[30] B.L. Yeo and B. Liu. Volume rendering of dct-based compressed 3-d scalar data. *IEEE trans on Visualization and Computer Graphics*, 1(1):29–42, March 1995.

[31] S.-Y. Chen and X. Shan. High-Resolution turbulent simulations using the Connection Machine-2. *Computers in Physics*, 6(6):643–646, 1992.

[32] V. M. Fernandez, N. J. Zabusky, S. Bhat, D. Silver, and S.-Y. Chen. Visualization and feature extraction in isotropic navier-strokes turbulence. In *Proceedings of the AVS95 conference*, Boston, April 1995.

[33] X. Wang and D. Silver. Volume Tracking – Video Animation, 1995. MPEG version on http://www.caip.rutgers.edu/vizlab.html.

[34] Arunava Banerjee, Haym Hirsh, and Tom Ellman. Inductive learning of feature-tracking rules for scientific visualization. Workship on Machine Learning in Engineering(IJCAI-95), March 1995.

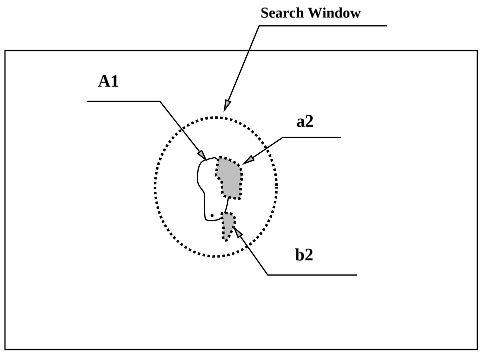

Figure 15: A search window to catch drifting objects. A_1 is from t_i, a_1 and a_2 are from t_{i+1}.

to Dr. N. Zabusky, Dr. V. Fernandez, J. Ray, D. Epstein, B. Yeo and R. Samtaney for some valuable discussions. The authors are grateful to Dr. V. Fernandez and S.Y. Chen for providing the data set. The lab acknowledges the support of ARPA HPCD (DABT-63-93-C-0064), DOE (DE-FG02-93ER25179.A000), NASA (NAG 2-829), and the CAIP Center. The parallel computations were performed on the CM5 at the National Center for Supercomputing Applications. The video animations can be seen on our web site, *http://www.caip.rutgers.edu/vizlab.html*

References

[1] Al Globus. A software model for visualization of unsteady 3-D computational fluid dynamics results. AIAA 95-0115, AIAA 33rd Aerospace Sciences Meeting and Exhibit, Reno, NV, January 1995.

[2] O.N. Boratav, R.B. Pelz, and N.J. Zabusky. Reconnection in Orthogonally Interacting Vortex Tubes: Direct Numerical Simulations and Quantifications. *Physics of Fluids A*, 4(3):581–605, 1992.

[3] R. Samtaney, D. Silver, N. Zabusky, and J. Cao. Visualizing Features and Tracking Their Evolution. *IEEE Computer*, 27(7):20–27, July 1994.

[4] P. Woodward. Interactive Scientific Visualization of Fluid Flow. *IEEE Computer*, 26(10), October 1993.

[5] J. Jimenez, A. Wray, P. Saffman, and R. Rogallo. The structure of Intense Vorticity in Isotropic Turbulence. *J. Fluid Mech.*, pages 65–90, 1993.

[6] J. G. Brasseur and Wen-Quei Lin. Structure and Statistics of Intermittency in Homogeneous Turbulent Shear Flow. *Advances in Turbulence*, 3, 1991.

[7] D.H.Ballard. *Computer Vision.* Prentice-Hall,Inc, Englewood, New Jersey, 1982.

[8] B. Jahne. *Digital Image Processing.* Springer Verlag, 1991.

[9] I. Carlbom, I. Chakravarty, and W. Hsu. Integrating Computer Graphics, Computer Vision, and Image Processing in Scientific Applications. *Computer Graphics*, 26(1):8–16, January 1992. SIGGRAPH '91 Workshop Report.

[10] A. Pentland and B. Horowitz. Recovery of nonrigid motion and structure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(7):730–742, 1991.

[11] D. Metaxas and D. Terzopoulos. Shape and nonrigid motion estimation through physics-based synthesis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(6):580–591, June 1993.

[12] Stan Sclaroff and Alex Pentland. Modal matching for correspondence and recognition. Technical Report 201, The Media Laboratory, Massachusetts Institute of Technology, May 1993.

[13] Gabriel Taubin and David B. Cooper. Recognition and positioning of rigid objects using algebraic moment invariants. volume 1570, pages 175–186. SPIE–The International Society for Optical Engineering, SPIE, 1991.

[14] Y. Arnaud, M. Desbois, and J. Maizi. Automatic tracking and characterization of african convective systems on meteosat pictures. *Journal of Applied Meteorology*, 31(5):443–453, May 1992.

[15] 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.

[16] D. Silver, N. J. Zabusky, V. Fernandez, M. Gao, and R. Samtaney. Ellipsoidal Quantification of Evolving Phenomena. In N. M. Patrikalakis, editor, *Visualization of Physical Phenomena. Proceedings of Computer Graphics International '91 Symposium*, pages 573–588. Springer-Verlag, June 1991.

[17] T van Walsum. *Selective Visualization on Curvilinear Grids.* PhD thesis, Delft University of Technology, 1995.

