Spot Noise
Texture Synthesis for Data Visualization

Jarke J. van Wijk
Netherlands Energy Research Foundation ECN
P.O. Box 1, 1755 ZG Petten, The Netherlands

ABSTRACT

The use of stochastic textures for the visualization of scalar and vector fields over surfaces is discussed. Current techniques for texture synthesis are not suitable, because they do not provide local control, and are not suited for the design of textures. A new technique, *spot noise*, is presented that does provide these features. Spot noise is synthesized by addition of randomly weighted and positioned spots. Local control of the texture is realized by variation of the spot. The spot is a useful primitive for texture design, because, in general, the relations between features of the spot and features of the texture are straightforward. Various examples and applications are shown. Spot noise lends itself well for the synthesis of texture over curved surfaces, and is therefore an alternative to solid texturing. The relations of spot noise with a variety of other techniques, such as random faults, filtering, sparse convolution, and particle systems, are discussed. It appears that spot noise provides a new perspective on those techniques.

CR categories and subject descriptors: I.3.3 [**Computer Graphics**]: Picture/Image generation; I.3.7 [**Computer Graphics**]: Three Dimensional Graphics and Realism - color, shading, and texture.

Keywords: texture synthesis, scientific visualization, flow visualization, fractals, particle systems.

1 INTRODUCTION

Scalar and vector fields over surfaces have many applications, ranging from common scalar functions of two variables, used in many disciplines, to the distribution of pressure and velocity over a ship hull or the wings of an airplane. The topic of this paper is the use of texture, loosely defined as the local variation in visual properties, for the visualization of fields over surfaces. Tufte [33] has shown that the use of fixed patterns leads to poor results. A better result can be expected if the texture is based on a stochastic, rather than a deterministic model. Several terms are used for such textures: stochastic textures, random fields, and noises.

Applications of stochastic texture in scientific visualization are rare. Krueger [16] has used texture to show the differences between related data sets. In the context of flow visualization it has been noted [34, 35, 38] that the simulation of particle convection leads to texture. If many particles are used, the individual particles cannot be distinguished any more and clouds, smoke and other typical textures that are well known in experimental flow visualization are perceived. These applications show that texture is a useful concept for scientific visualization, but it is not clear how the proposed techniques should be used for other applications.

In this paper a more general approach to the design and synthesis of texture for scientific visualization is presented. In section 2 the requirements are drawn up, and current techniques for texture synthesis are discussed. It appears that the techniques used for the synthesis of realistic textures do not fulfill these requirements. In section 3 *spot noise* is introduced. Its synthesis is based on the principle that the random placement of a small pattern, the spot, over a surface leads to texture. In section 4 it is shown that this technique is very appropriate for the design of textures, because the relations between the features of the spot and those of the corresponding texture are straightforward. In section 5 various applications of spot noise are presented, for data visualization and for image synthesis. In section 6 spot noise is compared with existing techniques, and directions for further research are indicated. Finally, in section 7 conclusions are drawn.

2 TEXTURE FOR DATA VISUALIZATION

2.1 Requirements

Figure 1 shows a data flow diagram of texture synthesis for data visualization. The parameter values for the texture synthesis process are determined in two steps. First, the data are retrieved that correspond to the texture coordinates; second, these data are converted into parameter values according to a data mapping specified by a designer. The term *designer* is used here functionally: it can be an expert in visual communication, but also a researcher that wants to visualize his data. With this diagram in mind, the requirements on texture synthesis can easily be derived. They fall into two categories: texture generation and design.

The synthesis technique has to allow for non-stationary textures to express the variation in the data. Further, the model has to allow for a wide range of textures. The aim of realism is replaced by the aim of expressiveness: it must be possible for the designer to choose a texture that matches with the nature of the data, and variations in the data must lead to clear variations

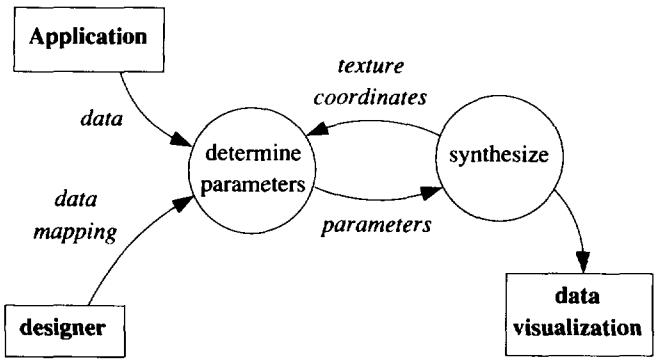

Fig. 1 Data flow diagram texture synthesis for data visualization

in the texture.

The selection of a suitable data mapping is an iterative design process. The efficiency of this process depends on several aspects. Obviously, the synthesis of the texture has to be efficient. Another way to improve efficiency is to use previews, simplified versions of the final result, during the design phase [13]. Finally, the number of iterations can be reduced if the relation between the specification and the resulting texture is clear to the designer [36]. The closest relation is one-to-one, but in order to limit the designer's work we further require that the specification should be of a suitable level: instead of drawing the texture himself, the designer must be enabled to specify the features of the texture and their relation with the data on a higher level.

Summarizing, a synthesis technique is required for non-stationary textures that can be specified in a simple, predictable way. In section 2.2 an overview is presented of the main current techniques for texture synthesis. In section 2.3 they are tested against our requirements.

2.2 Texture synthesis

Stochastic textures are realizations of a statistical model. There is a general consensus, supported by evidence from perception research [14], that second-order, or pairwise statistics suffice for the description of textures that can be discriminated by human observers. Techniques that use the full second-order statistics [23, 9] are very general and give impressive results [9]. However, they are also quite involved and have a brute-force character.

A convenient simplification is the restriction to so-called Gaussian textures. This simplification is similar to the common simplification for first-order statistics: if a normal or Gaussian distribution is assumed, the distribution can be fully described by its mean and variance. Gaussian textures are described by their autocorrelation function $C_f(\tau)$, which is the correlation of two random samples of f at an interval τ. For a one-dimensional stochastic function $f(t)$ with zero mean it is defined as

$$C_f(\tau) = \langle f(t) f(t+\tau) \rangle ,$$

where triangular brackets denote averages over many samples. $C_f(0)$ equals the variance σ^2 of f. If $C_f(\tau) = 0$, the function f is completely uncorrelated for samples at distance τ. For common stochastic functions, $C_f(\tau)$ approaches 0 with increasing τ.

