Intelligent Feature Extraction and Tracking for Visualizing Large-Scale 4D Flow Simulations

Fan-Yin Tzeng* Kwan-Liu Ma*

Department of Computer Science
University of California at Davis

ABSTRACT

Terascale simulations produce data that is vast in spatial, temporal, and variable domains, creating a formidable challenge for subsequent analysis. Feature extraction as a data reduction method offers a viable solution to this large data problem. This paper presents a new approach to the problem of extracting and visualizing 4D features within large volume data. Conventional methods requires either an analytical description of the feature of interest or tedious manual intervention throughout the feature extraction and tracking process. We show that it is possible for a visualization system to "learn" to extract and track features in complex 4D flow field according to their "visual" properties, location, shape, and size. The basic approach is to employ machine learning in the process of visualization. Such an intelligent system approach is powerful because it allows us to extract and track an feature of interest in a high-dimensional space without explicitly specifying the relations between those dimensions, resulting in a greatly simplified and intuitive visualization interface.

Keywords

feature extraction, feature tracking, flow visualization, hardware acceleration, machine learning, artificial neural networks, time-varying volume data, user interface

1. INTRODUCTION

With increasingly accessible supercomputing facilities, scientists are given unprecedented power to model natural phenomena at greater accuracy, resulting in an explosive growth of data that they must study. In particular, modeling complex flow is an important topic in many scientific and engineering applications. Examples include supernova simulations, climate modeling, aircraft design, combustion simulations, and human organ modeling. All these physical phenomena and chemical processes are time-varying in nature, and the simulations can routinely generate terabytes of data, creating a formidable challenge for subsequent data analysis tasks. Several parallel visualization solutions are available for this large data problem [1, 12, 13, 14, 28, 29]

While we can directly visualize a flow field, it is often more economical and helpful to extract and track features of interest and focus on the corresponding smaller subset of the data, especially when the data size is large. Feature extraction generally refers to the separation of structures/materials based on their data properties, whereas feature tracking involves following a set of extracted features over time.

Conventional methods requires either an analytical description of the feature of interest or tedious manual intervention throughout the feature extraction and tracking process. In this paper, we present new feature extraction and tracking methods making use of machine learning with a focus on visualizing time-varying volume data. We show that it is possible for a visualization system to *learn* to extract and track features in complex 4D flow field according to their *visual* properties, location, shape, and size. Our intelligent system methods are integrated with direct volume rendering, which mainly uses transfer functions to make features of interest visible. We show how to conduct feature extraction and tracking in both transfer function space and data space.

We introduce an intelligent adaptive transfer function that can effectively adapt to the temporal changes in the data. With this method, the user specifies features/regions of interest with a traditional one-dimensional transfer function for selected time steps, and then an adaptive transfer function for the whole time sequence is automatically generated using the machine learning system. In this case, the adaptive transfer function takes into account the scalar field and the data distribution variations over time. We also introduce a volume feature tracking technique based on such intelligent transfer functions. The user-selected features are highlighted during rendering as the tracking proceeds, and a variety of highlighting criteria may be used to enhance the features of interest.

Intelligent feature extraction and tracking can also be done in the data space. This approach is especially powerful than previous methods because the machine learning engine can take high-dimensional data directly but the scientists do not need to specify explicitly the relationship between these different dimensions. The system will "learn" about scientists' knowledge about the features through scientists' interaction with the system. Essentially, the feature extraction function is thus revised through an iterative training process following the temporally changing properties of the tracked features.

*IDAV & Department of Computer Science, University of California at Davis, One Shields Avenue, Davis, CA, 95616. {tzengf,ma}@cs.ucdavis.edu

We have designed an intuitive, multi-view user interface to facilitate interactive, intelligent volume feature extraction and tracking in spatial, temporal domains of the data. Using a painting metaphor, the scientist specifies a feature of interest by marking directly on the 2D or 3D images of the data through this interface. The effectiveness of our techniques is demonstrated using time-varying flow data sets with a wide range of characteristics.

The contribution of this paper is the introduction of a fundamentally different approach to the pressing large data visualization problem, suggesting scientists to rethink about their existing data analysis procedure and scientific discovery process. We feel the intelligent system approach will play a major role in the next-generation data analysis and visualization systems.

2. RELATED WORK

Most of the research in time-varying volume visualization has focused on rendering performance or storage issues. In contrast, few methods of specifying transfer functions or classifications for time-varying data have been invented. Bajaj et al. [2] introduce the Contour Spectrum method, which can capture isosurfaces of interest, chosen from different time steps. Jankun-Kelly and Ma [9] present automated methods for generating a minimum set of transfer functions to visualize time-varying volume data. They categorize time-varying data behaviors into regular, periodic, and random/hot spot. Different approaches are used for data sets with different properties. In contrast, we introduce a method that studies the characteristics of user specified data, and use machine learning to create an adaptive transfer function that takes into account shifts in a data value position in the histogram distribution.

Various techniques have been developed for feature extraction and tracking of time-varying data. Region growing is one of the basic segmentation methods, and has been widely used in 3D volume extraction. In [22, 18, 24], the features of interest are extracted by flood-fill type algorithms so that the features are defined as connected nodes that satisfy a certain criteria. In our work, the criteria for region growing are in the form of an arbitrary-dimensional classification function rather than a particular threshold value. This allows more flexible extraction.

Silver and Wang [22, 23] extract the features, and organize them into a octree structure to reduce the amount of data during tracking. Chen et al. [3] introduce the "feature tree" data structure, which allows the tracking to work between refinement levels, time steps, and processors. Reinders et al. [20] calculate the basic attributes for the features of interest which are used to track features with a prediction and verification scheme. Ji et al. [10] present a volume-tracking technique using higher-dimensional isosurfacing. They define features as connected isosurfaces or interval volume components, and find the overlapping components in higher-dimension space at the next time step. Our work defines the features of interest using high-dimensional region growing to capture the connected features in the temporal domain. Several of other feature extraction and tracking techniques are described in an excellent survey [19].

Work employing machine learning to scientific visualization problem has been sparse. Most of the previous efforts focused on data segmentation and classification problems for 2D medical images [6, 7, 16] or medical volume data [25]. In this paper, we discuss how to employ machine learning for 4D flow visualization problem.

3. MACHINE LEARNING

Machine learning is a branch of artificial intelligence that involves the acquisition of knowledge through experiences. After a training process, a machine learning engine is able to apply the knowledge to solve similar problems. In our work, we employ an artificial neural network, which is one of the more traditional machine learning techniques, to perform learning of scientists' knowledge about features of interest in their simulations.