[18] Deborah Silver. Object-oriented visualization. *IEEE computer graphics and applications*, 15(3), May 1995.

[19] Baining Guo. Interval set: A volume rendering technique generalizing isosurface extraction. In *Proceedings Visualization '95*, pages 3–10, Atlanta, Georgia, October 29-November 3 1995. IEEE Computer Society Technical Committee on Computer Graphics, IEEE Computer Society Press.

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

[21] J. D. Buntine and D. I. Pullin. Merger and Cancellation of Strained Vortices. *J. Fluid Mech.*, 205:263–295, 1989.

[22] J. Helman and L. Hesselink. Visualization of Vector Field Topology in Fluid Flows. *IEEE Computer Graphics and Applications*, 11:36–46, May 1991.

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

[24] Xin Wang, Deborah Silver, and Smitha Bhat. Visualization Tools for Feature Extraction and Quantification in 3D Dataset:User Manual. Technical Report TR-199, CAIP center, Rutgers University, CoRE Building-Frelinghuysen Road, Rutgers University, Piscataway, NJ 08855-1390, 1996.

[25] 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.

[26] H. Samet. *The Design and Analysis of Spatial Data Structure.* Addison-Wesley, Reading, Massachusetts, 1989.

[27] Jane Wilhelms and Allen V. Gelder. Octrees for Faster Isosurface Generation. *ACM Trans. on Computer Graphics*, 11(3):201–227, July 1992.

[28] Frank J. Post, Theo van Walsum, Frits H. Post, and Deborah Silver. Iconic techniques for feature visualization. In *Proceedings Visualization '95*, pages 288–295, Atlanta, Georgia, October 29-November 3 1995. IEEE Computer Society Technical Committee on Computer Graphics, IEEE Computer Society Press.

[29] Al Globus. Octree optimization. In *Symposium on Electronic Imaging Science and Technology, SPIE/SPSE.* SPIE/SPSE, 1991.

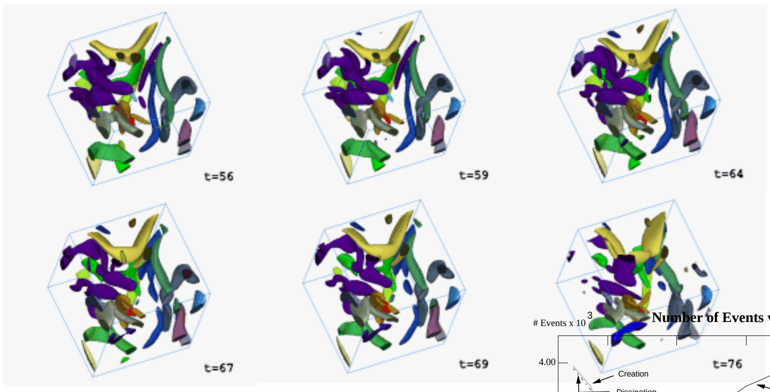

Figure 14: Simulation 2: tracking

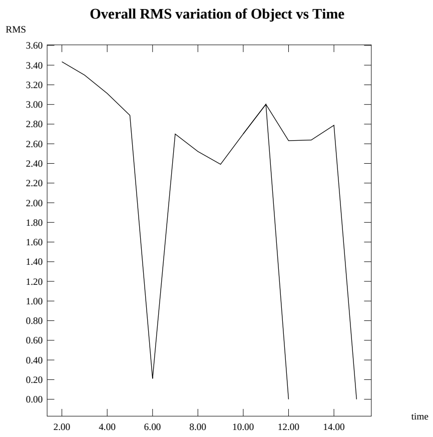

Figure 12: RMS variation of red feature.

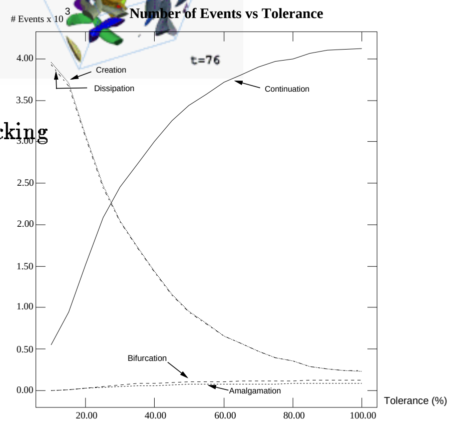

Figure 13: Events vs tolerance.

coded visualization, the evolutionary history of features can be represented as a directed acyclic graph or arrow plot.

7 Conclusions and Future Work

The algorithm presented in this paper is dependent upon a high sampling rate. However, this condition is not always available. Furthermore, to reduce the total size of time-varying datasets, we may want to use fewer time steps. We are investigating methods which can relax the basic assumption on the sampling frequency. Obviously, the tolerance is sensitive to Δt, and is also domain dependent. Because of the dependence upon overlapping between datasets, objects which drift will not be tagged as continuing. A search window around the object being tested can be used to locate non-overlapping neighboring objects (See Figure 15), which can be added to the candidates for a best matching test. The objects must then be transformed (by their centroid and moments) before a difference is computed. This type of operation would also be useful to compute a true RMS between evolving features. The size of the search window is based on the sampling frequency.

