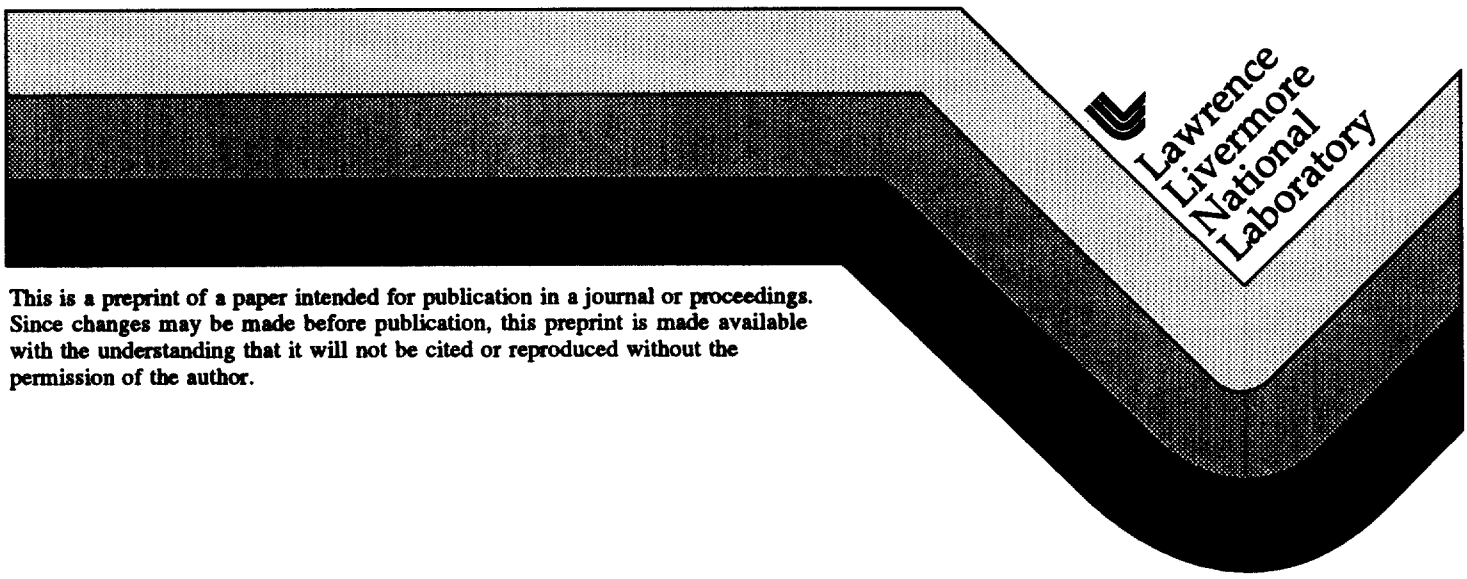
UCRL-JC-123623
PREPRINT

Multivariate Volume Rendering

R.A. Crawfis
N. Max

This paper was prepared for submittal to the
Institute of Electrical and Electronics Engineers Visualization
San Francisco, CA
October 27-November 1, 1996

March 1996



Multivariate Volume Rendering

Roger A. Crawfis
Nelson Max

Lawrence Livermore National Laboratory

Category: Research

Abstract

This paper presents a new technique for representing multivalued data sets defined on an integer lattice. It extends the state-of-the-art in volume rendering to include non-homogeneous volume representations. That is, volume rendering of materials with very fine detail (e.g. translucent granite) within a voxel. Multivariate volume rendering is achieved by introducing controlled amounts of noise within the volume representation. Varying the local amount of noise within the volume is used to represent a separate scalar variable . The technique can also be used in image synthesis to create more realistic clouds and fog.

Background on Multivariate Representations

Several techniques for representing multi-dimensional data, such as that resulting from census data, can be found in [1]. The emphasis with these techniques is the analysis of scattered data with many independent variables. Few techniques have been explored for representing several continuous fields defined on data grids. One common approach for two-dimensional data sets is to color-code one variable using a hue-ramp and another using a saturation or value-ramp [1]. The resulting color is then defined by taking the hue from the first variable-mapping and the saturation (or value) from the second while holding the value (or saturation) constant. Similar schemes have been explored using a separate red-ramp, green-ramp and blue-ramp for three different variables. These techniques can be extended only in a limited fashion to volume rendering in three-dimensions, due to the compositing or additive nature of the colors. For two-dimensional data sets, texture mapping techniques can be applied to a three-dimensional surface plot of a variable. Crawfis and Allison[2] illustrate several techniques for representing 2D multivariate data using texture mapping.

Volume Rendering Background

Volume rendering can be used to represent a scalar variable, such as those generated from MRI or CAT scans, and those generated in computer simulations. Drebin,

Carpenter and Hanrahan[3], and Levoy[4] introduced a technique for ray-tracing these scalar fields to yield a meaningful representation. Upson and Keeler[5] describe both a ray-tracing solution and a scan-conversion solution for volume rendering. Max, Hanrahan and Crawfis[6] introduce a scan-conversion technique for arbitrary polyhedra. Shirley and Tuchmann[7] describe a hardware assisted approximation of the volume rendering by scan-converting the projected triangles of a tetrahedron.

Westover has proposed two methods of a technique he coined "splatting" to produce volume rendering. Here, each data point's contribution to a continuous signal is calculated and composited separately onto the image plane. Weighted reconstruction functions, centered at each data point are added together to form a continuous 3D signal. Integrating the reconstruction function or kernel along the viewing direction, produces a 2D kernel that Westover called a footprint function, or splat. In his first method [8], the color and opacity footprint functions for each voxel are composited one by one in back-to-front order (in regard to their center points). In the second [9], the color and opacity splats for all the voxels in a layer are summed into an accumulation buffer, and then composited as a whole into the image. This second method prevents the opacity interactions of the voxels within one layer, and eliminates any possible small glitches from the change in sorting order within a layer during rotation. However, it may introduce larger glitches when the choice of the layer direction (most perpendicular to the viewing direction) changes. Laur and Hanrahan [10] approximate each splat with a collection of polygons which can easily be scan-converted by specialized graphics hardware.

Crawfis and Max [11] extended the splatting technique to use texture mapping hardware. They developed an optimal reconstruction footprint for volume rendering, with a minimal footprint area. This footprint image is made into a texture map representing the opacity. A single texture mapped square is then composited onto the image at each projected data point in back-to-front order. They then extended the field of volume rendering by introducing distortions into the texture map to represent vector fields. Small local distortion preserved the overall footprint function, while adding the ability to represent the projected vector field direction. A table of these splats was constructed for animation of the vector field.

The algorithm described in this paper attempts to represent several scalar variables defined on a regular three-dimensional mesh. Volume rendering is extended to yield noisy-looking density clouds. The amount of noise within a cloud is controlled to represent a separate variable from the variable defining the density cloud. This approach is a new technique which we have called Noise Splats.