There are more than 50 kinds of neural network in use today. The neural network topology we have used is a three-layer perceptron, and it is trained with the Feed-Forward Back-Propagation Network (BPN) algorithm. The back-propagation algorithm, which is designed for supervised training, was introduced by Paul Werbos [27], and has been widely used since the work of Rumelhart and McClelland [21].

The training sets, which include a small number of corresponding inputs and outputs, in our work, are provided by the user through an interactive visualization interface. The output value usually indicates the level of certainty about a feature of interest. After training, the neural network can be applied to similar data sets by feeding the new data to the trained neural network.

We utilize neural networks as the machine learning approach for adaptive learning primarily because of its simplicity and generality. There are other supervised machine learning techniques such as Support Vector Machines [8], Bayesian networks [4], and Hidden Markov Models [15] usable for our purpose. In the context of intelligent visualization, the cost and performance tradeoffs for each of these methods remain to be evaluated.

4. INTELLIGENT FEATURE EXTRACTION

Feature extraction is typically done by explicitly separating the feature of interest from the raw data, and creating either a volumetric or geometric representation of the extracted feature. One benefit is therefore the potential reduction of data to a more manageable size. In the context of our study for volume data visualization, the results of feature extraction are directly presented as visualizations, though the trained neural network can direct the construction of a compact representation of the features as needed. In our study we focus on identification of features rather than their representations.

Feature extraction can be done by looking at the data values, topology, size, shape, location, and context of the feature as well as the relationship between two or more variables. Some of the properties cannot be easily defined and specified, and sometimes unknown. In this section, we show how learning-based feature extraction can be done in both the transfer function space and the data space.

4.1 Transfer Functions and Feature Extraction

In volume visualization, transfer function specification plays an important role in allowing scientists to visualize, explore, and discover. By manipulating the mapping between the original input data and the output color and opacity, they can enhance the visibility of the specific spatial structures they would like to see, and reduce the opacity of material that could obscure regions of interest. When visualizing time-invariant data, one transfer function is usually sufficient to capture a feature of interest since the data range is fixed. For time-varying data sets, however, the data values constituting a feature likely change over time, making it difficult to capture and track with a single transfer function. Sometimes, the range of the data values can vary so dramatically that we can easily lose track of features in a time sequence. The user could find that the features fade and disappear over time simply because of the transfer function, even though the features in fact still exist in the domain.

Furthermore, a data set might contain several features with very similar data properties, in which case when a single transfer function alone is not capable of differentiating these features, tracking these features in the temporal space would make it possible to isolate some of them.

4.2 Extraction in the Transfer Function Space

Conventional methods for transfer function design can be roughly divided into data-driven and image-driven techniques [17, 11]. Data-driven transfer function design consists of defining a transfer function by using the original volume data or derived data properties. Image-driven transfer function design is a much more trial-and-error process, where the user specifies a transfer function, sees the rendered result, and further modifies the transfer function until the desired visualization is achieved. Image-driven transfer function specification has become increasingly popular with advancements in hardware-accelerated volume rendering techniques which have made interactive volume visualization increasingly powerful and affordable. The user-centric interactive nature of image-driven transfer function specification does not make it entirely well suited for time-varying transfer function specification. Although the user can easily specify a transfer function for one or a couple of time steps, it could quickly become a difficult and tedious task to adapt the transfer function over time for large time sequences. Thus, in our work we make use of a hybrid of the two techniques.

Figure 1 illustrates the process of using our approach. The user first selects a couple of key frames and assigns a 1D transfer function for each key frame to define the features of interest. These 1D transfer functions are then sent to a machine learning engine for training. Training is an iterative process and the trained machine learning engine is able to generate an adaptive transfer function by using these image driven transfer functions, as well as the data-driven properties of the data set. During rendering, the adaptive transfer function is used to assign opacity to each voxel. The user can visualize the rendered results using the adaptive transfer function and add new key frames when needed.

4.2.1 Cumulative Histogram

A histogram shows the distribution of values in a data set, and is a powerful aid in data driven transfer function specification. For a given data set, the value of a voxel's cumulative histogram is the number of voxels in the data set that have scalar value less than or equal to that voxel. If the primary temporal changes in a volume are changes in position or global shifts in intensity, the cumulative histogram values of a feature of interest would remain constant over time. One could therefore conceive of specifying a transfer function with an input consisting of the cumulative histogram information, rather than the scalar value itself. Such a transfer function could be better suited to adapt to changes in the values of features over time. However, such a transfer function would not properly classify those features that have constant value, but vary in *size*. Such features, could dramatically shift with respect to the cumulative histogram, resulting in shifts in transfer function classification.

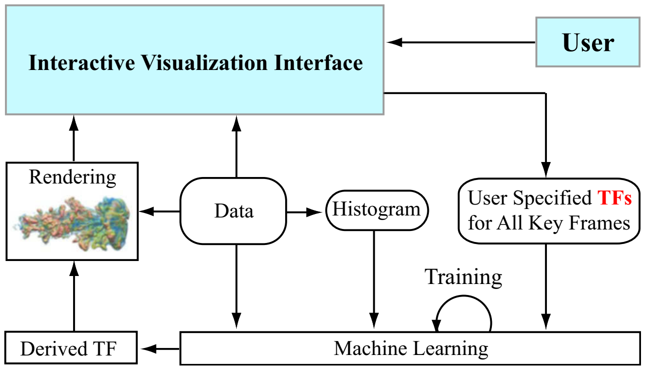
Interactive Visualization Interface
User
Rendering
Data
Histogram
User Specified TFs for All Key Frames
Training
Derived TF
Machine Learning

Figure 1: The process of transfer function based approach. The user first assigns 1D transfer functions for a couple of key frames, and the machine learning engine automatically generates an adaptive transfer function for capturing features of interest over time.

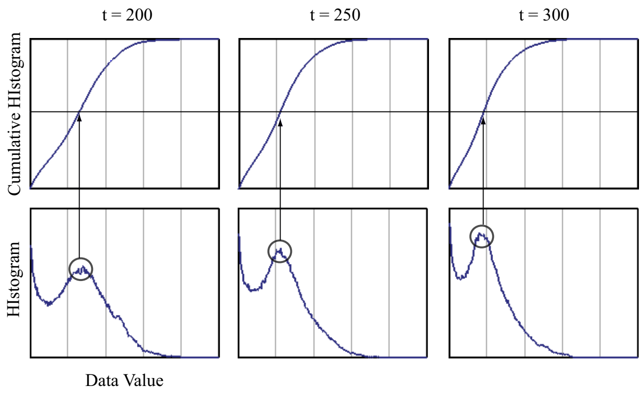

Figure 2: The histograms and cumulative histograms of the argon bubble data set. A feature's data value and histogram can change over time, however, the cumulative histogram value remains similar. This make the cumulative histogram useful for specifying transfer functions.