A strongly related technique is spectral modeling, which is based on the use of power spectra. The power spectrum $P_f(\omega)$ of a stochastic function $f(t)$ is

$$P_f(\omega) = \lim_{T\to\infty} \frac{1}{T} |F_T(\omega)|^2 ,$$

where $F_T(\omega)$ is the Fourier transform of a sample of $f(t)$ with length T. According to the Wiener-Khintchine relation [3], the autocorrelation function and the power spectrum provide equivalent information, because they are a Fourier transform pair:

$$P_f(\omega) = \int_{-\infty}^{\infty} C_f(\tau) e^{-i\omega\tau} d\tau .$$

The standard approach of spectral modeling is to filter white noise (with a constant power spectrum) with a transfer function $H(\omega)$. Voss [27] has used this technique to generate fractal textures and terrains: noises with power spectra $f^{-\beta}$. These noises are generalizations of Brownian motion ($\beta = 2$), and are called fractal Brownian motions (fBm) [21, 27]. The first simulations of fBm were based on considering Brownian motion as the cumulative displacement of a series of independent jumps or pulses [21]. This technique is generalized to surfaces by using random faults instead of random jumps.

Fournier, Fussel, and Carpenter [7] use stochastic subdivision to generate fractal terrains. Lewis [18] has generalized the stochastic subdivision technique for arbitrary power spectra and autocorrelation functions.

In [17] a technique is described for the synthesis of textures for digital painting. These textures are the result of weighted additions of a displaced, windowed texture sample, where the weights and displacements are chosen randomly. This process is equivalent to an out-of-order convolution of the sample with a sparse, white noise, hence it is named sparse convolution.

Perlin [28] generates solid textures through the composition of non-linear functions. For stochastic textures he defines the function *Noise*(**x**) as a modeling primitive. This function is band-limited, statistically invariant under translations (stationary) and statistically invariant under rotations (isotropic). Fractal textures are modeled as linear combinations of the scaled noise function:

$$f(\mathbf{x}) = \sum_k \frac{Noise(2^k \mathbf{x})}{2^k} .$$

In a similar way turbulence, marble and a variety of other natural textures can be modeled.

2.3 Evaluation

The issue of local variation of texture is not mentioned in most of the literature. An exception is Lewis [18], who states that local variations of the texture may be effected by varying the model parameters or by simple postprocessing techniques, rather than by incorporating these variations in the original model. Both approaches are used by Musgrave et al. [24] for the synthesis of eroded terrain. His technique for the initial synthesis of the fractal terrain is based on Perlin's: the weights for the *Noise* function are functions of the altitude. This works well for isotropic textures, but the implementation of anisotropic textures with local variations is less straightforward. For instance, if we want to visualize a 2-D flow velocity field **v**(**x**) with an anisotropic texture, such that the dominant direction aligns with the direction of the flow, a natural solution would be:

$$f(\mathbf{x}) = Noise(\mathbf{x} - (\mathbf{x}\cdot\mathbf{v})\mathbf{v}) .$$

Here the primitive *Noise* texture is stretched according to the magnitude and direction of the velocity. If **v**(**x**) is constant, this gives the desired result, in most other cases, however, it does not. Other solutions in the same spirit could be devised, but they all share the same deficit: local deformations of texture cannot be modeled by global transformations (scaling etc.) of a

texture.

The direct spectral approach and the random fault technique are not suitable for local, spatial control. With the other techniques it is indeed possible to vary the parameters as a function of space, but the literature does not make clear how this should be done to realize a desired effect.

This is related to the next point of our evaluation: the design of textures. As most authors aim at realism, again this issue is not mentioned often. The examples given in [28] for the construction of solid textures are based on more or less simplified models of physical processes. In [23, 9] the second-order statistics are derived by sampling real-world textures, for the generation of fractal terrains the power spectrum $f^{-\beta}$ is used as a starting point [21, 7, 27].

Let us therefore consider possible ways for a designer to specify a texture. For spectral modeling three options are available. First, a designer can enter a sample of the desired texture, from which the desired parameters are derived. However, it is not simple to render a suitable texture by hand. Further, the tautological character of this solution strongly suggests rejecting it. Second, the designer could enter an autocorrelation function. Advantages of the autocorrelation function are that the spatial domain is more familiar to most people than the spectral domain, and it directly reflects features such as the scale, period of oscillation, and directional tendencies. Although this is certainly true if a given autocorrelation function is analyzed, the design of an autocorrelation function, especially in two dimensions, is far less simple. Another severe problem is further that not every autocorrelation function leads to a realizable texture, because its Fourier transform, the power spectrum, must be nonnegative. As a third approach, the designer could specify the power spectrum. In [17] it is stated that it is possible to acquire an intuitive feel for the relation between a painted spectrum and its corresponding texture. The author of that article could reliably paint spectra to simulate some textures, but this might be different for an arbitrary designer.

Besides the specification of a standard texture, the designer also has to specify how the texture has to be varied as a function of the data, which aggravates the problems of the three discussed options.

As a conclusion, we can state that no current technique for texture synthesis provides an easy solution that satisfies our requirements for data visualization. This can be explained from the difference between the applications. Traditionally, the focus is on the synthesis of realistic, stationary textures, whereas for the application discussed here clarity, ease of design, and local control are the main requirements. In the next sections, a texture synthesis technique is described that was developed with those requirements in mind.

3 SPOT NOISE

3.1 Definition

In this section a texture for data visualization is presented: *spot noise*. Spot noise has strong relations with the techniques discussed in the previous sections. The specific advantages of spot noise will emerge in practical applications, discussed in section 4 and 5.

Spot noise is the spatial analogue of shot noise. Shot noise [3] is a special kind of random function that has many applications in engineering. It is produced by the successive repetition at random intervals of independent pulses. If each pulse produces the profile $a_i h(t-t_i)$, the resultant function $f(t)$ is thus

$$f(t) = \sum_i a_i h(t-t_i) ,$$

where the values t_i of the independent variable (e.g. time) form a random sequence. The power spectrum of $f(t)$ is directly related to the energy spectrum

$$S_h(\omega) = |H(\omega)|^2$$

of $h(t)$, where $H(\omega)$ is the Fourier transform of $h(t)$. If a_i has zero mean, and if on average there are ν repetitions per unit time, then

$$P_f(\omega) = \nu < a_i^2 > S_h(\omega) .$$

The spatial analogue also has many applications, for instance in diffraction theory. For the application discussed here, the pulse $h(\mathbf{x})$ is considered as a spot that is dropped on the plane, hence we call the noise produced spot noise. The size of a spot is limited, and usually small compared to the size of the texture segment to be synthesized. In analogy with shot noise, spot noise is defined as

$$f(\mathbf{x}) = \sum_i a_i h(\mathbf{x}-\mathbf{x}_i)$$

where $\mathbf{x}_i$ are random positions on the plane. If on average there are ν repetitions per unit area then

$$P_f(\mathbf{k}) = \nu < a_i^2 > S_h(\mathbf{k}) ,$$

where $\mathbf{k}$ is the two-dimensional frequency vector.

3.2 Synthesis

The last relation of 3.1 is valuable for the synthesis of spot noise. It states that the power spectrum of the texture and the energy spectrum of the spot are the same, except for a scale factor. So, realizations of spot noise can be constructed in the frequency domain via the multiplication of the Fourier transform $H(\mathbf{k})$ with a scale factor and addition of a random phase shift $\alpha_\mathbf{k}$ to $H(\mathbf{k})$.

