√

Conf-931080--3

UCRL-JC-113626
PREPRINT

RECEIVED
JUL 19 1993
OSTI

Texture Splats for 3D Vector and Scalar Field Visualization

Roger A. Crawfis
Nelson Max

This paper was prepared for the submittal to the
Visualization '93
San Jose, CA
October 25-29, 1993

April 6, 1993

Lawrence
Livermore
National
Laboratory

This is a preprint of a paper intended for publication in a journal or proceedings. Since changes may be made before publication, this preprint is made available with the understanding that it will not be cited or reproduced without the permission of the author.

[illegible]

DISCLAIMER

This document was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor the University of California nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial products, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or the University of California. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or the University of California, and shall not be used for advertising or product endorsement purposes.

Texture Splats for 3D Vector and Scalar Field Visualization

Roger A. Crawfis (crawfis@llnl.gov)
Nelson Max (max2@llnl.gov)
Lawrence Livermore National Laboratory
P.O. Box 808 / L-301
Livermore, CA 94551

Abstract

*Volume Visualization is becoming an important tool for understanding large 3D datasets. A popular technique for volume rendering is known as **splatting**. With new hardware architectures offering substantial improvements in the performance of rendering texture mapped objects, we present **textured splats**. An ideal reconstruction function for 3D signals is developed which can be used as a texture map for a splat. Extensions to the basic splatting technique are then developed to additionally represent vector fields.*

Introduction

Westover has proposed two methods of using *splatting* to produce volume rendering. In the first method [Westover89], the color and opacity filter kernels for each voxel are composited one by one in back to front order (in regard to their center points). In the second [Westover90], the colors and opacities 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.

We have implemented the earlier method, taking advantage of the texture mapping and compositing features in our Silicon Graphics VGX rendering engine. As in [Westover90], we put the color and opacity values for a high resolution master splat into a texture map, and then use the texturing hardware to interpolate sampled values from these maps. We also use the RGBA compositing hardware to modify the frame buffer with each splat. The *Explorer* product from Silicon Graphics, Inc. uses the compositing scheme, but not the texture mapping in their volume rendering. We have added extensions to *Explorer* for texture mapped splats for both scalar and vector volume rendering.

Ideal Reconstruction Functions

Laur and Hanrahan [Laur90] also used the fast compositing hardware of the SGI VGX, but approximated each splat by a collection of polygons. Mach bands are visible at the polygon edges, and individual splats are visible, because they do not overlap smoothly. Each splat is typically created from fifteen to twenty-one triangles, or a triangle mesh. For architecture's that have hardware support for texture mapping, we can replace these many polygons with a single texture mapped square. Max [Max92] developed an ideal reconstruction function for images. We used this kind of function rather than a gaussian, since its overlap extent is well know and the function goes to zero in a minimum extent. This function was designed for the reconstruction of 2D signals (images), not 3D signals. By focusing on the reconstruction of three-dimensional signals, we have developed a reconstruction function for 3D splats that is accurate from all viewing directions. This function is tri-cubic, offering additional degrees of freedom in the optimization. Hence it is as accurate as [Max92] for orthogonal views perpendicular to the axes.

MASTER
[illegible]

xb

We have mathematically optimized the splats to give a smooth overlap, from any viewing direction, with the desired property of minimal extent. If h(x,y,z) is the 3D reconstruction filter kernel for a voxel at (0,0,0), its 2D footprint f(x,y) is given by:

$$f(x,y) = \int_{-\infty}^{\infty} h(x,y,z)dz \qquad (1)$$

As in [Westover89], we restrict ourselves to rotationally symmetric filter kernels, such that $h(x,y,z) = g(\sqrt{x^2 + y^2 + z^2}$, for a function g(r) of a single radius variable r. Our goal is to chose g(r) such that 1) g(r) $\in$ $\mathbf{C}^1$, eliminating mach bands, and 2) the splats overlap into a smooth density, hiding the structure of the individual splats.