A traditional scalar value transfer function is much better suited for features that vary in this manner.

Figure 2 shows the histograms and cumulative histograms for three time steps of the argon bubble data set. In each histogram, the peak circled in black indicates the feature of interest. Notice that in the bottom row the data value and height of this peak changes over time. However, for the cumulative histograms, the feature's data value maps to similar cumulative histogram values in all three time steps.

We therefore introduce the use of data value, cumulative histogram, and time in an adaptive transfer function. Such a transfer function is able to adapt to shifts in feature value over time by taking into account the cumulative histogram value, while it can also remain invariant with respect to cumulative histogram value by relying on scalar value. Specifying a transfer function that takes into account the relationship between these three attributes, can be extremely difficult for a user since the temporal relationship between these three attributes can be complex and unintuitive.

4.2.2 *Transfer Function Generation*

The neural network used for generating the adaptive transfer function is first trained with a set of inputs and desired outputs. The training inputs can be randomly selected from the key frame volumes. The random selection provides data with similar distribution over the whole volume. That is, the training inputs would have similar probability distribution function as the histogram. However, when the volume size is large or many time steps are used, it can be time consuming to load the volumes for training since not all the data can fit in core. Another problem is that when the feature of interest is small, more likely data values of non-interested features are selected. This not only wastes the time for training unimportant data, but might lead to poor results due to the lack of generalized training samples.

In our implementation, the training data is collected from the transfer functions user specified. For each data value in a key frame transfer function, a vector $< data, histogram(data), t >$ is created where $data$ is a data value, $histogram(data)$ is the corresponding cumulative histogram value, and t represents the time step. The corresponding desired output is the opacity specified by the user from the key frames' 1D transfer function. In this case, all the key frame transfer function can remain in core for fast access, and each entry in the IATF has the same amount of training.

The training is performed iteratively in the system's idle loop since it is typically the most time consuming step for a machine learning method. That is, the user can visualize the current rendered result with the adaptive transfer function, and continue to interact with the system by specifying new key frames as training progresses. During rendering, a new 1D transfer function is created by the current neural network for each frame. The value of each element in the transfer function is obtained by passing that element's index (a scalar value), cumulative histogram value and time to the trained neural network.

4.2.3 *Case Studies*

Figure 3 and Figure 4 are examples created with the argon bubble data set. The same color map is applied to all the time steps in an example. This data set shows the result of the simulation of a shockwave applied to a bubble of argon gas surrounded by air creating a swirling torus-shaped "smoke ring" along with smaller turbulence structures.

If IATF were not used, the most straightforward method to generate a transfer function for a time step between two key frames is to linearly interpolate the transfer functions of the key frames. However, in most cases, linear interpolation is not sufficient to capture the dynamic behavior of the time-varying features. Figure 3 is a rendered example showing the difference between linear interpolation and our adaptive learning technique. On the left, the transfer function and rendered results of the first key frame are shown where the transfer function is set to capture the ring structure of the argon bubble. The second key frame is shown on the right. For an intermediate time step, two methods are applied to obtain the transfer function for extracting the same feature of interest. The lower image is the result of linearly interpolating the transfer functions of the key frames. Instead of preserving the ring structure at the intermediate time step, the linear interpolated transfer function combines two separated features from the two key frame transfer functions with reduced opacity. However, with our method, the cumulative histogram can capture the global data value changes and thus show the single ring structure for the middle time step.

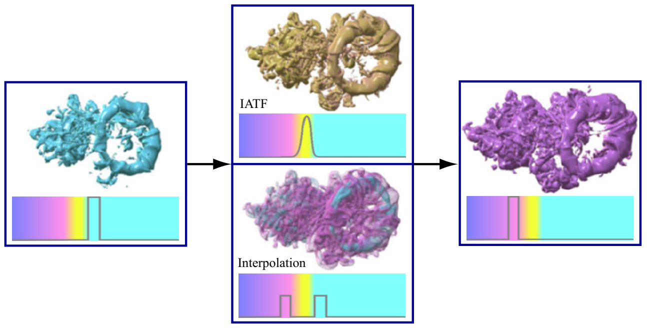

Figure 3: The left and right most images are the 1D transfer functions and rendered results of two key frames which attempt to capture the ring structure within a small range of data value in the argon bubble data set. For an intermediate time step, the rendered results using adaptive transfer function and linear interpolation are shown in the middle. The adaptive transfer function is able to capture the ring structure better than the the interpolation method can, as shown.

The top row of Figure 4 shows the key frame transfer functions specified by the user. The gray curves are the mapping functions from data value (x axis) to opacity (y axis). The color map is shown as the background color, which stays constant over time. The three transfer functions in the first row are defined to preserve the ring structure for the three key frames (time step 195, 225, 255). Since the data range changes significantly over time, a transfer function set to visualize an earlier time step is unsuitable for the later time steps and loses the features of interest as shown in the lower left images. With our system, the user can define new transfer functions based on later time steps to capture the smoke ring and turbulence. The images bounded in blue presents the rendered results using the intelligent adaptive transfer function. The ring structure is completely preserved over the time period between the three key frames, while the change in color indicates the change in data value, and the transfer function images on the right illustrate the changes of data range of interest over time.

Figure 5 shows the results of using IATF with a DNS (Direct Numerical Simulation) turbulent combustion data set. DNS is a mature and productive research tool in combustion science. Because of the high demand for computational power, current stat-of-the-art (terascale) DNS must run on parallel supercomputers. The simulation models a temporally evolving turbulent reacting plane jet. There is fuel, which flows between two counter-flowing air streams, resulting in turbulence which distorts the reacting mixing layer of fuel and air. Each time step is a $480 \times 720 \times 120$ volume with multiple variables.

The vorticity magnitude is used in Figure 5, which cannot be captured with a single transfer function for all the time steps. For example, a transfer function is specified to capture the turbulence in the upper left image (time step 8). However, the same transfer function fails to capture most of the features in time step 128 due to the data range change. Two more transfer functions are then set for key frame 64 and 128, where those transfer functions do not perform well in earlier time steps. With IATF, the machine learning system is able to modify the opacity mapping function so that the feature of interest can always be extracted from the volume as shown in the fourth row in Figure 5.

