

MARCHING FLOW

Edward Angel
Albuquerque High Performance Computing Center
Computer Science Department, University of New Mexico
David Munich
Albuquerque High Performance Computing Center
Patricia Crossno
Sandia National Laboratories

ABSTRACT

Interactive visualization of large flow data sets requires new methods that are fast, can be run in parallel, and are interactive. In this paper we propose a method based on the use of color and local pattern matching. The pattern matching is inspired by the *marching cubes* algorithm for scalar field visualization. For flow visualization, there are many more patterns, but at least for two-dimensional flow, the number of cases is manageable. For three-dimensional flow, using a subset of the cases can produce informative visualizations. We show results on both data from global ocean circulation models and from artificial data sets.

INTRODUCTION

Visualization of flow data is one of the most important applications of scientific visualization. In spite of many years of research in the area and the development of a variety of methods, including streamlines [9], stream tubes [3], and line integral convolution (LIC) [1], none of the present techniques are adequate for dealing with large data sets.

In this paper, we are concerned with visualization of the large data sets that can arise in applications such as ocean and climate modeling. In such problems, we can have flow data over a three-dimensional grid consisting of 10^9 points. Data may be computed or measured daily, or hourly, for periods of up to 100 years [10].

Clearly such problems present a host of difficulties, which range from the amount of computation required, to the volume of data stored, to data interpretation. From the visualization perspective, however, we need to think about interactive techniques that can be used for data exploration because it will not be possible to process every datum in each data set. We must seek methods that are hierarchical and can be made interactive, thus allowing the scientist to identify areas of interest rapidly. Once identified, these regions can then be explored later by slower more detailed methods. Just as clearly, we must use methods that are local and can be adapted to run in parallel.

Standard methods such as streamlines and LIC cannot easily be made useful for these applications. Problems include difficulties in parallelization, computational requirements and visual clutter.

A similar analysis holds for visualizing large scalar volumes. Although direct rendering methods, such as ray tracing, can produce informative images, these techniques require an enormous amount of computing and cannot be used interactively. Typically, such techniques require a complete recalculation of the output image for each motion of the object or change in the view. Hence, isosurface methods, which do not image all the data but are computationally efficient, are very popular.

Of the isosurface methods, *marching cubes* [8] is the dominant algorithm. It is local, looking only at a datum and its closest neighbors, and can be easily adapted to run in parallel. Moreover, it requires very little computation as it uses tables to recognize patterns in the data.

Our proposed method is inspired both by *marching cubes* and by the importance of color to human pattern recognition. Our method is based upon looking at the direction of flow at a point and its immediate neighbors. Considering symmetries in the patterns, we can reduce the total number of cases that must be considered. We allow the user to select which types of cases to view. For example, clustering cases by considering only the direction of flow results in a small number of distinct cases at each point. These cases can be encoded into colors.

This paper is organized as follows. First, we examine a combinatorial approach based only on the direction of flow, if the magnitude of the flow exceeds a user-determined threshold, at a point and its neighbors. We consider ways of classifying these cases. Then we show how we can display subsets of the cases using color encoding. Next, we present results for data from ocean circulation simulations and a mathematical function. Finally, we consider extensions from two- to three-dimensional flow.

MARCHING FLOW

Consider typical flow data on a structured grid. We can consider the flow in either two or three dimensions. For now, we will consider two-dimensional flow for clarity and will consider the modifications that we have to make for three dimensions later. We have an $n \times m$ array of two-dimensional vectors, the value of each vector determining the flow direction and magnitude at that cell.

If we look at a single cell and consider only the principle component of the flow vector we can assign one of four directions to the cell, as in Figure 1. We shall also

use O to denote a cell in which the magnitude of the flow is less than some threshold. If we look at the cell and its four neighbors as in Figure 2, we can identify 3125 ($= 5^5$) patterns between the cell and its neighbors. Some of these patterns are shown in Figure 3 for simple flow conditions. For comparison, *marching cubes* starts with 256 cell patterns.