The addition of a random phase shift $\alpha_\mathbf{k}$ is equivalent to multiplication with $W(\mathbf{k}) = e^{i\alpha_\mathbf{k}}$. The power spectrum of $w(\mathbf{x})$ is evenly distributed over all frequencies, so $w(\mathbf{x})$ is white noise. According to the convolution theorem, multiplication in the frequency domain is equivalent to convolution in the spatial domain, hence spot noise can also be synthesized via convolution of $h(\mathbf{x})$ with white noise.

An example of white noise is a set of random values on a grid. Spot noise can therefore be synthesized through the convolution of a randomly filled grid with the spot. This method can be compared to the filtering of a very noisy image with the spot as the filter kernel, a standard technique in digital image processing [10]. In the natural texturing model [15] a similar technique is used to synthesize texture.

Another example of white noise is a Poisson point process: a set of randomly scaled delta functions $a_i \delta(\mathbf{x}_i)$, randomly distributed over the plane. Here we close the circle: the convolution of a Poisson point process with a spot boils down to dropping spots on the plane, which is the original definition of spot noise. Random faults [21] and sparse convolution [17] are based on the same principle.

Variation of the texture for data visualization can be realized via variation of the spot. This requires a variable spot $h(p, \mathbf{x})$, whose properties are controlled by a set of parameters p. These parameters are determined via a data mapping m from the data $d(\mathbf{x})$ that belong to the texture coordinates $\mathbf{x}$. Spot noise for data visualization can thus be synthesized by using variable spots:

$$f(\mathbf{x}) = \sum_i a_i h(m(d(\mathbf{x}_i)), \mathbf{x}-\mathbf{x}_i) .$$

A drawback of this method is that the data to be visualized are smeared out. At each point several spots that correspond to different data values overlap. This is not a problem if the variation in the data is small relative to the size of the spot. Another solution is to use an alternative definition for variable spot noise:

$$f(\mathbf{x}) = \sum_i a_i h(m(d(\mathbf{x})), \mathbf{x}-\mathbf{x}_i) ,$$

i.e. the texture at a point $\mathbf{x}$ is considered as if it is part of a stationary texture constructed with identical spots that have the properties that correspond to the data at point $\mathbf{x}$. Another interpretation is that the spot is used as a (position dependent) filter-kernel for a Poisson point process. A possible implementation, though not very efficient for large spots, is via Perlin's approach. The preceding discussion reveals how this can be done: not via scaling, but via convolution of *Noise*:

$$f(x,y) = \sum_i \sum_j h(m(d(\mathbf{x})), x+i, y+j) Noise(x+i, y+j) .$$

As a final remark, the variance of spot noise is given by

$$\sigma^2 = v < a_i^2 > \iint h^2(\mathbf{x})\, d\mathbf{x} .$$

Note that in general the variance of a_i has to be adapted as a function of $\mathbf{x}_i$ if a constant variance of the texture is desired with a varying spot.

4 SPOT AND TEXTURE

In the preceding section we saw the strong relation between the energy spectrum of the spot and the power spectrum of the texture, and hence also between their autocorrelation functions. In this section the relation between a spot and the resulting texture is discussed from a designer's point of view. Given a simple spot, how are its features, such as size and shape, related to features of the texture, such as granularity and isotropy ? This will be shown with a number of examples, leading to rules of thumb that can be used for texture design. In this section all examples of textures are stationary, in the next section the use of space-variant spots will be discussed. The images in this section are made via addition of a random phase shift to the Fourier transform of the spot, followed by an inverse Fourier transform and normalization.

4.1 Size

A disk is the simplest spot that can be used. Figure 2 shows three disks with different radii and the corresponding textures. The differences in the textures can be explained from their power spectra. In figure 3 the power spectrum sinc^2 of the one-dimensional equivalent of a disk, the rectangular pulse, is shown. For the graphical display of a random function in the spatial domain, a finite band in the frequency domain has to be selected, because of limitations in resolution. The display of a large sample (narrow pulses) comes down to the selection of a band in the low frequencies, whereas the display of a small sample (wide pulses) is equivalent to selection of a band in the high frequencies. Thus, figure 3 shows that narrow pulses lead to white noise. For wide pulses, the right flank is dominant. This flank falls off with f^{-2}, which is the same as for Brownian motion. If the width of the pulse lies between those extrema, the corresponding random function is white noise that has been passed through a low-pass filter.

For two-dimensional signals, i.e. textures, a similar result can be derived in the frequency domain. However, a derivation in the spatial domain is instructive as well. If small spots are used, samples at different locations are uncorrelated, and hence the

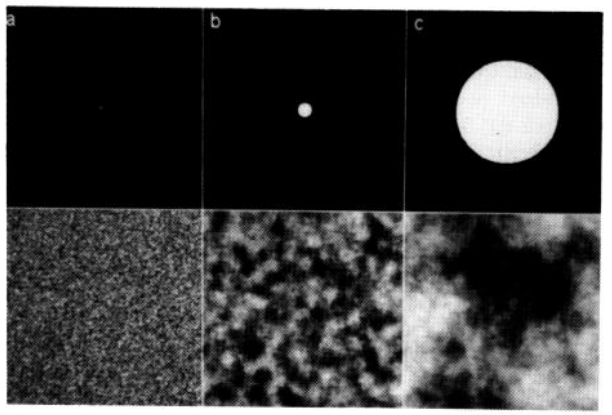

Fig. 2 Different sizes of spot

result is white noise. Large spots degenerate to random faults, so the result will be fractal. Here the texture is shown as a variation in the intensity, which gives a cloud-like result. For intermediate size spots these two effects occur simultaneously. At a large scale the result is white noise, while details have a fractal character.

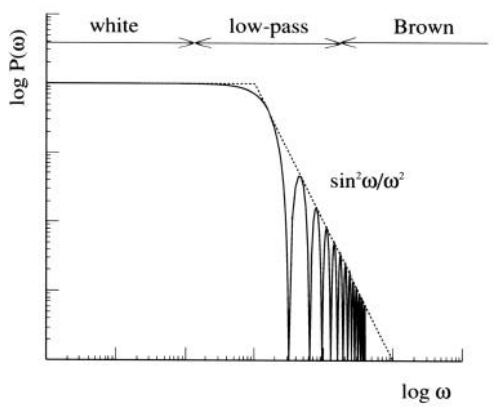

Fig. 3 Power spectrum rectangular pulse

For the autocorrelation function of two-level spots ($h(\mathbf{x}) = 1$ or 0), a simple geometric interpretation can be used: the normalized autocorrelation $K(\Delta)$ ($= C(\Delta)/\sigma^2$) is equal to the area of the overlap of a spot $h(\mathbf{x})$ and a displaced spot $h(\mathbf{x}-\Delta)$, divided by the area of the spot (fig. 4). In general, if two spots do not overlap for a displacement Δ, then the texture is uncorrelated for samples at a distance Δ. Therefore, the size of the spot determines the maximum correlation length or the granularity of the texture.