Adding Noise to the reconstruction texture

The basic approach we take is to extend the Textured Splats technique described by Crawfis and Max [11, 12]. For vector field visualization, they used an anisotropic texture for the splatting. Rather than using an anisotropic texture, we embed either noise or dots within the texture. By defining a table of textures with an increasing noise amplitude or number of dots, we can control the amount of noise within the volume using a second independent variable. Since the reconstruction kernels of the scalar splats overlap to produce a smooth function, the placement of dots within a reconstruction kernel needs to take into account this overlap to produce a homogenous amount of noise in the composited image. If the overlap is ignored, bands with a sharp increase or decrease in the density of dots will appear at the seams. Several different algorithms for generating this type of noise are described in the sections below.

Modulate texture with white noise

A reconstruction function is needed, that rather than being smooth, contains a substantial amount of high-frequency noise, while still preserving the overall kernel shape. Stated differently, we require the mean of the reconstruction function of the noisy splats to be equal to the reconstruction function of the scalar splat within a small neighborhood. By adding white noise with a mean of zero and amplitude $w * h(r)$ to the reconstruction function we produce the desired result. The noise is windowed by the footprint function $h(r)$ to avoid harsh edge effects on the texture mapped square, but also, to give the correct amplitude of noise when neighboring and overlapping splats are composited together. If the amplitude of the noise, ω, is large, the splat footprint function needs to be scaled by $(1-w)$, to ensure a valid color range. For semi-transparent splats and moderate noise amplitudes, this is usually not a problem. A noisy reconstruction function similar to that shown in Figure 1 is thus produced. By varying the amplitude w, different amounts of noise can be embedded into the texture.

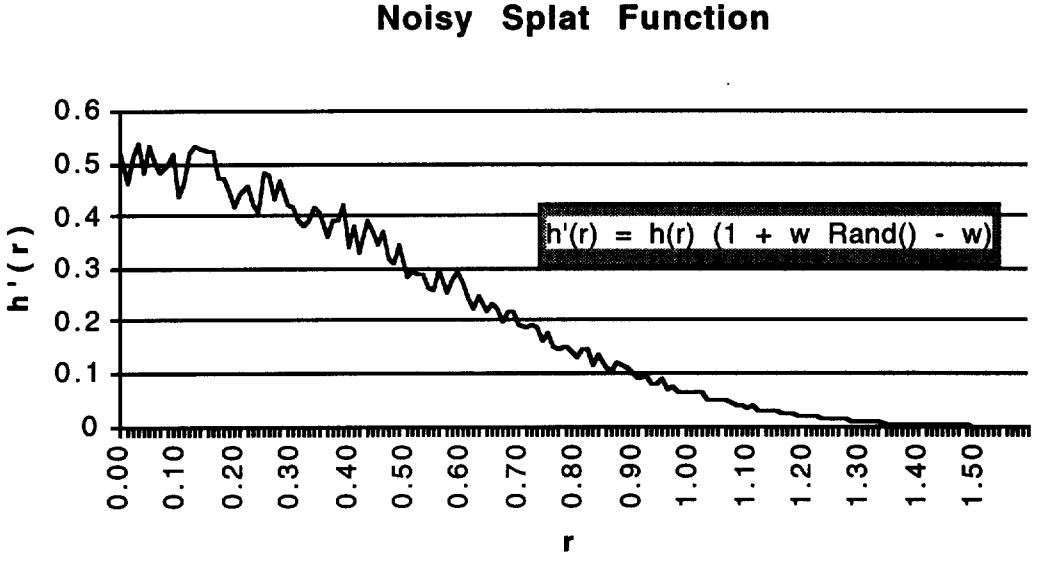


Figure 1 Noisy Splat Function

The resulting splats, called Noise Splats, need to maintain the overall property of the reconstruction function. Figure 2 shows a series of splats with increasing noise amplitudes. Since the noise has a mean of zero, the overall intensity is preserved across the splats and no extra considerations are necessary. Reconstruction of two constant functions with these noisy splats should still produce a volume rendering with a constant local-mean intensity, opacity and amount of variance. Figure 3 represents a single layer of splats with a constant amplitude appropriately composited together. The image is smooth, but noisy, yielding a faithful representation for the volume rendering and allowing the noise to represent another variable.

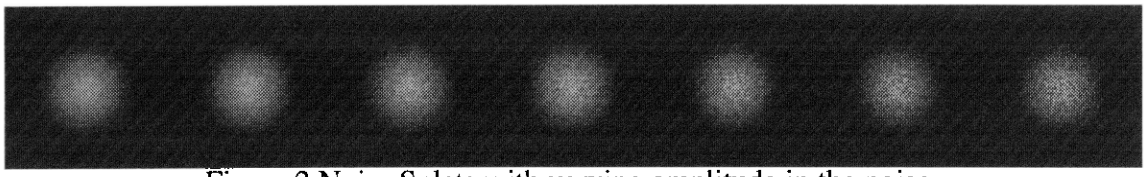


Figure 2 Noise Splats with varying amplitude in the noise.

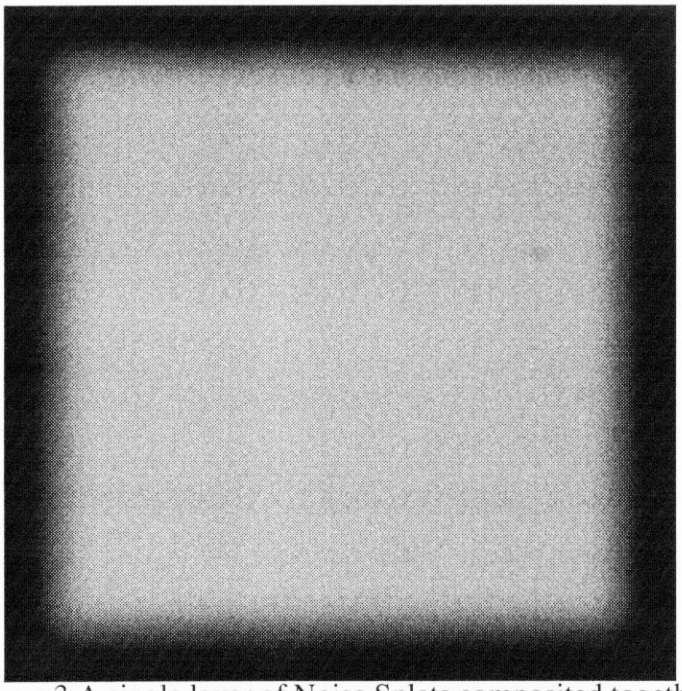


Figure 3 A single layer of Noise Splats composited together

The Noise Splats are rendered using the modulate blend function within the OpenGL [13, 14] graphics system. For each splat point, a splat color and an amount of noise is specified. The amount of noise varies between zero and one. A splat is selected from a precomputed table of splats with increasing noise amplitude, using the amount of noise at the splat point as a selector. A value of zero will select a splat with no noise, while a value of one will select the splat in the table with the highest noise amplitude. The maximum amplitude of the splats within the table can be specified by the user as part of the class specification. The size of the texture table (how many splats there are) can also be specified by the user. The textures are computed on the fly the first time the data is rendered and anytime the data or the settings (the maximum amplitude or the number of textures) are changed.