To understand this second criteria, consider a cube filled with *nxnxn* voxels, all with intensity one. The orthogonal volume rendering projection of this cube should be as constant as possible, within the region bounded by the projections of two parallel faces, where the ray path lengths are equal. Since this should be true for all *n*, and all viewing angles, it presents a difficult optimization problem. Instead, we have chosen to minimize the relative variance (r.m.s. deviation from the mean, divided by the mean) of the reconstruction c(x,y,z) of this constant function

$$c(x,y,z) = \sum_{-\infty}^{\infty}\sum_{-\infty}^{\infty}\sum_{-\infty}^{\infty} h(x-i, y-j, z-k) \qquad (2)$$

with voxel centers at the integer lattice vertices (i,j,k). Our filter kernel has a finite support, so each of these infinite sums is in fact, finite.

Except for edge effects where all terms in this finite sum are not present, the total density for two comparable pixels in our cube projection comes from two integrals of (2) along parallel segments of the same length, and should not vary much if the 3D reconstruction does not.

The 3D optimization is then a 3D version of the 2D optimization of [Max92]. We assume g(r) is zero for r > t, and is represented by two cubic polynomials, p(r) for $0 \le r \le s$, and q(r) for $s \le r \le t$ (see Figure 1.). The condition that h(x,y,z) be $\mathbf{C}^1$ at the origin means that the linear term of p(r) is zero, hence, p(r) takes the form:

$p(r) = a + br^2 + cr^3$

The condition that q(r) meets the zero constant function in a $\mathbf{C}^1$ fashion are r=t means that q(r) takes the form

$q(r) = d(t-r)^2 + e(t-r)^3$

The condition that p(r) and q(r) meet in a $\mathbf{C}^1$ fashion at r=s gives two more linear constraints on the variables a,b,c,d and e. We solved for c and e in term of the other variables, a, b and d. Finally, since we were only interested in relative variation, we set a=1. This gives four independent parameters, s, t, b and d. We minimize the relative variance of the sum in equation (2) using an unconstrained optimization algorithm [Gay83]. This algorithm estimates the gradients and Hessian matrices needed for minimization from function evaluations. There is no absolute minimum, since the relative variance can be arbitrarily small if t is arbitrarily large. Therefore, we searched for a local minimum with a reasonably small t, which we found at t=1.556228, and s=0.889392. The relative variance was 0.00119, and the maximum relative deviations from the mean were -.00233 at x=y=z=0.26, and 0.00534, at x=y=0.5 and z=0.

By dividing the constants a, b, c, d, and e by the mean of c(x,y,z), scaling the reconstructed function to values near 1, we get:

$$g(r) = \begin{cases} 0.557526 - 1.157743r^2 + 0.671033r^3 & 0 \le r \le s \\ 0.067599(r-t)^2 + 0.282474(r-t)^3 & s \le r \le t \\ 0 & s \ge t \end{cases}$$

Figure 1. shows a graph of this function with s and t indicated by tick marks on the horizontal axis.

The integral (1) can be computed in closed form, since

$$f(x,y) = \int_{-\infty}^{\infty} g(\sqrt{x^2+y^2+z^2})dz = \int_{-\infty}^{\infty} g(\sqrt{r^2+z^2})dz$$

where $r = \sqrt{x^2+y^2}$. (Polynomials of low degree in $\sqrt{r^2+z^2}$ appear in tables of indefinite integrals.) This closed form solution was used to compute the footprint function f(x,y). Figure 2. shows *2x2* periods of a 2D projection along the z axis of one layer of glowing, non opaque volume splats. (In this case the relative deviation is independent of the number of layers.) The maximum projected value was 1.00249 at (.5,.5), and the minimum was 0.99845 at (.25,.25). The intensities in figure 2. were scaled to exaggerate the deviation approximately 250 times. From this we can see that the total variation is only about one part in 256, or one bit with most 8-bit color accuracy's.

Combined Vector and Scalar Textures

We can integrate vector fields into the scalar reconstruction function, by adding a slight disturbance in the function, such as tiny vector particles, or scratch marks. For the master splat, these vector indications are created in the x-axis direction (Figure 3.). During the splatting process, the vector field direction for each splat is determined and transformed to viewing coordinates. The projection of this vector (v_x,v_y) is then used to determine a rotation of the polygon splat, and the splat is rendered.

