Theoret. Comput. Fluid Dynamics (1998) 11: 195–211

Theoretical and Computational Fluid Dynamics

Volume Visualizing High-Resolution Turbulence Computations[1]

John Clyne and Tim Scheitlin

Scientific Computing Division, National Center for Atmospheric Research,
Boulder, CO 80307-3000, U.S.A.

Jeffrey B. Weiss

Program in Atmospheric and Oceanic Sciences,
Campus Box 311, University of Colorado, Boulder, CO 80309, U.S.A.

Communicated by J.R. Herring

Received 30 April 1997 and accepted 27 August 1997

Abstract. Using several volume-visualization packages including a new package we developed called Volsh, we investigate a 25-Gbyte dataset from a 256^3 computation of decaying quasi-geostrophic turbulence. We compare surface fitting and direct volume rendering approaches, as well as a number of techniques for producing feature-revealing spatial cues. We also study the pros and cons of using batch and interactive tools for visualizing the data and discuss the relative merits of using each approach. We find that each tool has its own advantages and disadvantages, and a combination of tools is most effective at exploring large four-dimensional scalar datasets. The resulting visualizations show several new phenomena in the dynamics of coherent vortices.

1. Introduction

High-resolution computations of turbulence generate enormous four-dimensional datasets that contain detailed information on turbulent processes. Traditional statistical measures are easy to compute, but in reducing the data to relatively few numbers they filter out most of the information that has been computed at great expense. Visualizations of still images reveal important details about the instantaneous structure of the turbulence. Such images are necessarily two-dimensional projections of spatially complex three-dimensional fields, and it is often difficult to interpret the three-dimensional structure from a single image. Given the human brain's ability to process visual information, the most efficient method of examining large datasets and gaining understanding of turbulence is to view animations of the data. Animations of spatial rotations of fields at a single instant in time allow us to clarify the three-dimensional structure, while temporal animations, perhaps combined with spatial rotations, show the evolution of that three-dimensional structure. The software and hardware to create animations of large datasets is becoming increasingly available and easy to use. However, there are a number of issues that arise in visualizing high-resolution four-dimensional datasets. In this paper we describe our experiences in volume visualizing high-resolution computations of quasi-geostrophic (QG) turbulence, focusing on the relative merits of different visualization techniques and

[1] This work was partially supported by NSF ECS-9217394, DOC NA56GP0230, and the National Center for Atmospheric Research, which is operated by the University Corporation for Atmospheric Research under the sponsorship of the National Science Foundation. The dataset was calculated under NSF MetaCenter Grant MCA93S010P.

how the resulting animations have increased our understanding of QG turbulence. We discuss a few popular volume-visualization packages as well as a new volume-visualization tool, Volsh, that we produced to complement existing software. We expect that the experience we gained in visualizing these data will carry over to high-resolution computations of other turbulent flows.

The QG equations describe the asymptotic regime of rapid rotation and strong stable stratification and are thus an appropriate description for large-scale motions in the atmosphere and oceans (Pedlosky, 1987; McWilliams, 1991). The importance of rotation and stable stratification are measured by the smallness of the Rossby number, $R = U/fL$, and the Froude number, $F = U/NH$, respectively, where U is a horizontal velocity scale, f is the Coriolis parameter, L is a horizontal length scale, N is the Brunt–Väisälä frequency, and H is a vertical length scale. In so-called stretched coordinates, where the vertical dimension has been scaled by N/f, the equations for decaying QG flow with constant rotation (f-plane) and constant N are

$$\partial_t q + J(\psi, q) = D, \qquad q = \nabla^2 \psi, \tag{1}$$

where q is the potential vorticity, ψ is the streamfunction, $J(\psi, q) = \partial_x \psi \, \partial_y q - \partial_y \psi \, \partial_x q$ is the horizontal Jacobian, D is the dissipation, and $\nabla^2 = \partial_x^2 + \partial_y^2 + \partial_z^2$ is the three-dimensional Laplacian. It is important to note that the QG equations describe a flow with no vertical velocity; fluid parcels are only advected horizontally via the Jacobian. The streamfunction is related to the horizontal velocity (u, v) by $u = -\partial_y \psi$, $v = \partial_x \psi$. Despite being two dimensional, the advecting velocity field depends on the entire three-dimensional flow field through the three-dimensional Laplacian relating q and ψ.

In this paper we discuss visualizing the results of a direct numerical simulation of (1) with periodic boundary conditions in all three directions, starting from random initial conditions. Dissipation is modeled using a hyperviscous operator, $D = -\nu \nabla^4 q$, where ν is a hyperviscosity coefficient. Hyperviscosity allows the simulation to reach a higher effective Reynolds number than would Newtonian viscosity. In two dimensions, hyperviscosity does not significantly affect the aspects of the solution we are interested in here; while no detailed comparisons have been done for the QG equations, there is no reason to believe that hyperviscosity introduces artifacts. The equations were integrated on grid resolutions of up to 320^3; here we discuss visualization of a 256^3 computation which shows essentially the same phenomena as the higher-resolution case. The integration was carried out on a Cray T3D using a parallel multigrid algorithm (Baillie *et al.*, 1995; Yavneh and McWilliams, 1996).

The statistical properties of QG turbulence and the general form of the coherent structures are now well established (Charney, 1971; Herring, 1980; Hua and Haidvogel, 1986; McWilliams, 1989; McWilliams *et al.*, 1994; McWilliams and Weiss, 1994). QG turbulence is characterized by an inverse cascade of energy, where energy flows from small to large scales. As a result, the energy is removed from the small scales where dissipation acts, and it is conserved. Enstrophy (mean square vorticity) undergoes a direct cascade to small scales and is dissipated. Traditional theories of QG turbulence ignore the possibility of coherent structures and predict isotropic spectra (in stretched coordinates) with an inertial-range energy decay proportional to k^{-3} (Charney, 1971).

Previous analysis of these computations showed that QG decaying turbulence deviates significantly from the theoretical predictions, and that this deviation is associated with the emergence of coherent vortices (McWilliams, *et al.*, 1994). Before vortex formation the spectrum decays as k^{-3}, but as vortices form the slope steepens. In addition, the assumption of spectrum isotropy is found to be false.

It is becoming increasingly clear that coherent vortices are the primary cause of the failure of traditional theories, and that turbulence is strongly influenced by the dynamics and evolution of the vortices. Coherent vortices also play an important role in determining the transport properties of turbulence by coherently carrying fluid parcels large distances (McWilliams and Weiss, 1994; Weiss, 1994; Babiano, *et al.*, 1994). Thus, it becomes important to understand the evolution and interactions of coherent vortices. The best method to attain a qualitative understanding of vortex behavior, and thus the first step to achieving a quantitative understanding, is to volume visualize the vorticity field.

Examining still images of the three-dimensional potential vorticity field at numerous times and from different spatial perspectives resulted in the following view of coherent vortex evolution (McWilliams *et al.*, 1994; McWilliams and Weiss, 1994). A random initial potential vorticity field self-organizes into a collection of nearly spherical, but slightly flattened coherent vortices. As the flow evolves, the population of coherent vortices changes through two primary interactions: horizontal merger and vertical alignment. This evolution is seen in Figure 1.