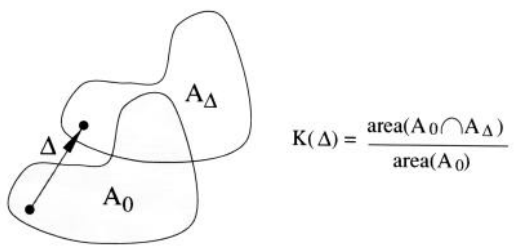

Fig. 4 Autocorrelation function for two-level spots

4.2 Edges

We saw that a sharp edge, i.e. a discontinuity at the transition from the interior to the exterior of the spot, leads to a fractal texture. Figure 5 shows the effect of the use of different types of transitions from the interior to the exterior of the spot. Besides a disk, a cone-shaped spot with a triangular cross-section and a spot with a Gaussian cross-section are used. The last two spots act as steep low-pass filters. The right flank of the power spectrum of a triangular filter falls off with f^{-4}, the power spectrum of a Gaussian filter falls off exponentially. The visual effect is that details below the scale of the spot are removed: a smooth texture is generated. Whereas in image processing usually such filters are preferred above the box or pulse filter, for texture synthesis this is a matter of taste. The smooth textures appear out of focus, whereas textures that result from spots with sharp edges affirm the theorem *fractal = natural*.

The difference between the use of a triangular and a Gaussian cross-section is small, whereas the difference between the smooth and the fractal texture is large. Textures between smooth and fractal can be synthesized via the use of spots with a trapezoidal cross-section.

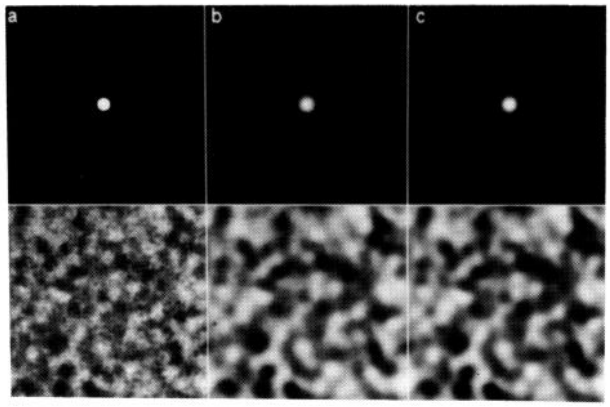

Fig. 5 (a) Disk, (b) cone, (c) Gaussian spot

4.3 Direction

The textures presented so far were invariant under rotation, i.e. isotropic. A texture will be isotropic if the spot is rotationally symmetric, or if each spot is, besides randomly positioned, also randomly rotated. The power spectrum $P_f(\mathbf{k})$ of the noise $f(\mathbf{x})$ that results from the use of a randomly rotated spot $h(x, y)$ with energy spectrum $S_h(k_1, k_2)$ is given by

$$P_f(\mathbf{k}) = \nu < a_i^2 > \frac{1}{2\pi|\mathbf{k}|} \int_0^{2\pi} S_h(|\mathbf{k}|\cos\alpha,\ |\mathbf{k}|\sin\alpha)\, d\alpha\ .$$

If no random rotation is used, the use of straight lines in a spot always leads to an anisotropic texture.

A simple way to generate an anisotropic texture is to scale a spot non-proportionally. Figure 6 shows the effect of the use of ellipses as opposed to disks. For elongated ellipses the texture has a fractal character in the direction of the longest axis, and a white noise character in the direction of the shortest axis. The effect of scaling the spot is not simply scaling the texture. Instead the texture is stretched locally, the large details in the texture remain at the same place.

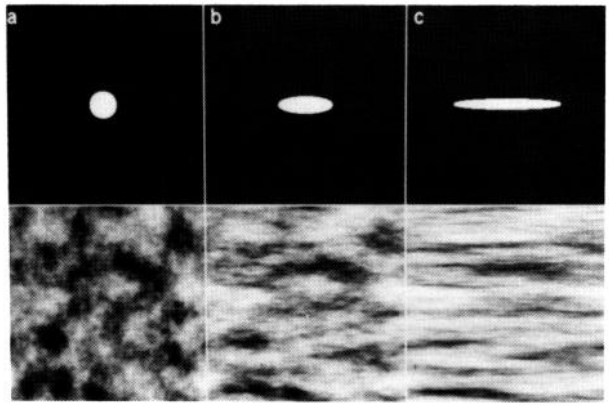

Fig. 6 Non-proportional scaling

4.4 Patterns

Many textures exhibit patterns, i.e. structures are repeated over some distance. Such patterns show up in the autocorrelation function as oscillations with a decreasing magnitude. Figure 7 shows that if the spot exhibits some regular pattern, the resulting texture also does. A spot composed of concentric circles leads to an isotropic, enamel-like texture, the use of a small sample of a grid as a spot leads to a textile-like texture. The corresponding autocorrelation functions are easily imaginable if the rule shown in fig. 4 is used.

Such spots have three levels of detail; each level corresponds to one feature of the corresponding texture. The global size of the spot determines the scale of the white noise component, the width and spacing of the lines determine the width and spacing of the pattern, and the sharp edges lead to fractal detail.

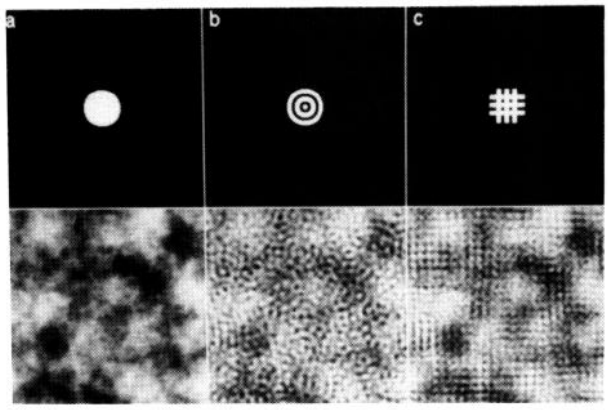

Fig. 7 Regular patterns

4.5 Shape

The preceding examples show that the shape of the spot strongly influences the texture. Some further examples are shown in fig. 8. The use of a square leads to a texture with strong horizontal and vertical patterns (fig. 8a). If the square is distorted into a diamond, the result is easily predictable (fig. 8b). The relation between the shape and the texture is not always so obvious. Fig. 8c shows a spot with the shape of a quarter circle. The resulting texture bears a strong resemblance with a top-view of a planet surface or lunar landscape (without craters).