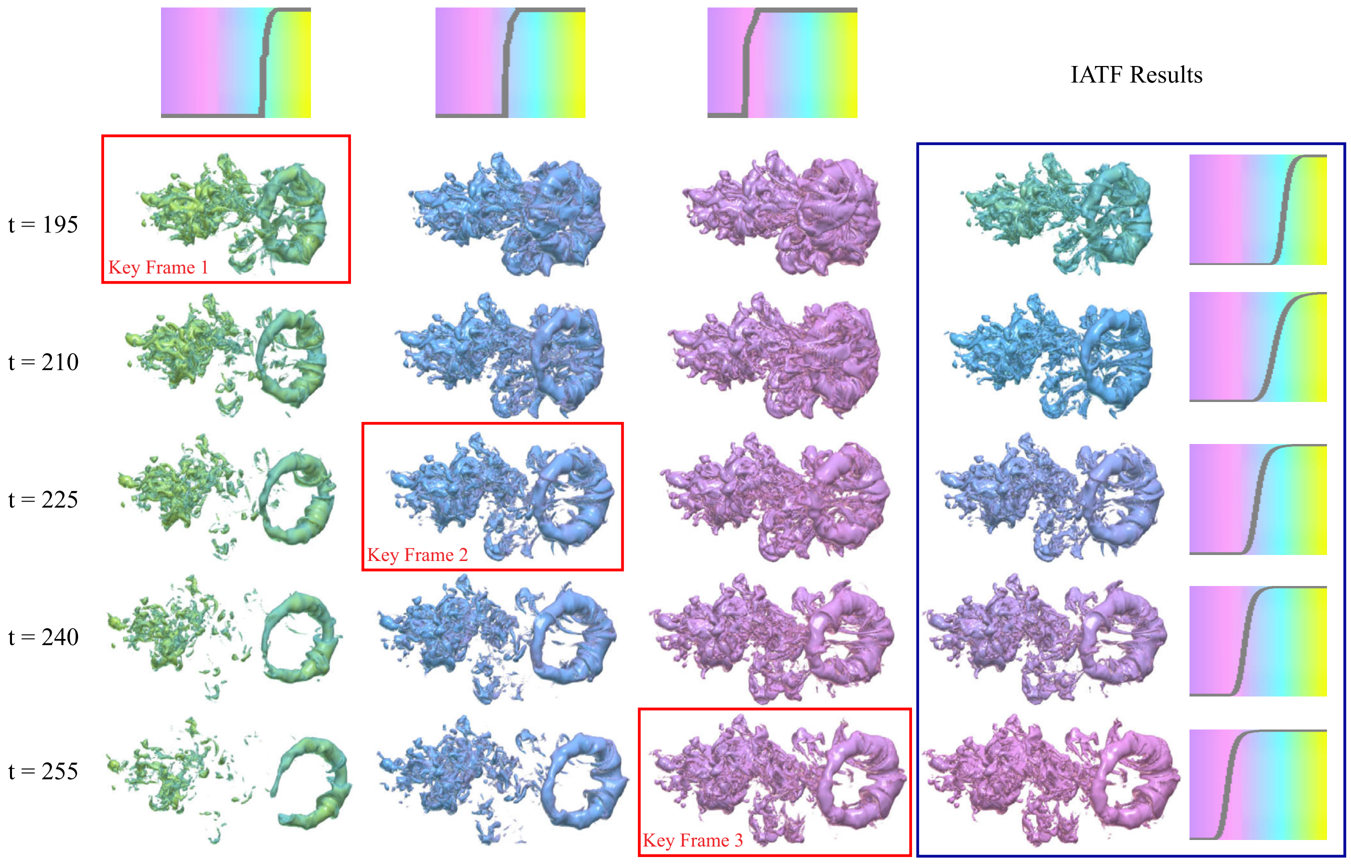

Figure 4: The top row shows the 1D transfer functions defined by the user to capture the ring structure in key frames. Each transfer function is applied to all the time steps shown below the transfer function. The images bounded in blue are the rendered results using IATF. The IATF adapts over time to follow the ring structure.

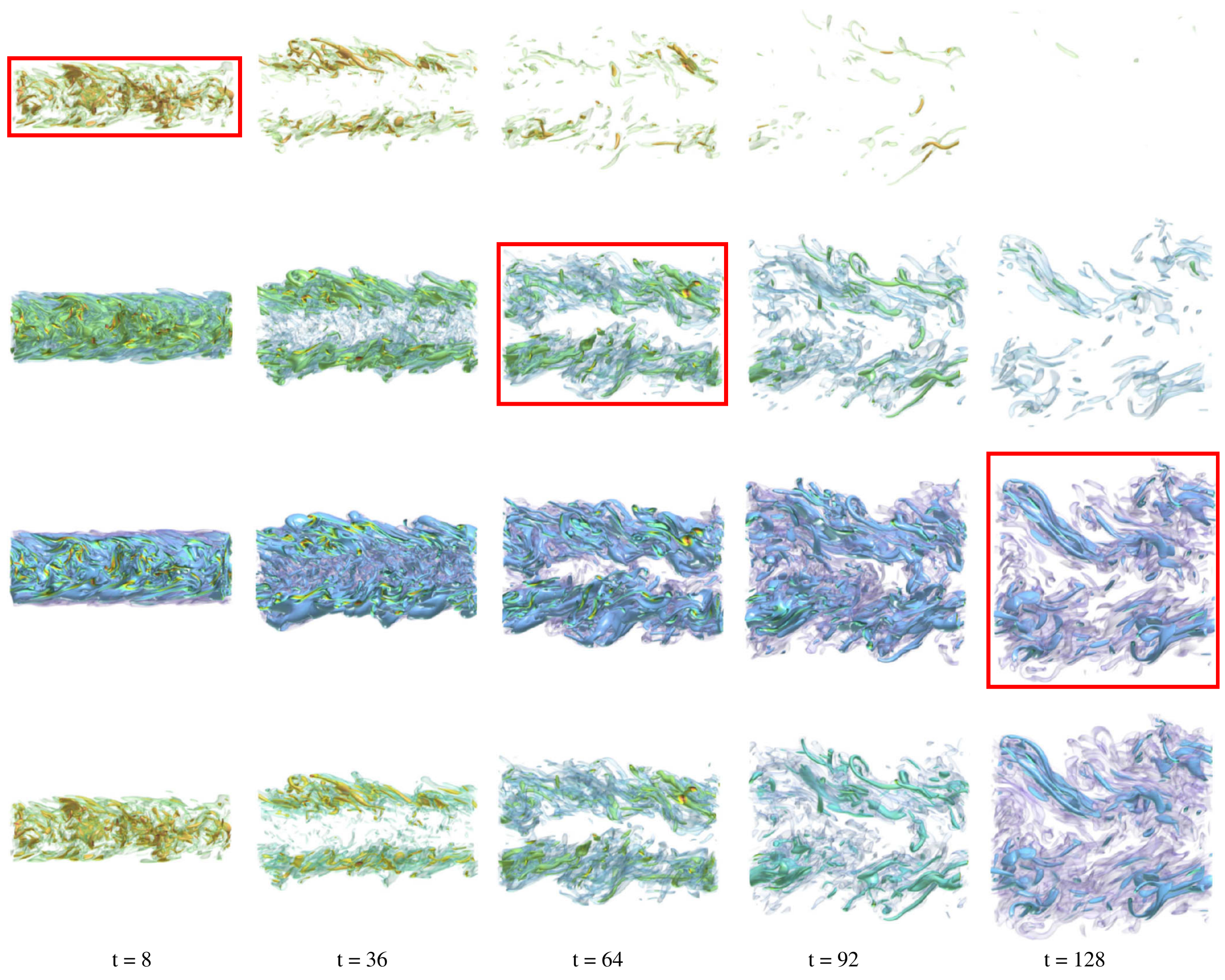

Figure 5: A DNS turbulent reacting plane jet data set using IATF. The key frames are bounded in red, and the transfer function used for each key frame is also applied to different time steps shown in the same row. The fourth row shows the rendered results using IATF where the vortex can be well extracted over the whole time sequence.

