Algorithms for Solid Noise Synthesis

J. P. Lewis

Computer Graphics Laboratory
New York Institute of Technology

Abstract

A solid noise is a function that defines a random value at each point in space. Solid noises have immediate and powerful applications in surface texturing, stochastic modeling, and the animation of natural phenomena.

Existing solid noise synthesis algorithms are surveyed and two new algorithms are presented. The first uses Wiener interpolation to interpolate random values on a discrete lattice. The second is an efficient sparse convolution algorithm. Both algorithms are developed for *model-directed synthesis*, in which sampling and construction of the noise occur only at points where the noise value is required, rather than over a regularly sampled region of space. The paper attempts to present the rationale for the selection of these particular algorithms.

The new algorithms have advantages of efficiency, improved control over the noise power spectrum, and the absence of artifacts. The convolution algorithm additionally allows quality to be traded for efficiency without introducing obvious deterministic effects. The algorithms are particularly suitable for applications where high-quality solid noises are required. Several sample applications in stochastic modeling and solid texturing are shown.

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, shadowing, and texture*.
General Terms: Algorithms, Graphics.
Additional Key Words and Phrases: Solid noise, texture, stochastic modeling, simulation of natural phenomena, texture synthesis, fractals.

1 INTRODUCTION

A solid noise is a random-valued function $f : R^3 \to R$. "Noise" is used to denote a random function with some known statistical properties. Solid noises are a subset of the concept of *solid textures* introduced in computer graphics by Perlin [17,18] and Peachy [16]).

Solid noises have been used for texturing three-dimensional objects by assigning the color at a visible point on the surface as a function of the noise value at that point in space. In this role, solid textures have several advantages over conventional texture mapping:

- Surfaces with Gaussian curvature can be textured homogeneously, without distortions such as poles that occur in texture mapping.
- The spatial nature of the noise correlation makes possible certain effects which would be difficult with texture mapping, for example, the "carved out of" effect [18] which uses the fact that noise features (e.g. veins in simulated rock) can cross overhangs in the object (Fig. 7).

Solid noises also have many potential applications in describing complex/irregular forms or movement; a few possibilities are shown in Section 5 of this paper.

2 SOLID NOISE ALGORITHMS

In all applications, it is desirable that a solid noise algorithm be controllable and free of artifacts. Consistent with recent work [18,9,11], the noise power spectrum is considered as a reasonably powerful and intuitive framework for developing control over the noise.

When the noise is used for surface texturing, efficiency is a major consideration, since the three-dimensional variants of even simple computations such as linear interpolation are fairly expensive when the computation is required at each pixel. It is desirable that the noise synthesis algorithm allow quality to be traded for efficiency where appropriate, e.g. for previewing or for background objects which do not need a high-quality noise. For most applications it is probably preferable to trade some control for efficiency rather than adopting an efficient method that has intrinsic artifacts.

For animation applications the solid noise should also be bandlimited. Although the aliasing of an improperly sampled noise function will often not be objectionable in a still picture (due to the same principle evident in stochastic sampling [5] - objectionable Moire patterns result from the structured sampling of a structured signal), the same aliased noise used in an animation will typically produce characteristic "shimmering" or "bubbling" aliasing effects.

While a large variety of particular three-dimensional random-valued functions are conceivable, most can be decomposed into a basic noise source and some functional or procedural transformation of this noise. As argued by Perlin [18], the noise source should be a controllable "primitive" that allows the user to define various ad hoc solid noise functions in terms of this noise primitive.

This paper presents two algorithms for the synthesis of high-quality solid noises with control of the noise power spectrum and (optionally) distribution functions. Considerations that lead to the selection of these particular algorithms are also described.

2.1 MODEL-DIRECTED SYNTHESIS

Although the linear filtering algorithms for obtaining noises having desired power spectra are well understood [13], these algorithms

```
#define RANTABLEN /* something prime */
float Rantab[RANTABLEN];
int Indx[ILEN],Indy[ILEN],Indz[ILEN];

float hash3(float x,y,z)
{
  int i = HASH( Indx[LOWBITS(x)], Indy[LOWBITS(y)], Indz[LOWBITS(z)] );
  return( Rantab[i % RANTABLEN] );
}
```

Fig. 1: Pseudocode for the lattice white noise function.

are not ideally suited to the requirements of computer graphic modeling and rendering. In particular, in place of the regular and ordered sampling that is fundamental to digital signal processing we require a *model-directed synthesis*, in which the noise function is constructed only at particular points determined by the object model, and in an order that may depend on the model or the viewpoint. In a texturing application, these points are the points on the object's surface that project without occlusion to pixels in a perspective projection of the object. Similarly, in a modeling application the noise may be constructed at a limited and irregular set of points, e.g., the vertices of a polygonal model.

Digital filters assume regular sampling and spatial or temporal ordering (causality) of the input signal and consequently cannot meaningfully operate at isolated points in space. The direct application of a digital filtering approach for solid noise synthesis would thus result in a solid region of filtered noise enclosing the points of interest. This is very costly in terms of storage, since the storage size of a solid noise varies with the cube of the resolution. The direct FFT or digital filter synthesis of a medium- or high-resolution solid noise is usually impractical in this respect. Also it would seem inefficient to construct the noise over a solid region when it is only needed at isolated points, though this may depend on the number of points required and on the respective algorithms. A third drawback of digital filtering approaches is that since the noise is sampled it needs to be interpolated from the sampling lattice to the locations of interest.

Model-directed synthesis can be achieved by constructing the input noise signal as needed at synthesis time, and by employing an acausal and metaphorically continuous rather than sampled filtering approach. Since the spatial ordering of the synthesis is unknown, particular regions may be visited multiple times. The input noise construction must be *internally consistent* (in the terminology of [8]): independent constructions of a particular point must produce the same value. While model-directed synthesis approaches are suited for many computer graphics problems, it is evident that they cannot easily use the coherence provided by regular sampling and consequently will be more costly than standard filtering approaches for constructing regularly sampled noises.

2.2 LATTICE WHITE NOISE

A consistent uncorrelated ("white") noise can be generated using a hash-like pseudo-random function of the mantissa bits of the location coordinates x, y, z. One such function was described in [4]. A variation of this function uses the low-order bits of each coordinate (scaled suitably) to index a corresponding randomly permuted 'indirection table' of indices into a second table of uncorrelated random values with the desired probability density. The three resulting indexes are hashed to form an index into the prime-length random value table (Fig. 1).

