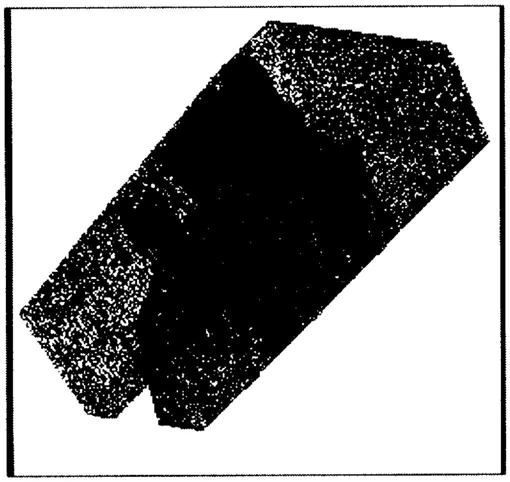
NASA-CR-202440

97728



3D Feature Extraction for Unstructured Grids

by D. Silver

Abstract

Visualization techniques provide tools that help scientists identify observed phenomena in scientific simulation. To be useful, these tools must allow the user to extract regions, classify and visualize them, abstract them for simplified representations, and track their evolution.

Object Segmentation provides a technique to extract and quantify regions of interest within these massive datasets. This article explores basic algorithms to extract coherent amorphous regions from two-dimensional and three-dimensional scalar unstructured grids. The techniques are applied to datasets from Computational Fluid Dynamics and those from Finite Element Analysis.

Introduction

Large scale experiments and numerical simulations generate massive amounts of data which must be interpreted for accurate prediction and understanding. This process involves searching and analyzing the data for features or effects: objects or structures which have space-time coherence so they are identifiable and have a finite lifetime (possibly a coherent region of high/low intensity). The persistence of an unexpected effect under variations of system parameters and external perturbations is a manifestation of something new. The scientist must then identify and quantify the effect: what is it; what is its cause; how does it evolve; how long does it persist, etc. The aim in the different disciplines is to study the evolution and essential dynamics of these objects and describe them for modified time periods, thus obtaining a partial solution or reduced-and-simpler model of the original problem. For example, one tracks the progression of a storm for weather prediction, the change in the ozone ‘‘hole" and the greenhouse effect, or the movement of air over an aircraft or automobile.

Visualization is an intermediate step in this process. It provides access to new domains and allows us to identify observed phenomena interactively. However, to fully study noted effects, feature based methods

http://www (caip.rutgers.edu/vizlab.htm) must be provided. These tools would enable the user to extract regions, visualize them, perform local measurements, classify and catalogue them, abstract for simplified representations (reduced models), and track their evolution.

Although each scientific domain has its own set of ''interesting'' features, generic procedures can be used to extract these regions, much like the rendering algorithms which are applicable to medical images, fluid dynamics, and meteorology (e.g.volume rendering or isosurface contouring). Many of the procedures can easily be incorporated into the standard packages as modules. These methods incorporate ideas from computer graphics, image processing, computer vision, and mathematical morphology.

Features are fundamental to the analysis/visualization process for a number of reasons:

- Reduction of Visual Clutter: important regions can be focussed upon and visualized unobstructed. In this image a 512^3 turbulent dataset using a volume rendering technique to display vorticity magnitude. A more informative view of the dataset would contain each region isolated and measured as in this image where a 128^3 section of the dataset has been extracted, and all the "features" in the data identified and measured. (See this paper for more details.)
- Localized Measurements: objects can be measured individually, i.e. volume, or surface area of a particular localized region;
- Structure Cardinality: the number of features found can provide insights into the evolving dynamics of the field;
- Classification: features can be automatically identified and classified (especially helpful for analyzing massive amounts of data);
- Tracking: once important regions are identified they can be tracked in time to understand their evolutionary paths;
- Juxtaposition: features can be compared or ''juxtaposed'' to other features from different datasets. For example, when comparing a simulation to a laboratory experiment, it is advantageous to compare individual regions instead of the entire dataset. Similarly, when comparing one simulation/experiment to a later one.
- Data Reduction: by focusing on features the amount of data to be analyzed is effectively reduced and only important features need to be saved;
- Database Management: to aid in the future of storing and retrieving datasets based upon contents, it is fundamental to define features and ''objects'' contained within the dataset.