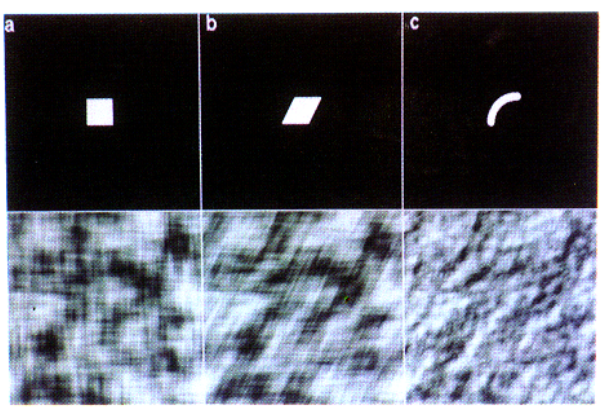

Fig. 8 Different shapes

4.6 Summary

In this section we have shown that the spot is a useful primitive for the design of stationary textures. In general, the relation between features of the spot and features of the texture is straightforward: size — granularity, edge — detail, rotational symmetry — isotropy, etc. Further, the spot is also of a suitable high level. Spots can, for example, be drawn with a simple drawing package. The amount of input is often minimal, because for practical applications often simple spots suffice.

5 APPLICATIONS

5.1 Scalar and vector fields

In this section various applications of variable spot noise are presented. For most images in this section, ellipses are used as spots. These ellipses are rendered via scan-conversion.

Figure 9 shows four examples of the use of texture for the visualization of scalar and vector fields. The colors indicate the value of a scalar field. A scale from saturated blue (negative) via grey to saturated red (positive) is used. The field was constructed by a B-spline approximation of randomly chosen values on a rectangular grid. In fig. 9a the variance of the texture indicates the absolute value. The type of texture indicates the sign: for negative values a ×-shaped spot was used, for positive values a +-shaped spot. In fig. 9b the gradients were emphasized by scaling the variance of the texture proportional to the norm of the gradient. For fig. 9c the scalar field was interpreted as a stream function $\psi(x, y)$, i.e.:

$$v_x = \frac{\partial \psi}{\partial y}, \text{ and } v_y = \frac{-\partial \psi}{\partial x}$$

The flow velocity $\mathbf{v}$ was visualized by using an ellipse shaped spot with the long axis proportional to $|\mathbf{v}|$, the short axis proportional to $1 / |\mathbf{v}|$, and the direction of the longest axis aligned with the direction of the flow. For fig. 9d the same principle was used, but here the scalar field was interpreted as a potential $p(x, y)$ that defines a vector field $\mathbf{v}(x, y)$, i.e.:

$$v_x = \frac{\partial p}{\partial x}, \text{ and } v_y = \frac{\partial p}{\partial y}$$

In these examples texture was used to emphasize some aspect or interpretation of the field in addition to the information provided by the color. The power of the textures can be judged from figure 10, where color is used to visualize a different and unrelated scalar field. For this example the stream function texture (fig. 10c) is the strongest, the global structure of the texture can easily be discerned in spite of the unrelated colors. With the other textures this is harder, but at least they still enable the observer to evaluate the value of the function depicted by the texture locally.

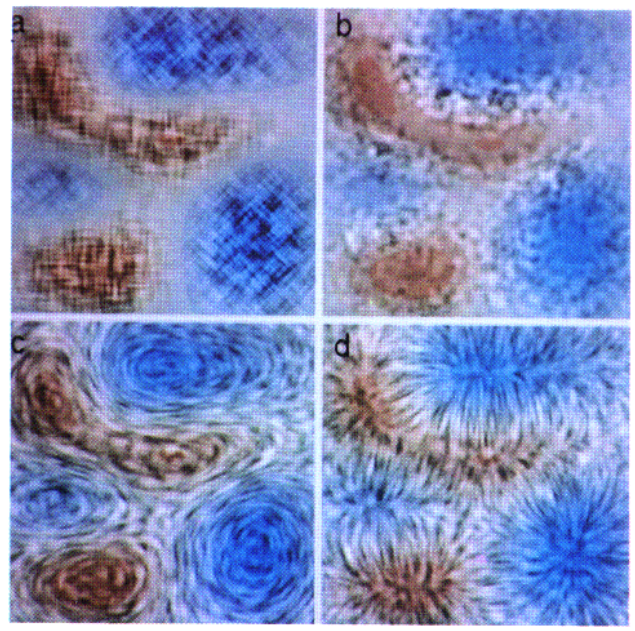

Fig. 9 Color denotes a scalar field. Texture is used to visualize attributes and interpretations of this field: (a) value, (b) gradients, (c) flow, (d) velocity potential.

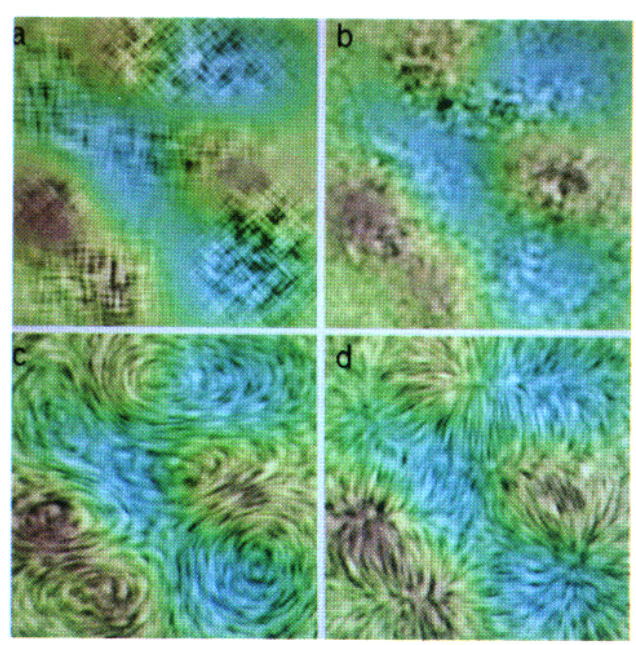

Fig. 10 Same texture as in fig. 9, color denotes an unrelated field.

5.2 Flow visualization

The preceding example shows how spot noise can be used for capturing flow in still images. However, animations give a better impression of dynamic phenomena such as flow. In [38] a technique is presented for the animation of stationary two-