The splat is rendered using a portion of Figure 3a as an intensity map, and the same portion of Figure 3b for an opacity map (The use of Figure 3 will be described more fully in following sections). There are several texture mapping possibilities in GL [8]. We chose a two-component texture using a BLEND operator. A polygon's color $(R_{in},G_{in},B_{in},A_{in})$, will be modified as follows:

$$R = R_{in} * (1 - I_{tex}) + R_{const} * I_{tex}$$
$$G = G_{in} * (1 - I_{tex}) + G_{const} * I_{tex}$$
$$B = B_{in} * (1 - I_{tex}) + B_{const} * I_{tex}$$
$$A = A_{in} * A_{tex}$$

Using the texture maps in Figure 3, this function will give us $(R_{const},G_{const},B_{const})$ colored vectors, with the appropriately colored splats, both of which are attenuated by the polygon's opacity A_{in}. The vector color can be changed for each splat, allowing the vectors to be color coded by magnitude, or offering an additional three-dimensional cue.

So far, we have only indicated the xy-projection of the vector direction. No indication of the vector magnitude or the component of the vector directed towards the eye is given. The later can be represented by a foreshortening of the vector based on the viewing direction component magnitude in relation to the overall vector length. If we are only representing a vector field or if we separate the vector and scalar splats and discussed above, and easy method of achieving this is to simply shorten the polygon in the vector direction. This is simply the x-axis direction of the base splat, since we use the transformation pipeline to orient the splat in the direction of the vector. This however leads to noticeable thin splats (or regions of no activity). A second alternative is to change the texture mapping coordinates in the x-axis direction (increasing the frequency content of the resulting image). Unfortunately there is no way to automatically have the resulting repetitive texture windowed using current hardware. A third methods which will also work for combined scalar and vector fields, is to create a table of textures indicating different amounts of foreshortening. This represented across the columns in Figure 3. V_z is used to index into a column in Figure 3.

The series of splats represented in Figure 3, also are used to provide animation of the flow field. Going up the rows of Figure 3, we have the vector component of the splats moving across the scalar reconstruction. It should be noted that the vector component is windowed, but given a slightly larger extent than the scalar splat. We assign each splat a random index into the rows. For animated flows, a changing phase shift is added to the indexing.

System Design

Inventor / Explorer

With the help of SGI, we have extended the *VolumeToGeom* and *Render* modules of the *Explorer* system from SGI. The main enhancement is the inclusion of Textured Splats, Vector Splats, and combined Vector and Scalar Splats. The Render module in Explorer uses *Inventor*, an object-oriented graphics library, from which we developed C++ subclasses for the various splats. By adding a timer Sensor, available in Inventor, we can change the phase of the splats. We have created a set of sixteen windowed vector texture maps, where each map has the vectors propagated forward (and cyclically) in the map before being windowed (Figure 3). We attach a Slider (another useful Inventor feature) to the Timer, that allows the user to control the speed of the animation of the vector field (see video). The use of C++ allows us to easily extend the capabilities of the volume rendering when using splats.

Performance

The performance of the algorithm is dependent on the overall size of the splats. For many small scalar texture splats the algorithm is actually faster than the gaussian splats used be Explorer. This comes about from the trade-off of rendering one texture mapped square versus rendering a Gourand shaded triangle strip consisting of ten triangles. As the amount of screen spaces the splats occupy increases, the performance reverses, and the gaussian splats are faster than the textured splats. The relative difference in speed is never very substantial for the test cases we have run. The quality of the image is however much better using the textured splats with the ideal reconstruction kernel described above.

The vector splats incur about a fifty percent performance penalty in the matrix multiply required with each vector to transform it to screen space. The matrix inverse of the viewing matrix is needed for this, but the four by four matrix inverse is computed only once for the entire octree. There is also a minor amount of additional work in calculating the rotation matrix and the vector foreshortening.