Feature Extraction

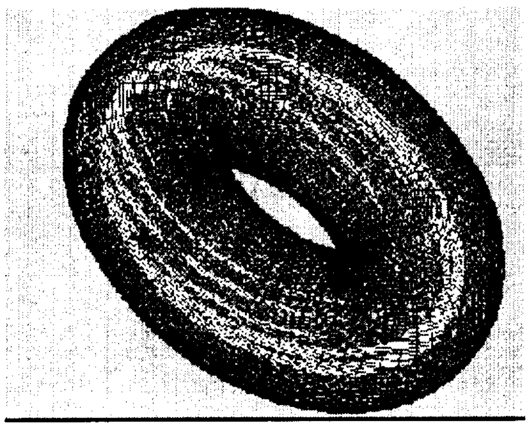
Every domain has a different definition for features, regions of interest, or objects. General methods to define features include segmentation [1], selective visualization [2], volume intervals [3]. (Specific features include vortex cores [4,5, 6], and critical points for vector fields [7], etc.) A basic definition of a feature is a region of interest consisting of voxels satisfying a set of pre-defined criteria. The criteria can be based on any quantities, such as threshold interval, shape, vector direction, and neighborhood connectivity. In [2,8] an algorithm for regular gridded datasets is given which partitions a dataset into connected thresholded regions and background. A region filling algorithm with a starting ''seed'' is used. The seeds are local extremal values. Each feature (set of voxels) is stored in an octree [9,10,11]. Interior nodes of the octree contain the extremal value of that node's subtree. In addition to the actual voxels, global properties such as the centroid, mass, moments, volume, etc.. are computed and stored for each separate feature [12, 13, 14].

These volume features can be visualized using volume rendering or fitting a surface around the boundary. (See here for visual examples.)

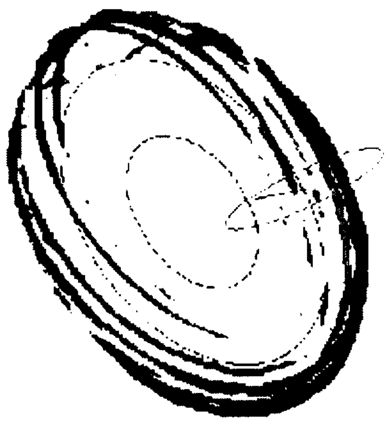
Curvilinear Grids

A curvilinear grid is a 3-dimensional rectilinear grid in computational space which is ''warped'' or subjected to non-linear transformations in physical space so as to wrap around a region of interest. For a dataset with this type of a grid, the object segmentation technique must include the periodicity of the dataset. If an object happens to extend along a boundary, it would be split into two objects if periodicity is not enforced. The object segmentation algorithm described above remains the same with only a change in the definition of the ''neighboring'' points. Points on the boundary of the dataset also have neighbors. These neighboring points are on the opposite face of the dataset. An example of this is illustrated below:

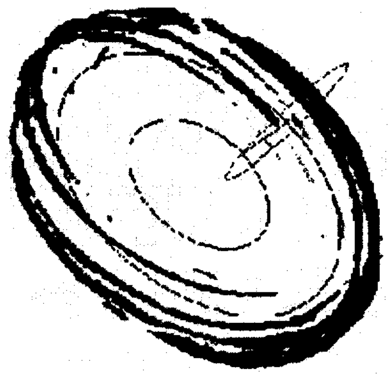
In these figures a curvilinear tokamak dataset is shown with dimensions 32 X 129 X 65. (The dataset was provided by S. Parker, PPPL.) The first image is a standard isosurfacing with indistinguishable features.



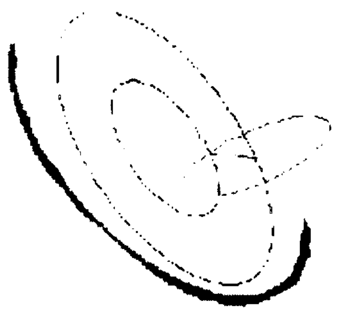
The next image contains segmentation without wraparound: (Note how objects which appear connected get a color change at the boundary.)