Mip-mapping can ensure that the noise remains in very small splats or that it fades into the reconstruction function average, depending on the user's preferences. As a splat's size approaches a single pixel, it can thus either represent the volume rendering or the variable encoded in the noise.

Figure 4 illustrates a volume rendering of a tornado data set using the Noise Splats. The x-velocity component is mapped to the noise amplitude, with a larger positive value increasing the strength of the noise.

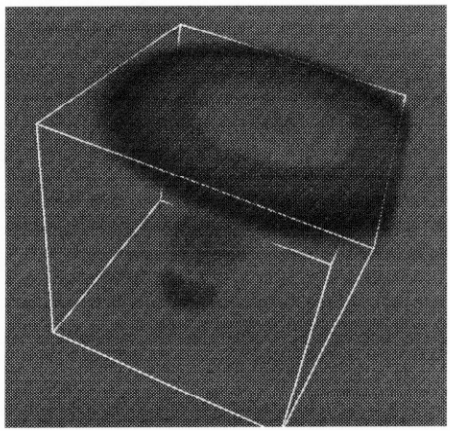


Figure 4. Modulated White Noise

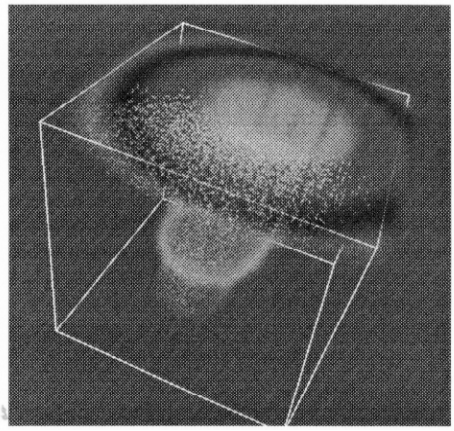


Figure 9. Stretched Point Distribution.

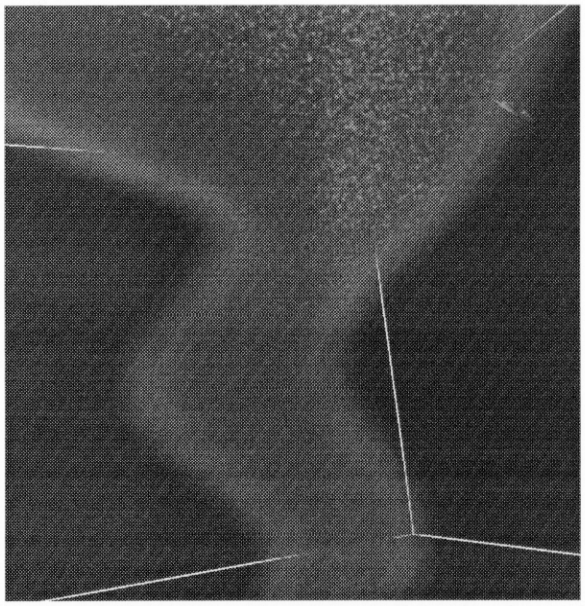


Figure 10. Z-position represented as the dot color.

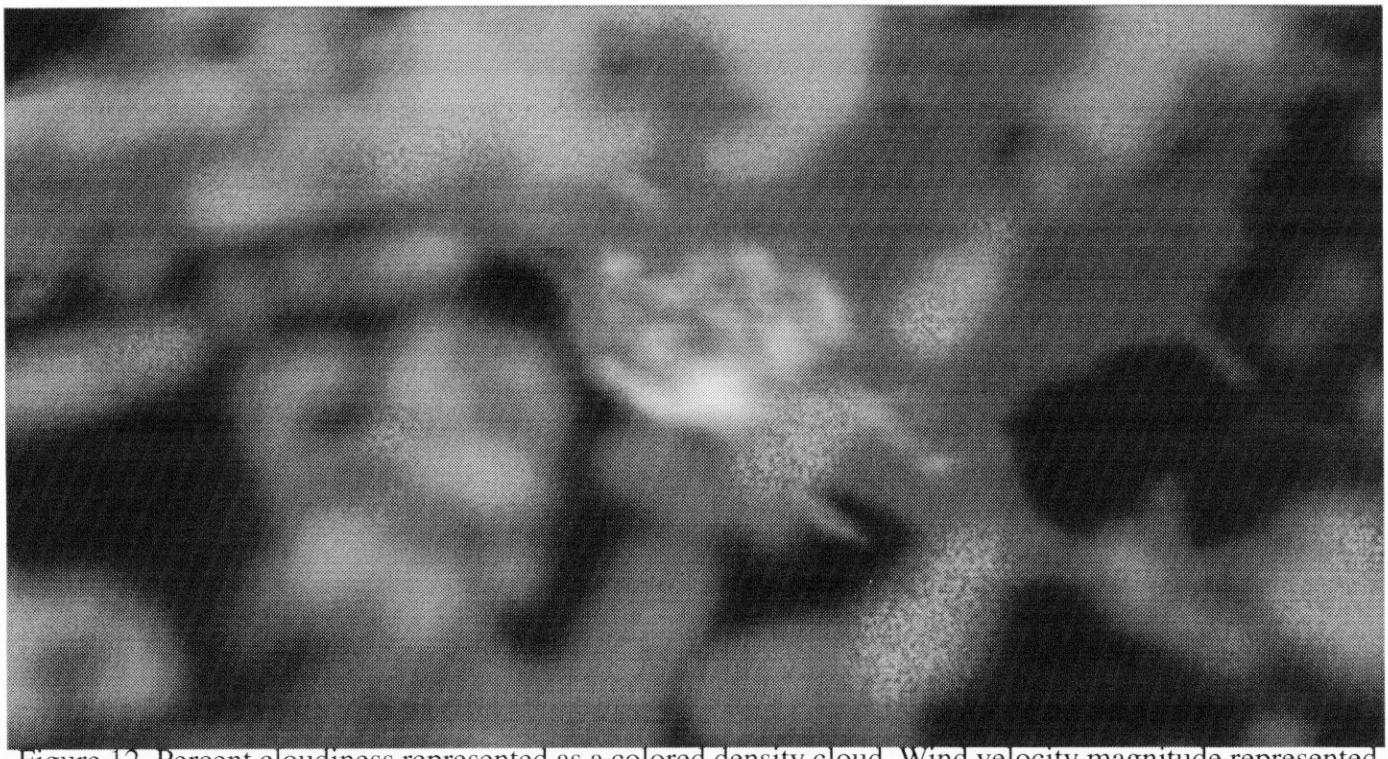


Figure 12. Percent cloudiness represented as a colored density cloud. Wind velocity magnitude represented using modulated white noise. Higher winds imply more noise.