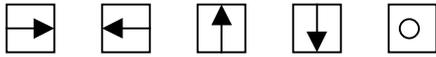


Figure 1: Five flow cases.

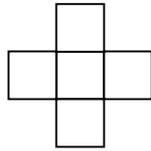


Figure 2: Center cell and four neighboring cells.

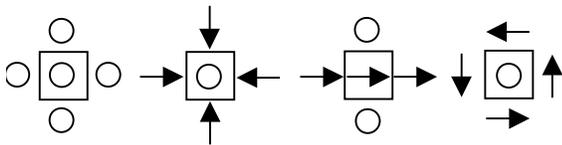


Figure 3: Some example flow patterns.

However, just as with *marching cubes* we can recognize symmetries in the patterns. For example, all the patterns in Figure 4 are equivalent in that they are characteristic of flow in one direction. These patterns can be computed as we march through the data set, thus leading to our naming this method *marching flow*.

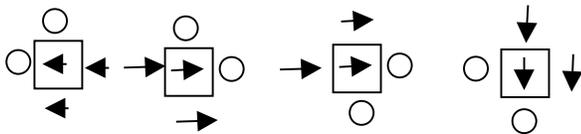


Figure 4: Equivalent flow patterns flowing in a single direction.

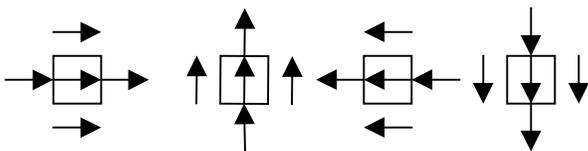


Figure 5: Symmetric patterns of flow.

Unlike *marching cubes*, in which all the cell configurations, other than the two trivial cases, generate a piece of the isosurface passing through the cell, here it is not clear how we should interpret and display the patterns. For example, although the four patterns in Figure 5 are symmetric, we may want to display each with a different color if we want to visualize the direction of flow. If we are interested only in whether or not the magnitude of the flow exceeds some magnitude over a small region, we might display these patterns in the same manner. Hence, although we recognize the rotational symmetry, we want

to count these as four distinct cases and decide how to display them later.

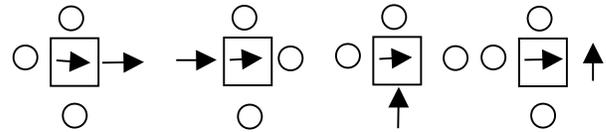


Figure 6: Removed symmetries.

However, if we count the patterns in Figure 5 as distinct cases, we will generate a large number of patterns even if we remove rotational symmetries and symmetries in the type of flow as in Figure 6. Here, we might argue that the two patterns on the right are equivalent, as are the two patterns on the left. However, this reasoning leads, for example, to 17 “distinct” cases in which there is no flow in three of the five boxes. For four and five boxes that have some sort of arrow, the number of patterns is so large that it does not lead to a simple program for implementing the method. In addition, with this approach it is difficult to distinguish between cases that are important and cases that probably will never appear in a data set, as they may make no sense physically. For example, the patterns in Figure 7 are unlikely to be important in any real application. However, it would help to start with a simple classification scheme. Suppose that we use an O to denote flow with a magnitude below some threshold and an X to denote any of the four flow directions. This simple classification yields the 12 unique patterns shown in Figure 8.

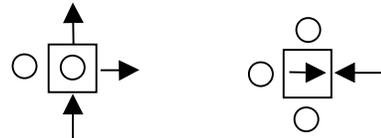


Figure 7: Improbable flow patterns.

