Group Dynamics in Scientific Visualization

Sedat Ozer[1] Jishang Wei[2] Deborah Silver[1] Kwan-Liu Ma[2] Pino Martin[3]

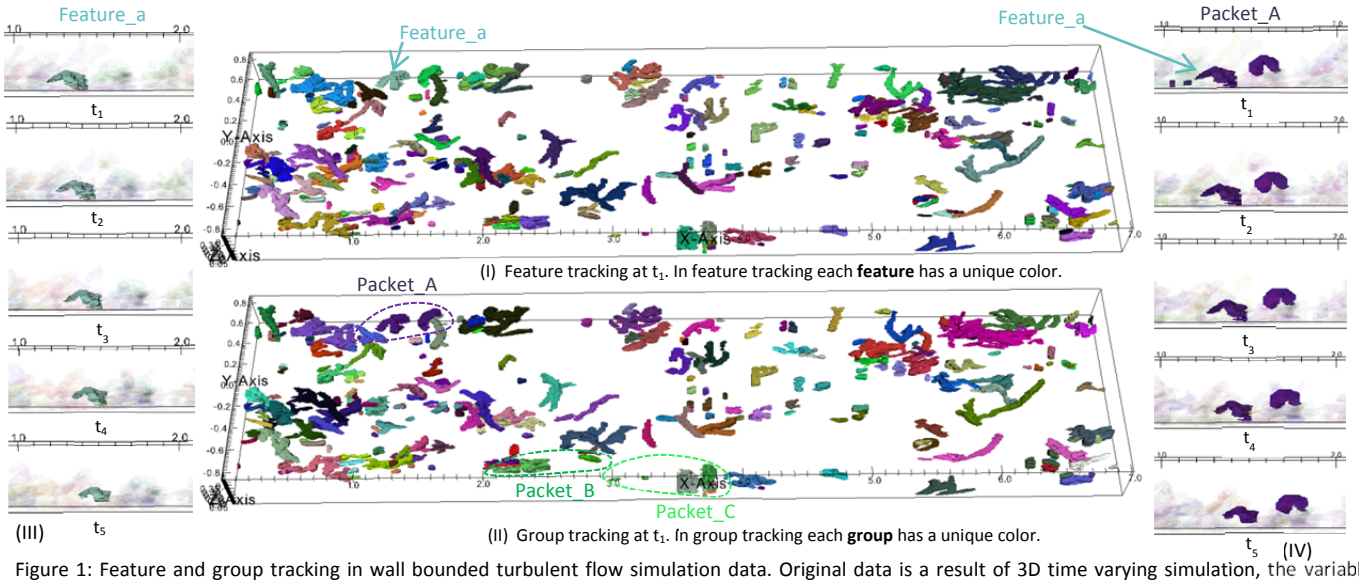

Figure 1: Feature and group tracking in wall bounded turbulent flow simulation data. Original data is a result of 3D time varying simulation, the variable being visualized is swirl magnitude; (I) Features extracted in t_1 where each feature has an automatically generated unique color (total 262 features); (II) Packets in t_1 where a total of 177 packets are identified and 3 sample packets (Packet_A, Packet_B and Packet_C) are circled. Packets are groups of features that travel together. (III) Feature tracking of Feature_a in the first 5 time frames; (IV) Group tracking of Packet_A in the first 5 time frames where Feature_a ϵ Packet_A. Features join and leave packets throughout a packet evolution.

Abstract— The ability to visually extract and track features is appealing to scientists in many simulations including flow fields. However, as the resolution of the simulation becomes higher, the number of features to track increases and so does the cost in large-scale simulations. Since many of these features act in groups, it seems more cost-effective to follow groups of features rather than individual ones. Very little work has been done for tracking groups of features. In this paper, we present the first full group tracking model in which we track groups (clusters) of features in time-varying 3D fluid flow simulations. Our model uses a clustering algorithm to group interacting features. We demonstrate our model on data output from a 3D simulation of wall bounded turbulent flow.

Index Terms— Feature tracking, group tracking, grouping, clustering, packet identification, scientific visualization.