Poisson disc distribution

For a stronger representation of the noise, we can introduce discrete dots within the texture. With individual points, we need a uniformly random looking pattern to avoid false patterns, but also need the frequency or density of points to be meaningful. A strictly random or pseudo-random set of points will not produce this. A first experiment at producing random pattern tried this and led to cases where the data could not be interpreted accurately. A Poisson disc distribution insures that the data points do not clump together. This can be accomplished by requiring the distance between a point and all previous points to be greater than some threshold. The threshold within a splat gradually gets smaller, starting with a wide distribution of points and then gradually increasing the point density.

For the purposes of splatting, we need the image resulting from several composited splats to produce a density of points that is meaningful. Much like the reconstruction functions for textured splats sum up close to a constant for areas with a constant functional value, the noise distribution needs to be fairly uniform in areas with a constant functional value. Unless the overlap is such that the same number of splats overlap everywhere, the Poisson disc distribution will not produce the proper distribution when the splats are composited. To overcome this, a weighted Poisson disc distribution has been developed. The basic premise is to have a higher point density near the center of the splat, that gradually fades to zero at the edges of the splat. The summation of the expected densities from splat points composited on top of each other should then be fairly constant. Since the reconstruction functions sum approximately to one, the same function can be used as the desired density function. The threshold in the Poisson disc distribution calculation is changed to be inversely proportional to the splat footprint or reconstruction function. Hence, the probability of adding points near the center is greater than at the edges of the splat texture. A series of splats with an increasing number of points is represented in Figure 5. The resulting composite of splats with the same number of points is represented in Figure 6. Three splats are calculated for each number of points to avoid regular patterns. Which of these three splats is used is chosen randomly.

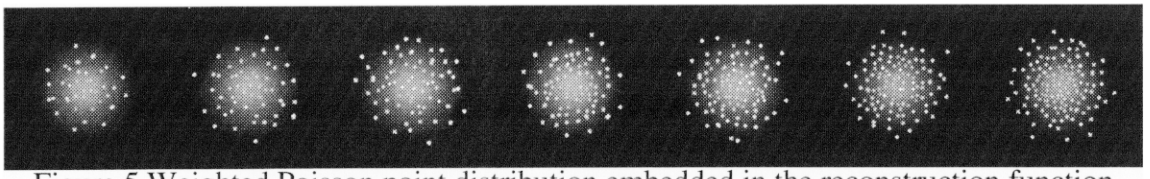


Figure 5 Weighted Poisson point distribution embedded in the reconstruction function

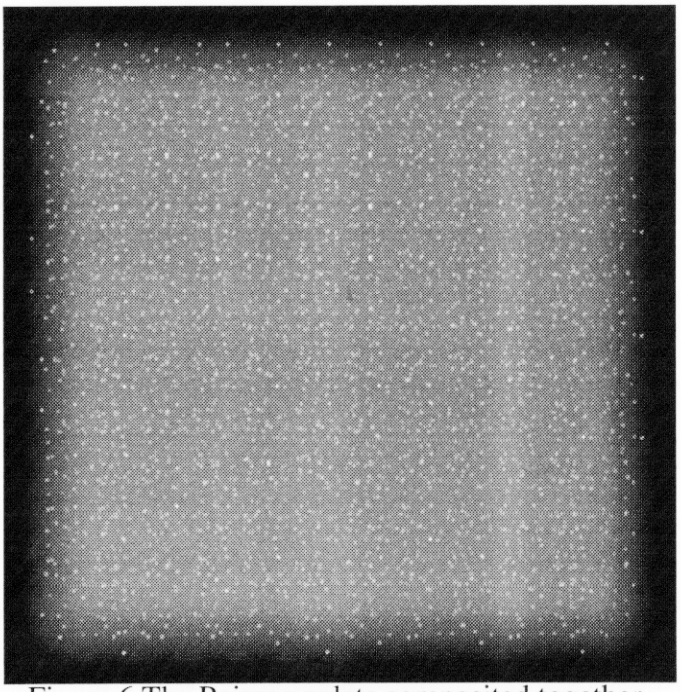


Figure 6 The Poisson splats composited together.

While this technique generates optimal distributions of the points, it is computationally expensive. The next two sections describe alternative schemes for generating point distributions such that the point density matches the reconstruction function. Several authors have developed algorithms for producing uniform distributions. However, none of these produce weighted distributions in two-dimensions.

Jittered Dithering Distributions

Mitchell [15] presented an algorithm for quickly generating a pseudo-uniform distribution based on the Floyd-Stienberg dithering algorithm. His algorithm used a regular 2D grid with sixteen times as many grid cells as the desired number of points. Each cell is visited in turn in a left-to-right and top-to-bottom fashion. A *potential* is calculated using the function:

$$D_{i,j} = (4D_{i-1,j} + D_{i-1,j-1} + 2D_{i,j-1} + D_{i+1,j-1}) / 8 \; + \; R$$

where,

$$R = \frac{1}{16} \pm \frac{1}{64} random().$$

If $D_{i,j}$ is greater than the threshold of one-half, a jittered point at cell (i,j) is selected and the $D_{i,j}$ potential is reduced by one. Thus, about 1/16th of the data points are selected.

To generate a weighted point density, two possible modifications to this algorithm were explored. The first was to adjust the threshold from a constant value of one-half to a value inversely proportional to the reconstruction function. Thus, it takes a higher potential to generate a point away from the center of the splat. The second approach was to keep the threshold constant, but change the incrementally added potential R to be proportional to the reconstruction function. Unfortunately, both of these techniques produced regular patterns in the splats and the number of dots deposited in a splat varied widely from the number specified and expected.

Stretched Uniform Distributions

Cook [16] presented a technique to generate weighted point densities in one-dimension by dividing the area under the distribution function into vertical strips of equal area. Consequently, this is equivalent to choosing a uniform distribution of samples, and then transforming the horizontal axis by a non-linear function that represents the "cumulative" integrated probability. Since our 2D reconstruction function is radially symmetric, we can generate a uniform point density and pull the points radially into the center. The function that "pulls" the points into the center depends on the reconstruction function. To determine this, recall that we want a point density to be proportional to the reconstruction function. Hence, we have

$h(r)$ = point density

and

$h\,da$ = expected number of points in a unit area.

