Texture Splats for 3D Scalar and Vector 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 data sets. 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 Silicon Graphics rendering engines. 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, 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 architectures that have hardware support for texture mapping, we can replace these many polygons with a single texture mapped square. Max [Max91] developed an optimal piecewise quadratic reconstruction function for images. We used this kind of function rather than a gaussian, since its overlap extent is well known 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 a piecewise cubic, offering additional degrees of freedom in the optimization. Hence it is as accurate as [Max91] for orthogonal views perpendicular to the axes.

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 n x n x n voxels, all emitting an intensity of one and with no opacity. The orthogonal volume rendering

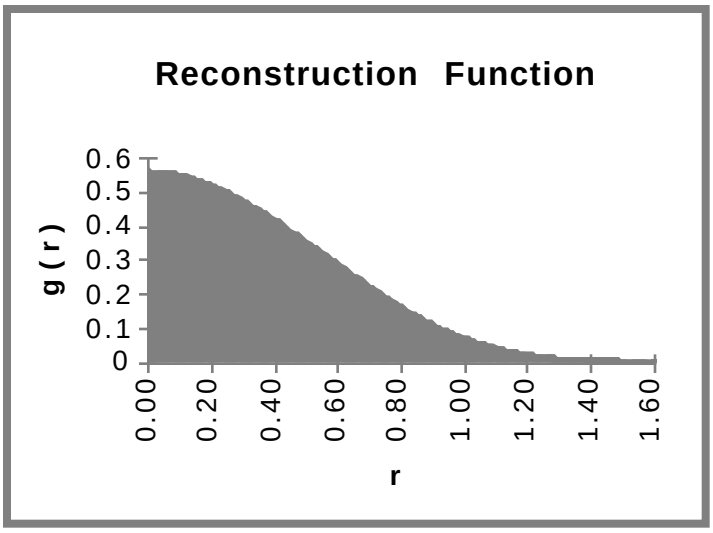

Figure 1. Ideal reconstruction function

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 [Max91]. 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 terms 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 minimized 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, and found one at $t = 1.556228$, and $s = 0.889392$. The relative variance was 0.00119, and the maximum relative deviations from the mean were -0.00233 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:

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

Figure 1. shows a graph of this function.

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 *2* x *2* periods of a 2D

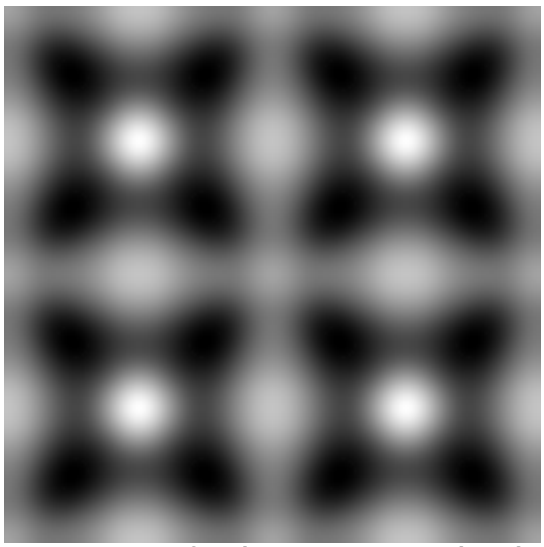

Figure 2. Error of volume reconstruction (x250)