The `HASH3` function generates an uncorrelated periodic noise, with the period determined by the number of coordinate bits which are retained. The function takes on new values only on the lattice defined by the low bits of the coordinate mantissas, thus "lattice noise". For most purposes it will be necessary to interpolate or filter the noise values on the lattice to obtain a continuous and correlated solid noise.

2.3 PERLIN ALGORITHM

Perlin [17,18] outlined a model-directed solid noise algorithm based on interpolating a location hashing function such as `HASH3`. The resulting noise is employed as a spectral basis function, with a desired noise $\hat{\eta}$ being approximated by a weighted sum of basis noises η_k at different scales:

$$\hat{\eta}(\boldsymbol{\rho}) = \sum a_k \eta_k(k\boldsymbol{\rho}) \tag{1}$$

The characteristics of the resulting noise are determined by the selected interpolation approach. The cubic polynomial interpolation suggested in [18] has certain disadvantages. Interpolation in several dimensions using the separable tensor product of a one-dimensional interpolation scheme results in preferred directions along the coordinate system axes; this artifact can only be avoided by using an intrinsically multi-dimensional interpolation approach. Cubic polynomial interpolation in three dimensions is also quite expensive. The direct tensor product scheme for cubic spline interpolation requires a support of $4^3 = 64$ points as well as $16(x)+4(y)+1(z) = 21$ spline evaluations [2]. The interpolation must be repeated for each basis function in the spectral summation (1).

One popular implementation of Perlin's approach employs Hermite interpolation, using the lattice noise to define gradients at the (eight) nearest-neighbor points on the lattice [10]. These values are separably interpolated using a cosine-like function. While this approach is considerably more efficient than a cubic spline interpolation, it has stronger directional artifacts (Fig. 2). Another drawback is that the noise value and second-derivative are both zero at the lattice points. The directional trends and regularly spaced zeros are visible (e.g. see Fig. 3), though it may be possible to disguise them through application of the summation (1).

In order to be approximately orthogonal, candidate spectral basis functions should be zero beyond a particular range of frequencies ("bandpass"). Approaches which use standard (meaning non-oscillatory, energy minimizing, spline-like) interpolation methods to interpolate an uncorrelated noise lattice produce a low-pass rather than a band-pass random function, however, since they do not remove or attenuate the low frequency portion of the original noise power spectrum (which has equal expected power at all frequencies) (Fig. 4.) This can be seen in part by considering the zero frequency: interpolation will not remove the mean. From a signal-processing viewpoint, standard interpolation methods have the effect of attenuating high frequencies [19] (also see problem 26 in [3]).

One misinterpretation of the summation (1) is that, by analogy with Fourier summation, a bandpass noise might be produced using a lowpass noise primitive by subtracting a more bandlimited noise from a given noise, in the hope of removing the low-frequency portion of the spectrum. Power spectra of mutually

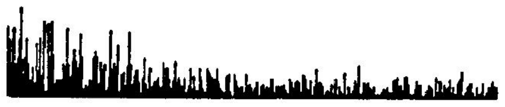

Fig. 4: Computed amplitude spectrum (zero frequency at left) of a long one-dimensional section of the noise shown in Fig. 2. The spectrum is not bandpass.

uncorrelated noises cannot be meaningfully subtracted however: $S(\eta_1 - \eta_2) = S(\eta_1) - 2C(\eta_1, \eta_2) + S(\eta_2)$where $C(\eta_1, \eta_2) = 0$ is the covariance between the noises. Eq. (1) is a spectral summation but not a Fourier summation.

The high-frequency (amplitude) spectrum of the Perlin basis noise using cubic interpolation falls off as λ^{-4} since the amplitude spectral envelope of a C_n function is λ^{-n-2} [3]. This also is not ideally bandlimited. We conclude that polynomially interpolated noise does not provide an ideal spectral basis.

The various disadvantages of polynomial interpolation for a spectral synthesis approach are avoided in the Wiener interpolation algorithm presented in section 3 below.

2.4 GARDNER AND PEACHY ALGORITHMS

Gardner [9] developed a naturalistic texturing function based on a modified Fourier series. This approach is unusual in its use of a conceptually *deterministic* function to simulate irregular texture.[1] The function appears as a product of two one-dimensional series in x and y (as described it is two-dimensional but an equivalent three-dimensional function can be formulated). A *separable* function $f(u, v) = f_u(u) f_v(v)$ has strong directional artifacts that make it unsuitable for simulating a naturalistic texture even if the component functions f_u, f_v are characteristic sections of the desired texture. Gardner overcame this ‘checkerboard effect’ by coupling the phases of each term in the u-series to v, and conversely. The resulting texture is not separable and is sufficiently complex that it mimics a random texture when applied carefully.

The spectrum of the Gardner function has not been analyzed but is not (as might be supposed) directly defined by the Fourier series coefficients. This can be seen by considering the kth term of the u-series evaluated along a diagonal profile with u and v varying:

$$f_k(u) = a_k \sin(\omega_k u + I \sin(\omega_{k-1} v))$$

This is a form of frequency modulation. From [1]

$$\sin(\theta + I \sin\beta) =$$
$$J_0(I) \sin\theta + \sum_{k=1}^{\infty} J_k(I) \left\{ \sin(\theta + k\beta) + (-1)^k \sin(\theta - k\beta) \right\}$$

so although the Gardner texturing function has a line spectrum, it is more complex than suggested by its Fourier series resemblance (also it is evident that it is not strictly bandlimited).

Peachy [16] proposed solid function generation by the composition (e.g. sum or product) of several lower-dimensional functions. If the functions are random the result is a solid noise. As in the Gardner algorithm, the composition function can be designed to eliminate separability but the absence of an intrinsically three-dimensional correlation structure may be visually evident.

[1]While the computer implementation of any random process is necessarily deterministic, there is a practical as well as a conceptual difference, in that the period of an n-term Fourier series is $2n$ samples whereas the period of a simulated random process is usually considerably larger, as determined by the period of the pseudo-random number function.

3 WIENER INTERPOLATION ALGORITHM

Wiener interpolation differs from other interpolation approaches in that it is based on the expected correlation of the interpolated function. Since the autocorrelation or autocovariance function is equivalent information to the power spectrum, Wiener interpolation is particularly suited for noise synthesis where control of the noise correlation and spectrum is required. Control of the noise spectrum is intrinsic to Wiener interpolation, so problems with band-limiting and the expensive spectral summation (1) are avoided.