Ideally, if we can obtain heuristics about how the object evolve or interact in subsequent datasets, we can use that information to perform a correlation test. The potential candidates of correlated objects in the next dataset may be predicted with an inductive learning algorithm and the information lost by sub-sampling can be recaptured. We have started experimenting with some inductive learning methods [34]. We hope we can relax the sampling assumptions and make the program more robust.

The process of feature extraction and feature tracking is an intensive computational process. To deal with high resolution datasets, parallel or distributed algorithms are needed to speed up the process and meet the huge memory requirement. We have been working on the implementation of the feature tracking while the simulation is being computed.

In this paper, we have presented a volume tracking algorithm for 3D time varying datasets and have shown how it can substantially improve the quality of the visualization. The work we presented here is an expansion of our previous work [3] and we have succeeded in producing more accurate tracking results. We exploited the octree data structure to improve the performance of the algorithm. In addition, because of the exhaustive nature of the tracking, all features are categorized and all events are labeled. This catalogue of information can now be used for event classification and searching of massive 4D data sets.

8 Acknowledgment

The work presented here was done with the help of the members of the Laboratory for Visiometrics and Modeling and the CAIP center at Rutgers University. Special thanks

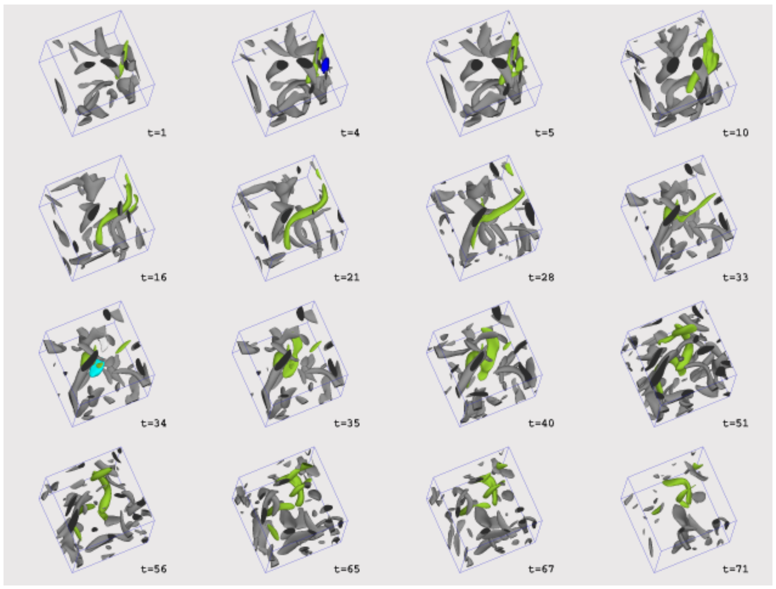

Figure 7: Feature tracking (Simulation I). The evolution of the light green feature (from t=1) is highlighted. All other features are colored grey. When an object is about to merge with the light green one it is given a different color (as in t=4).

Events	Number of events
Continuation	3722
Creation	658
Dissipation	652
Bifurcation	108
Amalgamation	77

Table 1: Number of events classified for Simulation I of 100 turbulent datasets with 128^3 resolution.

the RMS variation for the object tracked in Figure 11 over time. Between t_5 and t_6 the feature barely changes (which can be seen in both the mass plot and RMS plot). It then breaks up before dissipating. Features can also be transformed (translated by their centroids and rotated) to get better results.

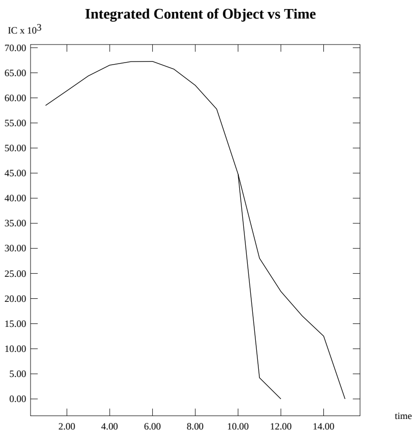

Figure 11: Mass change of red feature from $t = 1$. Split line indicates bifurcation.

6.1 Tolerance

The tolerance is sensitive to the rate of change in the volume of features in the time-varying datasets and the drifting speed of these features. The tolerance chosen depends on the nature of the datasets being investigated, as well as the sampling time. For the datasets used in this section, we found that choosing the tolerance between 50% - 60% can best account for the event classification. Some errors exist when small objects are moving too fast. Either the tolerance is too low, or the overlapping condition is not satisfied. This can be easily overcome with a higher sampling frequency. (Determining the "correct" tracking is difficult without doing exhaustive testing. Visual testing, while useful, is not always accurate.)

We studied the sensitivity of our algorithm to the tolerance by changing the tolerance and counting events. If the tolerance is too low, many more features will be tagged as dissipation/creation instead of continuation. The number of events as a function of tolerance is shown in Figure 13. Similarly, when the tolerance is set too high, regions get marked as continuation when they are not.

6.2 Example 2

Simulation 2 is another pseudo-spectral simulation of coherent turbulent vortex structures. The variable being visualized is vorticity magnitude at 48% of maximum threshold value. This simulation runs from t_30 to t_80. Six datasets from this simulation are given in Figure 14. (The full animation is in [33]). All of the features are tracked and the events are classified. Note the breakup of the large purple region.

