

CASE STUDY: VISUALIZING OCEAN CURRENTS WITH COLOR AND DITHERING

Patricia Crossno

Sandia National Laboratories
P.O. Box 5800, Albuquerque, NM 87185-0822
pjcross@sandia.gov

Edward Angel

Albuquerque High Performance Computing Center
Department of Computer Science, University of New Mexico
Albuquerque, NM 87131
angel@cs.unm.edu

David Munich

Albuquerque High Performance Computing Center
1601 Central NE, Albuquerque, NM 87131
dmunich@ahpcc.unm.edu

ABSTRACT

This case study presents several related approaches to visualizing flow information from large vector volumes generated by ocean circulation modeling. Flow vectors are mapped to colored pixels to enable global views of dense three-dimensional vector fields. Each of the approaches starts by classifying vector direction into a small number of colors. One approach then uses scaled linear interpolation to blend between adjacent directional colors. Two other approaches use half-toning and dithering methods to rapidly display flow information. By using opponent colors for our directional encoding, we can blend colors, either through linear interpolation or the user's visual system, into intermediate colors without expressly calculating them by a conversion to polar coordinates.

CR Categories: I.3.3 [Computer Graphics]: Bitmap and framebuffer operations; I.3.8 [Computer Graphics]: Applications; J.2 [Physical Sciences and Engineering]: Earth and atmospheric sciences.

Keywords: flow visualization, vector field visualization, color mapping, dithering, half-toning.

1 INTRODUCTION

Vector visualization arises in many applications. Perhaps the most important is in visualizing flows in gases and fluids. In this case study, we address the problem of visualizing global ocean currents.

Predicting the global climate is one of the grand challenge problems. Much of the effort in this area has been in simulating ocean currents. These simulations use three spatial dimensions because the slow thermohaline circulation patterns involve currents that rise and fall as the salinity and temperature of the water change. The simulations must be carried out at a resolution, both for numerical stability and to capture local phenomena. At the present, models are computed at up to 1/10 degree resolution and 80 depth levels [11]. Researchers may want to run

simulations over periods up to a century with at least one time sample per day.

The analysis of such huge data sets will have to be done in ways other than by examining every datum. One likely method will be to use visualization to identify areas that require detailed analysis. Our approach of using color is meant to be a first step in the analysis. Thus, we seek a fast local method that can be spread across multiple processors. The interactivity will allow users to discard large uninteresting areas and apply slower more detailed analysis to the remaining data. This paper reports on our work to provide such a method.

Over the past twenty years a number of powerful techniques have been developed for vector visualization. Streamlines [10] and stream tubes [2] follow the path a weightless particle would follow in steady state flow. Icon-based methods, such as hedgehogs, show the direction and magnitude of the vector field at selected points [10]. Methods such as spot noise [14] and line integral convolution (LIC) [1] use texture to show flow direction over surfaces. As elegant and informative as these methods can be, they are unsuitable for many large-scale visualization problems, such as our application.

For three-dimensional vector fields of any significant size, icon-based approaches break down because visual clutter overwhelms the viewer once the size of the icon exceeds the sampling rate of the data set. Texture methods are computationally very expensive to implement, and, without the addition of color, are limited to representing directional information.

The best approach for our application, given the data size and the desire to view the data in its totality, is to use a colored-pixel approach. This approach maximizes the amount of information that we can present on the screen by representing each vector as a single pixel. We present several approaches for combining a simplified vector-to-color mapping with fast methods of generating color displays. We have tried using error-diffusion and dithering to generate binary colors. The eye of the viewer then acts to blend the color information into intermediate colors without explicitly color encoding more than a small number of vector directions.

We organize our paper as follows. First, we review previous approaches to using color for flow visualization. Next, we look at

a direct color mapping. Then we discuss methods that are based upon dithering of the color. Lastly, we expand our methods from two dimensions to three. These methods combine ideas from previous work with techniques that emphasize efficiency and hardware implementation.