Wiener interpolation has many other potential applications in computer graphics (e.g., as the basis for an improved stochastic subdivision method [11], or possibly as an approach to resampling stochastically sampled images for display). Some additional characteristics and advantages of Wiener interpolation are:

- The data can be arbitrarily spaced.
- The algorithm applies without modification to multi-dimensional data.
- Wiener interpolation of discrete data is simple, requiring only the solution of a linear equation.
- In an estimation application the algorithm provides an error or confidence level associated with each point on the interpolated surface.
- The algorithm is optimal by a particular criterion (see below) which may or may not be relevant.
- The interpolation can be made local or global to the extent desired. This is achieved by adjusting the covariance function so that points beyond a desired distance have a negligible correlation.
- The interpolation can be as smooth as desired, for example, an analytic covariance function will result in an analytic interpolated curve or surface.
- The interpolation need not be “smooth”, for example, the correlation can be negative at certain distances, oscillatory, or (in several dimensions) have directional preferences.

(The last three properties result from the direct spectral control provided by Wiener interpolation.)

There are a number of formulations and variations of Wiener interpolation [20,7]. A simple probabilistic formulation suitable for solid noise interpolation will be used here. The description requires two concepts from probability:

- The correlation of two random variables is the expectation of their product, $\mathbf{E}[xy]$. The autocorrelation or autocovariance function of a random process (noise) is the correlation of pairs of points from the process:

  $$C(t_1, t_2) = \mathbf{E}\left[\eta(t_1)\eta(t_2)\right]$$

  For homogeneous noise, this expectation is a function only of the distance between the two points: $C(t, t+\tau) = C(\tau) = \mathbf{E}[\eta(t)\eta(t+\tau)]$. The variance is the value of the autocovariance function at zero. (Auto)covariance refers to the correlation of a process whose mean is removed and (usually) whose variance is normalized to be one.

- Expectation behaves as a linear operator, so any factor or term which is known can be moved "outside" the expectation. For example, assuming a and b are known,

$$\mathbf{E}\{a\eta + b\} = a\mathbf{E}[\eta] + b$$

Also, the order of differentiation and expectation can be interchanged, etc.

Wiener interpolation estimates the value $\hat{\eta}$ of the process η at a particular location as a weighted sum of the values η_j observed at some number of other locations:

$$\hat{\eta} = \sum a_j \eta_j \tag{2}$$

The weights a_j are chosen to minimize the expected squared difference or error between the estimate and the value of the "real" process at the same location:

$$\mathbf{E}\left\{(\eta - \hat{\eta})^2\right\} \tag{3}$$

The reference to the "real" process in (3) seems troublesome because the real process may be unknowable at the particular location, but since it is the *expected* error which is minimized, this reference disappears in the solution.

Wiener interpolation is optimal among linear interpolation schemes in that it minimizes the expected squared error (3). When the data have jointly Gaussian probability distributions (and thus are indistinguishable from a realization of a Gaussian stochastic process), Wiener interpolation is also optimal among nonlinear interpolation schemes.

3.1 DERIVATION

By the *orthogonality principle* [21,14], the squared error of a linear estimator is minimum when the error is orthogonal in expectation to all of the known data, with "orthogonal" meaning that the expectation of the product of the data and the error is zero:

$$\mathbf{E}\{(\eta - \hat{\eta})\eta_k\} = 0 \text{ for all } k$$

Substituting $\hat{\eta}$ from (2),

$$\mathbf{E}\left\{(\eta - \sum a_j \eta_j)\eta_k\right\} = 0 \tag{4}$$

$$\mathbf{E}\left\{\eta\eta_k - \sum a_j \eta_j \eta_k\right\} = 0$$

The expectation of $\eta\eta_k$ is the correlation $C(t - t_k)$, and likewise for $\eta_j\eta_k$, so:

$$C(t - t_k) = \sum a_j C(t_j - t_k)$$

or

$$\mathbf{Ca} = \mathbf{c} \tag{5}$$

This equation can be solved for the coefficients a_j. The coefficients depend on the positions of the data η_j through the covariance function, but not on the actual data values; the values appear in the interpolation (2) though. Also, (5) does not directly involve the dimensionality of the data. The only difference for multi-dimensional data is that the covariance is a function of several arguments: $\mathbf{E}[pq] = C(x_p - x_q, y_p - y_q, z_p - z_q, \ldots)$.

3.2 COST

From (5) and (2), the coefficients a_j are $\mathbf{a} = \mathbf{C}^{-1}\mathbf{c}$, and the estimate is $\hat{\eta} = \eta^t \mathbf{C}^{-1}\mathbf{c}$. The vector $\mathbf{c}$ changes from point to point, but $\eta^t \mathbf{C}^{-1}$ is constant for given data, so the interpolation cost is a dot product

$$\hat{\eta} = < \eta^t \mathbf{C}^{-1}, \mathbf{c} > \tag{6}$$

of two vectors whose size is the number of data points.

3.3 EVALUATION

The spectral definition possible in Wiener interpolation is proportional to the number of data points considered in the interpolation. For simple spectra requiring a small neighborhood of points (e.g. 3^3 or 4^3 points), the computation (6) appears to be considerably more efficient than polynomial spline interpolation. A fair degree of spectral control can be achieved if larger neighborhoods are used, for example, oscillatory (bandpass) noises are possible. The expensive spectral summation (1) is also avoided

The disadvantage of this algorithm is intrinsic to the approach of *interpolating* an uncorrelated noise lattice: the three-dimensional covariance function, centered at a noise lattice point, should strictly be zero when sampled at any other lattice point, since these points are not correlated. For an isotropic covariance, this requires that $C_1(\tau) = 0$ at distances $\tau = \delta\sqrt{i^2 + j^2 + k^2}$ for all lattice offsets $i + j + k >= 1$ (δ is the lattice spacing). The covariance structure is thus artificially constrained. If the specified covariance does not satisfy this constraint, the interpolation error (confidence measure) will be non-zero and the realized covariance will be somewhat different than that specified.

4 SPARSE CONVOLUTION ALGORITHM

A second algorithm avoids the covariance function constraints of the noise lattice interpolation approach but retains the direct spectral control of the Wiener interpolation approach. In addition, it has the advantage of conceptual simplicity.