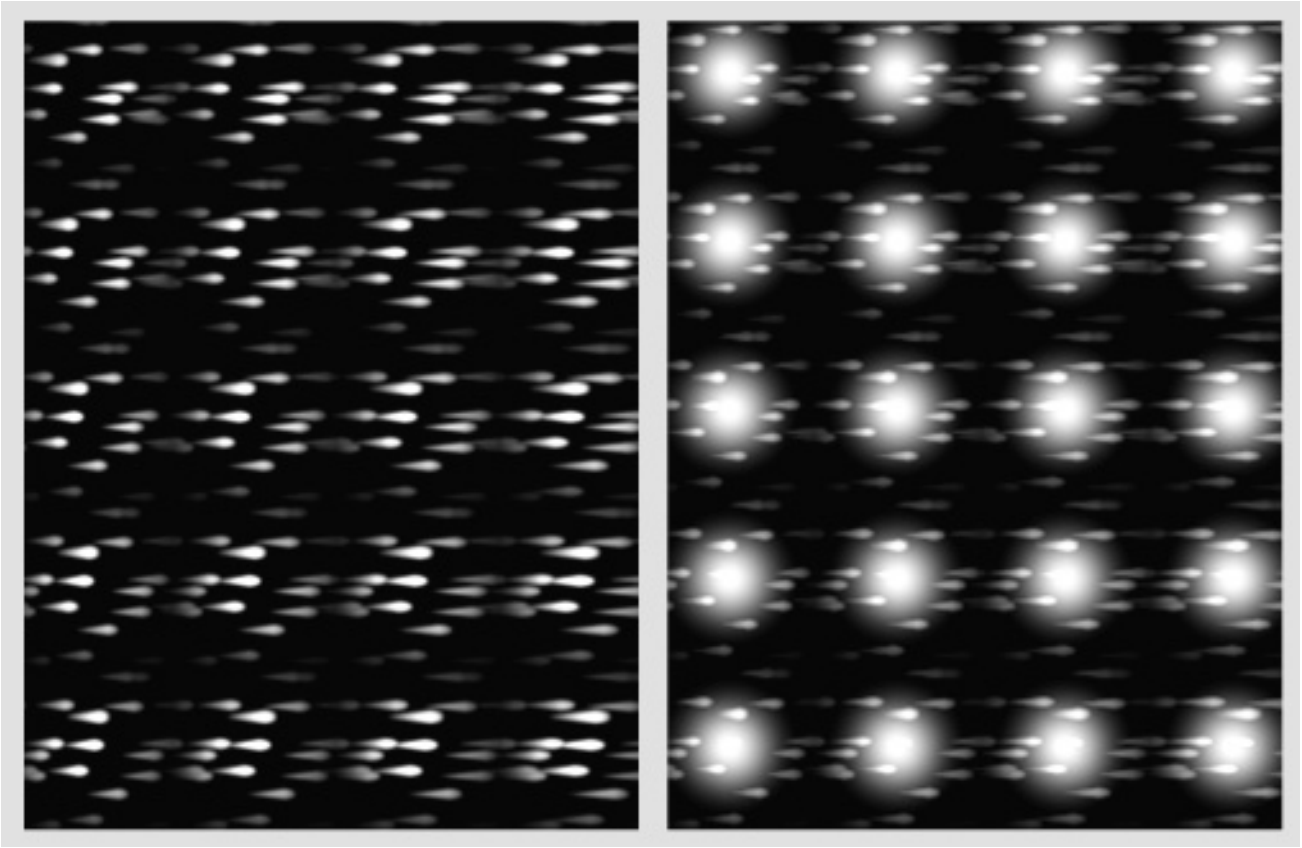

Figure 3: a) Portion of Intensity Table b) Portion of Opacity Table

projection along the *z* axis of one layer of glowing, completely transparent 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.

We use this function to generate a texture that is used as the splat. The texture is generated with an extent of 1.6 and the splats are built up in back to front order. Figure 4b was generated using this technique.

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. illustrates a series of twelve such splats. These twelve are part of a larger table that will be explained later in this paper. For the splatting of vector splats, two additional calculations must be carried out. First, 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 matrix for the polygon splat.

The splat is rendered by selecting a splat from the splat table in Figure 3a as an intensity map, and for scalar fields, the same splat in Figure 3b for an opacity map. There are several texture mapping possibilities in SGI's GL graphics library [GL91]. We chose a two-component texture consisting of an intensity and an opacity component for our texture splats. Different texture mapping operations are used for representing only a vector field versus a scalar and vector field.

The MODULATE operator [GL91] multiplies the polygon's color by the splat's intensity and opacity. This operator is used for representing a single vector field. The splats in Figure 3a are used for both the intensity and opacity. The resulting splat's color will thus be:

$$R = R_{in} * I_{tex}$$
$$G = G_{in} * I_{tex}$$
$$B = B_{in} * I_{tex}$$
$$A = A_{in} * A_{tex}.$$

The polygon's color () (the resulting splat's color) can be used to convey the magnitude of the vector field. Alternatively, we can map a separate scalar field or use an axes coordinate as an additional spatial cue.

The BLEND operator [GL91] will produce a dissolve between the polygon's color (R_{in},G_{in},B_{in}) and a specified constant color using the splats intensity (I_{tex}) as the fraction to take from each. The polygon's opacity (A_{in}) is modified as with the MODULATE operator. The resulting splat's color will thus be:

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

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 by tying the color mapping to the splat's world coordinate position.

When representing a single vector field, we can stretch the polygon in the vector direction, producing a "streaky" or paint brush affect.

These operations define the texture mapping used to create a data dependent splat with the proper transparency and colors. This splat is what is then composited into the final image.

So far, we have only indicated the xy-projection of the vector direction. No indication of the component of the vector directed towards the eye is given. This can be represented by a foreshortening of the vector based on the viewing direction component in relation to the overall vector length. If we are only representing a vector field or if we separate the vector and scalar splats, an 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. 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 method which will also work for combined scalar and vector fields, is to create a table of textures indicating different amounts of foreshortening. This is represented across the columns in Figure 3. The *z* component v_z of the vector direction 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. These modules implement the octree volume rendering scheme of Laur and Hanrahan [Laur91]. 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 (rows of 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 the accompanying Video Proceedings). The use of C++ allows us to easily extend the capabilities of the volume rendering when using splats.

Performance

The performance of the algorithm on scalar data sets 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 by Explorer. This comes about from the trade-off of rendering one texture mapped square versus rendering a Gourand shaded triangle strip consisting of many triangles. As the amount of screen space the splats occupy increases, the performance reverses, and the polygonal 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 when using the textured splats with the optimal 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 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 4a and 4b are volume renderings of a sample Explorer data set. Figure 4a was generated using the polygonal mesh with 3 radial samples and seven azimuth

samples to approximate the gaussian. Figure 4b was generated using our ideal reconstruction function and texture mapped squares. Both images used a splat size of 1.6. Artifacts of the polygonal mesh and the rapid cutoff of the gaussian are clearly evident in Figure 4a.