1 Introduction

Following the interactions of physical phenomena over time is an important problem. There have been many papers that deal with time varying phenomena focusing on individual phenomenon or features, examples can be found in the papers [1], [2], [3], [4],[5], [6] and [7]. Today's advanced simulations generate high resolution data at peta-byte or exa-byte scale. As datasets get larger, the number of features in the simulation grows. Similarly, as the resolution becomes finer, some features that appeared large in coarser resolution runs become finer. The change in resolution affects the identification of feature dynamics since some features merge at a lower resolution [9]. This is similar to identifying finer features as a single structure, i.e. group, at a lower resolution. Various interactive navigation and exploration tools proposed to help scientist explore the large scale data at different resolutions as in [9] and [10].

Groups also exist in scientific simulations regardless of the resolution. In many domains, features tend to act coherently in meaningful groups. Generally speaking, a group is a set of coherent structures (features) that are related to each other in certain ways or "act" together (See Section 3). (This is analogues to birds that both act individually and fly together in flocks). Understanding both the individual feature evolutions and the group evolutions is important to understanding the dynamics of scientific phenomena. Examples of scientific group dynamics include groups of cells and studying their collective behaviour in biochemistry [8], groups of galaxies (halos) in cosmology [11] and groups of hairpin vortices (packets) in wall bounded turbulent flow [12]. What is common among all the above examples is that they all have a hierarchical structure where a set of smaller structures (e.g., a set of voxels) forms a high level structure (e.g., a feature) and a set of these high level structures (e.g., features) forms an even a higher level structure (e.g., a group).

While feature tracking follows the interaction and evolution of individual features, the dynamics of groups of features has not been fully addressed. Our focus here is on unconnected or disambiguated features (although our model can be applied to groups of connected features as well). While many features in scientific simulations can be connected at very low thresholds, our goal in this work is to understand the dynamics of groups of individual features.

[1] *Vizlab, Electrical & Computer Engineering, Rutgers University, E-Mail: {sedat, silver}@rutgers.edu*
[2] *VIDi, Computer Science, University of California at Davis, E-Mail: {jswei, ma}@cs.ucdavis.edu*
[3] *Crocco Lab, Aerospace Engineering, University of Maryland, E-Mail: mpmartin@umd.edu*

In this work, we present the first full group tracking model in which we track groups as well as the individual features that comprise them. In order to track groups, we first extract features, track them and then group them based on the grouping criteria. Once features are grouped at each time step, we track the groups to see how they change and interact over time. A clustering algorithm is employed to cluster (group) the features. One benefit of employing a clustering algorithm is the ability to isolate and map the group definitions from various domains to the computational domain via similarity functions (See Section 3). This property of clustering algorithms helps us to build a generic model for group tracking. We apply our group tracking model on a wall bounded turbulence-Direct Numerical Simulation (DNS) and demonstrate various visualizations.

2 Related Work

Early studies of group tracking consider movement of features within radar data a list of such papers can be found in [13]. Computer vision studies also address group tracking. McKenna et al. [14] proposed a computer vision based tracking system. The system utilized color to disambiguate occlusions and to qualitatively estimate the depth ordering and position during occlusion. The system tracks groups of people through mutual occlusions as they form groups and then separate. Gennari and Hager [15] introduced a general class of partitioning functions to define a group, and a set of rules to split and merge groups. Based on the group definition, they proposed a modified PDA estimator to track groups of objects. They reported that they can detect the groups of people that merge and split. Mucientes and Burgard integrated Multi-Hypothesis Tracking (MHT) with Murty's algorithm to tack clusters of people [16]. Joo and Chellappa targeted on solving the data association problem in object tracking using the multi-hypothesis approach [17]. Lau et al. extended the MHT method to hypothesize over both, the group formation process (models) and the association of observations to tracks (assignments) [18].