Results

Figures 4 and 5 are volume renderings of a sample Explorer data set. Figure 4. was generated using the polygonal mesh with 3 radial samples and seven azimuth samples to approximate the gaussian. Figure 5 was generated using our ideal reconstruction function and texture mapped squares. Both images used a splat size of 1.6 or sixty percent overlap. Artifacts of the polygonal mesh and the rapid cutoff of the gaussian are clearly evident in Figure 4.

Figure 6 show another variation on this data set with a vector field inserted into it as a test case for the vector splatting. The scalar field is still adequately represented, while the additional vector field can easily be noted. Figure 7. uses this technique on climate data over North

America. The clouds are represented by the scalar field, and the winds (could) be represented by the vector field (I say could because we just got the valid data into the system the day this paper was being shipped off. Figure 8. has real vector data).

Figure 8 represents just the vector field. Here, rather than using a blend texture operation, we use a modulate texture operation, splatting only the vectors onto the image. The color of the vectors is given again by the underlying polygon, which we have mapped to the vectors magnitude.

Figure 9 is a representation of the clouds over North America, using the smooth texture splats for a scalar field.

Future Work

Most notably, better texture needs to be experimented with for better representation, particularly textures with an animated component. The texture used here was our first try, more effective textures certainly exist. Much work could also be done to improve the performance of the texture-mapping, paying particular attention to the amount of available texture map memory. We have many different application areas that we will be testing these techniques out on in the next few months. These include: fluid flow, electro-magnetics, underground water contamination, and structural mechanics. Since we can easily extend our C++ classes, multivariate splats, tensor splats, and possible combinations can be added. Our current plans include working on splats to represent relationships between two or three scalar variables and possibly a vector field, as well as representing two vector fields, such as the electrical (E) and the magnetic (H) fields in electro-magnetic simulations.

Acknowledgments

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract number W-7405-ENG-48, with specific support from an internal (LDRD) grant. The authors would like to thank SGI for their cooperation, in particular, Roy Hashimoto and Bob Brown. The Global Climate data is courtesy of the Program for Climate Model Diagnosis and Intercomparison and the European Centre for Medium-range Weather Forecasts.

References

[Gay83] Gay, David, *Algorithm 611*, Collected algorithms of the ACM, also in **ACM Transactions on Mathematical Software** Vol 9, No 4 (1983) pp. 503 - 524.

[Laur91] Laur, David, and Pat Hanrahan, *Hierarchical Splatting A Progressive Refinement Algorithm for Volume Rendering*, **Computer Graphics** (SIGGRAPH 91), Vol 25 No. 4, pp 285 - 288.

[Max91] Max, Nelson, *An Optimal Filter for Image Reconstruction*, in **Graphics Gem II**, James Arvo (ed), Academic Press, New York, pp. 101 - 104.

[Westover89] Westover L., (1989) *Interactive Volume Rendering*, **Proceedings of the Chapel Hill Workshop on Volume Visualization**, Department of Computer Science, University of North Carolina, Chapel Hill, NC, pp 9 - 16.

[Westover90] Westover L., (1990) *Footprint Evaluation for Volume Rendering*, **Computer Graphics** (SIGGRAPH 90), Vol 24 No. 4, pp 367 - 376.