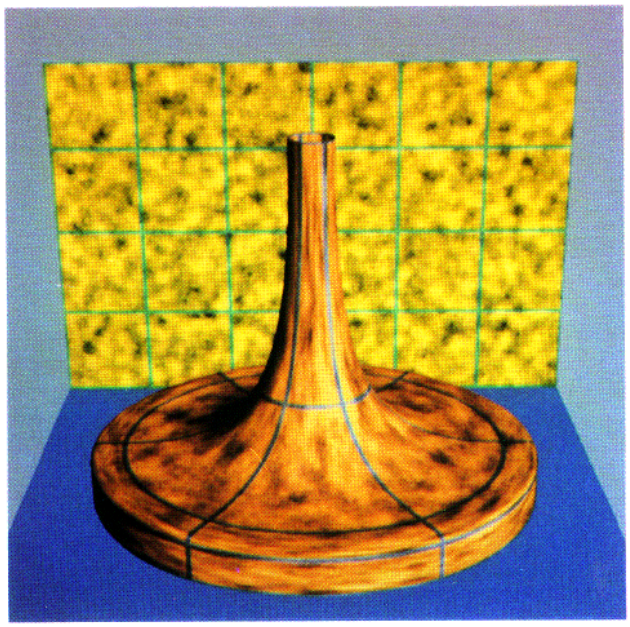

Fig. 11 Mapping of isotropic texture leads to distortion

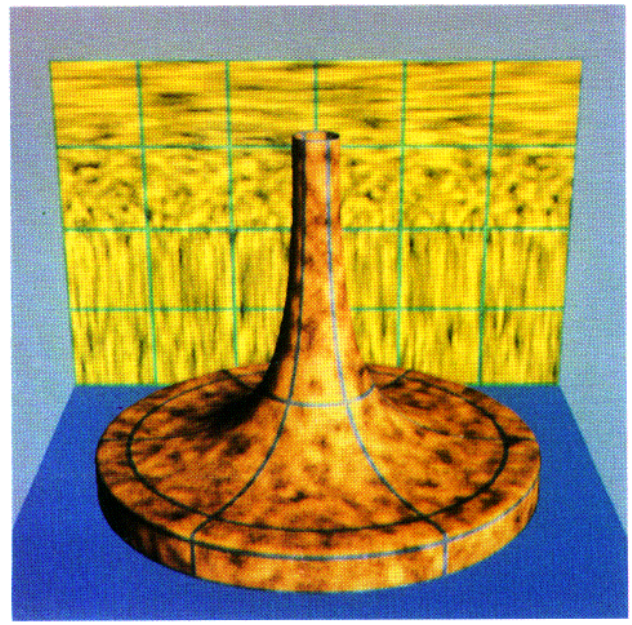

Fig. 12 Mapping of distorted texture

dimensional flow, based on particles and cyclic display. The particles used were point-shaped and no motion blur was applied, so that the flow was visualized as a noisy, moving texture. The same technique can easily be used with spot noise. Hereto we redefine the spots as moving particles. During a lifetime T, we let each spot glow up and decay, while it moves over a small distance. Besides a random position and a random maximal intensity, each spot also has a random time at which it starts its cycle.

Various shapes of spots can be used. The use of circles gives isotropic textures. If ellipses are used, two options are open. If the longest axis of the ellipse aligns with the direction of the flow, the visual effect is that of motion blurred particles. If the longest axis of the ellipse is perpendicular to the direction of the flow, the visual effect is that of short-crested waves, perpendicular to the flow direction.

5.3 Texture mapping on parametric surfaces

The preceding examples were two-dimensional. However, in many cases the data have to be visualized over curved surfaces in space. Typical applications are found in computational fluid dynamics. This requires the mapping of a texture on a surface. Texture mapping was introduced in 1974 by Catmull [2], and since then many techniques and refinements have been developed. For an overview see [12].

Figure 11 shows a problem of standard texture mapping techniques: the isotropic texture (yellow) is distorted when it is mapped onto a parametric surface (orange). The solid texturing technique [25, 28] does not suffer from these artifacts. In section 3.2 we have shown how spot noise can be generated with this method. This technique has some disadvantages however. The proposed convolution is not very efficient, and for animations the texture has to be calculated anew for each frame.

An alternative solution, similar to the one proposed in [20], is shown in figure 12. Here the texture is distorted in texture space (u, v), so that when it is mapped onto the parametric surface, the resulting texture is stationary in object space (x, y, z). Spot noise lends itself very well to this kind of local distortions.

Figure 13 shows a practical application of the techniques presented : the visualization of the flow around a ship calculated by the DAWSON package of MARIN [29]. The colors and the white contour lines indicate the hydrodynamic pressure on the ship hull: red denotes high, and blue denotes low pressure. The shape of the ship is visualized via shading and black equidistant cross-section lines. The technique used for rendering the lines with constant width in object space is described in [37]. The flow velocity on the hull is visualized via spot noise. Ellipses were used as spots, with the longest axis aligned with the flow direction. Texture distortion was used to compensate for the distortion due to the parametrization of the surface.

5.4 Image synthesis

Figure 14 shows an application that lies in between scientific visualization and realistic image synthesis. It is a frame of an animation of the dynamics of a ship in sea waves. For the modeling of the dynamics of large ships, only waves with a long wavelength (50 m and more) are of interest [4]. Further, the steepness of such waves is small, compared to shorter waves. The straightforward visualization of the results of those simulations with flat shading gives disappointing results. First, the shape of the sea surface is hard to grasp, and second, the sea surface does not look realistic, which is important when the results are presented to principals. Both problems could be solved with spot noise. Here bump mapping [1] was used for mapping the texture onto the surface. The amplitude of the spots was varied via a tent function of time to obtain a turbulent sea surface. A risk of polishing scientific results in this way, is that it can add false information. Therefore, the position of the spots was fixed, and the size of the spots was chosen such that the texture does not interfere with the results of the simulation.

For the rendering of waves, many techniques have been proposed that are more accurate and more elaborate [22, 8, 26]. The simple technique used here, however, serves its purpose in the sense that a fairly realistic result is achieved, and that the shape of the waves is presented more clearly.

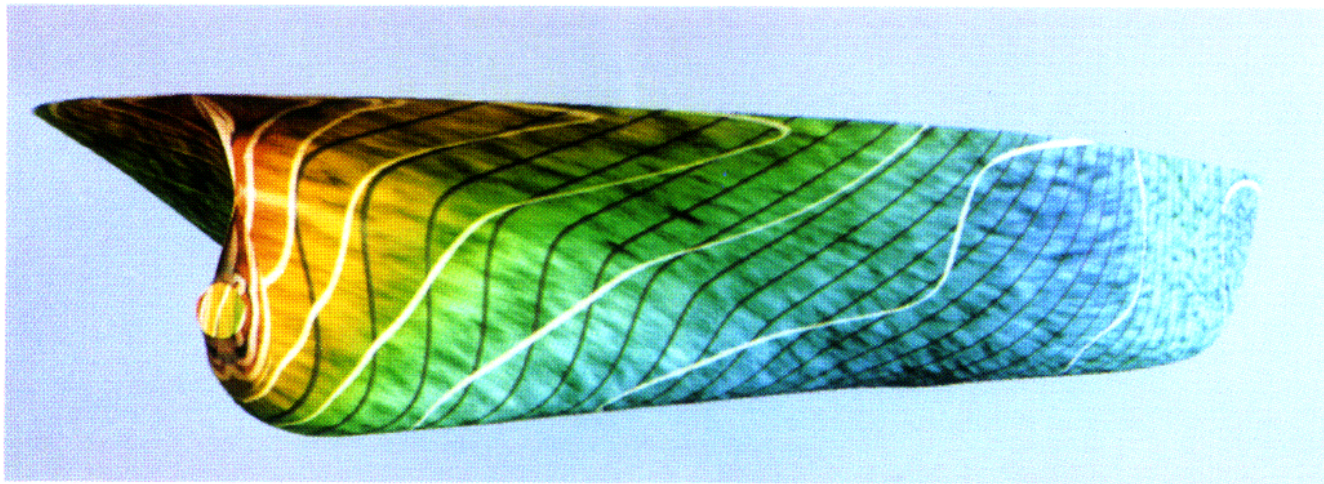

Fig. 13 Visualization of velocity and pressure on ship hull. Texture indicates flow velocity, white contour lines and color indicate pressure, black lines are cross-section lines. *Courtesy data: H.C. Raven, MARIN*

Figure 15 shows an application of space-variant spot noise that is not related to scientific visualization. First, a low-resolution letter was drawn with a painting package. Next, a height field was constructed by approximation of the values in the bitmap with a B-spline surface. Spot noise was used to give the impression of distorted material. The texture was generated with the same technique as used for fig. 9c, and bump mapping was used for mapping the texture onto the surface.