2 PREVIOUS WORK

There are three noteworthy earlier papers that map vector information into colored pixels. The first two of these operate in three dimensions, and the last is a planar algorithm. There are subtle differences in the mappings used and the color spaces employed.

Usselton [13] visualized perturbation velocity vectors generated by computation fluid dynamics simulations by volume rendering the results of a conversion from vector space to color space. He first transformed the vectors into polar coordinates, then represented the rotational angle by hue, while the elevation angle was depicted by saturation. These angles were determined with respect to the “free stream” velocity and resulted in highly saturated values highlighting flow at angles of 180 degrees to this reference velocity. By mapping the magnitude of the perturbation vector to opacity, the majority of the free stream flow is removed, leaving only the flows of interest.

Hall [5] also presented a vector visualization technique that mapped vector direction and magnitude into a volume of color values that he then volume rendered. He developed a perceptually based, spherical color space arranged in accordance with the opponent theory of color vision. This color space places colors that are perceptually opposite on opposite sides of the sphere, so that blue is opposite yellow and red is opposite green. Fully saturated colors lie along the equator, with black and white making the vertical axis. The center of the sphere is gray, with color increasing with distance outward along the radius. Direction is mapped to colors on the sphere, while magnitude is mapped to distance along the radius. Opacity is used to segment the volume and provide a data windowing capacity. A reference rendition of the color sphere is provided to help interpret the direction-to-color mapping.

Johannsen and Moorhead [7][8] did earlier work not only in vector-color mappings, but their application area was also visualization of ocean flow. However, they rejected the three-dimensional aspects of Hall’s work saying that for ocean models, two-dimensional visualization simplified cognition and that viewers would have difficulty distinguishing small changes in saturation and luminance. They also disliked having gray mapping to null-vectors, which they felt should be black. They called their approach the *color wheel* technique. Using the hue, saturation and value (HSV) color model, the direction of flow is encoded from a circular color lookup table and the magnitude scales the saturation and/or value of the color. The coloring is mapped onto a surface along with the color wheel, which acts as a legend for the interpretation of the colors.

In considering each of these papers, we find that there are problems with each approach. Although we like the idea of mapping magnitude to opacity as Usselton did, we prefer Hall’s choice of an opponent color space. We agree with Johannsen and Moorhead that using saturation and luminance changes to distinguish up and down directional differences is a poor choice. Yet, we disagree with them that a two-dimensional solution is adequate for viewing ocean models when we are looking for a global circulation model that will probably not exist in a single layer. Another objection to all of these methods is that they

require a transformation to polar coordinates in order to do their color mappings.

A less closely related work, that maps second-order tensor fields into color using a spherical color map, is Kindlmann and Weinstein’s *hueballs* algorithm [9]. Coherent structures within the tensor field are then seen as areas of uniform color.

3 DIRECT COLOR

Although we also perform a mapping between direction and color, we do so much more simply than the earlier work because we do not try to provide the appearance of a continuous mapping with a different color for every possible vector direction. Instead we look at the grid cell in which the vector is conceptually contained. In two dimensions this cell is a square, and in three dimensions it is a box.

To illustrate, let us look at the problem in two dimensions. If we look only at a single vector and consider the direction of flow in relation to the cell faces, we have only the five cases of flow shown in Figure 1. Notice that either there is no flow, or the direction of flow passes through one of the four faces. Therefore the flow can be classified as being either up, down, right, or left. Each direction is assigned a color.

We use the maximum component of the vector as our direction of the flow and assign the pixel the corresponding color (up = yellow, right = red, down = blue, left = green). The user provides a threshold on the component value to define the cutoff for the no flow case. No flow cases are then rendered as white pixels in two dimensions. Consequently, our color conversion is done at very little computational cost because we avoid converting to polar coordinates.