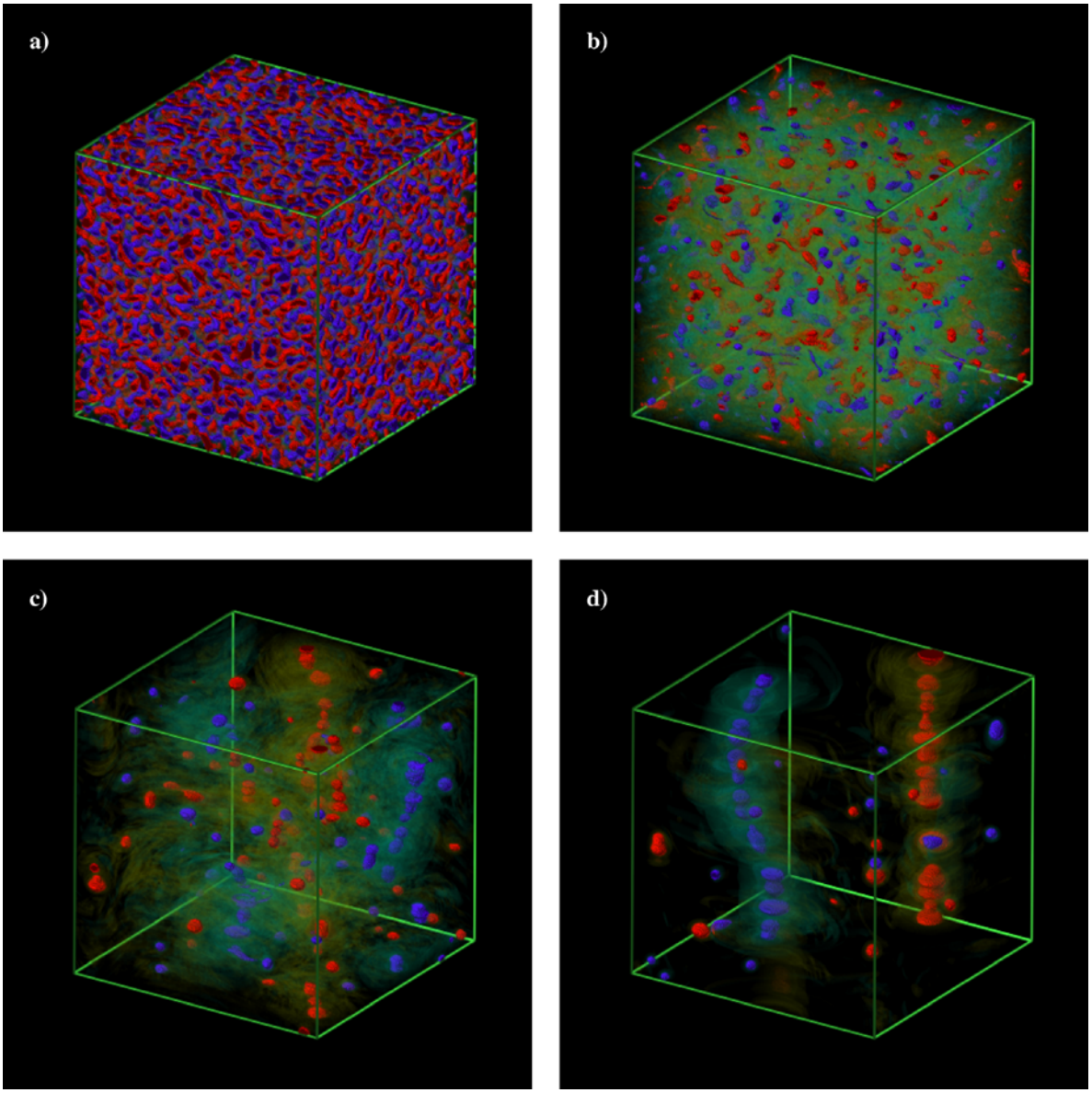

Figure 1. Time evolution of the potential vorticity field showing the emergence and evolution of the coherent vortices. The field is shown at (a) the initial time, $t = 0$, (b) $t = 5.4$, (c) $t = 26.9$, and (d) the final time, $t = 80.0$.

Merger occurs when two same-sign vortices at roughly the same vertical level approach closely. They then merge to form a larger vortex and, in the process, shed vortex filaments, which are stretched out to small scales, resulting in enstrophy dissipation. Vortex merger is relatively well understood due to its similarity with merger in two-dimensional turbulence.

Vortex alignment is an intrinsically three-dimensional phenomenon and is poorly understood. When two same-sign vortices are vertically aligned, they sometimes remain aligned in their subsequent evolution even though the potential vorticity field falls to very small values between the two vortices. Because the motion is purely horizontal, the vertically separated vortices can never merge. The process of alignment is poorly understood and the stability of aligned vortices in response to vertical shears is not known. The temporal animations discussed here shed significant new light on the alignment process.

As a result of horizontal merger and vertical alignment, the turbulence evolves until it eventually consists of two columns, one with positive and one with negative potential vorticity. The columns are not barotropic

(i.e., independent of z), but consist of individual vortices separated by regions of very small potential vorticity. It is possible that the vertical scale of the individual vortices within the column is set by an instability recently studied by Dritschel and de la Torre Juárez (1996). The axes of the final columns are not perfectly vertical but support transverse waves. The animations discussed here show, for the first time, some of the possible modes and how these oscillations are excited.

Because the previous analysis relied on static images, the process was quite tedious and only a small fraction of the computed data could be viewed. Further, it did not reveal features of the dynamics that only become apparent when viewing temporal animations. In Section 2 we review the basics of volume visualization and present the framework for the discussion in Section 3 of the environment and tools used to visualize the QG data. In Section 4 we present the results of our analysis and describe how volume visualization has yielded new insights into the dynamics of QG turbulence.

2. Volume Visualization

Volume visualization is the subfield of scientific visualization that is concerned with the representation, manipulation, and display of scalar fields in three dimensions. In essence, volume visualization is a process for projecting a three-dimensional volume of data onto a two-dimensional image plane for the purpose of providing insight into the features and structures contained within. Though its origins are strongly rooted in the medical field, volume-visualization techniques may be applied to scalar data from any scientific discipline. The algorithms covered here are generally constrained to uniformly spaced, scalar data on a Cartesian grid. Papers addressing volume visualization of nonuniformly structured, or unstructured, data may be found in (IEEE Computer Society Press, 1996).

2.1. Volume Visualization Pipeline

There are two fundamental volume-visualization approaches currently in practice: surface fitting (SF), also called isosurfacing or surface-based rendering, and direct volume rendering (DVR). In SF, a threshold value is specified and a surface detector applies this value to the data volume and "fits" geometric primitives to the isovalued surface. The geometric primitives may then be displayed using conventional geometry rendering techniques. SF is considered an indirect approach because of the required conversion of volume data elements into geometric primitives prior to rendering (Kaufman, 1991). DVR methods, in general, require no such intermediate representation: the sample array is projected directly to the image plane without first fitting geometric primitives to the data (Levoy, 1988).

Both SF and DVR share a number of processes, called the *volume-visualization pipeline* (Kaufman, 1991). This pipeline attempts to capture the logical sequence of operations necessary to visualize a volume of data. In practice, the order in which these stages are presented may differ, or in some implementations, multiple stages may be combined. The intent here is not to provide a rigorous architecture, but only to provide a sense of the processes involved.

Lastly, the reader should bear in mind that to produce artifact-free, accurate visual representations of the data, it is imperative that at each step in the pipeline every effort is made to preserve the natural continuity of the data (Drebin *et al.*, 1988). Nonlinear operations, such as thresholding or all-or-none decisions, are to be avoided wherever possible.

The first step in the volume-visualization pipeline is data preparation, beginning with data acquisition. The QG data discussed herein were obtained through numerical simulation of the fluid equations (1) followed by a discretization of the potential vorticity field from floating-point values down to byte values, i.e., an integer from 0 to 255. In the language of volume visualization, the data samples are now referred to as *voxels*.[1] A voxel, or *volume element*, is the three-dimensional analogy to a *pixel*, or *picture element*.

In the next step the voxels must be classified. Classification, the most user-demanding stage of the pipeline, is the process of determining the visibility and possibly the appearance attributes (e.g., color) of

[1] There is some variation in the definition of a voxel with regard to its dimensionality throughout the volume-visualization literature. In the context of this paper, a voxel refers to a dimensionless sample point, located at an integer grid coordinate.

the individual voxels. Classification determines which features or structures in the data will be visible in the resulting image, and in the case of DVR, the classification process may also partially determine how those features will appear.

For SF techniques, classification implies selecting a threshold value that defines the surface of interest. For DVR techniques, classification implies defining a mapping between voxel values and opacity values, and possibly between voxel values and appearance attributes as well. This mapping is most often accomplished via a transfer function table (e.g., a color or opacity map). By choosing appropriate transfer functions one can highlight or hide important aspects of the data (one also runs the risk of finding only what one is looking for). Transfer functions should generally be constructed to be smooth and continuous if the continuity of the data itself is to be preserved.

Classification is typically a highly interactive procedure. Choices are made, results are checked, and the process is repeated. Because the results depend so sensitively on the transfer function, we found that several different transfer functions are sometimes required to get an accurate picture of the data. Familiarity with the data can greatly reduce the number of iterations involved (Elvins, 1992).