As the data set grows in either size or number of time steps, it becomes impractical to load the entire data onto a single computer for feature extraction specification and rendering. Since our IATF generation is only dependent on the key frame volumes, the user can select a few time steps for specification, create an IATF that is suitable for all the time steps, and send the IATF to parallel systems or remote machines for rendering.

4.3 Extraction in the Data Space

Flow feature extraction is generally done by using the set of scalar and vector data defining the flow field. In Section 4.2, we have demonstrated how the learning approach makes use of transfer functions to succinctly extract time-varying features. In some applications, the size of the feature can play a crucial role in the enhancement and extraction of features. However, it is not a trivial task for the scientist to specify "size", and there is generally no systematic and robust way to measure the size of a 3D feature. A more viable approach is to let the scientist see 4D flow field from different views and at different levels of details, and interactively select the features with the desired sizes. With a learning capability, the scientist only needs to select a few representative ones and then let the learning-based system to capture the rest.

Since it is not practical to explicitly measure the size of each selected feature, an alternative is to use the data values associated with not only the selected feature but also the neighborhood of the feature. Again, this becomes a high-dimensional classification problem. Note that a selected feature likely contains multiple voxels; however, the training is done in per-voxel basis. In our implementation, we use a shell rather than the whole volumetric neighborhood of the feature to cut down the cost. That is, we do not use all the voxel values in the neighborhood; only those voxels a fixed distance away from the feature of interest are used, and this distance is data dependent and derived according to the characteristics of the selected features so far. The time step information is also used in training so that the size of the tracked feature can be different over time.

To visualize the whole data set, similarly, the scientist goes through a small number of time steps of the data to specify the features of interest. The trained system is then applied to the complete data set which might consist of several hundreds or even thousands of time steps. To adapt the feature extraction over time, the trained network in fact takes as input a feature vector which consists of data values of the feature, neighborhood information, and the time step number. Figure 6 illustrates the process of performing intelligent feature extraction in the data space.

4.3.1 *A Case Study*

An example of the need of extraction by size is the visualization of astrophysics simulations performed by scientists at the Princeton Plasma Physic Laboratory. One such simulation is for studying the reionization of the universe. Scientists want to observe the larger structures but were distracted by the large number of surrounding tiny features in the data. These tiny features are considered as "noise" in the data.

Figure 7 contrasts the results of using different methods to remove those small features. In the first image, a 1D transfer function is applied to the data set. We first specify a different transfer function to remove the small features. Because many of the small features have data values similar to the large structure, the 1D transfer function which only takes into account the data value for mapping cannot

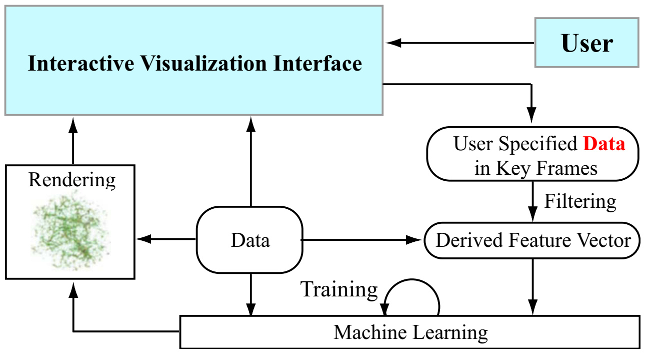

Figure 6: The process of performing intelligent feature extraction in the data space. A feature vector is derived and used as the input to the machine learning engine.

separate the small features from the large-scale features. The third image is the result of using a conventional filtering method to repeatedly smooth the data. This could remove those noise but at the same time the fine details on the large features would be taken away too. The fourth image shows the result of using our learning-based method. Note that in the right image how fine details of large structures are preserved while many of the tiny features are suppressed. The scientist can interactively validate and refine the results as immediate visual feedback is provided.

Figure 8 shows the results of using time step 130 and 310 to train the neural network, and then applied the trained network to other time steps of the data. As shown in the middle image for time steps 250, the small features are invisible and large features are retained over time. This level of detail retained can be controlled by the user easily through an intuitive visualization interface.

5. FEATURE TRACKING

Often a user would like to follow the temporal progression of a select set of features. Feature tracking is the process of capturing all the events for one or more features. With traditional tracking techniques, the features are typically tracked based on a constant threshold. By applying the IATF method from Section 4.2 as an adaptive criterion, our approach allows more flexible and powerful tracking, and provides the user the control to track features with data values that change over time.

In our method, we first make the assumption that there is sufficient temporal samplings for the matching features to overlap in 3D space for consecutive time steps for feature tracking. Because of this overlap, tracking can be achieved by using 4D region growing where the forth dimension is time, and the adaptive transfer function is applied to feature tracking. The user specifies the tracked data value range using a transfer function interface, and the adaptive transfer function is created with the previous method and is used as the region growing criteria. The region growing result is then saved in a 3D volume texture for rendering.