And the next contains the proper segmentation:



In this case, each of the objects can be isolated as shown in:



Unstructured Grids

The feature extraction algorithm has been extended to work with unstructured grids (with tetrahedral cells) 15.

To extract features in the dataset, values that meet the threshold criteria are first selected. A seed growing technique is then used to locate the features in the dataset. i.e. regions are grown around seeds which are initially the extrema values in the dataset. The advantage of this method is that for more than one threshold value, or for mapping out the entire threshold sequence of an entire dataset, this algorithm is efficient because it retains knowledge, i.e. when changing thresholds from one threshold to a lower one, existing structures tend to "grow" and therefore should be used as starting points. The position of the neighboring values is not implicit as in the case of a regular dataset. For a given point, its neighboring points are the other three vertices of the tetrahedron it is part of. Therefore, it is necessary to keep track of the adjacency lists.

To avoid scanning the dataset each time for information about connectivity, a list of tetrahedra that have a given point in common is maintained as part of the data structure. All the points associated with the tetrahedra that form the region are flagged as ''used''. A new set of seed points is selected from the maxima of the remainder of the points. This algorithm is executed over the dataset iteratively until there are no more ''active'' points (i.e. separate growing regions). For multiple threshold values, at the end of each iteration, different objects are merged if needed, since regions which were separate for one threshold may merge for a lower one. The final result of the algorithm is a set of distinct objects each represented as a list of tetrahedra. A marching tetrahedra algorithm is then applied directly to each object to obtain the bounding surfaces. These surfaces are represented as triangular polygons which can be directly rendered by standard graphics toolkits.

Once the objects have been identified, they need to be quantified. Parameters like Volume, Mass, Centroid, Moments, etc. as described earlier are computed for each of the distinct objects found.

Memory requirement and simplicity have been the key considerations for the design of the data structures used to implement the segmentation algorithm. The dataset is read into a structure that is composed of four lists. The elements in each list can be addressed by an index into the respective arrays.

- The Data-List: This is an array of data values at each point of the dataset. Each point can have up to three data values, which can be accessed by an offset into this list. The list consists of N x M elements, where N is the number of points in the dataset and M is the number of data values.

- The Position-List This is the list of all points in the dataset. This list contains the x, y, z coordinates of the point, a set of flags and a pointer to the Cell-List. The Cell Pointer points to the list of tetrahedra which have this point in common (an adjacency list). This list is created once at startup.

- The Cell-List: This is a list of all the tetrahedral cells in the dataset. Each cell is represented by pointers to the four vertices of the cell. Each pointer is an index into the Position-List described above. This list is also generated only once at startup from the input data file.

- The Object-List: This is a list of objects that have been generated by the segmentation algorithm. The Cell pointer points to a linked list of tetrahedral cells that comprise this object. This list can be used directly to calculate the mass, volume and other physical parameters for the objects. For visualization, a bounding surface can be generated and the list of triangles for the surface is pointed to by the Polygon Pointer. Each triangle consists of the x, y, z coordinates for each of the vertices.

```
Preprocess the data, to create the data, cell, and position lists.
(only vertices above the threshold value are processed into the position list).
While there are no points marked USED
  Determine Set of Points M(i) with highest data
    value and not marked USED.
      This value is the highest value not under the threshold desired.
  For each point in M(i) perform the following
     Create a new entry in the Object List, O(i).
     Add all the cells in the adjacency list of M(i) to O(i)
   Now grow the objects.
  For each Object, O(i), in the Object-List
    For each Cell in the Object-List
      Determine the member vertices from the Cell-List (mark value as used).
      Determine the Cells connected to this vertex from the
             adjacency List of the Position-List.
      Add all the adjacent cells to the Object-List.
```

If a sequence of thresholds is desired, the algorithm is executed for each threshold value in descending order to take advantage of the "growth" of regions from highest threshold to lowest (or vice versa for variables which contain the minima as the extremal values). In this case, objects may "merge". Additional testing is needed when growing the object. This is done with a flag on each cell stating its object number. Once the object list is created, a marching tetrahedra algorithm is implemented. For a full description of the algorithm, see 15.