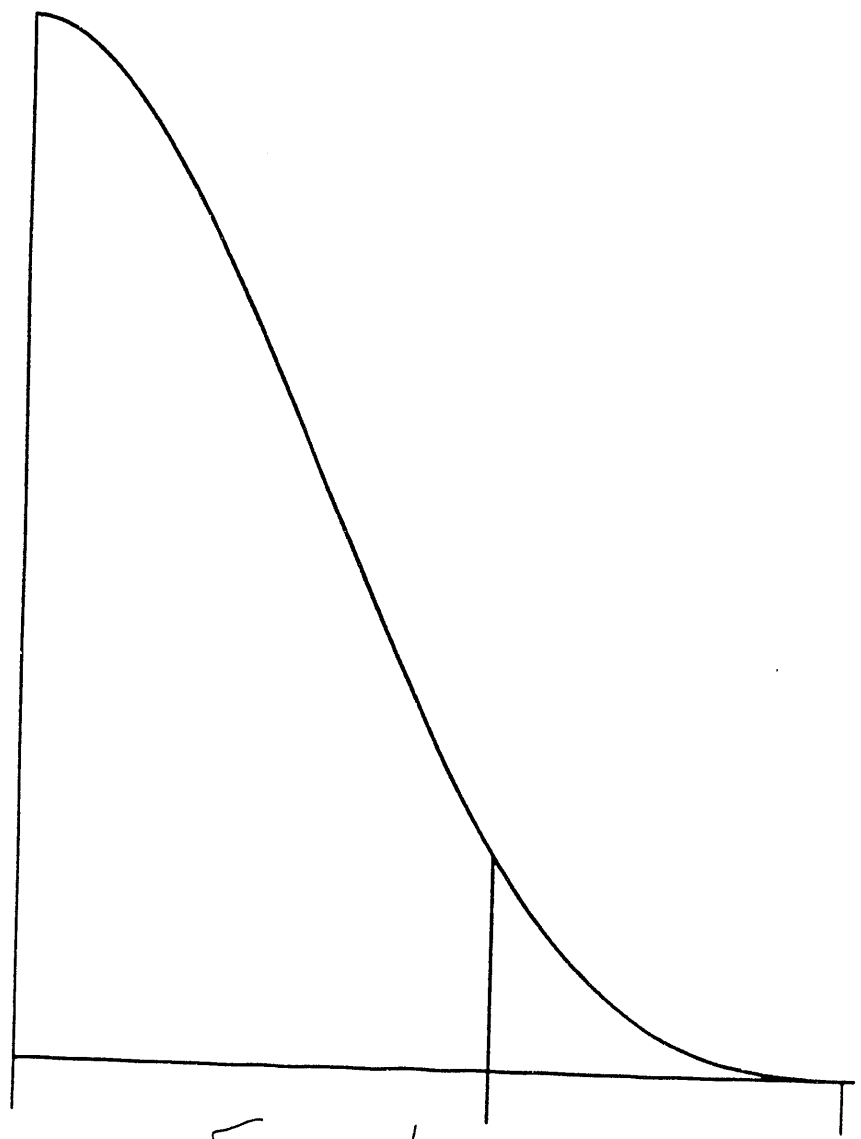

Figure 1

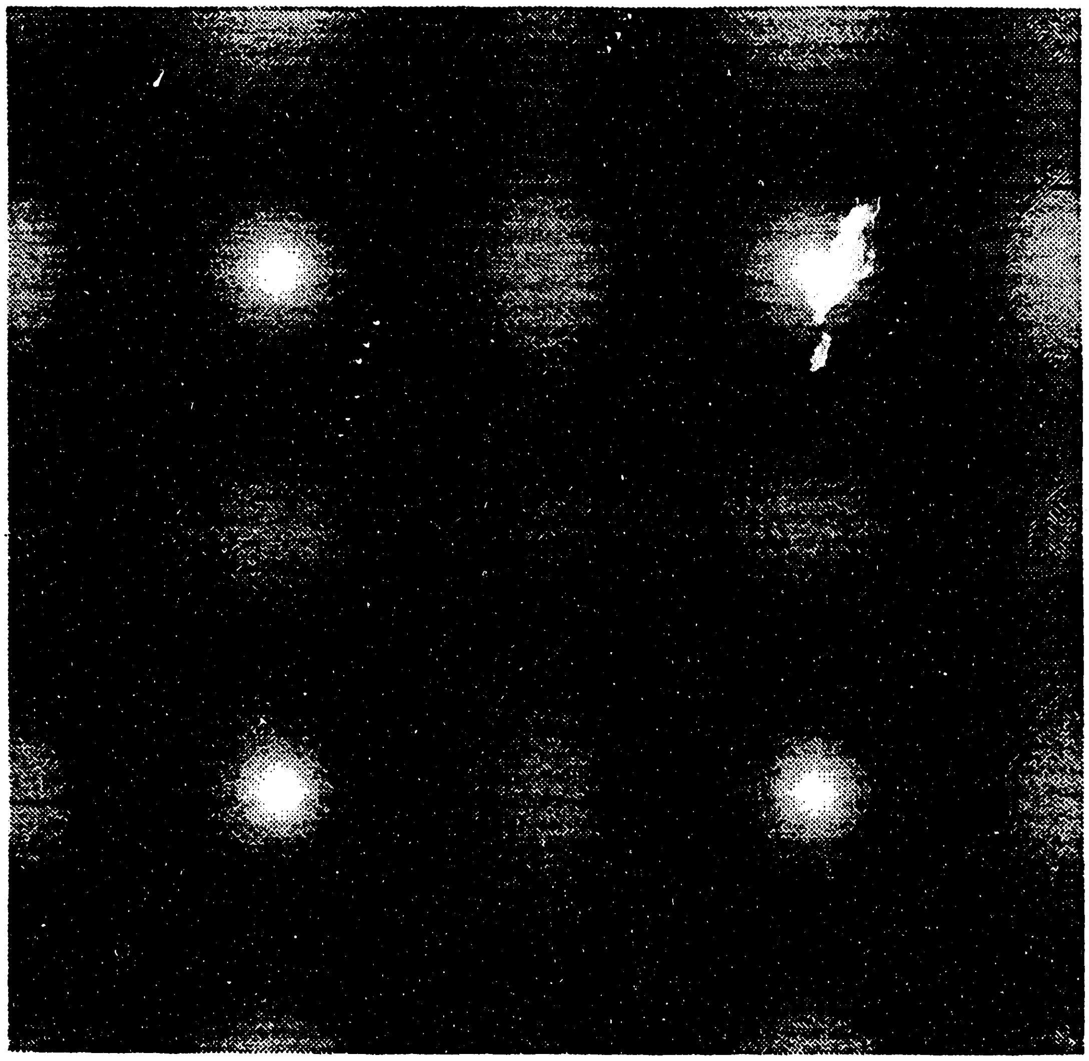

Figure 2

Figure 2

7

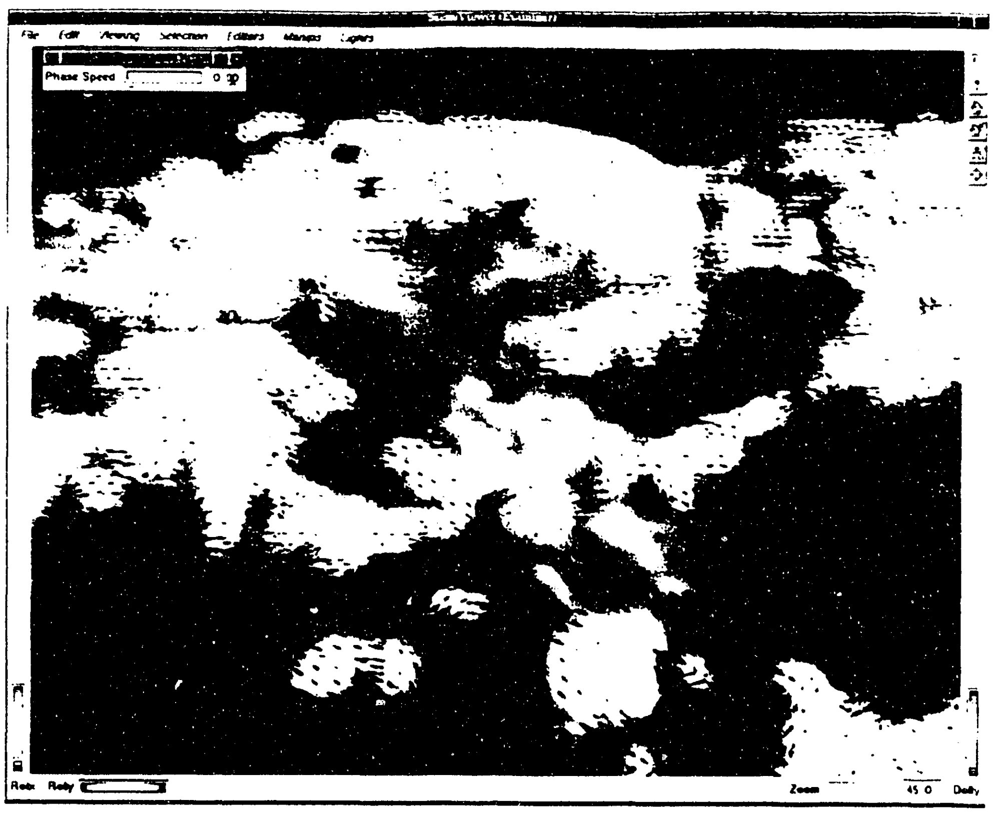