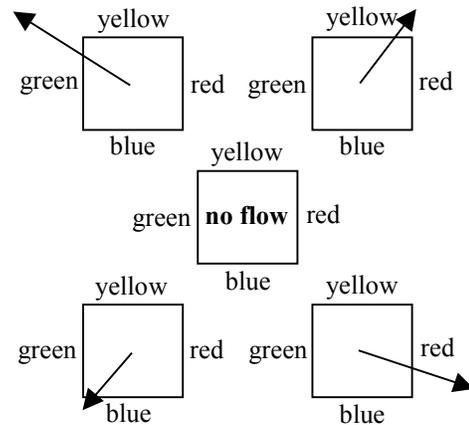


Figure 1: Five cases for flow in two dimensions.

The directional colors we have chosen take into account perceptual issues. By using red and green for one pair of opposite faces, and blue and yellow for the other pair, we keep opposing colors from being adjacent. This is important when we use dithering. Based on the opponent theory of color vision, we want to avoid blending green and red, or blue and yellow because perceptually those blends do not exist [6].

In Color Plate 1, which shows flow for one plane of the North Atlantic, we have taken this one step further and blended the colors of the adjacent faces. Each vector, whose magnitude is above the threshold, has been assigned a color, which is a linear combination of the two colors for the direction of the planar flow. The resulting color is then scaled in intensity by the magnitude of the vector.

Note the appearance of vortices in the Caribbean and the Mediterranean. A close up of two vortices in the Caribbean is shown in Color Plate 4. This technique is similar to that of Johannsen and Moorhead [7], but without use of the color wheel. However, this method extends more easily to three-dimensional flow. We have adopted their approach of including a color legend within each image that shows the mapping between flow direction and color.

4 DITHERING

We use color dithering or half-toning to display our flow patterns. Color dithering is a method of displaying shades of color using only pure colors. The basic idea behind all dithering algorithms is to distribute dots or squares of pure color in such a manner that the average shade over a small area is statistically correct. Although dithering methods are normally used to produce the visual appearance of smooth shading where the hardware cannot do so directly, as with printers, we are attracted by the simplicity of the method.

We started with the error diffusion method of Floyd and Steinberg [3]. The basic idea of the method can easily be understood for a gray scale image. Suppose that we have an image with gray level values in the range between zero and one. We assign white to be one, and black to be zero. If the first pixel we want to display has a gray level g , we output a white pixel if g is greater than a half, and a black pixel otherwise. If we output a white pixel, it is overly white by $1.0-g$. If we output a black pixel it is overly black by g . We can distribute or diffuse this error to pixels that we have not yet output by making them lighter or darker depending on the sign of the error and weighting factors. If we produce the image from top to bottom and left to right, we can diffuse the error down and to the right, as in Figure 2.

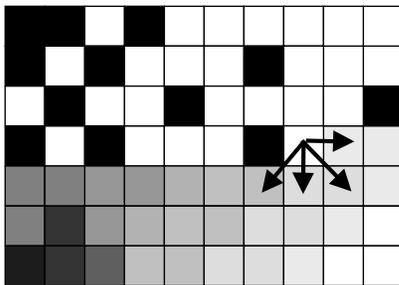


Figure 2: Diffusing error to neighboring pixels. The top few rows show pixels that have already been operated on. The current pixel is centered under the arrows. The error will be distributed to the neighboring pixels pointed to by the arrows.

We can do this technique on a primary-by-primary basis or, in our case, on a color-by-color basis. Thus, in two dimensions, our primary colors are red, blue, green, and yellow, but we use them in pairs. Hence, if the flow is in the red-blue direction, we can output a blue pixel or a red pixel. Each would be a full magnitude and the error in each primary would then be diffused to other pixels. This technique is shown in Color Plate 2 for the North Atlantic data.