We have also redone the tracking for the simulation presented in [3] with improved results [33]. In addition to a color

color of a feature is inherited from its parent. When features merge, different schemes can be employed for the coloring of the merged region. Generally, the dominant feature gets to chose the color. The dominance can be calculated using volume, mass, or a local extremal value. For example, if we are interested in understanding the behavior of volume in the evolutionary process, and two features merge, the one with the larger volume assigns the color. Alternately, colors can be added when two features merge. With the color-coded information, the interaction of objects can be understood and detected. In Figure 6 and in the animation [33] the feature with the larger volume assigns the color. The statistics of the different events for the entire simulation (tolerance=60%) are shown in Table 1.

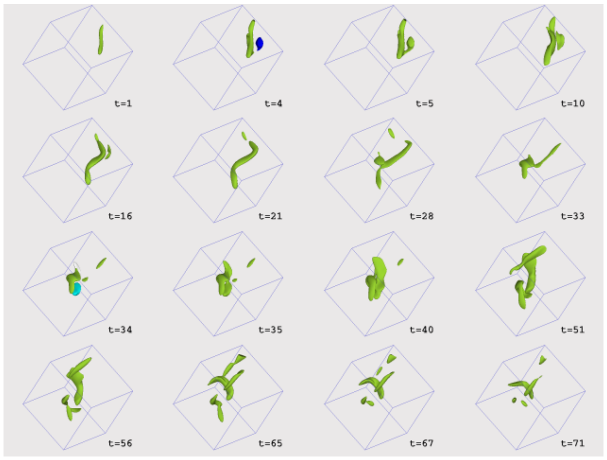

Figure 8: Simulation I: Light green feature is isolated.

Since all regions are tracked and all events are classified, any of one of these can be mapped back to the visualization. In Figure 7, the light green feature of Figure 6 is tracked. Here, 16/100 datasets are shown with one feature colored and all others in grey. When a feature is about to merge with the green one, it is given another color, as in $t = 4$, where a blue object is visible. Because the green is larger, the final merged feature gets the green color. This process can easily be seen in Figure 8 where this event has been isolated. Another complicated process can be observed in t_{33}. The green object in t_{33} splits into three objects in t_{34}. One of which then merges with other objects in t_{35}. The two smaller green objects dissipate in t_{35}. In addition to the visualization, important quantifications can be represented. These are necessary for any thorough understanding of the ongoing evolutionary processes. In Figure 9, the mass change of the green object as a function of time is presented. The split lines indicate bifurcation. A sharp rise in mass can generally be attributed to amalgamation. The number of features as a function of time is shown in Figure 10. This is dependent upon the threshold values chosen for the segmentation process. In this example, the absolute thresholds change as a function of the maximum value in the dataset. Another example of mass tracking is shown in Figure 11. The red cone shaped feature from Figure 6 is tracked. It has a short life and breaks up before dissipating. This type of quantification will help elucidate the energy transfer mechanisms in turbulence as well as the characteristics of turbulent mixing.

In addition to the volume difference between features, a

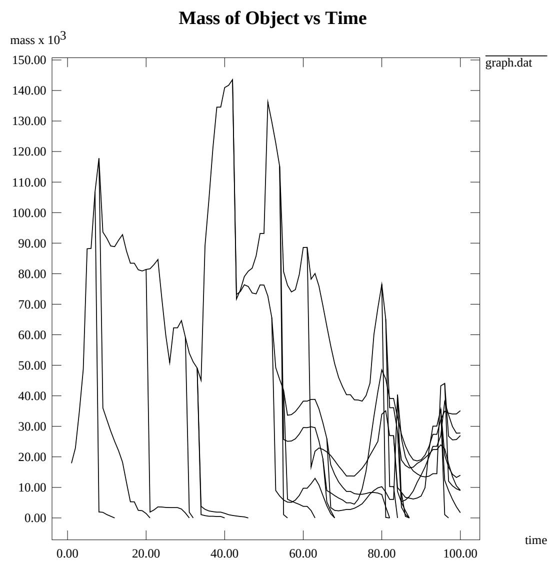

Figure 9: Track of light green feature, mass vs time. The split lines indicate bifurcation. When mass decreases to zero this indicates that the feature has dissipated.

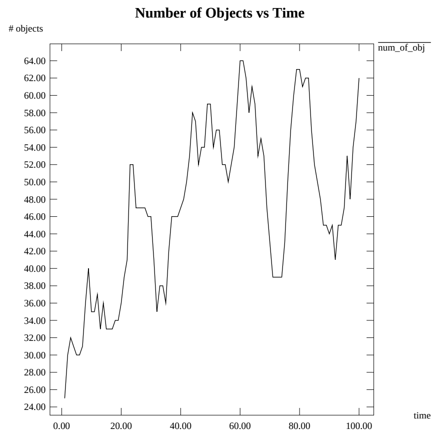

Figure 10: Simulation I: Number of features vs time.