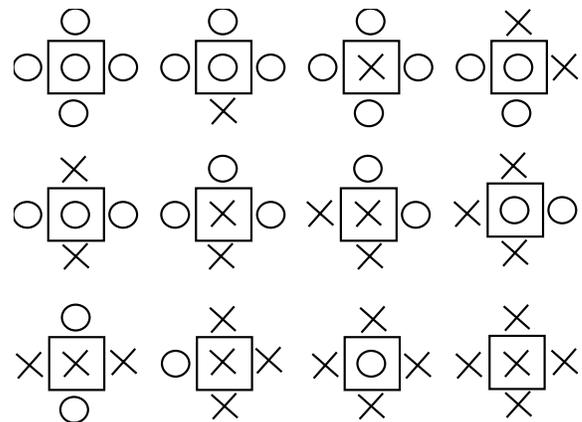


Figure 8: General flow cases.

Nevertheless, a typical implementation must use the direction of flow and distinguish between important and unimportant, or physically impossible, cases. We found it useful to start with the cases that have the most coherence between the arrows. The first cases that we consider are

the ones with five arrows, all pointed in the same direction as in Figure 5. Next we consider, all the cases with four arrows in the same direction. In a typical application, we might choose to treat these patterns in the same way as those where all five arrows are aligned. We then move to cases with three aligned arrows and so on. The advantage of this sieve-like algorithm is that it allows us to treat the symmetries in an orderly fashion without creating a complex table of cases.

The idea of displaying patterns is actually an old one. Klassen [7] assigned small arrow patterns as black dots in a small white box. Although their work showed the coherence that could be seen from the directions, the loss of resolution did not lead to a viable method, especially when high-resolution color displays became available.

COLOR ASSIGNMENT

Generally, we would like to decide interactively, through some sort of user interface, which patterns we would like to display. Our display technique is to assign a unique color to each data point so that we do not lose resolution.

A simple display technique would be to display “strong flows” by display of those patterns that have a high degree of coherence with three or more aligned arrows. If we are looking for sources or sinks, we might display only those patterns where there are opposing arrows. But in other applications, we are looking for particular patterns.

Our base method of color assignment is to use a small set of carefully chosen colors. These colors should obey some standard practices to ensure compatibility with properties of the human visual system [4][6]. In particular, we want to choose highly saturated colors but we do not want opposing colors to be adjacent. For example, a basic display pattern would be to use color to show direction of flow. We might color all cases in which three or more arrows point in the same direction with the same color. If we were to choose colors equally spaced on the color wheel, we might have red opposite yellow and green and blue almost opposite. On a perceptual basis, we are better off if we use red/green and blue/yellow for the opposite directions. If we want to display more than four types of patterns we tend to select colors that are far from blue/green/yellow/red in color space and include colors such as black, white, brown, and gray.

EXAMPLE: VISUALIZING OCEAN FLOW

The problem that motivated this work is the visualization of flows in the ocean from simulation data. Prediction of global climate is one of the grand challenge problems. Global ocean models have been increasingly sophisticated over the last 20 years. These models now compute a resolution of 1/10 degree, which means there are 3600 points around the equator. Although there is some scaling that reduces the number of points needed at the poles we have two-dimensional data sets that consist

of approximately ten million points. At each of these points, we might have values for the flow, temperature and salinity.

While surface models with two spatial dimensions and time are important in many studies, many others require three spatial dimensions. One of the major problems is the study of thermohaline circulation patterns. Because salt heavy water drops and warmer water rises, there are circulation patterns that must take into account the depth of the ocean. The latest models use 80 levels in depth, in addition to the 1/10 degree resolution along the surface. Because these thermohaline circulation patterns involve slow flows, oceanographers are interested in modeling these patterns over periods measured in the tens of years. In addition, some of the most important features of these flows are small branches of flow that leave the large current circulating the Antarctic.

One of the problems in displaying this type of data is that we want to show both land masses, patterns of flow, and areas of small flow. Generally this requirement forces us to use two colors, sometimes black and white, for the land and low-flow regions.