The problem with error diffusion is that due to its direction of diffusion, small features can be displaced down and to the right along with the error. There are many variants of dithering algorithms ranging from the use of halftone masks to random algorithms. Each method gives slightly different local behavior, while preserving the same average behavior. Our next approach preserves feature location at the expense of possibly having small

features be lost in the error distribution. If the features being sought are larger than the size of the features that could be lost, this is an acceptable trade-off.

We next implemented dispersed-dot ordered-dither to distribute the error over a small neighborhood [4]. This approach uses the halftone matrix in Figure 3, which is an array of values that acts as a threshold for deciding whether or not to color a pixel. The matrix is tiled across the output. If the color component value exceeds the matrix value at a particular output location, the color is then added to that location's pixel computation. The color added is the color used for that direction (red, green, blue, or yellow). The results of this approach are shown in Color Plate 3 for the North Atlantic data.

8	136	40	168
200	72	232	104
56	184	24	152
248	120	216	88

Figure 3: Halftone matrix.

Looking at the color plates, we can make a few observations. If we use a weighted-average of the chosen colors, we get results that are visually similar to those of the color wheel, but with less effort and more flexibility in how to assign colors. Using pure software implementations, the dithering and direct color methods required about the same amount of work. However, this is less than the work required for using the color wheel, since we avoid calculating square roots.

Evaluation of the dithering results is more difficult. Certainly, they capture the large-scale flow features but so do the other techniques. Although dithering appears to add noise to the image, these patterns convey detail in the data. For example, if we look at the area between South America and Africa in the color plates, we can observe that although the blended color method produces the most visually pleasing image, because the colors are scaled, detail is lost in areas of low flow. Were we not to scale the colors, we would lose magnitude information. The dithered image shows a pattern of mixed green and blue pixels, which indicate that the flow is not unidirectional in these regions. While we could have obtained this information in other ways, it would have required more computation.

5 THREE-DIMENSIONAL FLOW

Expanding our approach to three-dimensional flow, we need two more distinct colors to represent up and down flows. Additionally, we would like to reserve two colors for background and land mass silhouettes. If we use white and black for background and land, and the red-green and blue-yellow pairs for flow in the plane, the colors for the up-down flows cannot be colors that would be blends of our colors in the plane, such as orange or purple. We would tend to confuse these colors with our blended planar directions.

Consequently, whatever colors we choose must be composed of all three primaries (red, green, and blue). The hue, luminance, and saturation (HLS) and HSV color systems do essentially this by using the black to white continuum as the third dimensional axis in their cones. However, since we are not looking for a continuum of colors, and because we have already removed black and white from consideration, we are free to select any two other colors that are composed of the three primaries.

We have chosen brown for the downward flows, and gray for the upward flows. In addition to being well separated in color space and recommended for color encoding [15], these two colors

blend well with each of the planar directional colors, whether we are using interpolation or a dithering technique to combine colors in the vertical direction.

Pixel opacity is determined by the magnitude of the vector windowed over a range of interest. This windowing permits visualization of flows of lesser intensity without the visual clutter of having to include the higher intensity flows. Vectors whose magnitudes fall either above or below the window range are transparent.

Once the vectors have been replaced by colors (RGBA), we have a three-dimensional color volume that then must be rendered into a two-dimensional image. We are currently using texture memory to do fast volume rendering of the smaller flows. For larger flows, we can use a scalable parallel volume renderer, such as PVR [12].

Color Plate 5 shows flow with the gray-brown axis indicating the up and down currents, respectively. In this image, we did a direct color mapping of the primary directional component without blending or dithering. Lesser flows are rendered transparent, leaving the major circulation features against an uncluttered three-dimensional backdrop of the coastlines and the ocean floor. Since we only have 80 layers in the z direction, we scaled the z components by 3000 to enhance visibility. The display was created using a three-dimensional texture map.

6 CONCLUSIONS

We have presented several variations on the theme of vector-to-color mappings for flow visualization. This general approach is attractive for global viewing of very large, dense flow data sets, such as ocean and climate models, because of the compact representation afforded by a colored pixel representation of a vector. We have sought to simplify the mapping, compared to approaches presented in earlier work.