modified root mean squared (RMS) difference can be calculated, to highlight the change in the actual values (vorticity values). The RMS difference is calculated between the corresponding voxels in the same position of the two objects in two time steps (the value to value overlap). It provides us with the rate of evolving vorticity distribution between the isolated objects in different time steps and can be computed while the difference is being determined. The RMS variation is defined as:

$$\left(\frac{\sum (O_{A}{}^{i}_{\vec{x_i}} - O_{B}{}^{i+1}_{\vec{x_i}})^2}{Voxels(O^i_A \bigcup O^{i+1}_B)}\right)^{\frac{1}{2}}$$

where $(O_{A}{}^{i}_{\vec{x_i}} - O_{B}{}^{i+1}_{\vec{x_i}})$ is:

$$= \begin{cases} A_{\vec{x_i}} & \text{if } \vec{x_i} \in \text{Object A and } \vec{x_i} \notin \text{Object B} \\ B_{\vec{x_i}} & \text{if } \vec{x_i} \notin \text{Object A and } \vec{x_i} \in \text{Object B} \\ A_{\vec{x_i}} - B_{\vec{x_i}} & \text{if } \vec{x_i} \in \text{Object A and } \vec{x_i} \in \text{Object B} \end{cases}$$

$A_{\vec{x_i}}$ and $B_{\vec{x_i}}$ denote the scalar value of the data point at $\vec{x_i}$ in object A and object B respectively. Figure 12 shows

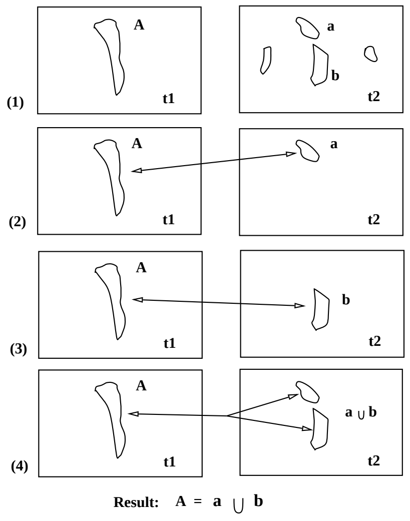

Figure 4: Bifurcation: In this example, feature A in t_1 overlaps with a and b in t_2. Therefore, the corresponding test is performed between object A in t_1 and all combinations of object a and object b in t_2.

it can still be exponential for very large regions breaking up into many small ones). For the turbulent simulation in Section 6, regions are localized and large overlaps do not occur. The ambiguous cases present in our previous algorithm [3], are also resolved because full volume matching is performed.

5 Memory Optimization

The octree implementation involves memory overhead. There have been approaches to reduce the memory overhead [29, 27]. The advantage of an octree, besides the ease at which difference and overlap can be computed, is that only a part of the octree needs to be in the main memory. The nodes in the memory can serve as a look-up table which can be used to obtain the knowledge of its subregion from the summarized information stored in it. The actual data in the octree is loaded into memory only when it is needed for computation.

Storage, manipulation and rendering of 3D time varying datasets is difficult because of the immense amount of data to be processed. Generally, more than one dataset cannot be loaded into main memory. Furthermore, storage of these large datasets is also a major difficulty. A number of different techniques can be incorporated to reduce the memory overhead. Segmentation and tracking can be performed while the computation is progressing (on the supercomputer), and the octree scheme outlined above could be employed. To reduce overhead, the datasets can be stored in a compressed format. Using a lossy compression scheme, we have compressed the datasets shown in Section 6, with a DCT-base compression scheme [30]. With the compression ratio of 24, the errors of attributes between compressed and uncompressed dataset were minimal. For mass, the average error was 0.17%, and for volume the average error was 0.2%. Larger errors were observed with small objects. When the feature tracking algorithm was applied to the compressed datasets, no differences were detected.

6 Examples

We demonstrate our algorithm on two problems from Computational Fluid Dynamics (CFD): both are pseudo-spectral simulations of coherent turbulent vortex structures with 128^3 resolution. Simulation I consists of one hundred datasets which were generated on a Connection Machine (CM5). The datasets contain vorticity. For the visualization and tracking, we use the scalar vorticity magnitude. (The simulation consists of an initial condition of six vortex tubes in parallel and orthogonal positions. A forcing scheme is applied to maintain the energy of the low wave number modes constant [31, 32].) In this example, the features represented are extracted at a threshold of 48% of the maximum vorticity magnitude in each dataset (each feature has its own threshold interval). In Figures 5, four of the one hundred datasets are shown. Note that it is difficult to identify interactions and follow regions (this is especially true in the animation [33]).

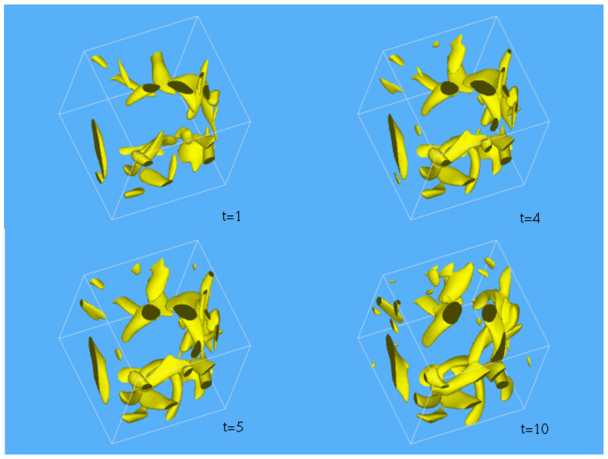

Figure 5: Simulation I: pseudo-spectral simulation of coherent turbulent vortex structures with 128^3 resolution (vorticity magnitude), isosurface threshold=48%. 4/100 time steps are shown.