The use of texture in image synthesis leads to much more realistic images, compared to the use of uniform surfaces without detail. Images such as fig. 15 suggest that this statement can be transposed: local variation in texture, as opposed to uniform texture, can lead to even more realistic results.

6 DISCUSSION

6.1 Relations with other work

Throughout this paper, several relations of spot noise with other techniques were mentioned. Here they are discussed in order.

First, spots can be viewed as degenerated *random faults*. A disadvantage of the use of random faults is its poor efficiency. If smaller shapes are used, this disadvantage disappears, but also the fractal property at all scales disappears. However, textures do not necessarily have to be fractal to be useful and interesting.

Second, spots can be viewed as *filter kernels* that are applied to white noise. A good example is fig. 8 in [5], where the effect

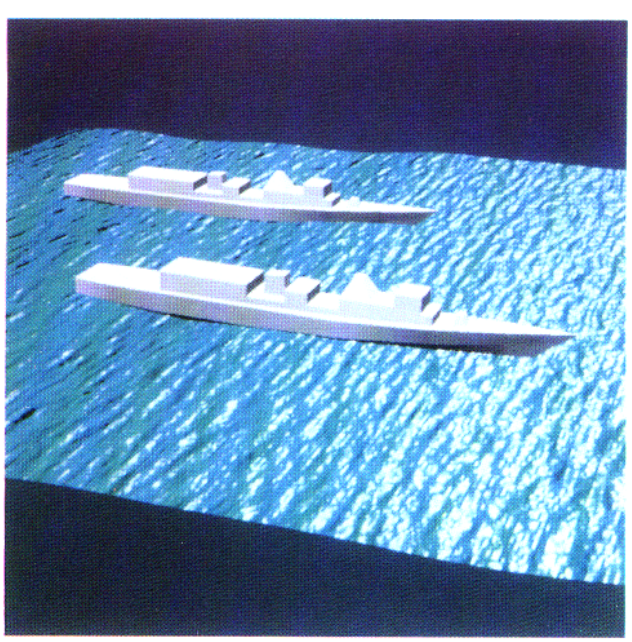

Fig. 14 Visualization of ship dynamics
Courtesy data: R. Dallinga, MARIN

Fig. 15 P.

of an adaptive filter width on white noise is shown in the context of stochastic sampling. For such applications, the aim is removal of high frequency noise, so the filter kernels are chosen from a standard set of rotationally symmetric filter kernels like the tent, the Gaussian, and the squared cosine. The observation that the use of a banana-shaped filter leads to lunar landscapes has not yet been mentioned in image processing literature.

Third, spot noise synthesis can be considered as *sparse convolution*. In contrast to using a texture sample [17], however, a spot is used, i.e. a simple geometric pattern. The examples in section 4 show that such a simple pattern is easier to use for texture design than the autocorrelation function and the power spectrum.

Fourth, spots can be viewed as *particles*, or brush strokes, and spot noise as the result of particle systems [30, 31, 32], or abstract image representations [11]. Although it has been noted before that the use of many overlapping particles leads to texture, this has not yet been analyzed in the frequency domain.

Fifth, spot noise can be viewed as an application of Perlin's *solid texturing* technique [28]. Perlin typically uses scaling of *Noise* to achieve certain effects, here it has been shown that for controlled, local variations convolution has to be used. Another relation between the concepts discussed here and *Noise* can be found in [19]. Here sparse convolution is used, i.e. convolution of a filter kernel with a Poisson point process, for the implementation of *Noise* itself.

We conclude from the preceding discussion that spot noise can be considered as a new concept that provides a new perspective on a series of so far unrelated techniques, and that provides an elegant basis for their analysis.

6.2 Texture for data visualization

The application aimed at for spot noise was texture synthesis for data visualization. We therefore test spot noise here against the requirements of section 2. Spot noise does allow easy local control, and a wide variety of textures can be synthesized. The specification of a texture by the designer requires few inputs, and in most cases the relation between his input and the resulting texture is straightforward. The synthesis process is reasonably efficient. Previews can easily be generated by using few spots, so that the separate spots can be distinguished. For images such as fig. 13, the use of previews, where the object of interest is covered with iconic representations of the texture, is very useful to establish the data mapping from velocity to spot.

Another point is whether the use of texture for data visualization itself is a useful concept. The use of texture means that resolution is sacrificed, i.e. the largest scale of the texture has to be smaller than the scale of variation in the data. However, in contrast to the use of color alone, it does allow the display of vector fields, and has more degrees of freedom. Further, the examples for flow visualization show that the resulting images are suggestive, if not natural and realistic. Thus, texture is probably more suited for global and qualitative visualization of data than for detailed and quantitative analysis, and it is more suited for external presentations than for regular use by researchers.

Spot noise was used here in its simplest form: straightforward mapping of intensity values. Several techniques can be used to enhance the results. The mapping of the data to the parameters of the texture used here was continuous, an alternative is to use discrete bands or bins. Non-linear transformations of the intensity values of the texture can be applied to achieve special effects. Examples are squaring the value of texture, clamping, and the use of the value of the texture as an index in a color table. Finally, besides intensity, also the hue and saturation of the spots can be varied as a function of the data.

6.3 Further work

An approach to gain more insight in the relation between the shape of the spot and the resulting texture, is to attempt to derive spots from sampled real-world textures. This step is the inverse of that from spot to texture. A spot $h(\mathbf{x})$ has to be constructed such that its energy spectrum is the same as the power spectrum of the texture, and such that it corresponds to the notion of the spot used here, i.e. satisfies some criterion such as minimal size or minimal variance.

If such a technique can be developed, the application of the spot might expand from texture design to texture analysis. It is an open question whether such derived spots provide additional insight above the autocorrelation function and the power spectrum.

7 CONCLUSIONS

— Texture is a useful visual primitive for data visualization;

— Spot noise satisfies the requirements for texture for data visualization: efficient synthesis with local control, and ease of design;

— Spot noise is an alternative to solid texturing for the synthesis of stochastic textures over curved surfaces;

— Spot noise provides a new perspective on a series of techniques: random faults, filtering, sparse convolution, particle systems, and solid texturing;

— Spot noise is a hot noise.