Color Plate 1 shows a visualization of flows in the North Atlantic. The data set is 1152 x 1280 and shows the flows on the surface layer. Here we encode land as gray and encode the patterns with five arrows all pointing in the same directions as red for right-moving flows, green for left-moving flows, blue for downwards flows, and yellow for upwards flows. The pattern of all zeros, corresponding to all vectors below threshold, is encoded as white.

One question that arises is whether or not this display conveys any more information than simply encoding the flow at each point using same color scheme [2]. For this choice of patterns, the major difference is that with *marching flow* we see distinct outlines at the borders between flow of different directions and between flows that are above and below a user selected threshold flow.

Color Plate 2 shows a different visualization of the same data set where different patterns are assigned the most distinct colors. All the coherent patterns are gray; all the low flow data are white; red shows coherent regions with three arrows in the same direction; green shows all the other cases, which include patterns with opposing flow and patterns with only two arrows in the same direction. This color assignment acts as an edge detector highlighting points where the flow changes or disappears.

EXAMPLE: MANDELBROT SET

The next example shows the advantages of *marching flow* for detecting complex patterns rather than coherent flows. The data are from the Mandelbrot set in which we have a complex number at each spatial location. For visualization purposes, we treat each complex number as a two-dimensional “flow” vector. In Color Plate 3, we show the flow encoded as in Color Plate 1 which shows a recognizable Mandelbrot set displayed in an unusual

manner. For Color Plate 4, we adopted the following color scheme:

red = opposing flow
blue = sink
yellow = source
gray = coherent cases
black = all other cases

Here we define opposing flow as a point in which there are different arrow directions among the point and its neighbors. A sink is place where the flow is below threshold at the point but the neighbors have one or more arrows pointing inward. A source is the opposite of a sink. Coherent flow includes those cases in which all the arrows point in the same direction.

THREE-DIMENSIONAL FLOW

In principle, we should be able to extend *marching flow* to three-dimensional flow problems without much difficulty. However, there are practical problems. In three dimensions, each point has 6 neighbors. If we classify flows as before, there are 7 choices for the point and its neighbors, resulting in 7^7 or 823543 patterns. Although there are symmetries and we could write a program to classify these patterns, it is not clear this would be a useful approach. Rather, we can proceed as with two-dimensional flow and classify patterns starting with the most coherent arrows and providing a user interface that allows selection of patterns.

CONCLUSIONS

Color methods have much to recommend them for visualizing large flow data sets, especially as part of a hierarchical process. They are fast, well-suited to properties of the human visual system, and easy to parallelize. The question is whether *marching flow* is better than direct color methods that do not make use of neighboring data points.

We argue that there are advantages. The *marching flow* method highlights changes in the flow patterns. This is analogous to the use of edge detectors in image processing. Although edges are in images, once they are displayed alone or enhanced in the original image, more information is conveyed to the user. For flows the situation is more complex as we are often interested in places where there is a source or a sink or places where there is turbulence. By choosing which patterns we display, we can offer the user a variety of displays, which should prove useful for interactive visualization of large flow data sets.

A final important point is to note that this method is not limited to just flow visualization, but may be applied to any vector data set.