Following classification, voxels are mapped into display primitives. With SF, voxels are mapped into geometric primitives (e.g., points, lines, polygons). Most common SF techniques are based on the Marching Cubes algorithm (Lorensen and Cline, 1987), which represents surfaces as a triangle mesh. With DVR, the voxels themselves are the display primitives: no mapping is required.

Display primitives created during the mapping step are projected to the two-dimensional image plane. Projection of geometric primitives is simply performed using conventional three-dimensional geometry rendering. Direct projection of volume primitives, required by DVR methods, necessitates the use of special volume-projection algorithms, loosely referred to as ray-casting (Drebin *et al.*, 1988; Lacroute and Levoy, 1994; Levoy, 1988; Sabella, 1988; Upson, 1988; Westover, 1990).

The projection stage determines which display primitives are visible to the observer, as well as their location on the two-dimensional image plane. Shading and illumination, the last step in the volume-visualization pipeline, determines the visible light contribution (color and intensity) each primitive makes toward the final image. Shading and illumination algorithms vary widely in how they impact the appearance of the final image.

Illumination models, which model the intensity of light at a point, are based on the premise that the appearance of an object is determined by the light the object reflects, transmits, and emits (Foley *et al.*, 1990). Shading algorithms compute the visible light that reaches an observer by employing an illumination model at select locations on the visible display primitives. The two most common shading models found in practice are Gouraud shading (Gouraud, 1971) and Phong shading (Bui-Tuong, 1975). Phong shading typically yields vastly superior results over Gouraud.

Illumination models are classified as either local or global. Global illumination models attempt to model the complex interaction of light with all visible surfaces. They are capable of creating global shading effects such as shadows and reflections. However, global models are significantly more computationally expensive, not necessarily advantageous for scientific purposes, and are consequently rarely used in volume visualization. See Sobierajski and Kaufman (1994) for an example of one volume renderer that employs global illumination.

Local illumination models are not able to capture global lighting effects but they are simple to implement and their computational costs are low enough to permit interactive shading (their principal attraction for use in visualization). Local illumination models perform their calculations using only localized information such as a surface normal, a point light source, and viewer position. Illumination models that take advantage of surface normal information are essential if the three-dimensional shape of a feature is to be accurately represented. In volume visualization, surface normals are generally recovered from the data by using a local gradient estimator. Central-differencing is adequate for most purposes and is probably the most common technique. More expensive methods, such as spline-based approaches, are also used (Bentum *et al.*, 1996).

2.2. Surface Fitting Versus Direct Volume Rendering

While the volume-visualization pipelines for SF and DVR are similar there are significant differences between these two volume-visualization methods. By using intermediate geometric primitives, SF techniques offer

one important advantage over DVR: the ability to project and shade intermediate geometric primitives using conventional graphics hardware. Commercial graphics accelerators, aimed at speeding up the rendering of geometric primitives, are inexpensive and widely available. With current technology it is possible to render static or even time-evolving geometric representations of complex surfaces interactively for moderately large (256^3) datasets.

DVR techniques, on the other hand, are memory and computationally expensive. Commercial products for accelerating the task are virtually nonexistent. Interactive rendering of static datasets is difficult to achieve. Interactive rendering of time-evolving data is possible only on the largest of supercomputers, and only with moderately sized individual time steps.

However, despite its computational costs, DVR offers several advantages over SF techniques. Amorphous features such as clouds or fog simply cannot be represented as surfaces. DVR handles these features nicely. The surface detection process itself is error prone, particularly if the original data do not contain clear surface boundaries. Visualization artifacts, particularly in high-frequency data, such as spurious surfaces (false positives) and erroneous holes (false negatives), are possible. Useful information that does not lie on a chosen surface is lost; only a small subset of the data is actually presented. Lastly, geometric primitives can only approximate a surface.

2.3. Spatial Cues

Perhaps the greatest challenge of any volume-visualization approach is to represent unambiguously the spatial relationships of a three-dimensional object, or objects, on a two-dimensional display. Volumetric display technology is an active area of research (Blundell *et al.*, 1994; Lucente and Galyean, 1995). However, no viable commercial volumetric displays currently exist. To discern the relative depth (distance from the viewer) of objects on the screen unambiguously, a number of spatial cues may be used. We conclude this section with a brief listing of some of the spatial cues used by the human visual system and discuss how these cues may be applied in computer graphics. For further information, see McAllister (1993).

Motion parallax: Motion of objects relative to each other can provide a strong sense of relative object depth. For example, as we move our head from left to right, or up and down, objects that are close appear to move more than objects that are further away. Motion parallax may be easily achieved in visualization by simply transforming the view point in an animation sequence (e.g., rotating about an axis parallel to the image plane).

Binocular disparity: Because of the approximate 2.5-in. horizontal separation between the left and right eye, each of our eyes has a slightly different viewpoint. The closer an object, the more pronounced this difference. Binocular disparity can be simulated by rendering separate left and right eye images and using special stereo display technology to view the left/right pairs. Binocular disparity is considered one of the strongest spatial cues (Julesz, 1971).

Shading: The amount of light reflected from an object toward the eye is strongly related to the object's surface normal, the location of the viewer, and the location of any light sources. The light reflected from a curved object varies depending on the surface normal at a given point, giving strong clues about the object's shape. Shading cues may be taken advantage of by using illumination models that incorporate surface-normal information, such as the Phong illumination model (Bui-Tuong, 1975).

3. Visualizing Quasi-Geostrophic Turbulence

In this section we first explore some of the QG data characteristics that influenced which visualization methods and techniques we chose for processing the QG data. Next we discuss the different visualization tools that were used and highlight some of their strengths and weaknesses in visualizing large, high-resolution, four-dimensional turbulence datasets. Finally, we discuss what visualization products were generated and mention some of the special tools and hardware that were employed to process and view the stereo imagery output.

3.1. Data Characteristics

Volume-visualization tools typically operate on small integer quantities. To accommodate the visualization process, the QG computation performed a quantization of the floating-point data to produce single byte, scalar values. Additionally, the temporal resolution of the model output was increased 30 times what traditional statistical processing would require to facilitate the creation of smooth visual animations. Thus the total size of the QG data was approximately 25 Gbytes, consisting of 1492 time steps of byte data dimensioned 256^3 on a regular, Cartesian grid. Even with this reduction in numerical precision, the data had to be subsampled or converted to other formats as part of the data preparation step, depending on which visualization tools were used to process the data. For some tools, the data could be used without any preprocessing.

The QG data were computed at the Pittsburgh Supercomputing Center and were transferred to the National Center for Atmospheric Research for analysis across the Internet in stages, a process that took approximately 10 days.[2] As the data arrived, they were immediately archived to a tape-based mass storage system to free up local disk space for the next stage of the transfer. Because of the large data size, this model of processing the data in stages was a persistent issue throughout each step of the visualization process. Consequently, all of the tools and procedures that we used had to accommodate this model.

The QG data can be characterized by their continuous nature and the existence of opposite-signed potential vorticity values contained within the stably stratified turbulent volume. Another important characteristic is the vorticity evolution that occurs as the data evolve from random initial conditions into well-defined coherent vortices at the end of the animation. It was important to consider how different visualization approaches would handle both the complex structures that existed at the beginning of the time series and the simpler, coherent structures that formed at the end.

3.2. Visualization Approaches

We describe three approaches that we used to visualize these data using the Brick of Bytes software (version 1.2), Volsh (version 1.0), and Vis5D (version 4.2).[3] Brick of Bytes (Bob) is logically a direct volume renderer. However, contrary to our previous definition, Bob does not treat voxels as display primitives. For each voxel, Bob draws a semitransparent polygon-oriented perpendicular to the dominant viewing axis. This "trick" allows Bob to take advantage of ubiquitous hardware geometry accelerators. Bob's illumination model is very simple: light sources are not supported and no gradient information is used. Consequently, objects rendered with Bob appear flat, lacking depth. Bob has a graphical user interface and can be considered interactive to some degree: viewpoints may be altered dynamically, and a dataset can be subsampled interactively, but moving temporally through the data is expensive. Bob's greatest assets are its speed and graphical user interface, which make Bob ideally suited as a volume-visualization browsing tool.