Figure 7. Clouds and Winds(??) over North America.

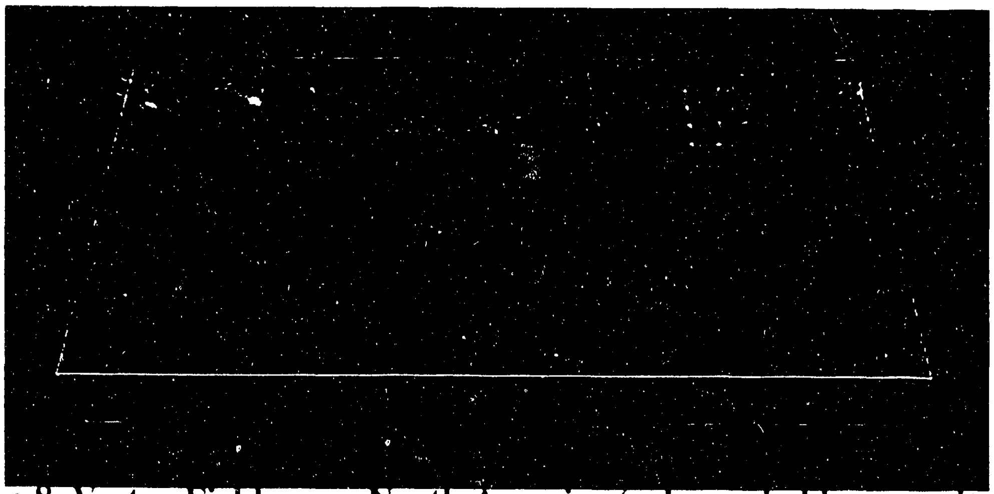

Figure 8. Vector Fields over North America (color coded by magnitude).