If we then look at an infinitesimal ring at radius r and width dr around the center of the splat, the area of the ring is $2\pi r dr$, and hence the number of points per infinitesimal ring is:

$h\,2\pi r\,dr$ = expected number of points in infinitesimal ring.

The number of points expected within an infinitesimal ring for a uniform point distribution would be:

$k\,2\pi\rho\,d\rho$ = expected number of points in infinitesimal ring in "flat land".

Since the goal is to map a ring in "flat land" to the raised surface of the reconstruction function, we want the point densities of these two infinitesimal rings to be equal:

$$k\,2\pi\rho\,d\rho = h\,2\pi r\,dr.$$

Integrating both sides, yields:

$$\int_0^{\rho_0} k\rho d\rho = \int_0^{r_0} h(r) r dr,$$

$$k\frac{\rho_0^2}{2} = \int_0^{r_0} h(r) r dr.$$

Solving for ρ_0, leads to the mapping function

$$\rho_0 = f(r_0) = \sqrt{\frac{2}{k}\int_0^{r_0} h(r) r dr}.$$

The inverse of this function is needed to map from ρ to r. If the reconstruction function is proportional to a gaussian, then this can be solved analytically. Let

$$h(r) = ce^{-\frac{r^2}{2}}.$$

Then

$$\int_0^{r_0} h(r)rdr = \int_0^{r_0} ce^{-\frac{r^2}{2}} rdr$$

$$= c\left(1 - e^{-\frac{r_0^2}{2}}\right)$$

and we have

$$\rho = f(r) = \sqrt{\frac{2c}{k}\left(1 - e^{-\frac{r^2}{2}}\right)}.$$

Solving for the inverse function yields:

$$r = f^{-1}(\rho) = \pm\sqrt{-2\ln\left(1 - \frac{k\rho^2}{2c}\right)}.$$

We will always want the non-negative value of this square root.

The algorithm is then to generate random points within a circle of radius $\sqrt{\frac{2c}{k}}$ centered at the origin and use the mapping above to pull the point towards the center. Figure 7 illustrates a series of splats with an increasing number of points. The resulting composition from splats with the same number of points is shown in Figure 8. Three separate splats were randomly used in the compositing. Only the discrete points are represented for clarity.

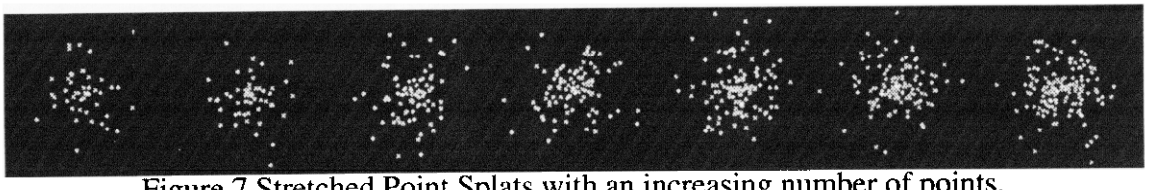


Figure 7 Stretched Point Splats with an increasing number of points.

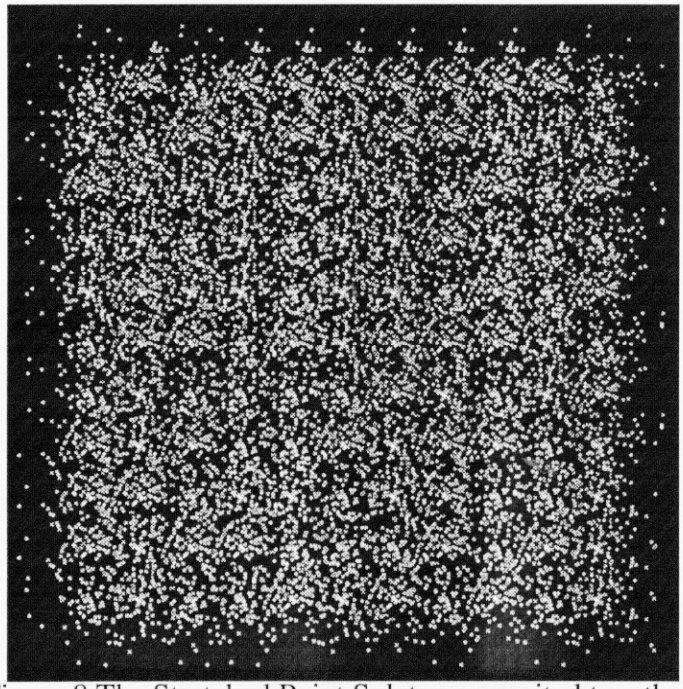


Figure 8 The Stretched Point Splats composited together.

Since random points were chosen rather than a Poisson disc distribution, some slight regular patterns are visible in the composited image. Starting with a more uniform distribution, such as a jittered grid, or using more than three splats in the compositing should alleviate some of this.

Figure 9 illustrates this technique on the tornado data set.

Animating the Noise Splats

Calculating a few (3-5) splats with the same amount of noise is useful in avoiding regular patterns as shown by the composited images in the previous section. By cycling through this set of splats, the noise can be animated, providing a stronger and dynamic representation of the variable mapped to the noise. A 2D table of splats is created with increasing amounts of noise down the columns. Each column is created independently using different sets of random numbers. By randomly selecting a column, the noise is animated. For small to mid-sized data sets that can be rendered interactively, this dynamic "sparkling" of the noise is also achieved interactively.

More than two independent variables

So far in this paper, we have only represented two variables with the noise splats. This is a very useful start for aiding the understanding of many complex simulations. It is also useful to be able to examine the relationships between three or more variables. The algorithms described in this paper can be extended to include at least one more variable. The blending operations described above have all used a constant color for the noise. By varying the color of the noise from splat point to splat point a third variable can be represented. Figure 10 illustrates the tornado using the stretched point distribution as in Figure 9, but here the color of the points are controlled by the z-coordinate. A mapping from black to red to yellow to white is used as z goes from the bottom of the tornado to the top.

By combining the anisotropic textures for representing scalar and vector field that were developed by Crawfis, with the splat textures developed in this paper, two or three independent scalar fields and one vector field can be represented. The vector icons in the texture must be of a different scale than the noise to avoid losing the vector icons into the noise. The same time dynamics from both the vector splats and the noise splats can be combined into the same textures. Thus a three-dimensional table of splats is needed if animation and vector foreshortening are desired. The axes of the table correspond to the vector foreshortening, the amount of noise and the time dynamics. Figure 11 illustrates a sample splat from this approach.