Bob runs only on Silicon Graphics Inc. (SGI) systems that have hardware support for alpha blending and Z-buffering. To maximize performance, Bob keeps three copies of the data in memory (one for each principal axis). A voxel in Bob is a single byte. Thus, Bob's memory requirements are approximately three times the number of data points (voxels).

Bob performs classification with an interactive color and opacity map editor called Icol. This is an easy-to-use utility for defining transfer functions, changing color and opacity values within the volume rendering, and for precisely adjusting image characteristics to highlight areas of interest.

Since the QG data ranged from relatively large negative potential vorticity values on one end of the spectrum to relatively large positive potential vorticity values on the other end, a two-hue approach was used in the color map. One end of the color map was set to a continuous range of red hues and the other end to blue hues so that the red hues mapped areas of negative potential vorticity, and the blue hues mapped areas of positive potential vorticity. The opacity map controlled the transparency coefficient applied to each value in the data volume. Both ends of the opacity map were set to nearly opaque values, whereas the center of

[2] As a subsequent experiment to test ATM network connectivity between the two locations, these same data were transferred across the vBNS in just a few hours.

[3] As this paper goes to press, the Brick of Bytes (Bob) software is freely available from the Army High Performance Computing Research Center (http://www.arc.umn.edu/gvl-software), Volsh is available from the National Center for Atmospheric Research (http://www.scd.ucar.edu/vg/Software), and Vis5D is available from the University of Wisconsin (http://www.ssec.wisc.edu/~billh/vis5d.html).

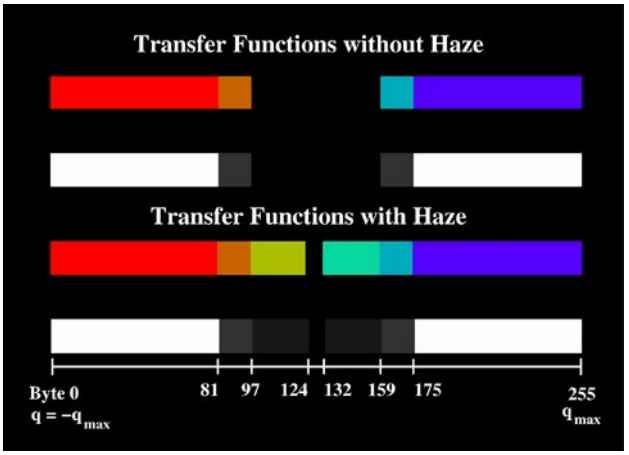

Figure 2. The two sets of transfer functions used in the ohter figures. The top pair are the color and transparency transfer functions used to render images without haze, and the bottom pair are the functions used to render images with haze. The transfer functions only differ in bytes 97–123, and 132–158. In each image the potential vorticity q is scaled by its maximum absolute value $q_{\max}$ so that $q = -q_{\max}$ is scaled to byte 0 and $q = q_{\max}$ is scaled to byte 255.

the map was set to a range of transparent values (see Figure 2). Therefore, areas of high potential vorticity were rendered mostly opaque giving an appearance of solid surfaces in the volume, and areas of mid to low potential vorticity were mapped to a semitransparent coefficient giving these areas a cloudy appearance in the final rendering. Areas of very low potential vorticity were mapped to an opacity coefficient of either zero, or a small value, and were thus rendered either completely transparent (without haze) or slightly transparent (with haze). At times early in the simulation the field is so complex that visibly rendering regions of low potential vorticity would completely obscure the structures; here the transfer functions without haze are used. At later times, the field simplifies enough that transfer functions with haze are used. We note that our transfer maps are somewhat discontinuous and that these abrupt transitions would have led to artifacts in the imagery were it not for the relatively high spatial resolution of the QG data.

Though Bob is interactive and well suited for data browsing, other tools were needed to perform noninteractive, production visualization tasks. We developed a new volume renderer, named Volsh, to meet these needs (see Figure 3(a), (d)). Volsh is designed to be compatible with Bob, allowing us to share the same data formats, color maps, and opacity maps between the two applications. Volsh works hand-in-hand with Bob in that we used Bob initially to explore the data interactively and to experiment with appropriate color maps and viewing angles. Once these were determined, the maps were saved and the viewing angles were recorded. These parameters were then fed into Volsh, and the full sequence was run out as a batch production process.

Volsh is a true direct volume renderer: no intermediate display primitives are necessary. The Volsh projection algorithm is highly optimized, based on Lacroute's Shear-Warp factorization rendering library, VolPack (Lacroute and Levoy, 1994). Volsh supports a local lighting model with gradient shading, i.e., Phong (Bui-Tuong, 1975). Volsh is noninteractive and uses a scriptable user interface based on the Tcl scripting language (Ousterhout, 1994). We saved the visualization parameters to a Tcl file and ran the rendering process across multiple systems, allowing us to divide the rendering time by the number of available CPUs currently at our disposal.

Volsh, like most DVRs, performs all rendering in software: no special graphics hardware is required. Volsh stores surface gradient information for each voxel in the dataset. Thus Volsh's memory requirements are four times the number of voxels: four bytes are used for each voxel.

The last visualization experiment that we conducted used Vis5D (see Figure 3(c)), a general-purpose visualization utility, aimed at the atmospheric sciences, that supports surface fitting based on Marching Cubes (Lorensen and Cline, 1987). Display primitive rendering and shading are performed using the OpenGL graphics language, which supports Gouraud shading and Phong illumination. Vis5D is portable to any system supporting OpenGL. More importantly, Vis5D is the most interactive of all the tools we tested. It supports a graphical user interface and permits temporal exploration of the data in real time.