In this algorithm a three-dimensional noise is synthesized by the convolution of a three-dimensional kernel $h(\boldsymbol{\rho})$ with a Poisson noise process γ

$$\eta(\boldsymbol{\rho}) = \int_{R^3} \gamma(\boldsymbol{\sigma}) h(\boldsymbol{\rho} - \boldsymbol{\sigma}) d\boldsymbol{\sigma} \tag{7}$$

The Poisson process consists of impulses of uncorrelated intensity distributed at uncorrelated locations in space:

$$\gamma(\boldsymbol{\rho}) = \sum a_k \delta(\boldsymbol{\rho} - \boldsymbol{\rho}_k)$$

($\boldsymbol{\rho}_k$ is the location of the kth impulse). This is a 'sparse' form of white noise, hence "sparse convolution".

The power spectrum S_y at the output of a linear time-invariant filter (expressible as a convolution) is related to the input spectrum S_x by [21]

$$S_y(\omega) = S_x(\omega) \left|H(j\omega)\right|^2$$

where H is the Fourier transform of the filter impulse response or kernel h. Since γ is uncorrelated its transform is a constant, so the spectrum of a noise synthesized by sparse convolution is simply the (deterministic) spectrum of the kernel, scaled by a constant.

4.1 EFFICIENCY

Sparse convolution has several advantages for digital computation. Because of the impulsive nature of the noise, the convolution integral (7) reduces to a summation over the impulses:

$$\eta(\boldsymbol{\rho}) = \sum a_k h(\boldsymbol{\rho} - \boldsymbol{\rho}_k) \tag{8}$$

Thus, the synthesis is reduced naturally to a computationally realizable form without requiring sampling (and subsequent interpolation) of the noise.

The quality of the noise can be varied as required for the application by varying the density of the Poisson noise. This is an important property, since e.g. background objects or interactive previewing applications may not require full quality noise. A density of less than one impulse per kernel volume produces a “lumpy” noise with little spectral definition. Typical applications require a density of several impulses per kernel volume, and noises produced with a density of 10 or more points per kernel are usually not distinguishable from those produced by convolving with a uniformly sampled (non-sparse) white noise, though the sparse convolution is considerably more efficient.

For an isotropic noise the kernel h is also isotropic and (assuming it is non-zero over a finite radius) can be approximately evaluated by a one-dimensional table lookup. In this case the summation (8) can be restricted to only those impulses γ_k within the kernel radius of the location $\boldsymbol{\rho}$. The problem then is to identify these points efficiently, in particular, without requiring examination of all points and an expensive distance computation requiring a square-root.

This can be accomplished with an appropriate construction of the Poisson process γ. A simple construction is to define a large but finite sampling lattice over the noise domain and approximate the Poisson process by choosing N Poisson-distributed impulses in each voxel. The voxels can then be numbered, and the voxel number serves as a random number generator seed for generating the impulses within that voxel. The lattice spacing and kernel radius are conveniently set to one (with space scaled accordingly). Then the impulses lying within a unit radius of a particular location $\boldsymbol{\rho}$ are those in the voxel containing $\boldsymbol{\rho}$ and in the adjacent voxels. The summation (8) is modified accordingly. Square roots are entirely removed by using the squared distance $|\boldsymbol{\rho} - \boldsymbol{\rho}_k|^2$ to index a pre-warped kernel table $\hat{h}(\tau) = h(\sqrt{\tau})$.

The author’s implementation of sparse convolution uses standard tricks such as fixed-point computation. In addition, the noise impulses $\boldsymbol{\rho}_k$ are stored in a cache array as they are computed, and are reused if the next location falls within the same voxel. With these optimizations, the algorithm using one impulse per voxel is slightly slower than the previously described Hermite implementation of Perlin’s algorithm, but does not have visible artifacts and provides some control over the spectrum.

The upper-left panel in Fig. 5 shows a planar section of a sample texture generated with the sparse convolution algorithm using a smooth cosine kernel $1/2+1/2\cos(\pi\tau),\ |\tau| < 1$.The texture does not reveal the synthesis coordinate system or display other artifacts.

5 APPLICATIONS

A solid noise algorithm is most useful as a primitive in a language that allows one to easily define functional or procedural transformations of the noise. An important characteristic of this language is that it should allow functions to be *dynamically defined* at modeling/rendering/animation time – the “user” should have the freedom to define an ad hoc function in the model, rather than requiring the original programmer of the graphics system to anticipate and implement libraries of special-purpose functions. This requires either an interpreted language or user-compiled functions that are dynamically linked with the graphics system. An interpreted language was described in [18], while the shade-trees approach [6] appears to use compiled functions that are dynamically linked to an interpreted expression evaluator.

In the language approach adopted by the author, a small and portable public-domain Lisp language interpreter was adapted to allow compiled C language functions to be dynamically linked and called from Lisp. This approach avoids the definition and implementation of a new special-purpose language, and permits functions to be implemented in either Lisp or C or some combination of these. Typically a function is developed in Lisp, and if needed the inner loops are reimplemented in C and dynamically “glued” together with Lisp.

Stepping back from the full power of procedural manipulation of the noise, we note that the special case of a functional transformation is useful from an analysis-resynthesis viewpoint, since a desired probability density (a commonly measured random texture characteristic) can be obtained by functional transformation [12].

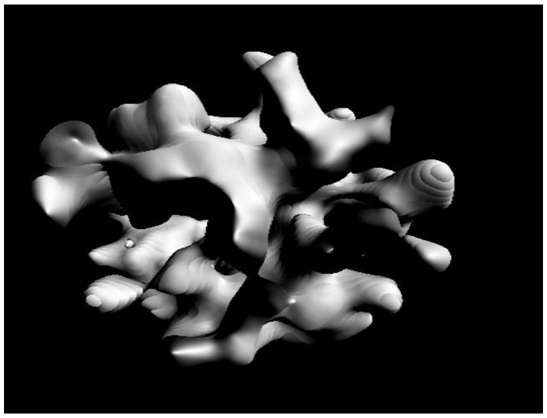
Fig. 8: Porous object model defined by the iso-density surface of a solid noise.