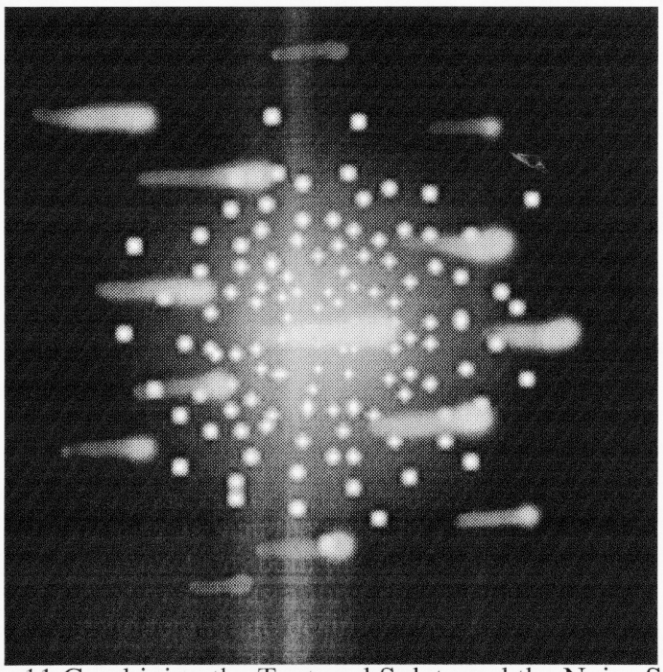


Figure 11 Combining the Textured Splats and the Noise Splats.

Finally, the animation of the noise can be controlled by simply cycling sequentially through the splats (with the desired amount of noise). By controlling the cycling speed individually for each splat, a fourth scalar variable can be represented.

Applications

Figures 12 and 13 illustrate the utility of this technique on data generated from a Global Climate simulation. The percent cloudiness is represented as a colored volume, with red and white representing the highest values of percent cloudiness (most surface occlusion). The wind velocity magnitude is represented using modulated noise in Figure 12 and discrete points in Figure 13. The discrete points in Figure 13 are colored according to the altitude, to give a better indication of the three-dimensional position of the cloud densities.

Figure 14 represents the density and the velocity magnitude of an astrophysical shock wave. The velocity magnitude is represented as the volume rendering, while the density is represented as varying amounts of noise. An iso-contour surface of density is also included within the density cloud.

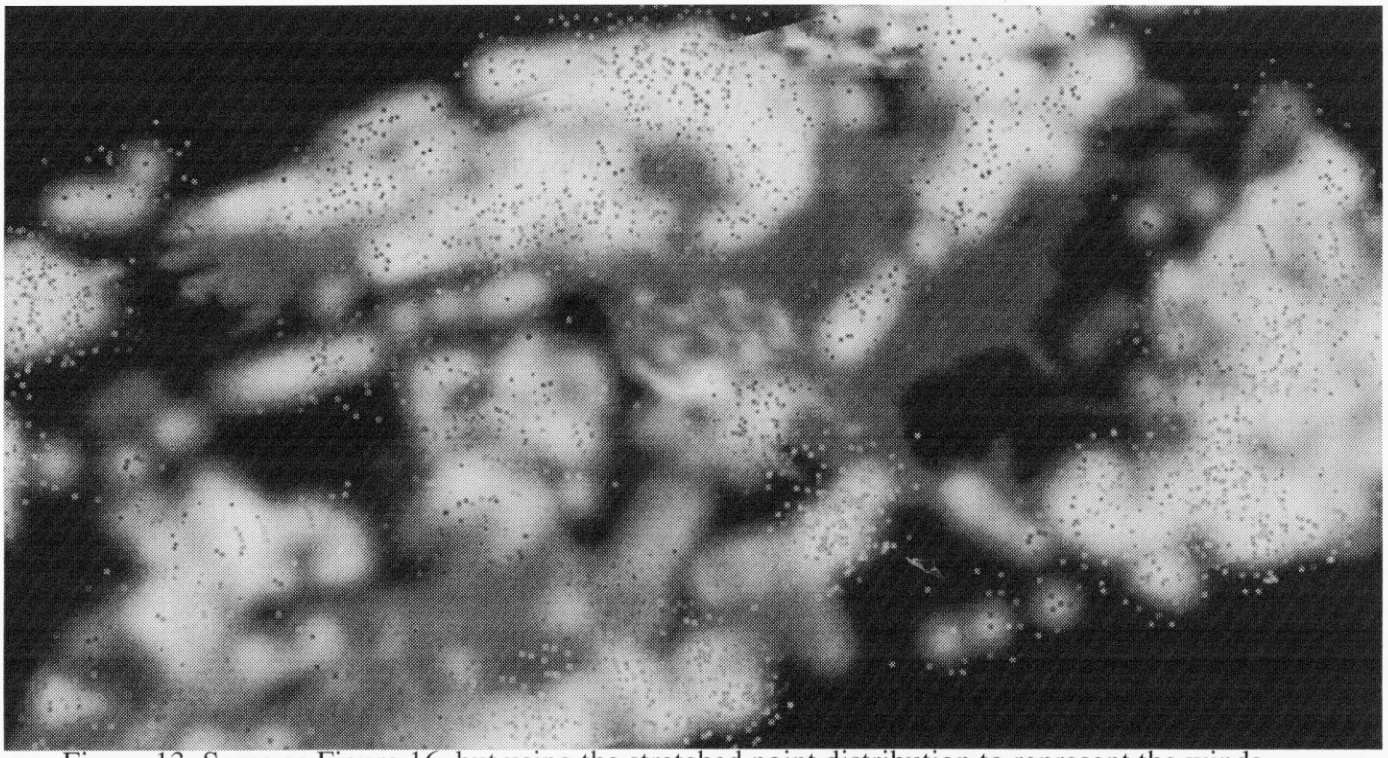


Figure 13. Same as Figure 16, but using the stretched point distribution to represent the winds. (Data courtesy of Jerry Potter).