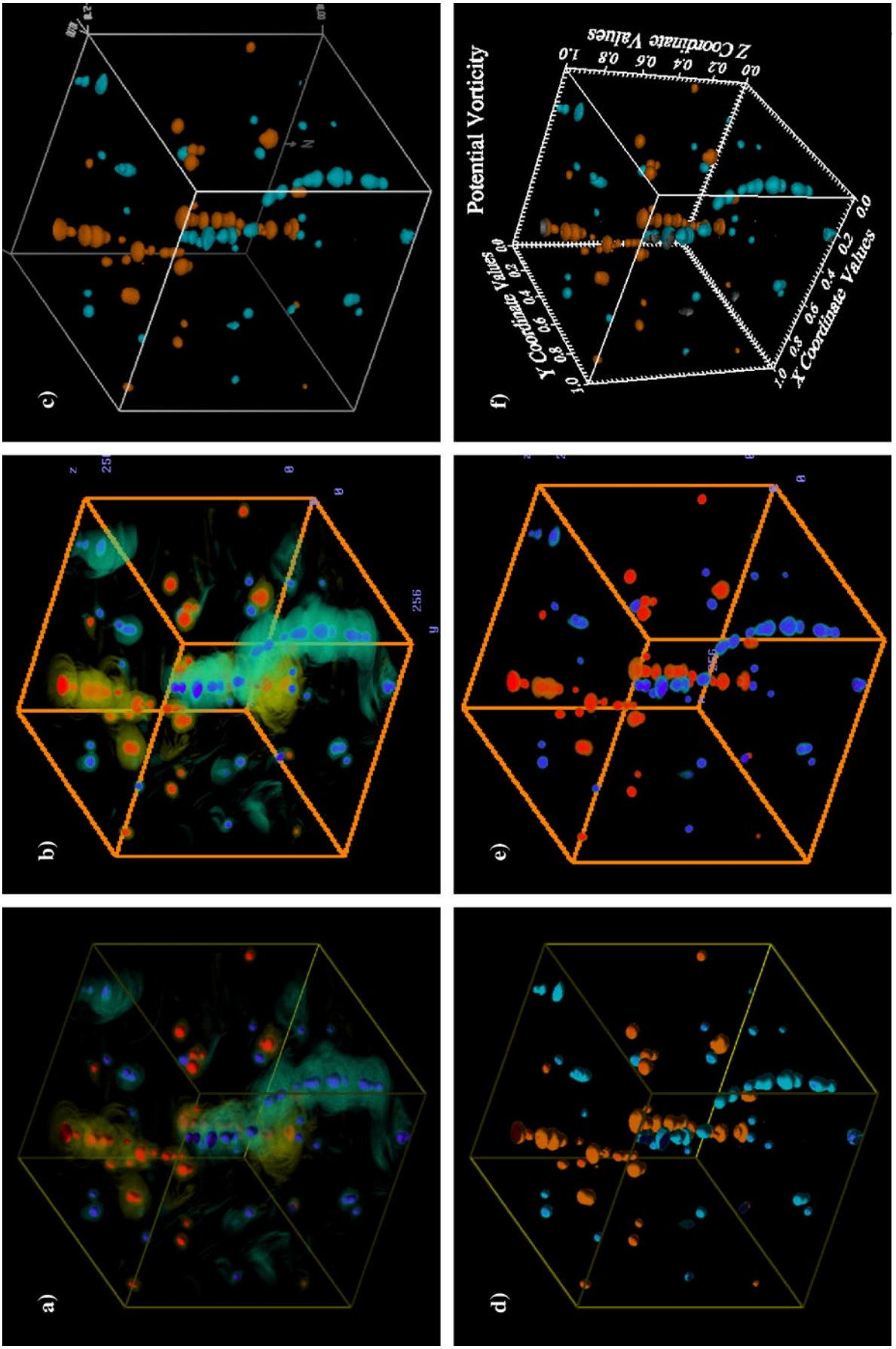

Figure 3. Identical datasets rendered with different tools and transfer functions. (a) Volsh DVR with haze. (b) Bob DVR with haze. (c) Vis5D isosurface rendering. (d) Volsh DVR without haze. (e) Bob DVR without haze. (f) NCAR Graphics TDPACK isosurface rendering.

We ran Vis5D on a 2-processor SGI Onyx system with 0.5 Gbyte of main memory and Infinite Reality Graphics. Even with a relatively powerful visualization platform like this, the rendering time was a crucial issue, especially when using Vis5D as an interactive tool. At full data resolution (256^3), it took 40 min. to extract isosurfaces for the least polygonally complex period (the last 100 time steps) of the model run. To reduce the preprocessing time and to improve rendering performance, we subsampled the data down to 128^3. At this lower resolution, it took only 5 min. to extract the isosurfaces and interactive rendering performance increased noticeably for the same 100 time steps.

To use Vis5D with the QG data, we had to increase the data size parameters in the Vis5D source code to accommodate larger data structures and internal arrays. We also added code to support stereo rendering and capture features. This gave us the ability to explore interactively the QG data in stereo mode while animating and to save the stereo imagery to files for playback later. Vis5D also provided features for adding multiple colored-slice planes at any angle and orientation in the data volume. These planes complemented the isosurface structures, helped show the continuous nature of the potential vorticity field, and better defined spatial relationships within the volume.

We also experimented briefly with a new NCAR Graphics utility named TDPACK (see Figure 3(f)). Like Vis5D, TDPACK also uses a surface-fitting algorithm, and it can be used in combination with other NCAR Graphics routines to label and annotate an image extensively. However, because TDPACK is strictly a batch utility with a complex Fortran program interface, we did not use this tool for QG data analysis to any great extent.

3.3. Visualization Products

For each of the visualization approaches discussed above, we saved either mono or stereo full-color images with a minimum of 480×480 image resolution. These images were then filtered, scaled, and quantized using a variety of image processing and movie composition tools to create mpeg and jpeg animation files, video (Betacam, S-VHS, and VHS) animations, and a variety of still-image formats for sharing results locally and on the Internet.

We used CrystalEyes active LCD stereographic hardware and a large screen front wall projection system for viewing the stereo imagery. With this technology, we had the option of viewing the imagery interactively with the stereo-enhanced software visualization packages, or we could play back the captured stereo images as digital animations. For stereo playback, we used a locally developed digital movie player tool that displays multiframe Sun-indexed image files.

4. Results

This section discusses the effectiveness of the various visualization approaches and tools employed in studying the QG calculation. We address the importance of interactive and production visualization capability, review the depth cues that help to resolve ambiguities in the spatial relationships of the time-evolving vortices, and conclude by discussing some of the new findings in QG turbulence made possible by the use of scientific visualization.

Quick-look, interactive visualization capability was critical to the visual analysis of the QG calculation. The ability to change classification functions quickly and easily, move forward or backward in time, change viewpoints, and zoom in and out on features of interest was essential. Vis5D, the most interactive of all the visualization utilities employed, proved to be a powerful exploration tool. Isosurface extraction, using opposite-signed isosurface values, clearly revealed the positive and negative rotating vortices in the QG dataset. A concern with Vis5D was that with the reduced resolution imposed by system memory requirements, we might lose too much information and not be able to create an enlightening visualization. However, even at the reduced resolution, we produced imagery that revealed important details about the turbulent structures. The horizontal vortex merger, vertical alignment, vortex filament shedding, and other QG turbulence dynamics were all clearly evident.

Though SF techniques provided much insight into the large-scale dynamics of the QG simulation, the isosurface approach was not well suited for representing the continuous range of data between large positive

and negative scalar values. Witness the wispy filaments visible in Figure 3(a), (b) and their absence in Figure 3(c)–(f). Note that the images in Figure 3(d), (e) were created with DVR tools using transfer functions without haze.

The QG data are highly continuous, and selection of isosurface values for surface fitting was somewhat arbitrary. Unlike less continuous data, where clear boundaries between features exist, such as the transition from tissue to bone in a medical computed tomography (CT) dataset, the QG data possess no distinct, implicit surfaces. Thus, while arbitrarily chosen isovalues show overall vortex organization, many phenomena were missed. Additionally, although isosurface renderings of the data easily resolved the simple vortex structures at the middle and end of the animations, this approach was not appropriate for the early time steps. The polygonal data structures that were needed to represent the numerous surfaces appearing early in the evolution were too large and complex to render efficiently, especially with interactive visualization packages. Thus a DVR approach, which neither stores nor processes polygonal information and which can effectively visualize highly continuous data, was still required. We note, however, that the relative crudeness of SF is partially what made the resulting images insensitive to the necessary subsampling.

The Army High Performance Computing Research Center's Bob software provided us with a quick-look, DVR capability that overcame the deficiencies of SF. Though not as interactive as Vis5D (time-evolutions are not smoothly presented by Bob in real time), Bob proved to be an excellent tool for performing data classification (creating color and opacity maps), exploring individual time steps, and testing viewing angles.

Despite these advantages, Bob was not particularly well suited for processing the entire 1492-frame sequence at once. Bob does not have adequate state saving or scripting capabilities, so it was not possible to restart the rendering process using the exact same viewing parameters. For this reason, it was also impossible to multiprocess the job across several different systems to reduce the amount of wall-clock time required to render the entire animation sequence. Without state-saving or scripting functionality, this visualization approach was time consuming and error prone. If the inherently batch-oriented operation had to be interrupted for any reason, then the whole process had to be restarted at the beginning for the viewing control parameters (e.g., viewing angles) to be identical in each frame of the animation. This was a particularly important issue since it took approximately 60 h to render the entire animation on a Silicon Graphics Indigo 2 system with Extreme graphics and a single 150-MHz R4400 processor. It was for these reasons that it was imperative for us to have the batch production processing capability provided by Volsh. Volsh, with its Tcl scripting interface, could easily be stopped and subsequently restarted precisely where it left off.

Easy-to-use exploratory browsing capability, combined with tightly controllable batch production rendering played a significant role in the QG analysis. Also of importance was the ability to project the data onto a two-dimensional image plane in a manner that preserved to the greatest degree possible the three-dimensional nature of the data. Several spatial cues were employed in the QG visualization to expose unambiguously the spatial relationships between vortices, and, to a lesser degree, to exhibit the spatiality, or shapes, of the individual vortices themselves.

The most effective technique for visualizing the spatial relationships between vortices proved to be stereo viewing. Monoscopic animations alone of the time-varying data, particularly from a fixed viewpoint, were not sufficient to resolve the spatial relationships of the evolving structures. Since the main physical evolution of QG turbulence involves increasing clustering of vortices, it is essential to distinguish which vortices are actually close together from those that merely appear close due to the projection onto the image plane. Stereo viewing of prerecorded animation sequences, interactive visualization sessions, and even static images greatly clarified the relative positions of vortices, allowing us to determine the true evolving three-dimensional spatial structure.