5.1 SOLID TEXTURE

Although spectral synthesis adequately simulates most homogeneous random textures, many textures have some structural features that cannot be simulated using this approach. Such textures can often be simulated using a procedural transform of a homogeneous texture. An example is the marble texture described in [18]. Another example of a procedural transform is the solid wood texture shown in (Figs. 6, 7). The structure of concentric rings parallel to the tree trunk is produced using a periodic or (preferably) quasi-periodic function $xtab$ of the radial distance from the z axis. This function describes the color variation across a radial section of a typical ring. Natural irregularity is introduced by perturbing the radial distance by a solid noise:

$$wood(\boldsymbol{\rho}) = xtab[\sqrt{\boldsymbol{\rho}.x^2 + \boldsymbol{\rho}.y^2} + \eta(\boldsymbol{\rho})]$$

(where $\boldsymbol{\rho}.x$ denotes the x-component of $\boldsymbol{\rho}$). Various refinements are possible, for example, the radial distance can be replaced by a random monotonic function of this distance, thereby creating radial regions of densely or sparsely separated rings to simulate periods of slow or fast growth. Fig. 6 shows planar sections of several solid wood simulations, where the perturbation noise η and the ring cross section function $xtab$ are altered to simulate different woods. Fig. 7 shows a figure model with a solid wood texture. This surface/texture combination would be difficult to achieve using texture mapping.

5.2 STOCHASTIC MODELING

An interesting stochastic modeling approach utilizing solid noise is to define objects as the equal-density surface of a solid noise. The overall shape of the object can be controlled by multiplying the noise by an analytic density function that tapers to zero outside of the desired object shape. This approach can produce porous and highly irregular shapes such as the coral-like form in Fig. 8.

5.3 STOCHASTIC DEFORMATION

This is a powerful stochastic modeling technique which uses a vector-valued solid noise (vector field) $v : R^3 \rightarrow R^3$ to perturb

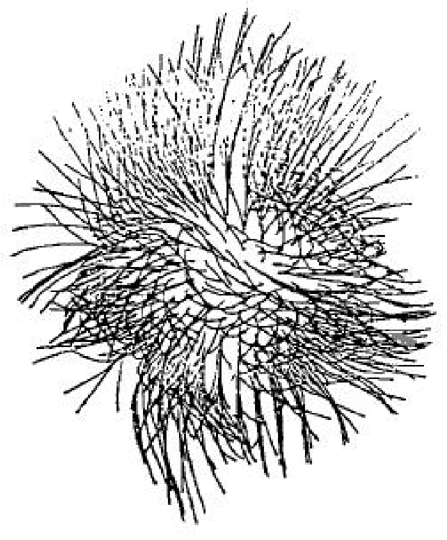

Fig. 12: Trajectories of a number of particles forced by a vector solid noise.

an existing object model. Three independent scalar solid noises form the components of the vector field. Stochastic deformation is particularly efficient for polygonal boundary-represented models, since only the vertices are perturbed.

Stochastic deformation can be used to simulate the individuality of natural objects by slightly deforming a prototype object model (Figs. 9, 11). The noise can be varied to produce either realistic or caricatural individuality.

Large-amplitude or iterated deformation can produce self-intersecting or twisted forms which do not resemble the original object (Fig. 10). A deformation can be animated by offsetting the object location by a continuously changing vector before perturbing, effectively moving the object through the noise.

5.4 CORRELATED FLOW

Solid noises may be employed as a correlated random environmental factor for many physically motivated simulations. For example, a solid noise can be used as a force field to produce turbulent trajectories or flow. Fig. 12 shows the trajectories of a number of particles obeying a simple dynamics equation $\ddot{\boldsymbol{\rho}} = \eta(\boldsymbol{\rho})$. The resulting collection of trajectories displays bifurcations and resembles animal fur and other natural structures. Fig. 13 is a frame from a brief animation in which the trajectories are animated by the previously mentioned technique of moving the model through the noise. The trajectories are rendered to produce the effect of an "organic fireball".

6 CONCLUSION

In addition to their demonstrated use in solid texturing, solid noises have direct applications in stochastic modeling. In both modeling and texturing it is desirable that the solid noise synthesis be controllable, efficient, and free of artifacts. Spectral synthesis provides a framework for assessing the control and quality of various synthesis approaches. Existing solid noise algorithms were surveyed from this viewpoint.

Two new algorithms were described and evaluated. Both algorithms provide improved spectral control and efficiency. The sparse convolution algorithm is attractive in that it allows a tradeoff between quality and efficiency as required by the application, without introducing gross artifacts. Several solid texturing and stochastic modeling examples visually illustrate the control and quality achievable with this algorithm.

Symbols

$S(\lambda)$	power spectrum
$C(\tau)$	autocovariance function
h	filter kernel
γ	uncorrelated noise
η	correlated synthesized noise
$\boldsymbol{\rho}, \boldsymbol{\sigma}$	locations in space
λ, ω	frequency, angular frequency

Acknowledgements

The figure model in Fig. 7 was developed by Dick Lundin and Susan VanBaerle. The head model used in Fig. 9 was developed by Fred Parke [15] with additions by Rebecca Allen, Steve DiPaola and Robert McDermott. Thanks to Paul Heckbert and Lance Williams for discussions.

References

[1] Abramowitz, M. and Stegun, I., *Handbook of Mathematical Functions*. Dover, New York, 1965.

[2] Bohm, W., Farin, G. and Kahmann, J., A Survey of Curve and Surface Methods in CAGD. *Computer Aided Geometric Design 1*, 1 (1984), 1-60.

[3] Bracewell, R., *The Fourier Transform and Its Applications*. McGraw-Hill, New York, 1965.

[4] Carpenter, L., Computer Rendering of Fractal Curves and Surfaces. Supplement to Proceedings of SIGGRAPH '80 (Seattle, July 1980). In *Computer Graphics 14*, 3 (July 1980), 180.

[5] Cook, R., Stochastic Sampling in Computer Graphics. *ACM Transactions on Graphics 5*, 1 (January 1986), 51-72.

[6] Cook, R., Shade Trees. Proceedings of SIGGRAPH '84 (Minneapolis, July 23-27 1984). In *Computer Graphics 18*, 3 (July 1984), 223-231.

[7] Deutsch, R., *Estimation Theory*. Prentice-Hall, New Jersey, 1965.

[8] Fournier, A., Fussell, D., and Carpenter, L., Computer Rendering of Stochastic Models. *Communications ACM 25*, 6 (June 1982), 371-384.

[9] Gardner, G., Simulation of Natural Scenes Using Textured Quadric Surfaces. Proceedings of SIGGRAPH '84 (Minneapolis, July 23-27 1984). In *Computer Graphics 18*, 3 (July 1984), 11-20.

[10] Heckbert, P., *Personal communication*.

[11] Lewis, J.P., Generalized Stochastic Subdivision. *ACM Transactions on Graphics 6*, 3 (July 1987), 167-190.