Examples

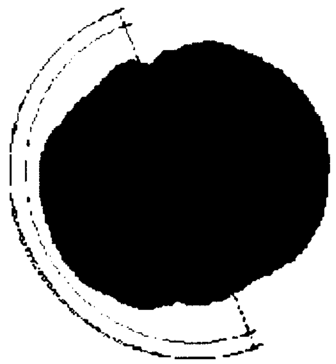
The segmentation technique was applied to three unstructured datasets. (In addition, for testing some regular datasets were converted to unstructured datasets.)

The first dataset was provided by Lloyd Treinish. This dataset is the global atmospheric temperature (ATM) described on a tetrahedral mesh. It has 311,040 tetrahedral elements and 74,095 points.

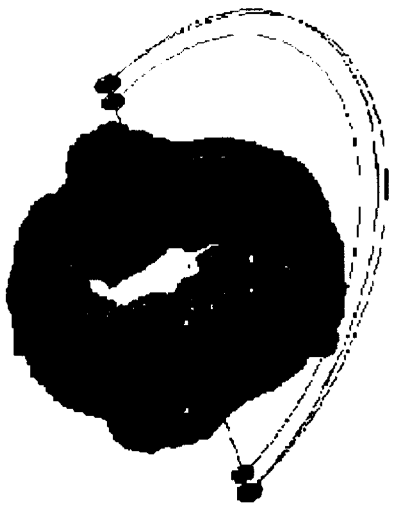
The second dataset was taken from the NASA web site provided by T. J. Barth and S. W. Linton. It has 595,536 tetrahedra and 112,551 points. This dataset consists of data for a multiple component wing computation (WING) of inviscid compressible flow (Mach = .2 and alpha = 0 deg).

The third dataset has 106,029 tetrahedra and 21,855 points (DATA3).

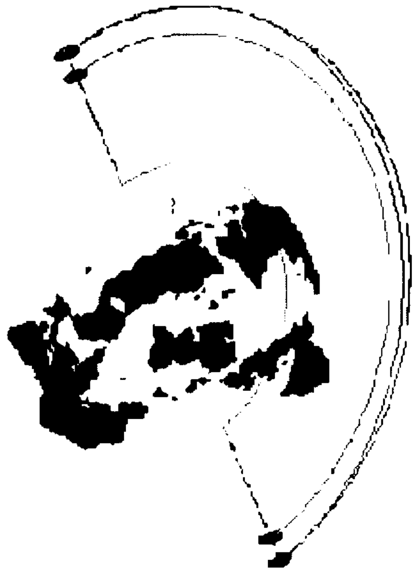
ATM dataset:



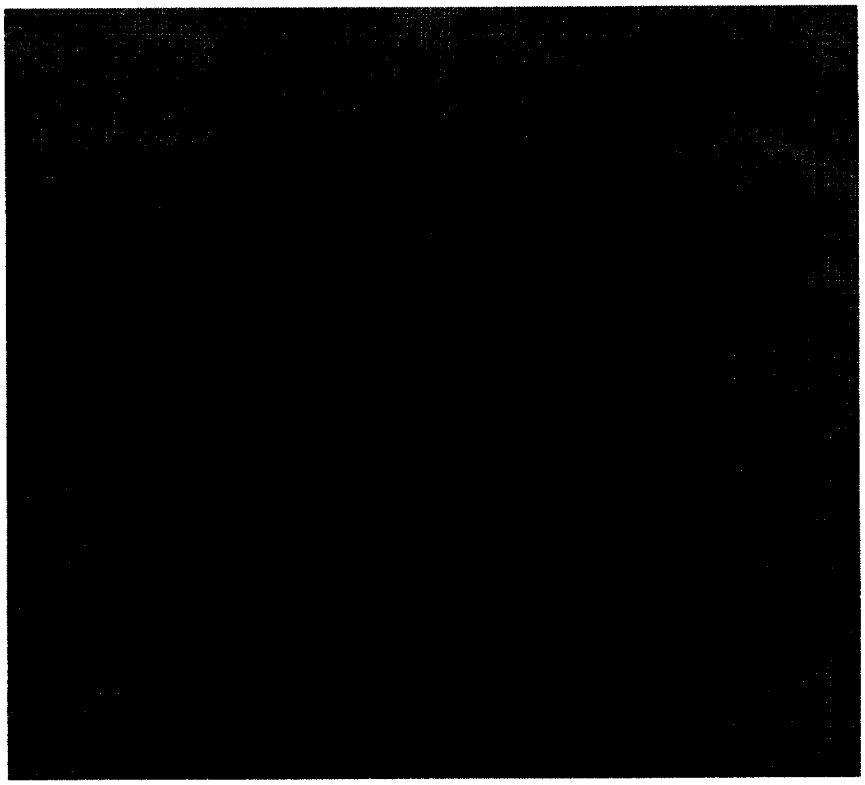
Threshold, 80% of maximum:


Threshold, 92% of max:

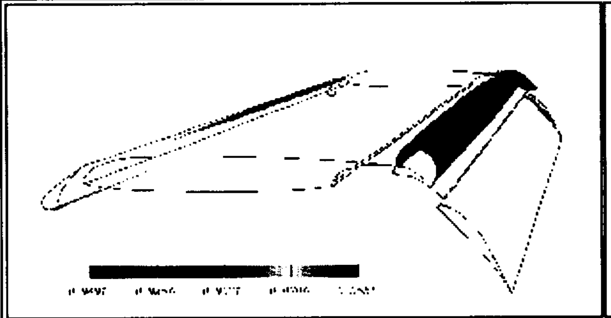
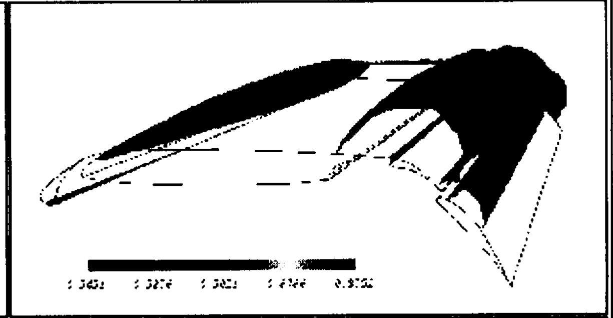
Threshold, 99% of maximum, note how each separate component is colored individually:

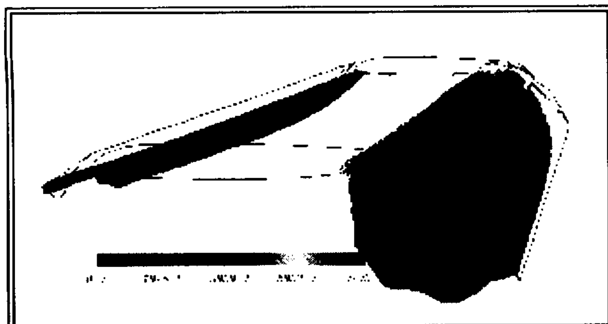
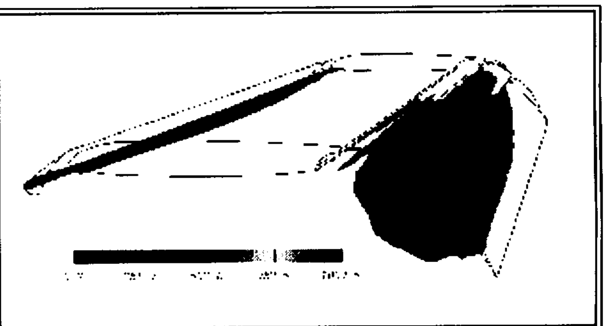
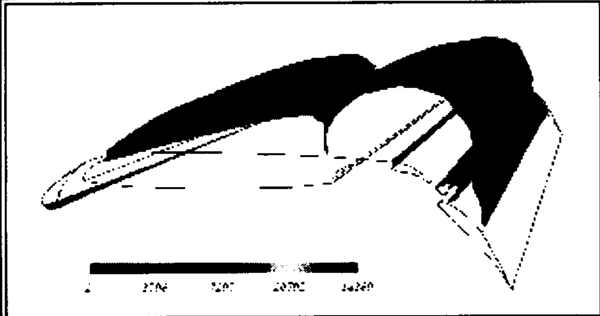
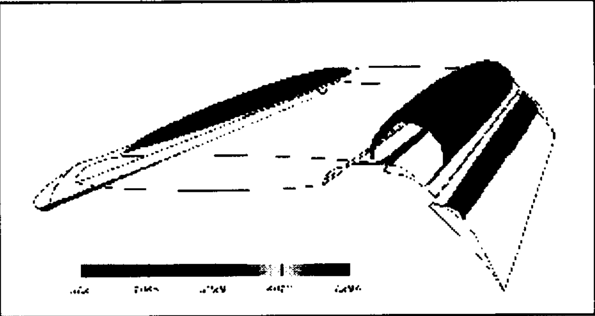