Shading methods whose illumination models took into account local gradient approximations were critical to portraying the three-dimensional shape of the individual vortices correctly. The interactive DVR software, Bob, incorporates only the simplest of lighting models; no surface gradient information is used. Hence, individual vortices appeared flat and featureless. The spherical shape of vortices is undetectable in the Bob renderings, and clearly visible in the Volsh renderings, which use lighting models supporting surface gradients (see Figure 4). The Volsh images also show more clearly the structure of the low-level vorticity rendered with haze.

By viewing stereo animations of the evolution, we were able to identify a number of events that shed light on the dynamics of vortex evolution. At early times there are a large number of vortices, and from still images it is nearly impossible to determine how the vortices are vertically aligned. By visualizing the field

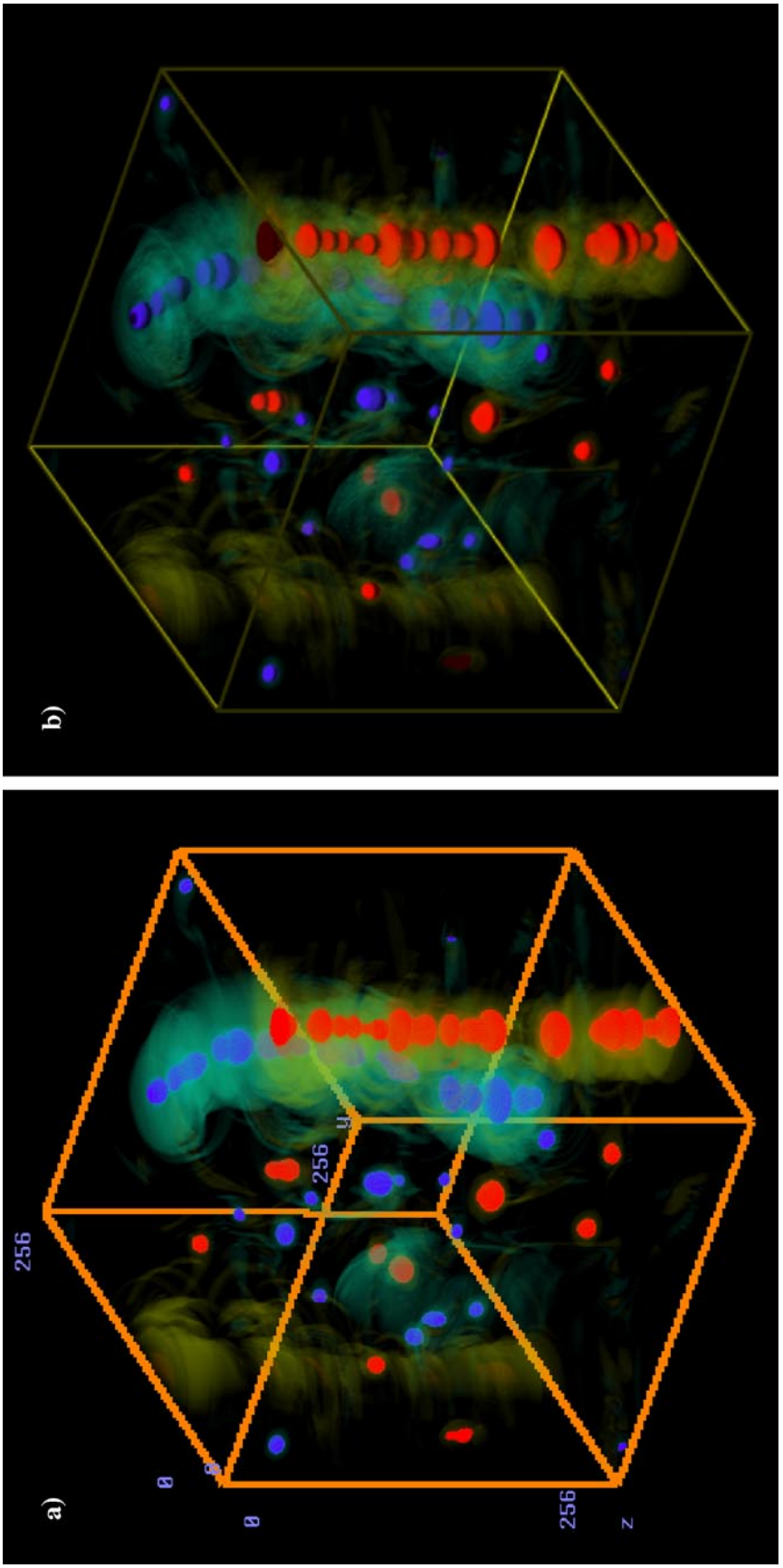

Figure 4. (a) Bob image with no lighting model. (b) Volsh image with Phong illumination model.

at a single time from several different perspectives and examining subvolumes, one can, with much effort, determine that some vortices seem to be grouped into vertically aligned clusters. However, it is impossible to tell whether these clusters evolve together or not.

Viewing stereo animations allowed us to see the nature of the vortex cluster dynamics at early times. As an example, the evolution of two vertically aligned vortex clusters at early times is seen in the Volsh renderings in Figure 5. In Figure 5(a), two separate clusters are identified by changing their colors: a green cluster containing two vortices and a yellow cluster containing four vortices. Both clusters have negative potential vorticity and were originally red. The clusters evolve coherently and are eventually brought close together as seen in Figure 5(b), (c). The top two vortices of the yellow cluster then merge with the green cluster, resulting in a single cluster, colored yellow (Figure 5(d)). In subsequent evolution this cluster is pulled apart by vertical shear (Figure 5(e), (f)), with little change in the component vortices. This is the first evidence that vortex alignment can be reversible, which is surprising since vortex merger is known to be irreversible. It is unlikely that this would have been discovered in this dataset without viewing animations of the evolution.

Another phenomenon first seen in the animations is the method by which the final columns of individual vortices assemble themselves. In Figure 6(a) two separate blue vortex clusters are seen, one roughly in the top third of the domain and the other in the lower two-thirds. The two clusters are nearly vertical. In Figure 6(b) the two clusters have been advected so that their horizontal separation is small but nonzero. Figure 6(c) shows that the two clusters join into a single column by tilting of the clusters. The result is a single cluster spanning the entire height of the domain. Note that the final blue column is not completely aligned in the vertical, but it supports a transverse wave. The orientation of this wave evolves in time, as seen by comparing Figure 6(c) with Figure 6(d).

The animations described here allowed us to see the excitation of transverse waves on the otherwise vertically aligned clusters. One excitation mechanism is described above: two horizontally separated clusters join through tilting. Another mechanism is that a single vortex (or a small cluster) can approach a column and pull the portion of the column at its vertical level out of alignment. When the single vortex is either advected away or merges with the column, the result is a column with a wave. The column acts like a string that is "plucked" by the single vortex.

Two different types of waves are visible on the final red and blue columns. Figure 6(d) shows that the wave on the blue column has a much larger amplitude. By viewing an animation from above (Figure 6(e), (f)), one sees that the wave on the blue column is in a single plane that rotates in time. The wave on the red column, however, is not in a single plane, but is rather in a helix that rotates in time. The waves here are necessarily transverse waves. Longitudinal waves are not allowed because the QG equations (1) do not allow vertical motion. Relaxing the QG approximation to allow gravity waves would presumably introduce longitudinal waves.

5. Conclusion

Currently, there is no single visualization tool that adequately meets all of our volume visualization needs. To examine a single time step in sufficient detail interactively, we found that the Bob software was the most effective. Bob's provision for quick data subsampling helped speed up interactive rendering times and allowed us to focus on a particular data subdomain. Vis5D provided other features, such as slice planes that could be drawn at any angle in the volume and then animated. More importantly, Vis5D's less computationally expensive surface fitting techniques permitted us to explore long time-evolving sequences interactively, provided the surfaces were not too complex. The visual fidelity sacrifices required by these hardware-dependent, interactive tools were acceptable for initial data interrogation purposes. Interactivity greatly facilitated the selection of appropriate classification functions and viewpoints, which could later be used by the batch-oriented, higher-quality rendering approaches necessary to capture all of the features of interest in the QG calculation accurately.