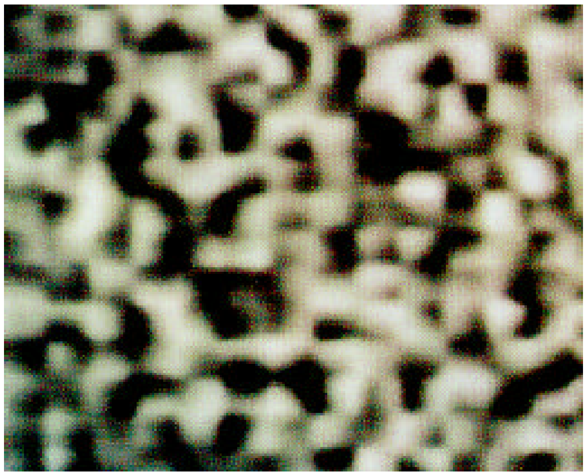

Fig. 2: Planar section of a solid noise synthesized by Hermite interpolation of a lattice of uncorrelated values. The noise shows directional artifacts and is zero at the lattice points.

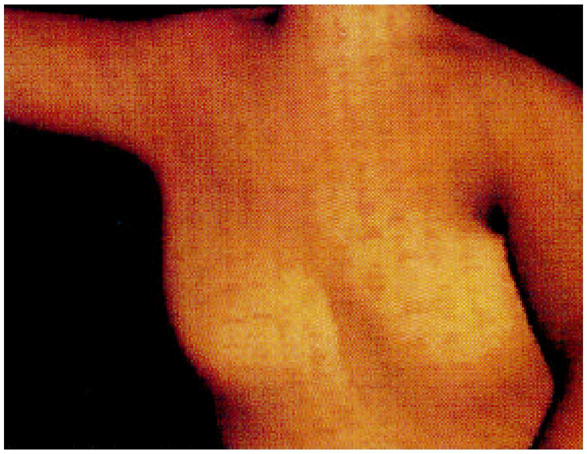

Fig. 3: Three-dimensional figure textured with a procedural solid wood texture which uses the solid noise shown in Fig. 2 as a primitive. Noise artifacts resulting from the regularly spaced zeros of this noise primitive are visible (compare with Fig. 7).

[12] Lewis, J.P., Methods for Stochastic Spectral Synthesis. In *Proceedings of Graphics Interface 86* (Vancouver, May 1986), 173-179.

[13] Oppenheim, A. and Schafer, R., *Digital Signal Processing*. Prentice Hall, Englewood Cliffs, N.J., 1975.

[14] Papoulis, A., *Probability, Random Variables, and Stochastic Processes*. McGraw-Hill, New York, 1965.

[15] Parke, F., Parameterized Models for Facial Animation. *IEEE Computer Graphics and Applications 2*, 9 (Nov. 1982), 61-68.

[16] Peachy, D., Solid Texturing of Complex Surfaces. Proceedings of SIGGRAPH '85 (San Francisco, July 22-26 1985). In *Computer Graphics 19*, 3 (July 1985), 279-286.

[17] Perlin, K., A Unified Texture/Reflectance Model. In SIGGRAPH '84 *Advanced Image Synthesis* course notes (Minneapolis, July 1984).

[18] Perlin, K., An Image Synthesizer. Proceedings of SIGGRAPH '85 (San Francisco, July 22-26 1985). In *Computer Graphics 19*, 3 (July 1985), 287-296.

[19] Schafer, R. and Rabiner, L., A Digital Signal Processing Approach to Interpolation. *Proc. IEEE 61*, 6 (June 1973), 692-702.

[20] Wiener, N., *Extrapolation, Interpolation, and Smoothing of Stationary Time Series*. Wiley, New York, 1949.

[21] Yaglom, A., *An Introduction to the Theory of Stationary Random Functions*. Dover, New York, 1973.

Fig. 5: Planar sections of solid noises synthesized using the sparse convolution algorithm.

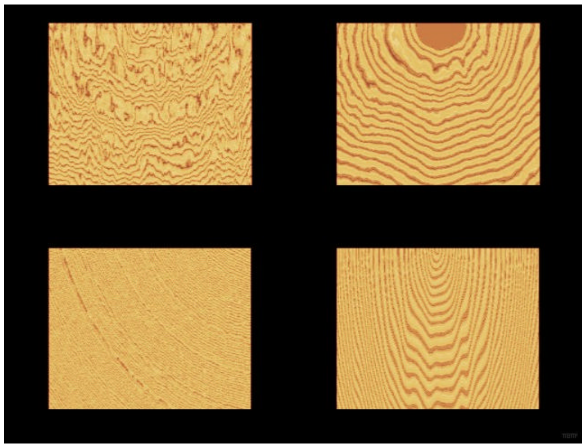

Fig. 6: Sections of solid wood textures synthesized by procedural transformation of suitable homogeneous textures.

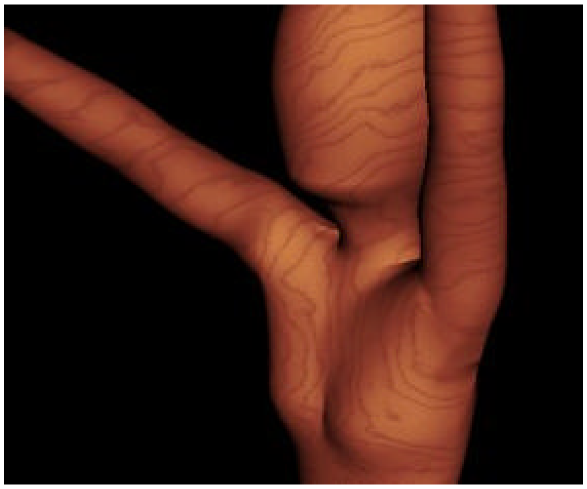

Fig. 7: Solid wood texture applied to a figure model.