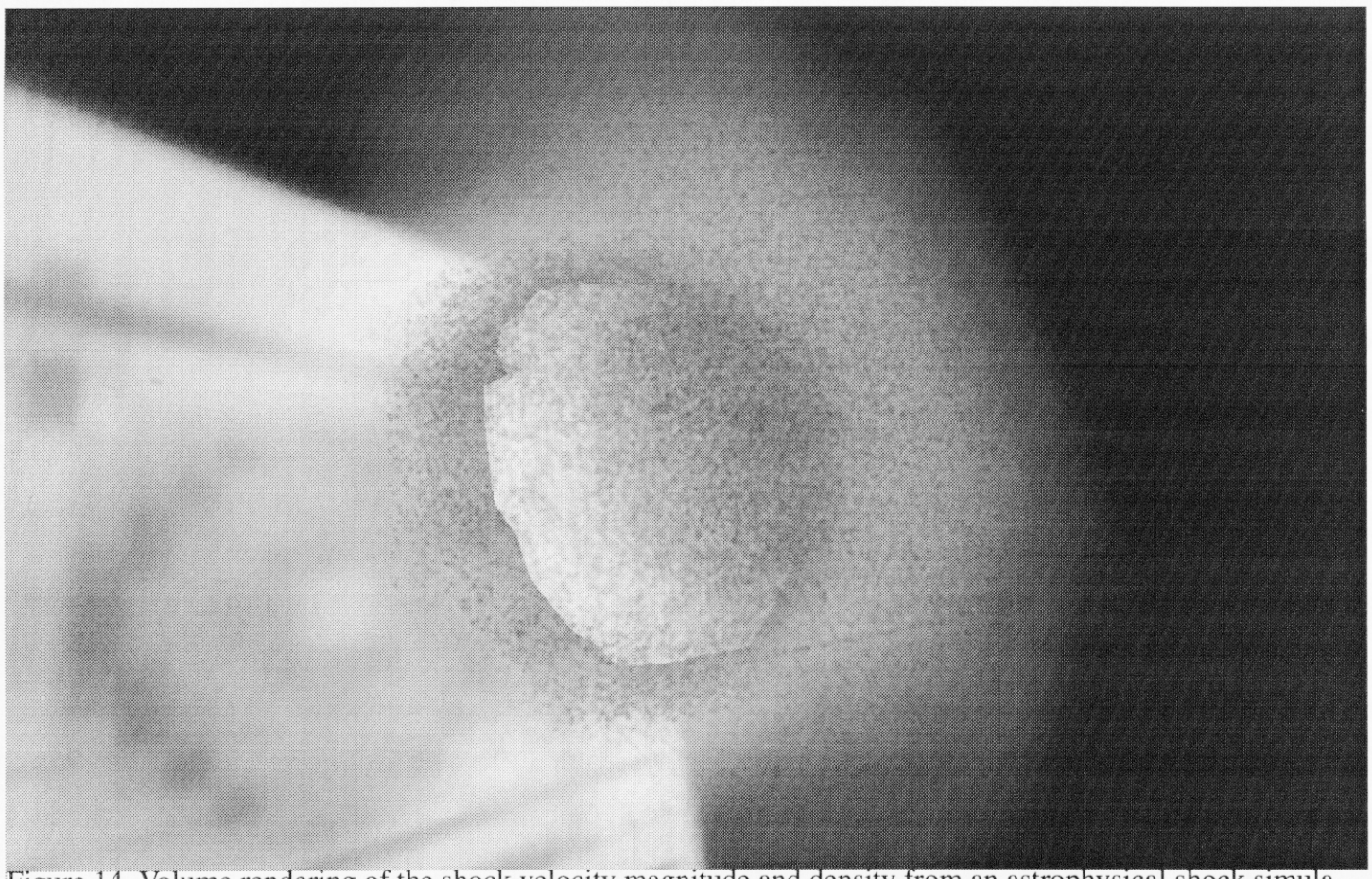


Figure 14. Volume rendering of the shock velocity magnitude and density from an astrophysical shock simulation. The velocity magnitude is represented using normal volume rendering, while the density is represented using varying amounts of Noise. (Data courtesy of Richard Klein).

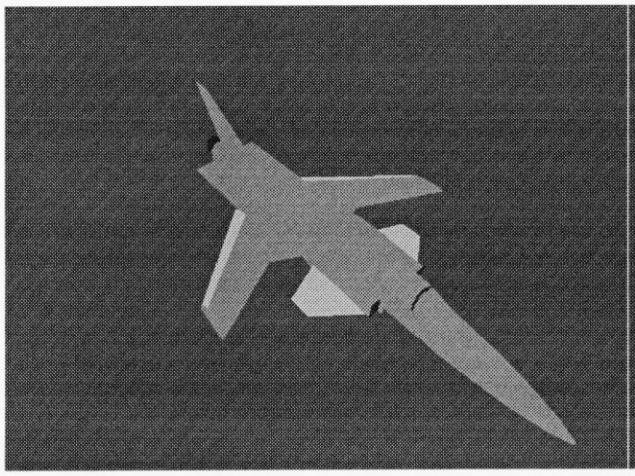


Figure 15. A simple rendering of a X29 Jet (model courtesy of Silicon Graphics, Inc.).

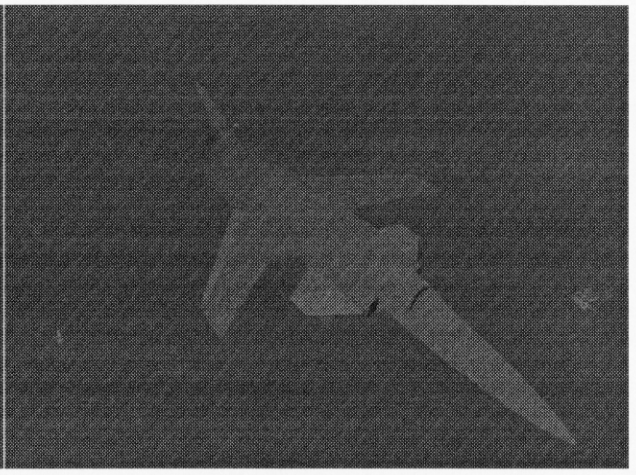


Figure 16. The X29 Jet with hardware fog added.

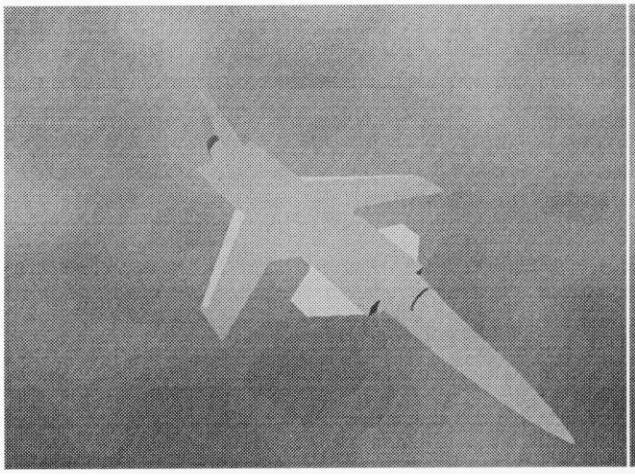


Figure 17. The X29 Jet with volume rendered clouds.

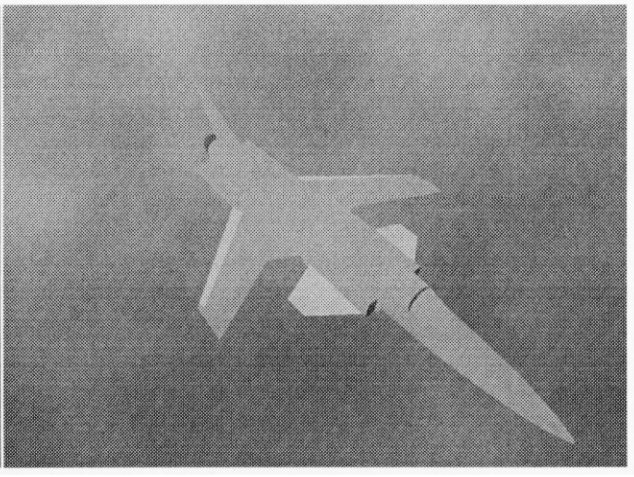


Figure 18. The X29 with volume rendered clouds using Noise Splats.