In two dimensions, we looked at a number of techniques, including direct color mapping, color blending, error diffusion and dithering. The direct color mapping, which was not shown in the figures due to lack of space, reduces all flows to just their primary directional color. This is useful for getting a sense of the main currents, but it is not a good choice for looking at smaller features. The blended color method is better, but some of the detail can be lost in the darkness when flow magnitude is small. Error diffusion captures small features and maintains feature brightness and visibility, but shifts them slightly relative to the coasts. Dithering does not cause shifting, but loses small features.

We then extended the techniques to three dimensions. Although any one of the two-dimensional blending or dithering approaches could have been used, we presented an example using direct color mapping without scaling, blending, or diffusing the errors. Important vertical flow features can be seen combined with vortices in the horizontal flow directions.

Although one criticism of color as a flow visualization technique has been the non-intuitive mapping between color and direction, we have found that after some use the user becomes trained in the interpretation of the colors. We conclude that the use of color is a fundamental technique for visualization of large flow data sets.

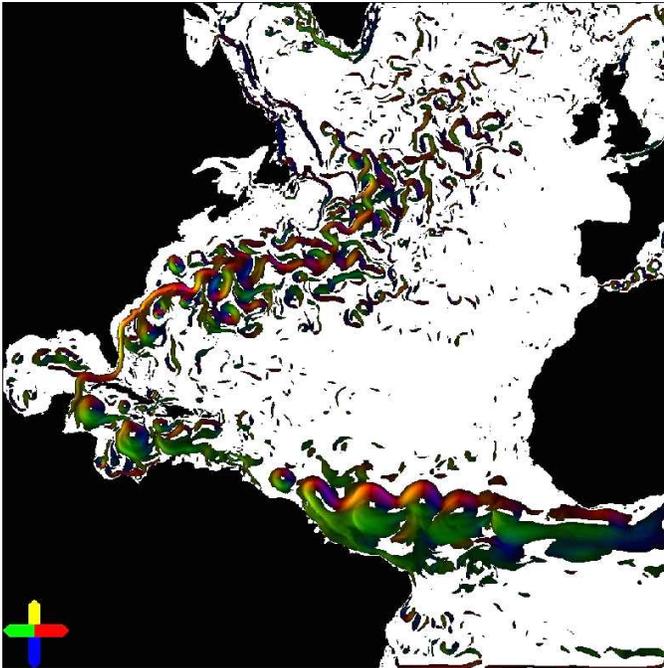
7 ACKNOWLEDGEMENTS

We would like to thank Pat McCormick at Los Alamos National Laboratories for providing the ocean model data set. The DOE Mathematics, Information, and Computer Science Office funded this research. The work was performed at the

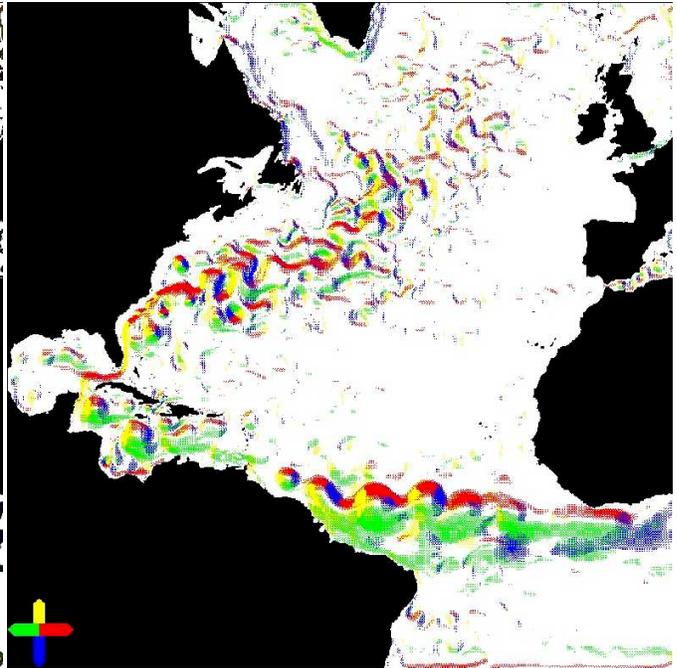
Albuquerque High Performance Computing Center and Sandia National Laboratories. Sandia is a multi-program laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under Contract DE-AC04-94AL85000.

