update rates essentially scale inversely with each of these 3 parameters. We limit the minimum drawing speed to 10 frames per second using the spray paint technique.

Figure 6 shows three examples of drawing with the texture nib. Its use for removing unwanted imagery embedded in a texture field is shown. Here we start with an image of a bee on three separate textures. By measuring the statistics of the underlying texture we are able to erase the bee in these images, even though bee is not segmented from the image in any way (no pun intended).

We have realized good performance of this algorithm over a broad range of textures, from stochastic to periodic textures. Textures with long range correlations, such as wood grain, are difficult for our technique since the Δx_{max}, Δy_{max} values must be kept relatively small to keep the CCM size small and pixel optimization time short.

Fig. 6. - Interactive texture synthesis controlled by a nib cursor.

6. Conclusions

We have introduced the capability of interactive drawing with textures that use contextual information in determining which colors to draw with. This is implemented by measuring second order spatial statistics of a desired stationary texture in the form of a four dimensional Color Co-Occurrance Matrix (CCM). Pixels are then drawn with colors that minimize the difference between a CCM representing pixels drawn and the measured, desired CCM. In this way the second order statistics of the pixels drawn approximate the second order statistics of the measured textures and hopefully fool the visual system into regarding the drawn texture as belonging to the same class as the desired texture. There is evidence that these second order statistics are significant for the visual system's perception of texture and this has motivated their use. We have described interactive software that implements this optimization with good performance.

Acknowledgements

We would like to thank Alex Sherstinsky for help with early texture nib prototype software. We also thank Jim Christy for photographing and scanning several of the textures that appear in this paper.

References

Brodatz,P. (1966) "Textures: A Photographic Album", Dover, New York.

Haralick, R.M., Shanmugam,K, Dinstein (1973) "Textural Features for Image Classification", IEEE Transactions on Systems, Man and Cybernetics, SMC-3, pp.610-621.

Gagalowics, A. and De Ma, S. (1985) "Sequential Synthesis of Natural Textures", Computer Vision, Graphics, and Image Processing, 30, pp. 289-315

Gagalowics, A. (1987) "Texture Modelling Applications", The Visual Computer, Vol. 3, pp.186-200.

Garber, D.D. (1991) "Computational Models for Texture Analysis and Texture Synthesis", Phd. Dissertation, University of Southern California, May, 1991.

Gervautz, M. (1990) "A Simple Method for Color Quantization: Octree Quantization", in Graphics Gems, edited by Glassner, A., Acedemic Press, San Diego, pp.287-293.

Gonzalez, R., Woods, R., (1992) "Digital Image Processing", Addison Wesley Publishing Company, Reading, Massachusetts.

Granger, E.M., Heurtley, J.C. (1973) "Visual Chromatic Modulation Transfer Function", Journal of the Optical Society of America, Vol. 63, pp.1173-1174.

Green, D.G. (1968) "The Contrast Sensitivity of the Colour Mechanisms of the Human Eye", Journal of Physiology, Vol. 196, pp.415-429.

Foley, J., Van Dam, A., Feiner, S., Hughes,J, (1990) "Computer Graphics, Principles and Practive, Second Edition", Addison-Wesley Publishing Company, Reading, Massachusetts.

Haralick, R.M., Shanmugam, K., Dinstein, I. (1973) "Textural Features for Image Classification", IEEE Transactions on Systems, Man, and Cybernetics, Vol. SMC-3, No.6, November 1973, pp. 610 - 621.

Haralick, R.M. (1979) "Statistical and Structural Approaches to Texture", Proceedings of the IEEE, Vol. 67, May 1979, pp.786-804.

Heckbert, P.S. (1982) "Color Image Quantization for Frame Buffer Display", Computer Graphics, Vol. 16. No. 3, pp.297-307.

Julesz, B. (1962) "Visual Pattern Discrimination", IRE Transactions on Information Theory, it-8, pp. 84-92.

Julesz, B. (1971) "Foundations of Cyclopean Perception", The University of Chicago Press, Chicago.

Julesz, B. (1973) "Inability of Humans to Discriminate Between Textures With Identical Third Order Statistics Revisited", Perception 2, pp. 391-405.

Julesz, B. (1975) "Experiments in the Visual Perception of Texture", Scientific American, April, 1975, pp.34-43.

Lewis, J.P. (1989) "Algorithms for Solid Noise Synthesis", Computer Graphics, Vol. 23, No. 2, July 1989, pp 263-270.

Lim, J.S. (1990) "Two-Dimensional Signal and Image Processing", Prentice Hall, Edgewood Cliffs, N.J., pp.606, 1990.

Martin, P., Baeverstad, H. (1990) "Turbo VRX: A High Performance Graphics Workstation Architecture", Proceedings of AUSGRAPH 90, September 1990, pp. 107-117.

Pratt, W.K. (1978) "Digital Image Processing", John Wiley and Sons, New York, pp.155- 160.

Pratt,W.K., Faugeras, O.D., Gagalowics, A. (1978) "Visual Discrimination of Stochastic Texture Fields", IEEE Transactions on Systems, Man and Cybernetics, Vol. SMC-8, No,11, November 1978.

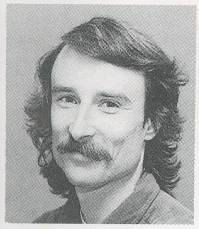
Pratt, W.K., Faugeras, O.D., Gagalowics, A. (1981) "Applications of Stochastic Texture field models to image processing", Proceedings of IEEE, Vol. 69, pp.542-551.

Press, W., Flannery, B., Teukolsky, S., Vetterling,W. (1988) "Numerical Recipes in C, the Art of Scientific Computing", Cambridge University Press, Cambridge, pp.224-228.

Roa A.R. (1990) "A Taxonomy for Texture Description and Identification", Springer Verlag, Berlin, Germany.

Segal, M., Korobkin, C., Widenfelt, R.V., Foran, J., Haeberli, P. (1992) "Fast Shadows and Lighting Effects Using Texture Mapping", Computer Graphics, Vol. 26 No. 2., July 1992, pp. 249- 252.

Zucker, S., Terzopoulos, D. (1980) "Finding Structure in Co-Ocuurrence Matrices for Texture Analysis', Computer Graphics and Image Processing, 12, pp. 286-308.



Tom Malzbender is a research engineer in the Media Technology Lab at Hewlett-Packard Laboratories. Tom joined Hewlett-Packard in 1982 and has developed and patented the Permuted Trace Ordering Scheme which is the basis of HP's current line of Graphics Tablets. His research interests include volume rendering, medical imaging, neural modelling and brain functioning. He received a B.S. degree in Electrical Engineering from Cornell University in 1982 and is a member of ACM.

Email: tom_malzbender@hplabs.hp.com

Susan Spach is a member of technical staff in the Media Technology Lab at Hewlett-Packard Laboratories. Her research interests include interactive computer graphics, parallel algorithms for computer graphics and medical imaging. She received her MS in computer science from the University of North Carolina at Chapel Hill in 1981. She is a member of ACM and the IEEE Computer Society.

Email: spach@hplabs.hp.com

Address: Hewlett-Packard Laboratories
1501 Page Mill Rd.
Palo Alto, California
94304, USA