There are methods [22, 20, 10, 19] developed for feature tracking which can reduce the cost in time or space after the feature extraction is done. Our method can employ similar approaches when the adaptive criteria are known in advance with additional cost of calculating IATF every time step. This involves obtaining results from the trained neural network for all the entries in the 1D transfer function, and can be done in sub-seconds. However, this is less

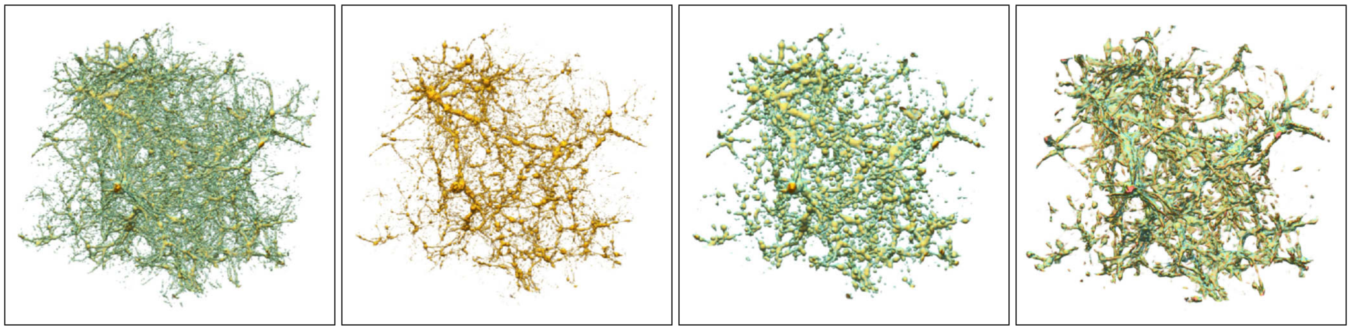

Figure 7: Time step 310. From left to right, the first image is the direct volume rendering using a 1D transfer function, the next two images are the results of changing the transfer function and bluring the volume to remove tiny features, and the right image shows the result of using our learning-based method which presents the large-scale structures more cleanly.

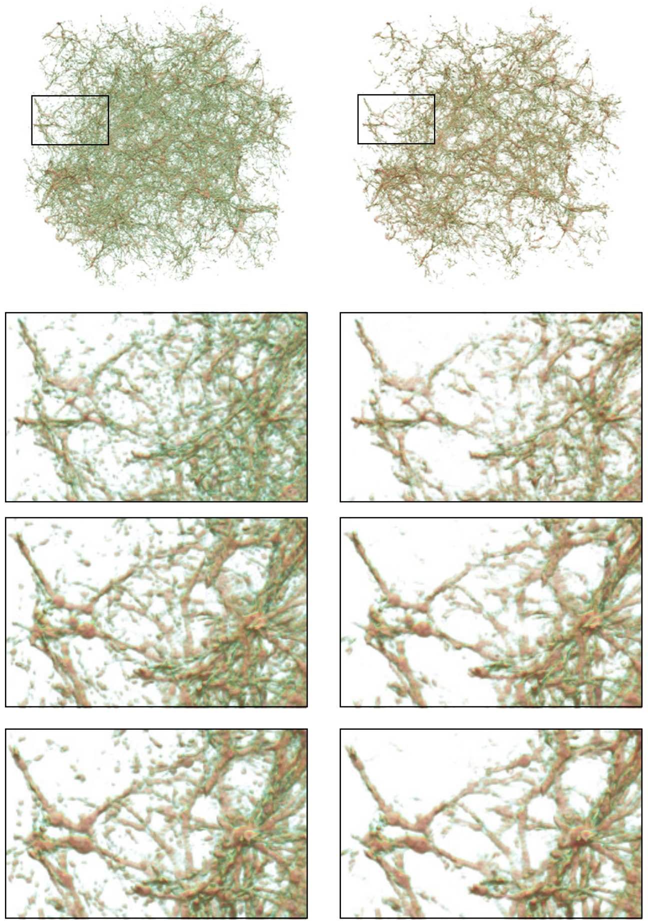

Figure 8: The top images show the entire data set with the selected region circled. The rendered results of this region are zoomed and shown from the second row. From up to down: Time steps 130, 250, and 310. Left: using a 1D transfer function. Right: Features in Time step 310 were learned using our method, and the two upper images were generated by the trained neural network. Note how the large features are retained over time while small ones are suppressed.

desirable in our method, which focuses on providing the freedom of changing tracking criteria over time.

5.1 Case Study

Figure 9 presents the feature extraction and tracking results using the turbulent vortex data set. From left to right, top to bottom, there are six rendering results from time step 50 to time step 74. The tracked feature is rendered in red with the original volume for providing content. The results show that the tracked vortex moves and changes its shape through time and splits near the end.

Figure 10 shows the result of tracking a feature (highlighted in red) in the swirling flow data set where the feature's data values decrease over time. The top row shows the feature tracked with a conventional fixed criterion over which data values constitute a feature. As the data values of the feature decreases with time, it eventually falls below this fixed criterion and no longer tracked, as shown in the top right. With our method, the user is given the ability to specify such a dynamic feature tracking criterion. By decreasing the tracked value range for the last key-frame, an adaptive transfer function tracking criterion is created that can track the feature across all the time steps between the two key frames.

In many cases, a feature is comprised of those connected voxels with the same properties. When the properties of the connected set of voxels no longer meet a preselected criterion, we consider the corresponding feature ceases to exist. The users must therefore be given the freedom to specify tracking criteria that are suited to their needs, whether that be based on absolute data value, relative position in the volume's histogram, shape or size of the feature, or a combination of them. Our intelligent system offers an intuitive means of providing this control.

6. INTERACTIVE VISUALIZATION INTERFACE

Even though the feature extraction and tracking tasks are performed in a high-dimensional space, the learning capability abstracts it from the user. As a result, the user interface of the learning-based system can become extremely simple. The user interface, which is shown in Figure 11, allows direct specification on the data through painting on three axis-aligned slices in the lower row of the interface. The user only needs to specify a few sample data of different classes with brushes of different color. The selected data is then used to look up corresponding data properties and sent to the machine learning function for training.

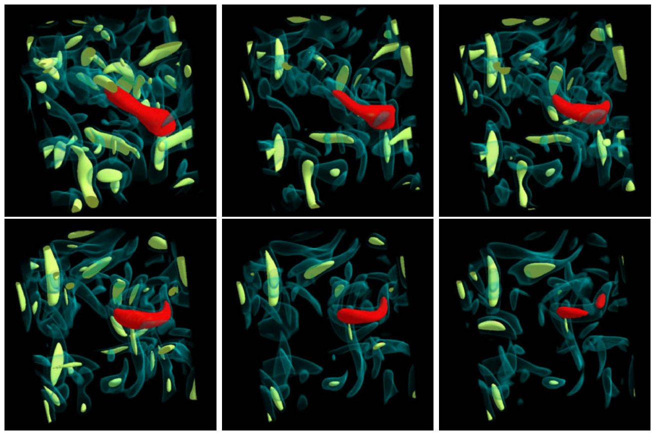

Figure 9: Results of feature tracking. Six time steps between time step 50 to time step 74 are shown. The tracked feature is rendered in red and the rendering performance is about 2 frames per second.