Figure 9. Smooth Splatting of Clouds over North America.

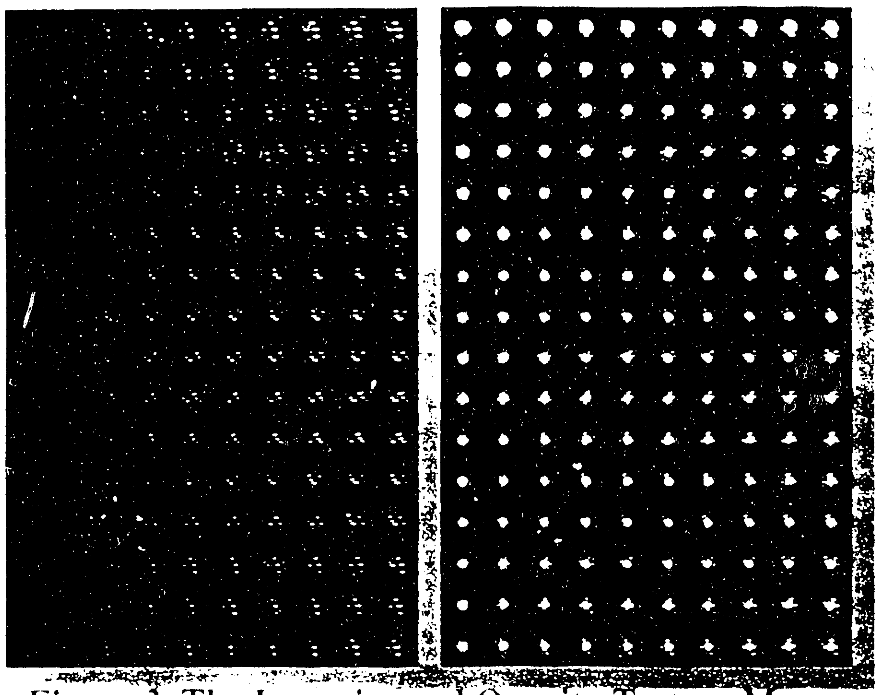
Figure 3. The Intensity and Opacity Texture Maps