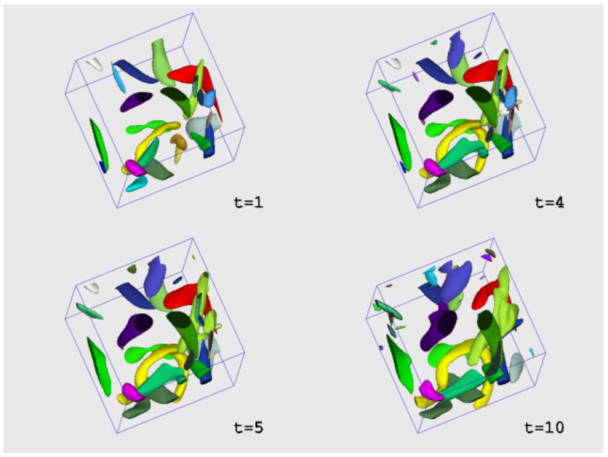

Figure 6: Simulation I: 4/100 time steps. All features are tracked and color coded.

Once the features are identified and color coded, evolutions can be followed. This is demonstrated in Figure 6. The

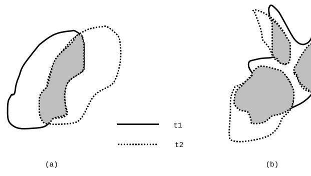

Figure 1: Corresponding objects overlap in consecutive time steps. (a) two features which overlap. (b) one large feature splits into three smaller ones.

we perform a volume difference test. Features O^i_A and O^{i+1}_B are considered matched if

$$\frac{O^i_A * -O^{i+1}_B}{max((O^i_A),(O^{i+1}_B))} < Tolerance$$

The difference, $*-$, is defined as

$$O^i_A * -O^{i+1}_B = max(O^i_A - O^{i+1}_B, O^{i+1}_B - O^i_A)$$

The tolerance is a percentage value normalized to the maximum volume of objects being tested. It can be chosen by the user and it is domain dependent. A tolerance is needed to catch for varying sampling frequencies. In Section 6, we discuss the sensitivity of the algorithm to the tolerance.

The difference function is a volume based operation and it is performed on the voxels of O^i_A and O^{i+1}_B. Since each of the features are stored in octrees, this is defined operation [26] which involves traversing both trees and detecting the differences. The difference operation, is shown in Figure 2.

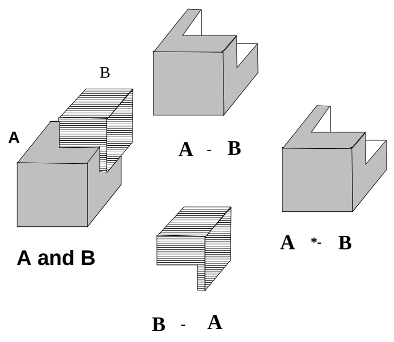

Figure 2: Boolean set operation: Difference

Features must overlap to be matched, so we only compute the difference operation on features which overlap. Since each extracted feature is stored as an octree, O^i_A, an overlap can be detected by "merging" an octree from t_i with the octree forest of t_{i+1}. For example, if O^i_A in t_i overlaps with O^{i+1}_a and O^{i+1}_b in t_{i+1}, the correspondence is determined among the combination of O^i_A with O^{i+1}_a, O^i_A with O^{i+1}_b, and O^i_A with $O^{i+1}_a \bigcup O^{i+1}_b$.

The first step of the feature tracking algorithm is to find out the features in one step which overlap the features in a subsequent time step to obtain potential candidates for testing. These are stored in an overlap list for each feature. The number of nodes of intersection (the difference) is also computed during the overlap test. The best matching test is performed only among the candidates found in the overlapping

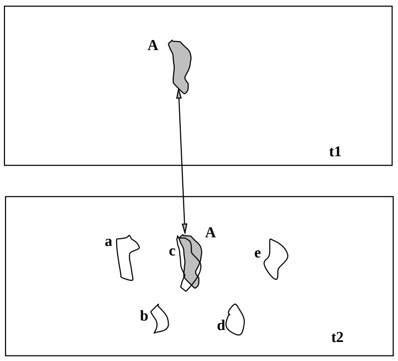

Figure 3: The difference test is performed only on overlapping regions.

test. The best matching proceeds by testing first for bifurcation and continuation and then amalgamation. For bifurcation and continuation, the difference between the template feature (from dataset t_i) and all combinations in the overlap list is computed. The combination with the smallest difference (and still satisfying the tolerance) is chosen, and all features are removed from the search space. Because an octree is used, the overlap test and all the difference calculation can be computed at the same time. The entire algorithm can be summarized as follows:

Feature Tracking

```
{
   For two consecutive time steps $t_i$ and $t_{i+1}$
      Extract all the features from the two datasets and
      store each feature in its own octree.
      Construct the octree forests $F_i = \bigcup_{p \in t_i} O^i_p$
      and $F_{i+1} = \bigcup_{q \in t_{i+1}} O^{i+1}_q$ ;
      *Use $O^i_p$ as a template for matching*
      For each feature $O^i_p \in F_i$ merge it into the octree
      forest, $F_{i+1}$ to
         identify all the overlapping regions of $O^i_p$ in
         $t_{i+1}$. Store this in a list called OverLap$O^i_p$[].
      For each feature $O^i_p$ in $F_i$
      *Determine bifurcation and continuation:*
         For all combinations of features in OverLap$O^i_p$[],
         Compute $O^i_p * -\bigcup OverLapO^i_p$.
            *If the lowest difference is below the tolerance,*
            *mark $O^i_p$ as bifurcating into the object or*
            *and remove them all from the search space.*
         Next $O^i_p$.
      *Else, Determine Amalgamation:*
      For each remaining feature in $O^{i+1}_q$ merge it into the
      octree forest, $F_i$ and test for amalgamation
         *This is the same as bifurcation with the inputs*
         Take the remaining $O^i_p$ in $t_i$ as dissipation;
         Take the remaining $O^{i+1}_q$ in $t_{i+1}$ as creation.
   }
```

Since the algorithm is running through many time steps, only one octree forest gets created and t_i is carried over from the previous iteration. Note that amalgamation is the "opposite" of bifurcation and can be detected using the same process as amalgamation, but in reverse. Using this heuristic, the number of corresponding test can be significantly reduced since only overlapping regions are tested. (However, handling topological changes. Some simplified representation approaches have also been proposed to extract properties from features [13, 14, 15] to ease the task of feature tracking.

In our previous work [3], features were tracked using their centroids and second order moments [16]. These parameters work well in describing most of the shapes of objects encountered in many turbulent simulations. However, this scheme was not always reliable, since this is a reduced model. As a result, errors can occur with concave features and small regions [3].

In this paper, many of these problems have been corrected by taking a full volumetric approach and performing *template matching.* Furthermore, all events are classified, and all parameters are computed and plotted (e.g. mass change over time, see Section 6). Finally, the information gathered, because it is global and pertains to each object and event, can be used to improve the rendering and visualization.

3 Feature Extraction

An important part of feature tracking is defining what features need to be tracked. Every domain has a different definition for *features, regions of interest, or objects.* General methods to define features include selective visualization [17], segmentation [18], volume intervals [19]. (Specific features include vortex cores [20, 15, 21], and critical points for vector fields [22], etc.) In many of these definitions, the features are defined with some sort of connectivity criteria which enable the method to partition the dataset into "important regions" and background. In this work, we assume features are regions 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. For the examples presented in Section 6, features are defined using a segmentation routine [23, 24, 25, 18] which divides the dataset into connected components above the threshold value 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 [26, 27]. Interior nodes of the octree contain the extremal value of that node's subtree. This property can be utilized as a time-saving strategy when performing the segmentation process for various thresholds. An advantage of the octree data structure is its simple and efficient set operations (union, intersection, and difference calculations, adjacency and membership testing, and transformations such as translation, rotation, and scaling). The efficiency comes from the octree's spatial indexing feature. The octree can also be used for memory optimization and object-oriented parallel computing. In addition to the actual voxels, global properties such as the centroid, mass, moments, volume, etc.. are computed and stored for each separate feature [16, 28].

Octree features can be visualized using volume rendering or fitting a surface around the boundary. An example is shown in Figure 5 and 6. The upper left image of Figure 5 shows a 128^3 turbulent dataset visualized using standard isosurfaces. In Figure 6, the same dataset is shown. However, segmentation was first performed so that each connected region has its own identifiable surface. The segmentation was based upon a threshold value, so an isosurface also results, however, now the surfaces can be colored by one of the computed properties of the feature such as mass, volume, local maxima, etc. (For other examples see [3, 18].)

The union of all the features in a particular dataset, t_i, where each feature, O_p^i, [1] is stored as an octree, is an *Octree Forest*, F^i and can be represented by

$$F^i = \bigcup_{p \in N} O_p^i$$

where N is the number of distinct extracted features from dataset t_i. An octree forest can be computed using a union operation [26].

4 Feature Tracking

Once we have defined features, we can characterize the evolutionary events present in time-varying scientific simulations as *Continuation*, *Creation*, *Dissipation*, *Bifurcation* and *Amalgamation* [3]. *Continuation* is where one feature continues from a dataset at time t_i to the next dataset at time t_{i+1}. Rotation or translation of the feature may occur and its size may remain the same, intensify (become larger - grow), or weaken (become smaller and begin to dissipate); *Creation* is where a new feature appears (i.e. cannot be matched to a feature in the previous dataset); *Dissipation* is when a feature weakens and becomes part of the background; *Bifurcation* is when a feature separates in two or more features in the next time step; and *Amalgamation* is where two or more features merge from one time step to the next.

The next step is to define how to classify each feature into the characterization above. This is known as the *correspondence problem*, where one has to match a feature in one dataset to one or more features in the next. A brute-force approach to the correspondence problem is to perform a matching test on features extracted from one dataset with all of the features extracted from the subsequent dataset, i.e. for each feature from dataset t_i, test it against all features from dataset t_{i+1} and all combinations of features from dataset t_{i+1} (for amalgamation/bifurcation) and choose the best match. The number of tests is exponential. Each domain may have different criteria for matching, and these are generally based upon volume, shape, distribution of values within the regions, and neighborhood.