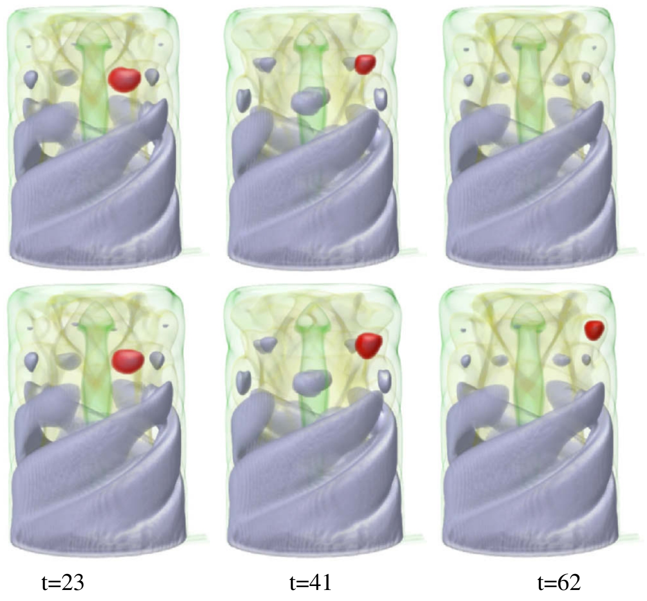

Figure 10: Results of our intelligent feature tracking technique using the swirling flow data set. Top: The rendered results of tracking a feature over time based on a specific data range. Bottom: The rendered results of tracking the same feature with the adaptive transfer function. The feature is still tracked even it's data range has changed.

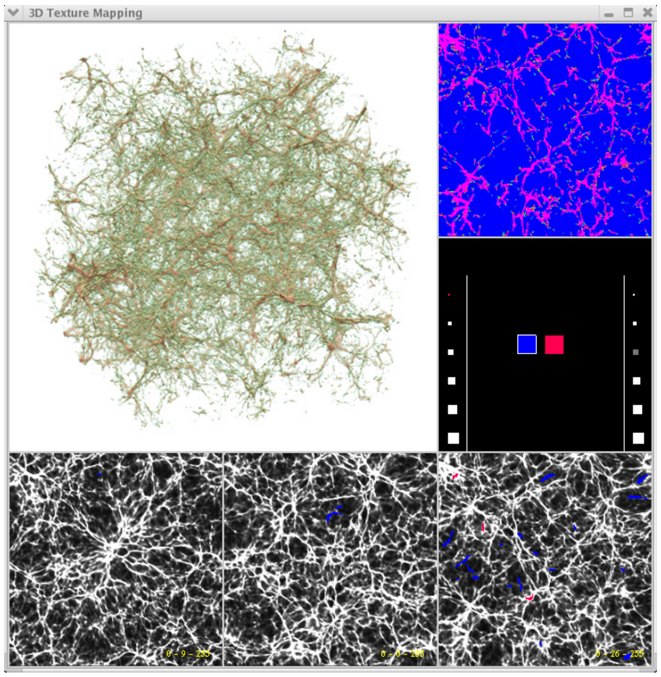

Figure 11: The visualization interface for the learning-based system.

During rendering, we provide real-time visual feedback to show intermediate results. That is, the user is able to interactively view the feature extraction results by applying the current neural network to classify individual slices, or the entire volume in real-time as shown in the upper right and upper left of the interface. The user can use this feedback to further revise the painting and add additional training data so that the system generates a more desirable result. Some features are easy to identify on a slice of the data set, but some features, especially small ones, can be difficult to find in a 2D space. Therefore, in addition to painting on the slices, the system also allows the user to select small features from the window of feature volume, and consider the selected regions as part of the unwanted feature.

The time needed for training the neural network and performing the feature extraction for the whole data set highly depends on the size of input vectors. The user can remove data properties in an input vector if they are considered unimportant based on the user's knowledge [26]. The size of the neural network changes according to how many different data properties are requested by the scientists. When the user considers less properties, the neural network becomes smaller and the input data for the previous network would be transferred to the new network. The user interface hides all these.

7. HARDWARE ACCELERATION

In our system implementation we have taken the advantage of the advanced features of contemporary consumer graphics hardware. Hardware accelerated rendering is performed using fragment programs and view aligned 3D textures [5].

We test our system on a PC with a 2.80GHz Pentium 4 CPU, 2GB main memory, and a GeForce 6800 GT graphics card with 256MB video memory. The performance for rendering in-core a

$256 \times 256 \times 256$ data set to a 512 by 512 window is 6 frames per second when the adaptive transfer function is recalculated every frame and the volume is rendered with shading.

For rendering the feature tracking results, when a voxel's value in the region growing texture is one, its color is set to red and its opacity is set to the opacity in the adaptive transfer function. Otherwise, the color and opacity looked up from the user specified 1D transfer function are shown. The tracked feature and the original volume are rendered using multiple passes. It takes about 4 frames per second for rendering with the hardware mentioned previously.

When performing extraction in the data space, the entire volume needs to be classified by the trained neural network. The classified result is stored as a 3D texture and used to assign opacity to each voxel. With our implementation, it takes 10 seconds to classify a $256 \times 256 \times 256$ data set, and can achieve 6 frames per second during rendering. However, the feature extraction task can be time consuming with the data size increases, and this can be improved by implementing the neural network classifier in a fragment program [25].

In flow visualization, color is typically used to communicate quantitative physical properties of scalar values such as density, velocity magnitude, or pressure. Shifting the assignment of colors could therefore give a misleading indication of variations in these properties. Therefore, our methods only apply to the opacity, when color is assigned by the original data value.

Even though it is possible to also implement the training step in hardware, the current software implementation provides an adequate performance to meet the needed interactivity. In fact, a more interesting and helpful capability is fast data decompression in hardware since one potential bottleneck for large data sets is the need to transmit data between the disk and the video memory. We will explore this option in the future.

8. CONCLUSION

We have applied machine learning to the extraction and tracking of time-varying fluid flow features in volume data. Our intelligent system is unique because feature extraction and tracking can be done in both the transfer function space and the high-dimensional data space through a multi-view, intuitive interface. In particular, that the system can take multivariate data as input opens a new dimension for scientific discovery.

In this paper, neural networks are used to generate all the examples. We have also used support vector machines [8] and obtained promising results for a different application [25]. We plan to experiment with other machine learning methods such as Bayesian network and study their performance. We are presently seeking a systematic way for the scientists to validate the feature extraction and tracking results. A promising direction is to use visualization.

Finally, even though processing one time step only take a few seconds, applying the trained network to thousands of time steps of the data can take a long time to finish. Since the processing of each time step is completely independent of other time steps, it is feasible and desirable to employ a large PC cluster to conduct the final feature extraction and rendering concurrently.

This paper contributes to the supercomputing community in the following two ways. First, we have demonstrated that learning-based visualization of 4D data is more flexible and powerful than conventional feature extraction methods. In particular, scientists can be more expressive in specifying features of interest. Second, the capability to directly extract and track features in a high-dimensional space offers scientists unprecedented explorability that will help them gain more insights in their data.