For batch processing the data, we used Volsh. This package, with its support for gradient-based lighting models and DVR, offered the greatest visual fidelity. Volsh also provided a Tcl scripting interface that was imperative for running out long sequences in a batch environment over multiple processors.

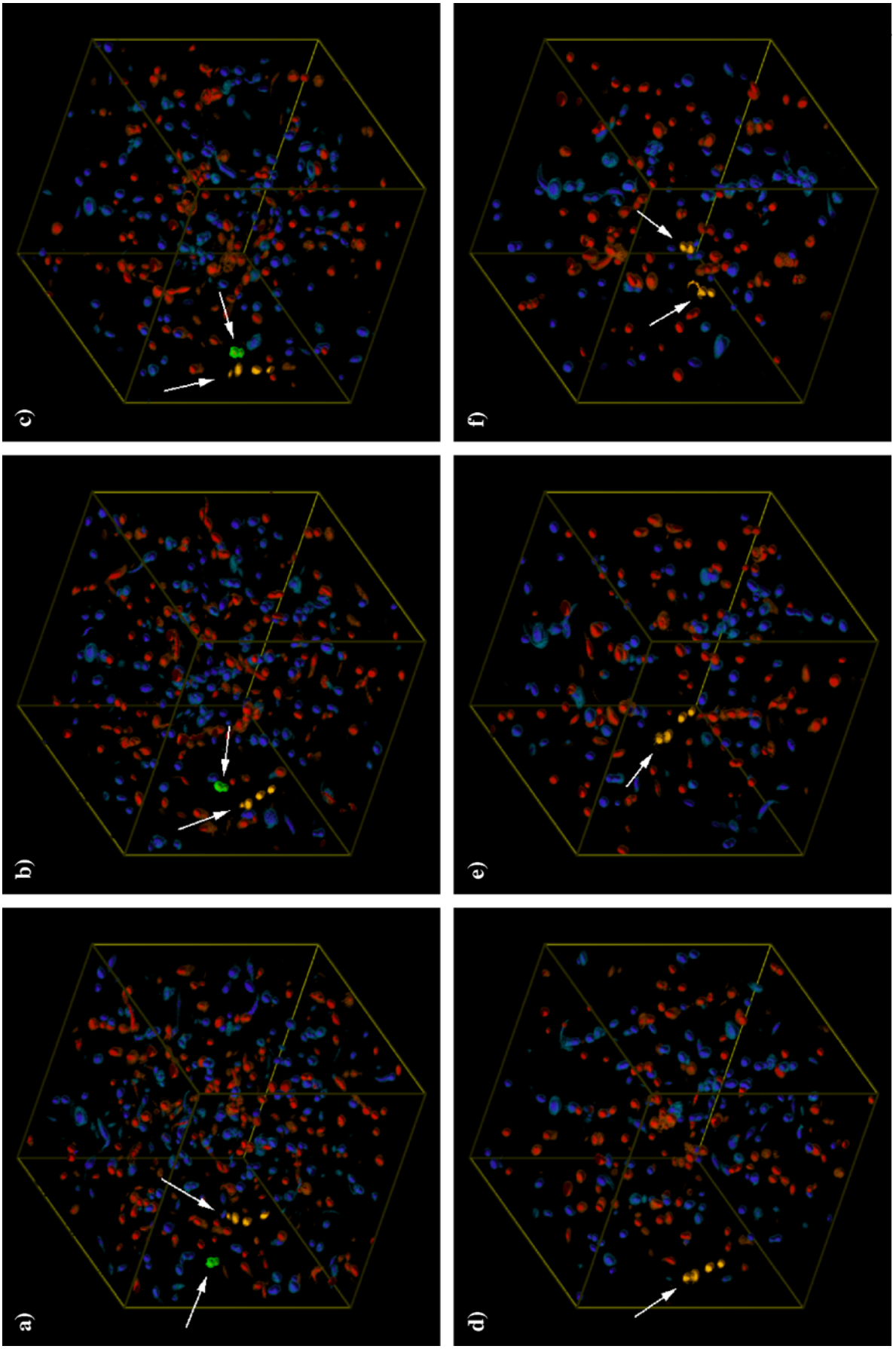

Figure 5. Evolution of vortex clusters. (a) A cluster of two vortices (green) and a cluster of four vortices (yellow) at t = 10.1. Both clusters have been recolored from the original red color. (b) The cluster move coherently but separately, t = 11.6. (c) At t = 12.9 the clusters are relatively close. (d) The top two vortices of the yellow cluster merge with the green cluster, leaving a single cluster at t = 14.7. (e) The cluster begins to be pulled apart at t = 18.4. (f) By t = 21.7 there are two separate clusters, each with two vortices.

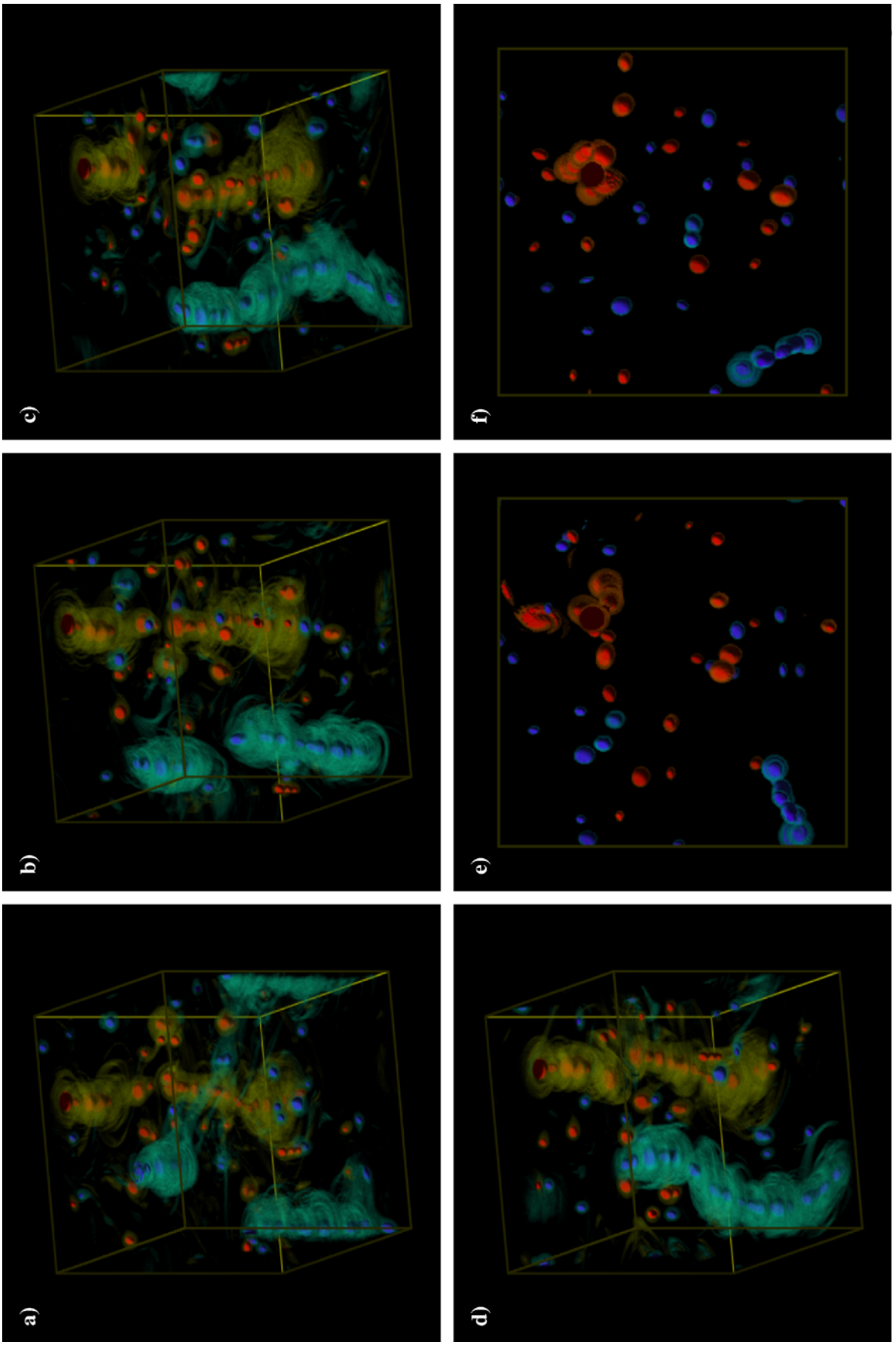

Figure 6. Two independent blue clusters at t = 46.1 (b) The blue clusters are closer together at t = 48.9. (c) By t = 52.7 the clusters have aligned, forming a single blue column spanning the entire domain. (d) By t55.8 the column has rotated. (e) View from above at t = 53.7. (f) View from above at t = 55.8.