REFERENCES

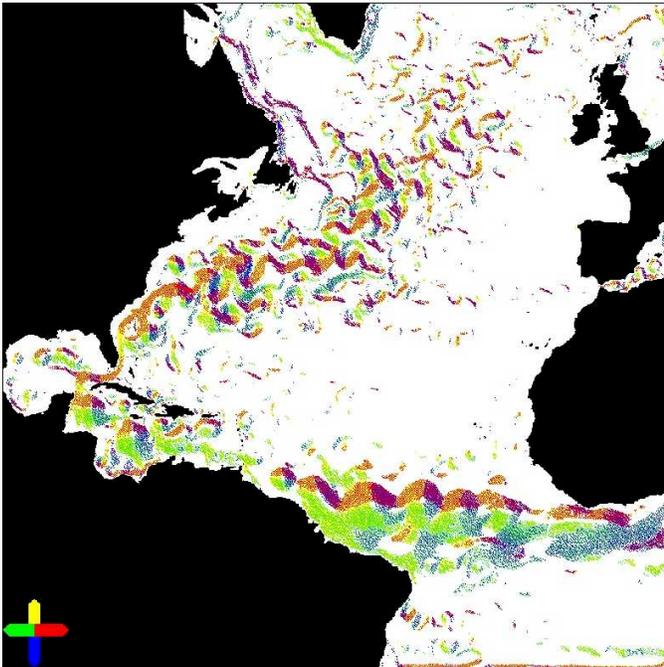
- [1] Brian Cabral, and Leith Leedom. Imaging Vector Fields Using Line Integral Convolution. In James T. Kajiya, editor, SIGGRAPH 93 Conference Proceedings, Annual Conference Series, pages 263-270. ACM SIGGRAPH, Addison Wesley, August 1993. ISBN 0-201-58889-7.
- [2] David Darmofal and Robert Haines. Visualization of 3-D Vector Fields: Variations on a Stream. In Proceedings AIAA 30th Aerospace Science Meeting and Exhibit, Reno, Nevada, January 1992.
- [3] R. Floyd and Lou Steinberg. An Adaptive Algorithm for Spatial Gray Scale. In *Society for Information Display 1975 Symposium Digest of Technical Papers*, pages 36-37, 1975.
- [4] James D. Foley, Andries van Dam, Steven K. Feiner, and John F. Hughes. *Computer Graphics: Principles and Practice Second Edition*. Addison-Wesley, pages 568-573, 1990. ISBN 0-201-12110-7.
- [5] Peter Hall. Volume Rendering for Vector Fields. *The Visual Computer*. 10 (2): 69-78. 1993.
- [6] Leo M. Hurvich. *Color Vision*. Sinauer Associates, 1981. ISBN 0878933360.
- [7] Andreas Johannsen and Robert Moorhead. Case Study: Visualization of Mesoscale Flow Features in Ocean Basins. In R. Daniel Bergeron and Arie E. Kaufman, editors, IEEE Visualization '94, pages 355-358. IEEE, October 1994. ISBN 0-8186-6627-7.
- [8] Andreas Johannsen and Robert J. Moorhead II. AGP: Ocean Model Flow Visualization. *IEEE Computer Graphics and Applications*. 15 (4): 28-33. July 1995. ISSN 0272-1716.
- [9] Gordon Kindlmann and David Weinstein. Hue-Balls and Lit-Tensors for Direct Volume Rendering of Diffusion Tensor Fields. In David Ebert, Markus Gross, and Bernd Hamann, editors, IEEE Visualization '99, pages 183-189. IEEE, October 1999. ISBN 0-7803-5897-X.
- [10] Will Schroeder, Ken Martin, and Bill Lorensen. *The Visualization Toolkit 2nd Edition*. Prentice Hall PTR, pages 167-174, 1998. ISBN 0-13-954694-4.
- [11] Albert Sementner. Ocean and Climate Modeling. *Communications of the ACM*, 43 (4): 80-89, April 2000.
- [12] Cláudio Silva, Arie Kaufman, and Constantine Pavlakos. PVR: High Performance Volume Rendering. *IEEE Computational Science and Engineering*, 3 (4): 18-28. Winter 1996.
- [13] Samuel P. Useton. Volume Rendering for Computational Fluid Dynamics: Initial Results. *Technical Report RNR-91-026*, NAS-NASA Ames Research Center, September 1991.
- [14] Jarke J. van Wijk. Spot Noise – Texture Synthesis for Data Visualization. In Thomas W. Sederberg, editor, *Computer Graphics (SIGGRAPH 91 Conference Proceedings)*, 25 (4): 309-318. Addison-Wesley, July 1991. ISBN 0-201-56291-X.
- [15] Colin Ware. *Information Visualization: Perception for Design*. Morgan-Kaufman, pages 135-136, 2000. ISBN 1-55860-511-8.