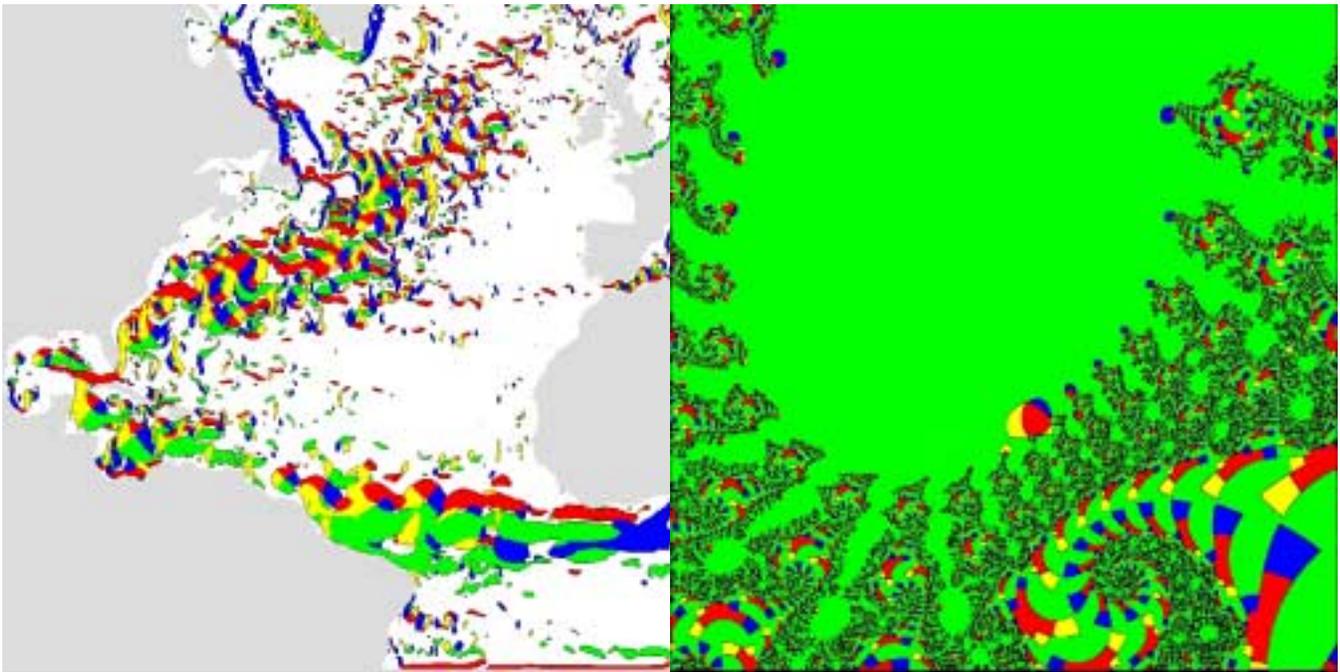
ACKNOWLEDGEMENTS

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.