Acknowledgments

This work has been sponsored in part by the U.S. National Science Foundation under contracts ACI 9983641 (PECASE), ACI 0222991, and ANI 0220147 (ITR), ACI 0325934 (ITR), and the U.S. Department of Energy under Lawrence Livermore National Laboratory Agreement No. B537770, No. 548210 and No. 550194. The authors would like to thank NCAR, Dr. Pierre Lalleman, Dr. Renyue Cen, and Dr. Jackie Chen from Sandia National Laboratories for providing the quasi-geostrophic turbulence flow data set, the swirling flow data set, the cosmological hydrodynamic simulation data set, and the turbulent combustion data set, respectively. The Argon Bubble data set was provided courtesy of John Bell and Vince Beckner of the Center for Computational Sciences and Engineering, Lawrence Berkeley National Laboratory.

9. REFERENCES

[1] James Ahrens, Kristi Brislawn, Ken Martin, Berk Geveci, C. Charles Law, and Michael Papka. Large scale data visualization using parallel data streaming. *IEEE Computer Graphics & Applications Special Issue on Large Data Visualization*, 21(4):34–41, July/August 2001.

[2] Chandrajit L. Bajaj, Valerio Pascucci, and Daniel R. Schikore. The contour spectrum. In *Proceedings of IEEE Visualization 1997 Conference*, pages 167–173, 1997.

[3] Jian Chen, Deborah Silver, and Lian Jiang. The feature tree: Visualizing feature tracking in distributed amr datasets. In *IEEE Symposium on Parallel and Large-Data Visualization and Graphics*, pages 103–110, 2003.

[4] Nir Friedman, Dan Geiger, and Moises Goldszmidt. Bayesian network classifiers. *Machine Learning*, 29:131–163, 1997.

[5] Allen Van Gelder and Kwansik Kim. Direct volume rendering with shading via three-dimensional textures. In *1996 ACM/IEEE Symposium on Volume Visualization*, October 1996.

[6] Erol Gelenbe, Yutao Feng, and K. Ranga R. Krishnan. Neural network methods for volumetric magnetic resonance imaging of the human brain. In *Proceedings of IEEE International Workshop on Neural Networks for Identification, Control, Robotics, and Signal/Image Processing*, 1996.

[7] Lawrence O. Hall, Amine M. Bensaid, Laurence P. Clarke, Robert P. Velthuizen, Martin S. Silbiger, and James C. Bezdek. A comparison of neural network and fuzzy clustering techniques in segmenting magnetic resonance images of the brain. In *IEEE Transactions on Neural Networks*, volume 3, September 1992.

[8] Marti A. Hearst. Trends and controversies: Support vector machines. *IEEE Intelligent Systems*, 13(4):18–28, 1998.

[9] T. J. Jankun-Kelly and Kwan-Liu Ma. A study of transfer function generation for time-varying volume data. In *Proceedings of Volume Graphics 2001 Workshop*, pages 51–65, 2001.

[10] Guangfeng Ji, Han-Wei Shen, and Rephael Wenger. Volume tracking using higher dimensional isosurfacing. In *Proceedings of IEEE Visualization 2003 Conference*, 2003.

[11] Gordon Kindlmann. Course notes: Transfer functions in direct volume rendering: Design, interface, interaction. *Siggraph 2002 Course*, 2002.

[12] Kwan-Liu Ma and David Camp. High performance visualization of time-varying volume data over a wide-area network. In *Proceedings of Supercomputing 2000 Conference*, November 2000.

[13] Kwan-Liu Ma and Tom Crockett. Parallel visualization of large-scale aerodynamics calculations: A case study on the cray t3e. In *Proceedings of 1999 IEEE Parallel Visualization and Graphics Symposium*, pages 15–20, 1999.

[14] Kwan-Liu Ma, Alex Stompel, Jacobo Bielak, Omar Ghattas, and Eui Joong Kim. Visualizing large-scale earthquake simulations. In *Proceedings of the Supercomputing 2003 Conference*, November 2003.

[15] Andrew McCallum, Dayne Freitag, and Fernando Pereira. Maximum entropy markov models for information extraction and segmentation. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 591–598, 2000.

[16] Leonid I. Perlovsky. *Neural Networks and Intellect: Using Model-Based Concepts*. Oxford University Press, 2000.

[17] Hanspeter Pfister, Bill Lorensen, Chandrajit Babaj, Cordon Kindlmann, Will Schroeder, and Raghu Machiraju. The transfer function back-off. In *Proceedings of IEEE Visualization 2000 Conference*, 2000.

[18] Frank J. Post, Theo van Walsum, Frits H. Post, and Deborah Silver. Iconic techniques for feature visualization. In *Proceedings of IEEE Visualization 1995 Conference*, 1995.

[19] Frits H. Post, Benjamin Vrolijk, Helwig Hauser, Robert S. Laramee, and Helmut Doleisch. The state of the art in flow visualization: Feature extraction and tracking. *Computer Graphics Forum*, 22(4):775–792, 2003.

[20] Freek Reinders, Frits H. Post, and Hans J.W. Spoelder. Visualization of time-dependent data using feature tracking and event detection. *The Visual Computer*, 17(1):55–71, 2001.

[21] David E. Rumelhart and James L. McClelland. *Parallel and Distributed Processing: Explorations in the Microstructure of Cognition*, volume 1. The MIT Press, 1986.

[22] Deborah Silver and Xin Wang. Tracking and visualizing turbulent 3d features. *IEEE Transaction on Visualization and Computer Graphics*, 3(2):129–141, 1997.

[23] Deborah Silver and Xin Wang. Tracking scalar features in unstructured datasets. In *IEEE Proceedings of Visualization '98 Conference*, 1998.

[24] Deborah Silver and Norman 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.

[25] Fan-Yin Tzeng, Eric B. Lum, and Kwan-Liu Ma. An intelligent system approach to higher-dimensional classification of volume data. *IEEE Transactions on Visualization and Computer Graphics*, 11(3):273–284, 2005.

[26] Fan-Yin Tzeng and Kwan-Liu Ma. Opening the black box - the data driven visualization of neural networks. In *IEEE Proceedings of Visualization '05 Conference, to appear*, 2005.

[27] Paul Werbos. *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*. PhD thesis, Department of Applied Mathematics, Harvard University, 1974.

[28] Brian Wylie, Constantine Pavlakos, Vasily Lewis, and Ken Moreland. Scalable rendering on PC clusters. *IEEE Computer Graphics and Applications*, 21(4):62–70, July/August 2001.

[29] Hongfeng Yu, Kwan-Liu Ma, and Joel Welling. A parallel visualization pipeline for terascale earthquake simulations. In *Proceedings of the Supercomputing 2004 Conference*, November 2004.