Figure 5 illustrates the vector splatting on a dummy test tornado data set. The magnitude of the vector field is used to control the opacity and a noisy color map of browns is used to add some variations. Figure 6 shows a global climate data set. Here, the jet stream winds are represented by the vector splats. The magnitude of the winds is mapped to both opacity and color, and a stretched vector splat is used to provide a wispy appearance in the wind field direction. Figure 7 uses the GL blend operation to add white vectors flowing through the volume rendering of the magnitude of an airflow through an aerogel (ultr light weight insulator) substance. Polygon data representing the aerogel fibers are embedded into the volume rendering. Figure 8. represents the scalar field percent cloudiness and the vector field of winds of the climate data over North America. The winds are color coded to give an indication of the placement within the altitude and are also restricted to where there is a high concentration of clouds.

Future Work

We need to experiment with different textures 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. The aerogel data is courtesy of Tony Ladd and Elaine Chandler, LLNL

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.

[GL91] **Graphics Programming Library Guide.** Silicon Graphics, Inc. 1991.

[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 4. a) Gaussian Splats b) Texture Splats with an optimal reconstruction texture.

Figure 5. Test Tornado

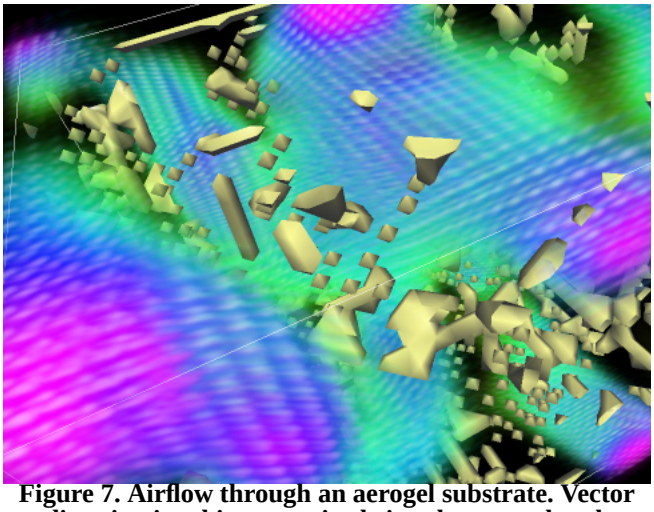

Figure 7. Airflow through an aerogel substrate. Vector direction in white, magnitude is volume rendered.

Figure 6. Wind Velocities (jetstreams). Color mapped to wind magnitude.

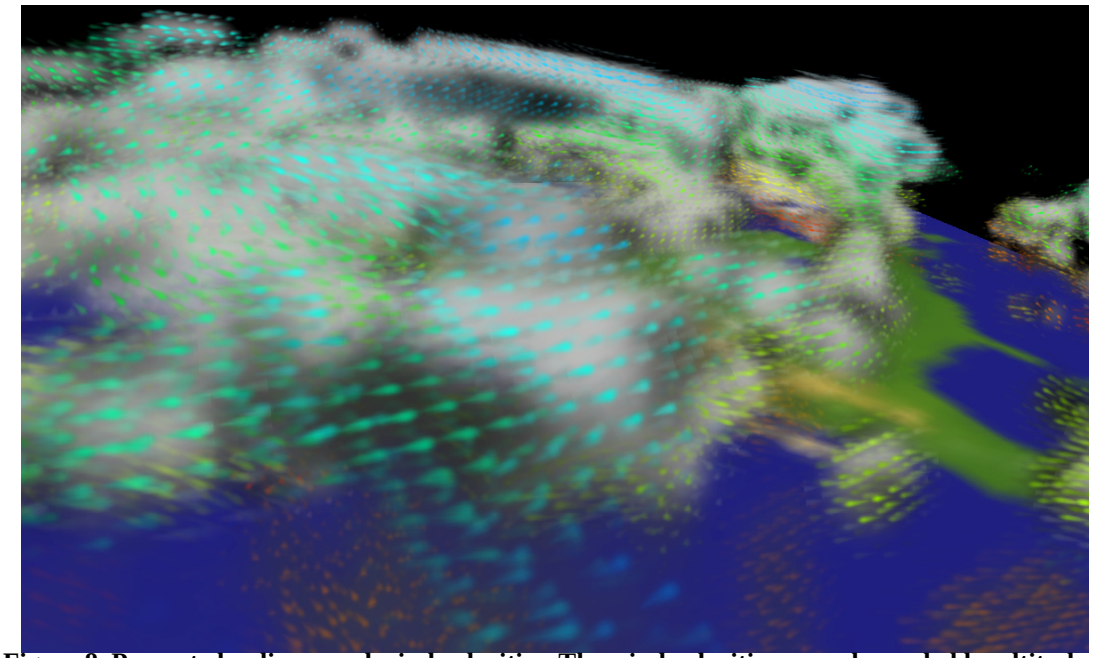

Figure 8. Percent cloudiness and wind velocities. The wind velocities are color coded by altitude.