WING (segmentation based on density, pressure, & mach number):



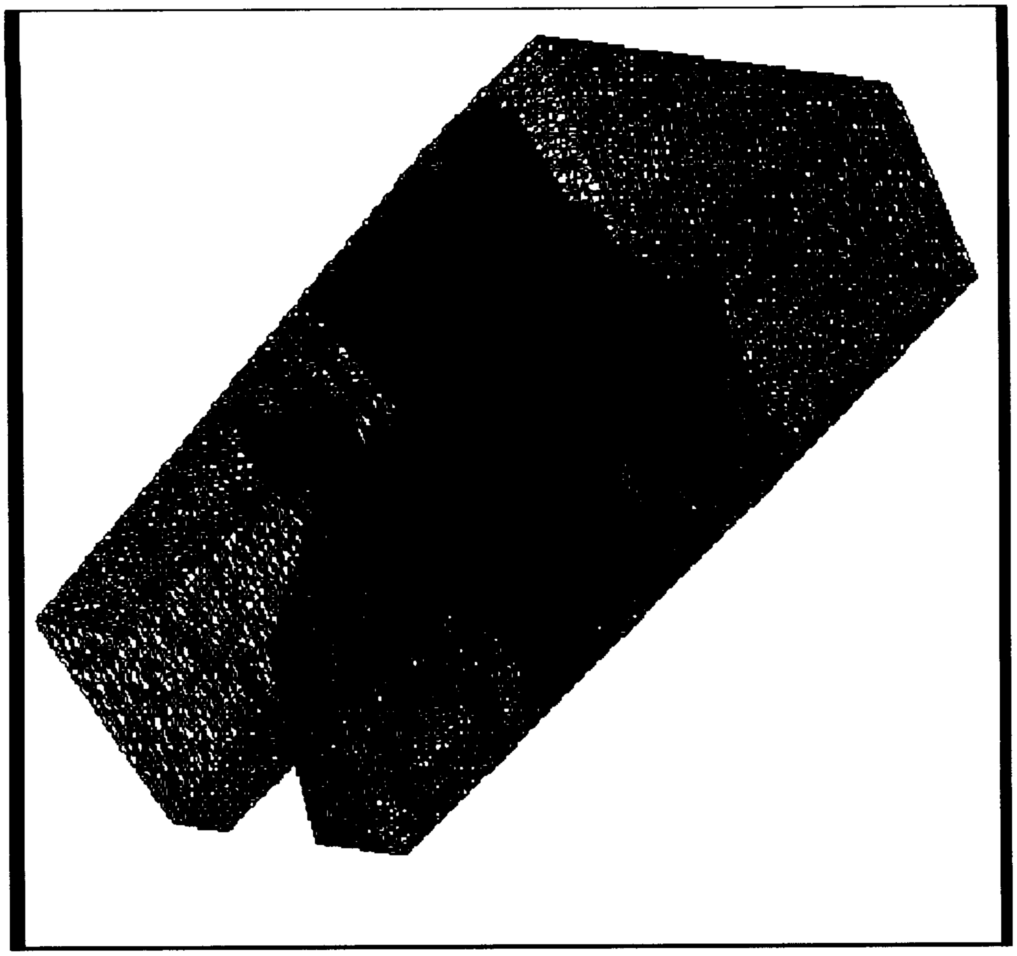
The following images represent the inner central region inside the dataset displayed above. The outline of the wing can be seen as a dark spot in the above dataset.