ACKNOWLEDGEMENTS

The discussions with Teun Burgers, Wim Rijnsburger (ECN), and Pieter-Jan Stappers (Delft University of Technology — DUT) were very helpful during the development of the work described here. Valuable comments on earlier versions of this paper were given by Wim Bronsvoort (DUT), and Gonno Leendertse (ECN). I further thank Hoyte Raven, Reint Dallinga, René Huijsmans, and Hans van der Kam (Maritime Research Institute Netherlands — MARIN) for providing interesting data sets, for the pleasant cooperation, and for their support of the ECN Scientific Visualization Project.

REFERENCES

1. BLINN, J.F. Simulation of wrinkled surfaces. *Computer graphics 12*, 3, (1978), 286-292.
2. CATMULL, E. *A subdivision algorithm for computer display of curved surfaces*. Ph.D. Thesis, Report UTEC-CSc-74-133, Computer Science Department, University of Utah, Salt Lake City, 1974.
3. CHAMPENEY, D.C. *Fourier transforms and their physical applications*. Academic Press, London, 1973.
4. DALLINGA, R. Seakeeping characteristics of SWATH vessels. In *Proceedings 13th WEGEMT Graduate school on design techniques for advanced marine vehicles and high speed displacement ships*, Delft University of Technology, 1989.
5. DIPPE, M.A.Z., AND WOLD, E.H. Anti-aliasing through stochastic sampling. *Computer Graphics 19*, 3 (1985), 69-78.
6. FOLEY, J.D., DAM, A. VAN, FEINER, S.K. AND HUGHES, J.F. *Computer graphics: principles and practice*. Second edition, Addison-Wesley, Reading, MA, 1990.

7. FOURNIER A., FUSSEL, D. AND CARPENTER, L. Computer rendering of stochastic models. *Communications ACM 25*, 6 (1982), 371-384.

8. FOURNIER, A., AND REEVES, W.T. A simple model of ocean waves. *Computer Graphics 20*, 4 (1986), 75-84.

9. GAGALOWICZ, A., AND MA, S.D. Sequential synthesis of natural textures. *Computer Graphics, Vision, and Image Processing 30* (1985), 289-315.

10. GONZALEZ, R., AND WINTZ P. *Digital image processing.* Second edition, Addison-Wesley, Reading, MA, 1987.

11. HAEBERLI, P. Paint by numbers: abstract image representations. *Computer Graphics 24*, 4 (1990), 207-214.

12. HECKBERT, P.S. Survey of texture mapping. *IEEE Computer Graphics and Applications 6*, 11 (1986), 56-67.

13. JANSEN, F.W., AND WIJK, J.J. VAN. Previewing techniques in raster graphics. *Computer & Graphics 8*, 2 (1984), 149-161.

14. JULESZ, B. Visual pattern discrimination. *IRE Trans. Inform. Theory*, IT-8 (1962), 84-92.

15. KRUEGER, W. Intensity fluctuations and natural texturing. *Computer Graphics 22*, 4 (1988), 213-220.

16. KRUEGER, W. Volume rendering and data feature enhancement. In Grave, M., and Y. le Lous (eds.), *Proceedings of the Eurographics Workshop on Visualization in Scientific Computing*, to be published by Springer-Verlag, Berlin.

17. LEWIS, J.P. Texture synthesis for digital painting. *Computer Graphics 18*, 3 (1984), 245-252.

18. LEWIS, J.P. Generalized stochastic subdivision. *ACM Transactions on Graphics 6*, 3 (1987), 167-190.

19. LEWIS, J.P. Algorithms for solid noise synthesis. *Computer Graphics 23*, 3 (1989), 263-270.

20. MA, S.D., AND GAGALOWICZ, A. Determination of local coordinate systems for texture synthesis for 3-D surfaces. In Vandoni, C.E. (ed.), *Proceedings Eurographics'85*, North-Holland, 1985, 109-118.

21. MANDELBROT, B.B. *The fractal geometry of nature.* W.H. Freeman and Co., New York, 1982.

22. MAX, N. Vectorized procedural models for natural terrains: waves and islands in the sunset. *Computer Graphics 15*, 3 (1981), 317-324.

23. MONNE, J., SCHMITT, F. AND MASSALOUX, D. Bidimensional texture synthesis by Markov chains. *Computer Graphics and Image Processing 17* (1981), 1-23.

24. MUSGRAVE, F.K., KOLB, C.E. AND MACE, R.S. The synthesis and rendering of eroded fractal terrains. *Computer Graphics 23*, 3 (1989), 41-50.

25. PEACHEY, D.R. Solid texturing of complex surfaces. *Computer Graphics 19*, 3 (1985), 279-286.

26. PEACHEY, D.R. Modeling waves and surf. *Computer Graphics 20*, 4 (1986), 65-74.

27. PEITGEN, H.-O., AND SAUPE, D. (eds.). *The science of fractal images.* Springer-Verlag, New York, 1988.

28. PERLIN, K. An image synthesizer. *Computer Graphics 19*, 3 (1985), 287-296.

29. RAVEN, H.C. Variations on a theme by Dawson. In *Proceedings of the 17th Symposium on Naval Hydrodynamics*, The Hague, 1988, 151-172.

30. REEVES, W.T. Particle systems - a technique for modeling a class of fuzzy objects. *Computer Graphics 17*, 3 (1983), 389-399.

31. REEVES, W.T., AND BLAU, R. Approximate and probabilistic algorithms for shading and rendering structured particle systems. *Computer Graphics 19*, 3 (1985), 313-322.

32. SIMS, K. Particle animation and rendering using data parallel computation. *Computer Graphics 24*, 4 (1990), 405-413.

33. TUFTE, E.R. *The visual display of quantitative information.* Graphics Press, Cheshire, Connecticut, 1983.

34. UPSON, C. The visual simulation of amorphous phenomena. *Visual Computer 1*, 2 (1986), 321-326.

35. UPSON, C. ET AL. The Application Visualization System: a computational environment for scientific visualization. *IEEE Computer Graphics and Applications 9*, 4 (1989), 30-42.

36. WIJK, J.J. VAN, BRONSVOORT, W.F., AND JANSEN, F.W. Some issues in designing user interfaces to 3D raster graphics. *Computer Graphics Forum 4* (1985), 5-10.

37. WIJK, J.J. VAN. Rendering lines on curved surfaces. In Grave, M., and Y. le Lous (eds.), *Proceedings of the Eurographics Workshop on Visualization in Scientific Computing*, to be published by Springer-Verlag, Berlin.

38. WIJK, J.J. VAN. A raster graphics approach to flow visualization. In Vandoni, C.E., and D.A. Duce (eds.), *Proceedings Eurographics'90*, North-Holland, Amsterdam, 1990, 251-259.