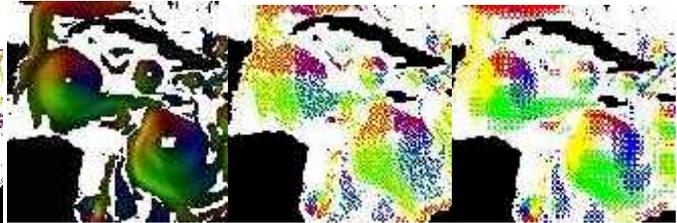
Color Plate 1: Blended color method where adjacent directional colors are blended using linear interpolation, then scaled by flow vector magnitude.



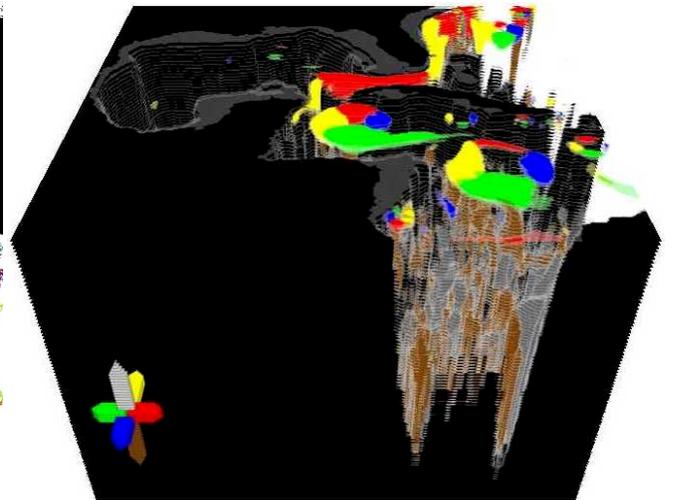
Color Plate 3: Dithering method.



Color Plate 2: Error diffusion method.



Color Plate 4: Close ups of vortices in the Caribbean for each technique (blended color, error diffusion, and dithering).



Color Plate 5: Direct color mapping of three-dimensional flow in the Gulf of Mexico without color scaling or blending. Note the vortices (also shown in Color Plate 4) and the up and down flow.

CASE STUDY: VISUALIZING OCEAN CURRENTS WITH COLOR AND DITHERING

Patricia Crossno, Edward Angel and David Munich