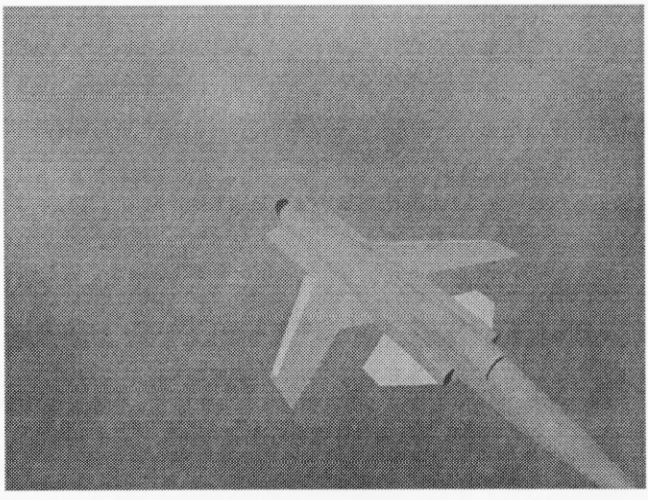


Figure 19. Using different noise settings produces slightly different effects.

Realistic Fog

The noise splats can also be used to model and represent more realistic clouds or fog. Fog models typically assume a constant and homogenous atmospheric attenuation. Figure 15 is a simple rendering of a X29 jet. Figure 16 adds hardware fog as a depth cue. True clouds and fog, are patchy, with varying densities. Volume modeling these density clouds and applying standard volume rendering techniques will produce a more realistic fog (Figure 17). A highly detailed volume model (such as that used in fractal cloud models) is needed to achieve truly realistic clouds. A fractal appearance can be achieved by volume rendering a low-resolution cloud density with the Noise Splats (Figure 18). This is achieved by setting the color of the noise to be close to the volume color and varying the amount of noise randomly. Figure 18 uses the same volume model as in Figure 17, but with discrete dots in the texture to add some detail. Figure 19, also uses the Noise splats, but here the noise color is made slightly pink. The Noise Splats also aid in representing clouds when animated. When flying through a cloud, individual water particles can be seen glimmering in the light. The noise splats can also simulate this by adding flickering "sparkles" in a normal volume rendering.

Conclusions and Future Work

Volume rendering is a useful technique in scientific visualization. Here, we have extended volume rendering for the representation of multiple scalar fields. This is useful for representing complex relationships between variables as well as providing more information per graphic. The varying noise gives a good indication of a secondary scalar variable, while allowing a primary scalar variable to be represented as a volume density cloud. A weighted Poisson disc distribution is too costly, even for the small number of points. We have explored several different techniques for generating distributions of points such that when *splatted* together would yield a uniform distribution. Radially pulling a uniform distribution of points in towards the center has given us a good distribution with minimal cost. The modulated white noise embedded into the splats provides a nice subtle cue, with a high degree of animation and works quite well in practice (especially on large high-resolution monitors).

We have also shown how Noise Splats are useful for adding detail and sparkle to clouds and fog. The same principles applied for the rendering of fractal clouds can be used to model and represent other complex materials. Arbitrary textures can be created and used to model a variety of textures.

Acknowledgements

The climate data is courtesy of Jerry Potter, the Program for Climate Model Diagnosis and Intercomparison at Livermore, and the European Centre for Medium-range Weather Forecasts. The astrophysical shock wave data was provided by Richard Klein at LLNL. This work was performed under the auspices of the US Department of Energy by Lawrence Livermore National Laboratory under contract number W-7405-ENG-48, with specific support from an internal (LDRD) research grant.

References

1. Keller, P.R. and M.M. Keller, *Visual Cues*. 1993, Los Alamitos, CA: IEEE Computer Society Press. 229.

2. Crawfis, R. and M. Allison. *A Scientific Visualization Synthesizer.* in *Visualization '91.* 1991. San Diego, CA: IEEE Computer Society Press.

3. Drebin, R.A., L. Carpenter, and P. Hanrahan, *Volume Rendering.* Computer Graphics, August 1988. **22**(4): p. 64-75.

4. Levoy, M., *Display of Surfaces from Volume Data.* IEEE Computer Graphics and Applications, May 1988. **8**(5): p. 29-37.

5. Upson, C. and M. Keeler, *V-BUFFER: Visible Volume Rendering.* Computer Graphics, August 1988. **22**(4): p. 59-64.

6. Max, N., P. Hanrahan, and R. Crawfis, *Area and Volume Coherence for Efficient Visualization of 3D Scalar Functions.* Computer Graphics, 1990. **24**(5): p. 27-33.

7. Shirley, P. and A. Tuchman, *A Polygonal Approximation to Direct Scalar Volume Rendering.* Computer Graphics (San Diego Workshop on Volume Visualization), Nov. 1990. **24**(5): p. 63-70.

8. Westover, L. *Interactive Volume Rendering.* in *Proceedings of the Chapel Hill Workshop on Volume Visualization.* May 1989. Chapel Hill, NC.

9. Westover, L., *Footprint Evaluation for Volume Rendering.* Computer Graphics, August 1990. **24**(4): p. 367-376.

10. Laur, D. and P. Hanrahan, *Hierarchical Splatting: A Progressive Refinement Algorithm for Volume Rendering.* Computer Graphics, July 1991. **25**(4): p. 285—288.

11. Crawfis, R. and N. Max. *Texture Splats for 3D Vector and Scalar Field Visualization.* in *Visualization '93.* 1993. San Jose, CA: IEEE Computer Society Press.

12. Crawfis, R., N. Max, and B. Becker, *Vector Field Visualization.* Computer Graphics and Applications, 1994. **14**(5): p. 50-56.

13. Board, O.A.R., *OpenGL Reference Manual.* Release 1 ed. 1992, Reading, MA: Addison-Wesley. 388.

14. Board, O.A.R., *et al.*, *OpenGL Programming Guide.* Release 1 ed. 1993, Reading, MA: Addison-Wesley. 516.

15. Mitchell, D.P., *Generating Antialiased Images at Low Sampling Densities.* Computer Graphics, 1987. **21**(4): p. 65-72.

16. Cook, R.L., *Stochastic Sampling in Computer Graphics.* ACM Transactions on Graphics, Jan. 1986. **5**(1): p. 51-72.

Technical Information Department • **Lawrence Livermore National Laboratory**
University of California • Livermore, California 94551