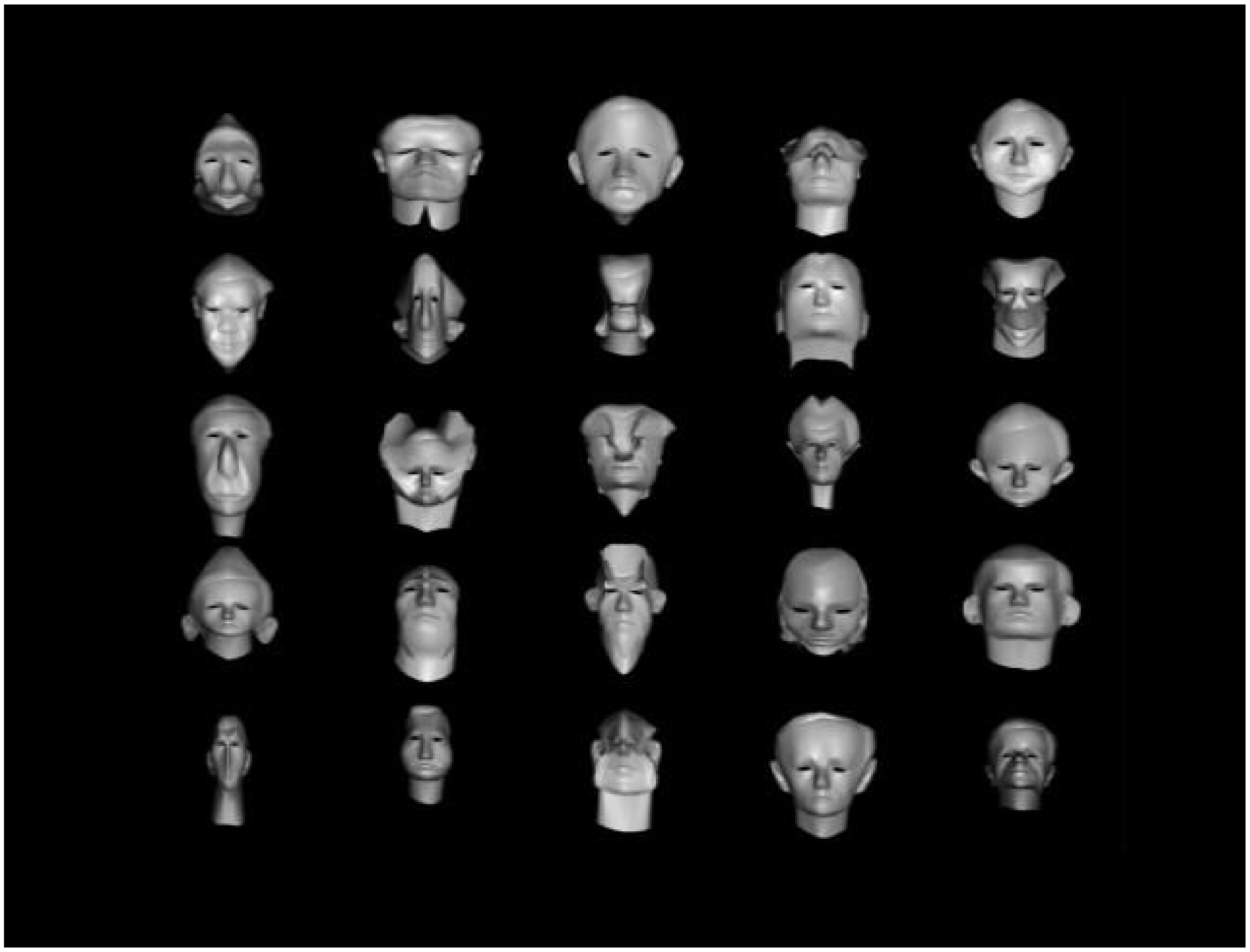

Fig. 9: Individual head models generated by displacing the vertices of a prototype polygonal head model with a vector solid noise.

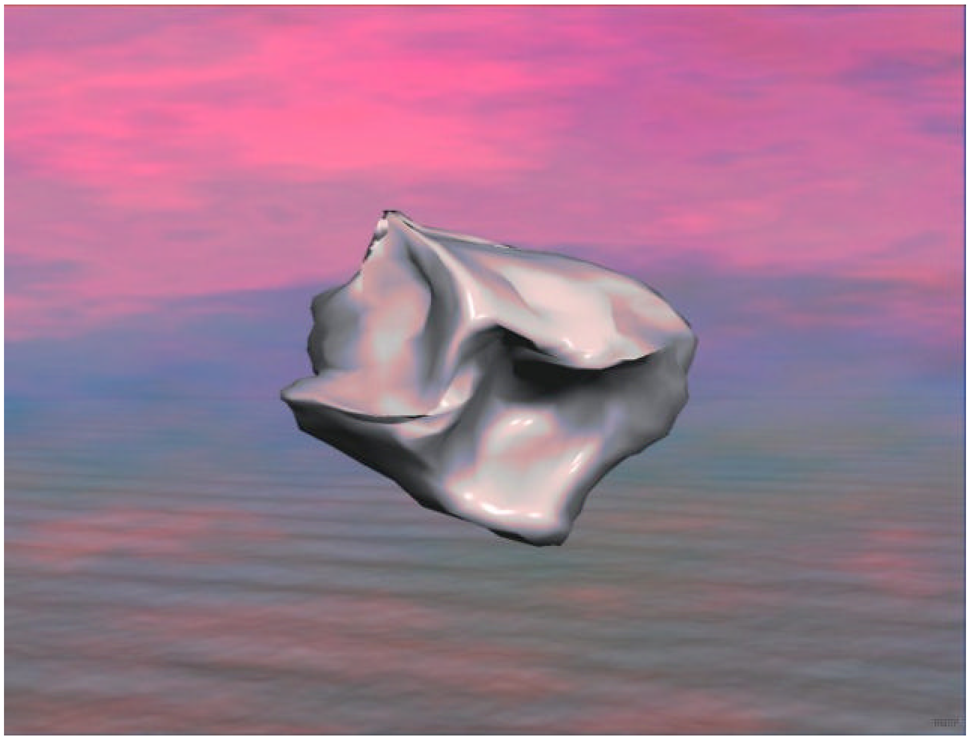

Fig. 10: A shape created by distorting a polygonal sphere by a vector solid noise.