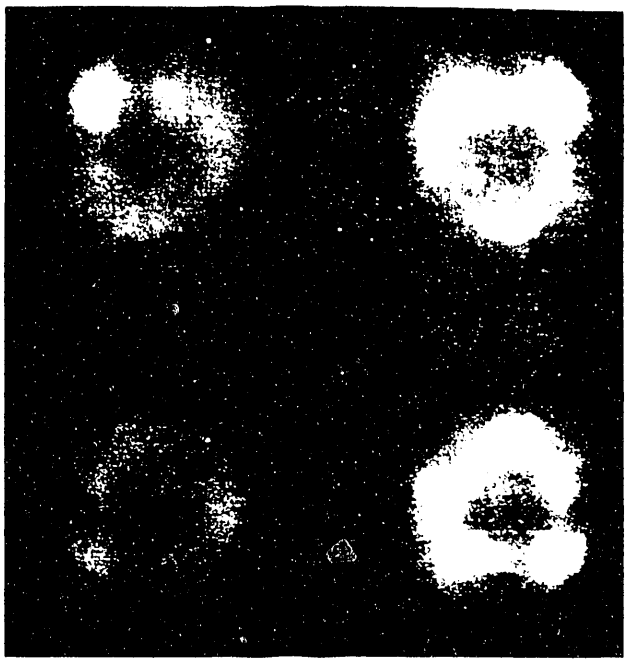
Figure 4. Polygonal Splats with a Gaussian Reconstruction Function.

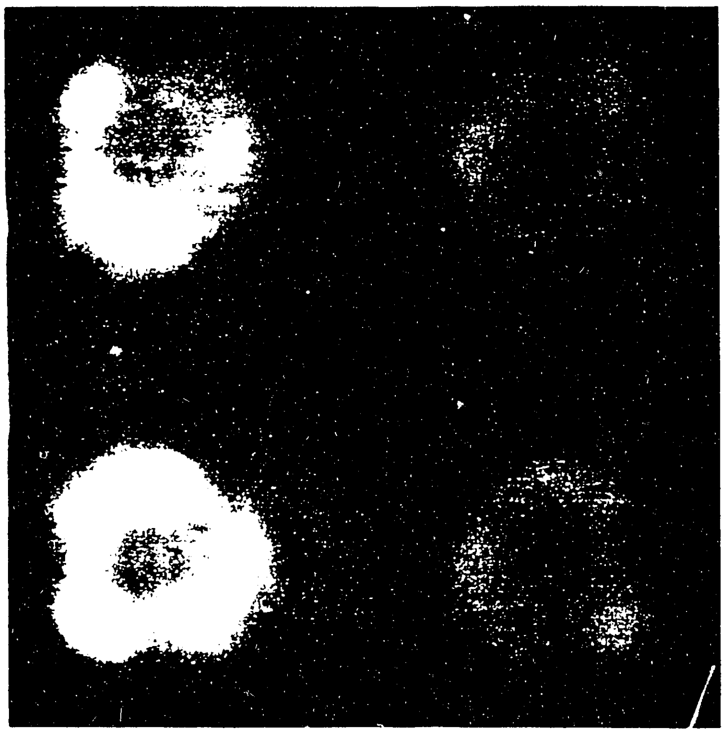
Figure 5. Textured Splats with an Ideal Reconstruction Function.

Figure 6. Vector and Scalar Splatting

END

DATE
FILMED

9/17/93