In this work, we take a general definition for both the correspondence problem and the matching test, i.e. one that is derived from viewing animations of 3D datasets. For our definitions, we make the basic assumption that we have a sufficient sampling to guarantee that matching features overlap in 3D space. We can therefore state the following observations:

Observation 1 *If feature O_A^{i+1} corresponds to O_B^i, then O_A^{i+1} overlaps O_B^i.*

Observation 2 *If an feature in t_i splits into a group of N features ($N > 1$) in t_{i+1}, then $O_{j \in N}^{i+1}$ overlaps with O_A^i.*

Observation 3 *If a group of N features ($N > 1$) in t_i merge into an feature in t_{i+1}, then $O_{j \in N}^i$ overlap with O_A^{i+1}.*

These cases are illustrated in Figure 1.

Since a feature may overlap with many features in a subsequent dataset, to choose which one is the best matching

[1]Notation:O_X^i refers to a particular object, labeled X, extracted from dataset, t_i. For a time-dependent simulation the datasets are referred to with their time stamp, so dataset $t_i : t = i$.

Volume Tracking

D. Silver and X. Wang

Dept. of Electrical and Computer Engineering and CAIP, Rutgers University

ABSTRACT

3D time-varying datasets are difficult to visualize and analyze because of the immense amount of data involved. This is especially true when the datasets are turbulent with many evolving amorphous regions, as it is difficult to observe patterns and follow regions of interest. In this paper, we present our volume based feature tracking algorithm and discuss how it can be used to help visualize and analyze large time-varying datasets. We also address efficiency issues in dealing with massive time-varying datasets.

Keywords: Scientific Visualization, Multi-dimensional Visualization, Feature Tracking, Computer Vision, CFD

1 Introduction

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. Studying the evolution and interaction of coherent amorphous objects is an essential part of understanding time-varying data sets. It is quite common for scientists in various disciplines to describe the evolution of objects over time. For example, scientists track a storm's progress for weather prediction, the change in the ozone "hole" for knowledge about the greenhouse effect, and the movement of air over an aircraft or automobile for better aerodynamic design. Coherent objects are easily recognizable when the data sets are viewed, because they are localized in space and persist over finite intervals of time. The overwhelming size of time-dependent datasets resulting from CFD simulation complicates the visualization task. The total size of time-dependent data sets in 3D CFD domain ranges from five to sixteen Gigabytes, and may involve thousands of time steps [1].

A feature based approach can greatly ease the process of investigating large datasets efficiently. Each domain has it own set of important features. Since they do appear in consecutive time steps, tracking them can enhance the visualization and provide a valuable analytical tool to study the behavior of these features. Furthermore, tracking can help automate searching for events and interactions. For example, part of the motivation for this research is to identify the *reconnection* event in turbulent CFD datasets [2]. Searching through hundreds (possibly thousands) of large 3D scalar and vector datasets where each data set can contain many amorphous structures (sometimes up to 1000 or more) is impractical (see Figure 5 for sample datasets from a turbulent simulation, in this example the structures are isosurfaces). While there are many criteria defining the *reconnection* event, tracking can help identify time-intervals for further investigation and reduce the amount of data to analyze.

Automatically tracking features is difficult because the features are continually evolving and interacting. Initial efforts in feature tracking have been presented [3] where the tracking was performed using approximations to features. In this paper, we present a different method for tracking using a volume based approach which takes a template from one datasets and correlates it to features in the subsequent dataset. We show how it is more robust and how it can be used for complex simulations. In Section 2, we give a short overview of related work. In Section 3, 4 and 5 we present our new algorithm for feature tracking and discuss efficiency issues. Three different examples are discussed in Section 6, including a 128^3 with over 100 time steps. The examples are taken from studies in turbulence (CFD). These are good candidates for tracking because the large number of amorphous evolving regions make it difficult to observe what is happening in the dataset. Other examples of turbulent datasets can be seen in [4, 5, 6].

2 Related Work

The computer vision community has been very concerned with 2D tracking (for introductions see [7, 8]), and many of the techniques developed in Computer Vision can be used in this domain to extract, identify and track features (these are discussed in [3, 9]). The main difference arises in the domain-specific knowledge needed to accurately analyze the data and the type of data available. The goal of visualization is to help understand and analyze the underlying physics or mathematical model. Because of this, the criteria for tracking features in a scientific domain are different than for most computer vision applications. In a scientific simulation, evolving features may split, merge or disappear. All the properties of these features must be computed, such as mass, volume, moments and their change over time. Furthermore, all the features must be tracked to identify interaction events and automate searching. Finally, the tracking information and computed properties should be used to provide better renderings of the datasets.

An important issue is feature representation. The representation should be general enough to handle a wide variety of shapes of features, yet simple enough to be usable for tracking and quantification. Many representation schemes have been proposed for object recognition and motion estimation. A polyhedral representation is one of the most general and intuitive methods in 3D object modeling, and they can be derived using isosurface contours. They must be connected to be tracked, but it is still difficult. Physically-based models have been used for tracking in computer vision and for 3D representations [10, 11, 12]. However, they generally do not classify all events and may have difficulty

[1]Dept. of Electrical and Computer Engineering and CAIP, Rutgers University, P.O. Box 909, Piscataway, NJ 08855-0909.
Email: silver, xswang@vizlab.rutgers.edu
Web: http://www.caip.rutgers.edu/vizlab.html