	
Threshold of 0.95 Kg/m ^{3} (variable: density): bellow threshold Coloring according to local minima	Threshold of 0.97 Kg/m ^{3} (variable: density): bellow threshold Coloring according to local minima

	
Threshold of 0.723 (variable: pressure): above threshold Coloring according to integrated content	Threshold of 0.73 (variable: pressure): above threshold Coloring according to integrated content
	
Threshold of 0.30 mach (variable: mach number): above threshold Coloring according to volume	Threshold of 0.33 mach (variable: mach number): above threshold Coloring according to volume

DATA 3 (pressure):

Threshold, 10% of maximum:



Conclusion

Features are crucial to understanding the results of complex numerical simualations. Above, we have discussed methods to extract simple features from both structured and unstructured grids. Once features are isolated they can be quantified and tracked. (See here and [16] for more information about tracking.)

This work was supported by NASA Ames (NAS) Research GRANT: NAG 2-829

For more information contact:

Deborah Silver
(silver@vizlab.rutgers.edu)
Smitha Bhat
(sbhat@vizlab.rutgers.edu)
Amos El-Roy
(someone@vizlab.rutgers.edu)

References

1 D.Silver. Object-oriented visualization. IEEE Computer Graphics and Applications, 15(3), May 1995.

2 T van Walsum. Selective Visualization on Curvilinear Grids. PhD thesis, Delft University of Technology, Delft, The Netherlands, 1995.

3 B. Guo. Interval Set: A Volume Rendering Technique Generalizing Isosurface Extraction. In Proceedings IEEE Visualization '95, pages 3--10, Atlanta, Georgia, October 29-November 3 1995.

4 B. Singer and D. Banks. Predictor-Corrector Scheme for Vortex Identification. Technical report, NASA Langley, March 1993.

5 J.Villasenor and A.Vincent. An Algorithm for Space Recognition and Time Tracking of Vorticity Tubes in Turbulence CVGIP: Image Understanding, 55(1):27--35, January 1992.

6 J.~D. Buntine and D.~I. Pullin. Merger and Cancellation of Strained Vortices. J. Fluid Mech., 205:263--295, 1989.

7 J. Helman and L. Hesselink. Visualization of Vector Field Topology in Fluid Flows. IEEE Computer Graphics and Applications, 11:36--46, May 1991.

8 M.Gao. Data Extraction and Abstraction in 3D Visualization. Master's thesis, Graduate School - New Brunswick, Rutgers, The State Uni versity of New Jersey, March 1992.

9 H. Samet. The Design and Analysis of Spatial Data Structure. Addison-Wesley, Reading, Massachusetts, 1989.

10 A. Globus. Octree optimization. In Symposium on Electronic Imaging Science and Technology, SPIE/SPSE. SPIE/SPSE, 1991.

11 J. Wilhelms and A. Gelder. Octrees for Faster Isosurface Generation. ACM Trans. on Computer Graphics, 11(3):201--227, July 1992.

12 D. Silver and N. J. Zabusky. Quantifying Visualizations for Reduced Modeling in Nonlinear Science: Extracting Structures from Data Sets. Journal of Visual Communication and Image Representation, 4(1):46--61, 1993.

13 X.Wang, D. Silver, and S. Bhat. Visualization Tools for Feature Extraction and Quantification of 3D Datasets: User Manual. Technical Report TR-199, CAIP center, Rutgers University, Piscataway, NJ, 1995.

14 F. Post and T. van Walsum and F. H. Post and D. Silver. Iconic Techniques for Feature Visualization. In Proceedings of IEEE Visualization '95, pages 288--295, Atlanta, Georgia, October 1995.

15 S. Bhat. Segmentation of Unstructured Grids. MS. Thesis, Dept. of Electrical and Computer Engineering, Rutgers University. Piscataway, NJ, 1996.