Determining which tool to use depended on whether we were browsing the data interactively or generating movie images in batch mode. We found that the tools tended to complement one another, and we used each one to process the data in a different way.

Viewing the entire suite of visualizations made it possible to isolate several phenomena in QG turbulence that had never been seen before. We discovered that the alignment of vortices into vertical clusters is a reversible process, and thus unlike irreversible horizontal vortex merger. These clusters form at relatively early times and are coherently advected. The joining of clusters into larger columns excites deformations on the column which can then oscillate in variety of patterns. These oscillations can also be excited by isolated vortices which are advected near the column, and then either advected away or merge into the column.

The visualization tools and approaches outlined in this paper were well suited for analysis of the QG data. We expect that other fluid datasets would also benefit from the techniques described. However, the QG data, which is strongly dominated by the presence of coherent structures late in the simulation, was an excellent candidate for exploration with SF techniques. Other, less feature-rich datasets, may find SF techniques not as useful. DVR should work well for these more amorphous data.

Finally, the tools and the visualization platforms that we are using today are adequate for processing and visualizing turbulence datasets with a resolution on the order of 256^3. However, it should be noted that these data were calculated in 1995, and the current state-of-the-art problem domain has grown to 1024^3. Our current visualization tools would be stretched to their capacity and perhaps even beyond if we tried to visualize a dataset of this size with a temporal resolution similar to the QG data. The volume-visualization packages available today are lacking in terms of their ability to handle and process the turbulence data sizes that are currently being generated. We anticipate that improvements in visualization tools will keep pace with the size of the datasets, but will not overtake them. Thus, volume visualization of high-resolution turbulence computations will continue to require a variety of tools, each having its own niche.

6. Acknowledgments

We would like to acknowledge James McWilliams, Irad Yavneh, and Clive Baillie for their work in helping create the dataset used in this paper, Don Middleton and Hongqing Wang for their Vis5D stereo development, Dave Kennison for his NCAR Graphics TDPACK imagery contributions, and Brian Bevirt for his editorial comments.

References

Babiano, A., Boffetta, G., Provenzale, A., and Vulpiani, A. (1994). Chaotic Advection in Point Vortex Models and 2D Turbulence, *Phys. Fluids*, **6**, 2465–2474.

Baillie, C.F., McWilliams, J.C., Weiss J.B., and Yavneh, I. (1995). Implementation and Performance of a Grand Challenge 3D Quasi-Geostrophic Multi-Grid Code on the Cray T3D and IBM SP2, *Proceedings of Supercomputing '95*, ACM Press, New York.

Bentum, M.J., Lichtenbelt, B.A., and Malzbender, T. (1996). Frequency Analysis of Gradient Estimators in Volume Rendering, *IEEE Trans. Visual. Comput. Graphics*, **3**, 242–253.

Blundell, B.G., Schwarz A.J., and Horrell, D.K. (1994). The Cathode Ray Sphere: A Prototype System to Display Volumetric Three-Dimensional Images, *Opt. Engrg.*, **1**, 180–186.

Bui-Tuong, P. (1975). Illumination for Computer Generated Pictures, *Comm. ACM*, **6**, 311–317.

Charney, J.G. (1971). Geostrophic Turbulence, *J. Atmos. Sci.*, **28**, 1087–1095.

Drebin, R.A., Carpenter, L., and Hanrahan, P. (1988). Volume Rendering, *Comput. Graphics*, **4**, 51–58.

Dritschel D.G., and de la Torre Juárez, M. (1996). The Instability and Breakdown of Tall Columnar Vortices in a Quasi-Geostrophic Fluid, *J. Fluid Mech.*, **328**, 129–160.

Elvins, T.T. (1992). A Survey of Algorithms for Volume Visualization, *Comput. Graphics*, **3**, 194–201

Foley, J., van Dam, A., Feiner, S., and Hughes, J. (1990). *Computer Graphics Principles and Practices*, Addision-Wesley, Readwing, MA.

Gouraud, H. (1971) Continuous Shading of Curved Surfaces, *IEEE Trans. Comput.*, **6**, 623–629.

Herring, J.R. (1980). Statistical Theory of Quasi-Geostrophic Turbulence, *J. Atmospheric Sci.*, **37**, 969–977.

Hua, B.L., and Haidvogel, D.B. (1986). Numerical Simulations of the Vertical Structure of Quasi-Geostrophic Turbulence, *J. Atmospheric Sci.*, **43**, 2923–2936.

IEEE Computer Society Press (1996). *1996 Symposium on Volume Visualization*, ACM, New York.

Julesz, B. (1971). *Foundations of Cyclopean Perception*, University of Chicago Press, Chicago, IL.

Kaufman, A. (1991). *Volume Visualization*, IEEE Computer Society Press, Los Alamitos, CA, pp. 1–9.
Lacroute, P., and Levoy, M. (1994). Fast Volume Rendering Using a Shear-Warp Factorization of the Viewing Transform, *Computer Graphics, Proceedings of SIGGRAPH '94*, pp. 451–457.
Levoy, M. (1988). Display of Surfaces from Volume Data, *IEEE Trans. Comput. Graphics Appl.*, **5**, 29–37.
Levoy, M. (1990). Efficient Ray Tracing of Volume Data, *ACM Trans. Graphics*, **3**, 245–261.
Lorensen, W.E., and Cline, H.E. (1987). Marching Cubes: A High Resolution 3D Surface Construction Algorithm, *Computer Graphics, Proceedings of SIGGRAPH '87*, pp. 163–169.
Lucente, M., and Galyean, T.A. (1995). Rendering Interactive Holographic Images, *Proceedings of SIGGRAPH '95*, pp. 387–394.
McAllister, D.F. (ed.) (1993). *Stereo Computer Graphics and Other True 3D Technologies*, Princeton University Press, Princeton, NJ.
McWilliams, J.C. (1989). Statistical Properties of Decaying Geostrophic Turbulence, *J. Fluid Mech.*, **198**, 199–230.
McWilliams, J.C. (1991). Geostrophic Vortices. In *Nonlinear Topics in Ocean Physics, Proceedings of the International School of Physics, Course CIX* (A.R. Osborne, ed.), North-Holland, New York, pp. 5–50.
McWilliams, J.C., and Weiss, J.B. (1994). Anisotropic Geophysical Vortices, *CHAOS*, **4**, 305–311.
McWilliams, J.C., Weiss, J.B., and Yavneh, I. (1994). Anisotropy and Coherent Vortex Structures in Planetary Turbulence, *Science*, **264**, 410–413.
Ousterhout, J. (1994). *Tcl and the Tk Toolkit*, Addison-Wesley, Reading, MA.
Pedlosky, J. (1987). *Geophysical Fluid Dynamics*, Springer-Verlag, New York.
Sabella, P. (1988). A Rendering Algorithm for Visualizing 3D Scalar Fields, *Comput. Graphics*, **4**, 51–58.
Sobierajski, L.M., and Kaufman, A.E. (1994). Volumetric Ray Tracing, *Proceedings of 1994 Symposium on Volume Visualization*, ACM Press, New York, pp. 11–25.
Upson, C., and Keeler, M. (1988). V-BUFFER: Visible Volume Rendering, *Comput. Graphics*, **4**, 59–64.
Weiss, J.B. (1994). Hamiltonian Maps and Transport in Structured Fluids, *Phys. D*, **76**, 230–238.
Westover, L. (1990). Footprint Evaluation for Volume Rendering, *Computer Graphics*, **4**, 367–376.
Yavneh I., and McWilliams, J.C. (1996). Multigrid Solution of Stably Stratified Flows: The Quasigeostrophic Equations, *J. Sci. Comput.*, **11**, 47–69.