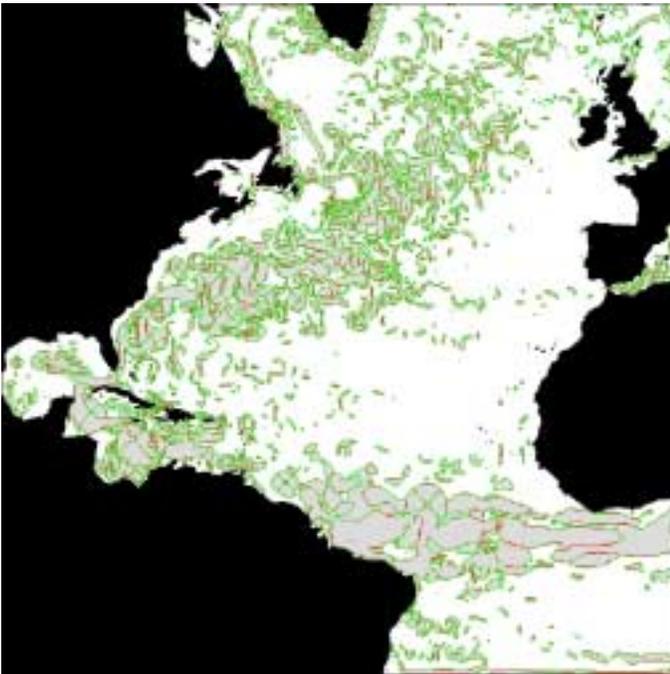
REFERENCES

- [1] Cabral, B. and L. Leedom. Imaging Vector Fields Using Line Integral Convolution. *Computer Graphics Proceedings Annual Conference Series*, pages 163-169. August 1993.
- [2] Crossno, P., Angel, E., and Munich, D., Case Study: Visualizing Ocean Currents with Color and Dithering, submitted to *IEEE 2001 Symposium on Parallel and Large-Data Visualization and Graphics*.
- [3] Darmofal, D. and Haines, R. Visualization of 3-D Vector Fields: Variations on a Stream. In *Proceedings AIAA 30th Aerospace Science Meeting and Exhibit*, Reno, Nevada, January 1992.
- [4] Hall, P. Volume Rendering for Vector Fields. *The Visual Computer*. **10** (2): 69-78. 1993.
- [5] Johannsen, A. and R. Moorhead. Case Study: Visualization of Mesoscale Flow Features in Ocean Basins. In *Proceedings of Visualization '94*, pages 355-358. IEEE, October 1994.
- [6] Johannsen, A. and R. Moorhead. AGP: Ocean Model Flow Visualization. *IEEE Computer Graphics and Applications*. **15** (4): 28-33. July 1995.
- [7] Klassen, R.V., and S.J. Harrington, Two-Dimensional Vector Field Visualization by Halftoning, *Scientific Visualization of Physical Phenomena*, pages 363-377, 1991.
- [8] Lorensen, W.E., and H.E. Cline, Marching Cubes, A High-Resolution 3D Surface Construction Algorithm, *Computer Graphics*, 21(4), 163-169, 1987.
- [9] Schroeder, W., Martin K., and Lorensen, B. *The Visualization Toolkit 2nd Edition*. Prentice Hall PTR, pages 167-174, 1998.
- [10] Semtner, A. Ocean and Climate Modeling. *Communications of the ACM*, Vol. 43. No. 4, pages 80-89, April 2000.
- [11] Uselton, S. Volume Rendering for Computational Fluid Dynamics: Initial Results. Technical Report RNR-91-026, NAS-NASA Ames Research Center, September 1991.

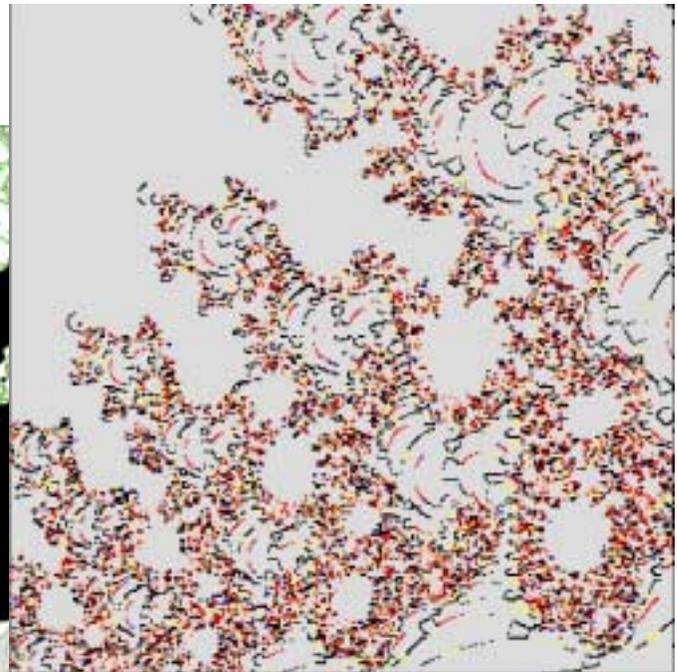


Color Plate 1: North Atlantic flows on surface layer.
Coherent flow shown by color:
 red = right, blue = down,
 green = left, yellow = up,
 gray = land,
 white = below threshold

Color Plate 3: Mandelbrot set using color-encoding scheme of Color Plate 1.



Color Plate 2: Edge detection for North Atlantic flows using the following color encoding scheme:
 gray = coherent patterns,
 white = low flow,
 red = coherent regions (3 arrows in same direction);
 green = all the other cases.



Color Plate 4: Close-up of Mandelbrot set using the following color encoding scheme:
 red = opposing flow,
 blue = sink,
 yellow = source,
 gray = coherent cases,
 black = all other cases.