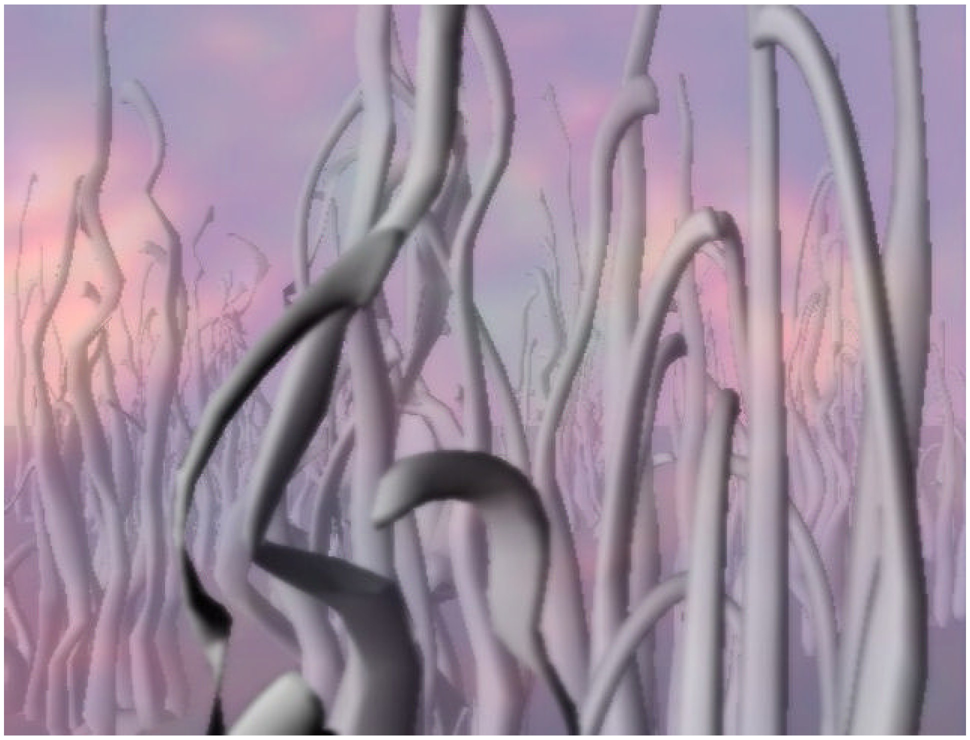

Fig. 11: Shrubbery created by stochastic deformation.

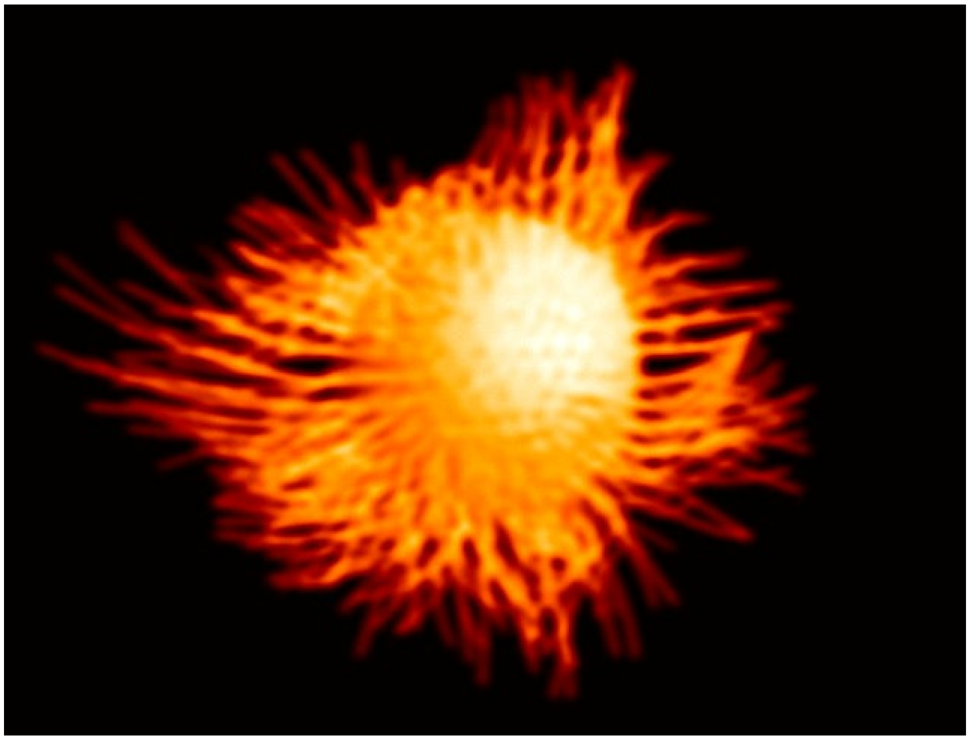

Fig. 13: An object rendered from trajectories as in Fig. 12.

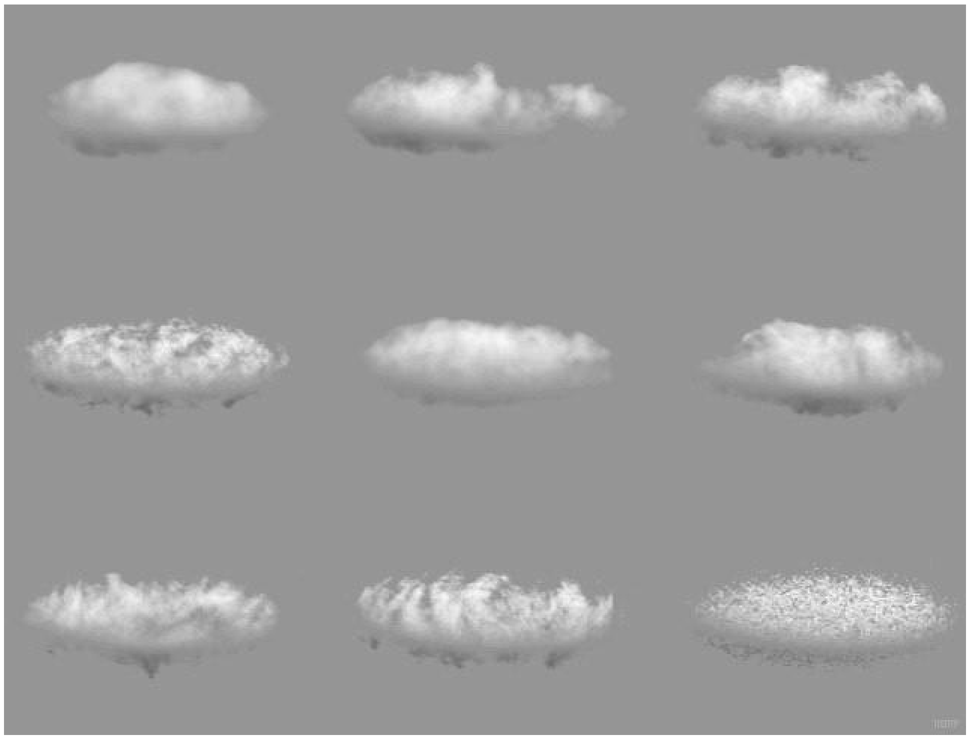

Fig. 14: Cloud studies using solid noises in a rendering algorithm similar to that in [9].