In scientific visualization, the first complete feature tracking framework was introduced by Samtaney et al. [1], as a solution to extract and track the volumetric features within time varying data sets. In their framework, they proposed the first feature tracking model and used centroid position, volume, mass and 2D circulation information to track features. In the same group, Silver and Wang extended the feature tracking model by introducing the volume tracking schema in [2], [19] and [20]. Then they observed that features overlap between two consecutive time frames when the frame sampling rate is high enough. By incorporating this observation in their work, they developed a more memory efficient algorithm which supported unstructured data sets as well. The next refinement, prompted by extending the feature tracking framework to a distributed Adaptive Mesh Refinement (AMR) data structures, allowed a viewer to isolate a multi-level isosurface and visualize its evolution spatiotemporally at different resolutions [21]. Next, the fitting of ellipsoids to the features was introduced for an efficient attribute computation and applied to the characterization and visualization of plumes [22]. Several studies use a predictive approach to tracking. In [7], Reinders et al. proposed a method that estimates the locations of the features in the next frame to improve the tracking quality by assuming that the features evolve predictably. Similarly, Muelder and Ma [5] proposed to include a prediction method. However, instead of extracting all the features first, they propose using the information of the current and previous frames to estimate the location of the feature in the next frame and they use that information to perform both segmentation and tracking at the same time. Other studies address different ways of abstracting the correspondence. In [3], Ji et al. proposed a method to track features by using higher dimensional isosurfacing. Thus, instead of extracting the isosurfaces from each time frame and then performing an overlapping test, they propose using the higher dimensional geometry to track user selected features in time by performing an isosurfacing process in R^4 as opposed to doing it in R^3. In [23], Tzeng and Ma applied neural networks to find the region boundaries by estimating the transfer functions in the feature tracking framework. Sohn and Bajaj [24], proposed a method to compute the correspondence in time varying contour trees. By applying this idea on feature tracking, they tracked user selected contours. In [25] Laney et al. propose to use Morse-Smale complex to segment bubbles in a hierarchical segmentation structure. Different attributes have also been used to correlate features. In [26], Ji and Shen proposed using earth mover's distance metric to track objects as an alternative to the volume overlapping or attribute based matching methods. In [27], Caban et al. proposed to use first and second order statistics with run-length matrices to capture textures and to distinguish them. Thus, by generating a feature vector, they perform a similarity search to find the best matches. In [28] Gezahegne et al. proposed a method that allows objects to retain their original labels for t frames, thus the algorithm can detect bubbles going through each other instead of being classified as merged and then split.

In earlier visualization works, clustering has been successfully employed to extract the groups for visualization. Examples can be found the papers [29], [30], [31] and [32]. However none of these papers address the issues related to group tracking.

In our work, we incorporate a number of these ideas to as we modify our previous feature tracking framework to track groups of features. Next we will give an overview of group extraction followed by details of some aspects.

3 Group Extraction

The group tracking framework flow diagram is shown in Figure 2. In this diagram, the first step is feature extraction (Figure 2a). A *feature* is the region of interest in the data. Features can be extracted by domain specific algorithms, or by using common techniques such as region growing, clustering, geometry or topology based algorithms as in [2], [3], [4], and [25]. Any of these techniques can be used in this framework. In addition to extraction of features, their attributes also have to be computed. *Spatial feature attributes* are computed after features are determined and include the geometric structure of the feature as well as the variable values. Examples are centroid, min/max (local extrema), volume, feature extends, surface information, etc. A list of different types of feature attributes is given in Table 1.

Spatial feature attributes	**Time-dependent feature attributes**	**Feature-to-Feature attributes**
Centroid, Max/min, Volume, Shape, Total number of particles, Mass, Self-Orientation, Moments, Extends, Mean/variance over voxels, Surface information.	Velocity, Acceleration, Self-rotation, Swirl, Extension, Expansion, Shrinkage, Change in a feature attribute, Mean/variance over time.	Rotation around a feature, Distance, Relative orientation, Nearest-neighbour information, Variable difference/sum, Mean/variance over features.

Table 1: Three different attribute categories and examples for each category

Next step is feature tracking (Figure 2b). Feature tracking correlates the features from the previous time step t_{i-1} to the next step t_i [2]. This correlation information is saved in a feature history. In this step *time-dependent feature attributes* can also be computed (examples are listed in Table 1). These are the attributes derived by jointly considering the current and previous *spatial-feature attribute*s. They can also define the change in a feature attribute over a specified duration. Accessing a specific feature's attribute in the previous time steps requires tracking information, i.e., feature

history. An example of this is the velocity. Other attributes are listed in the middle column of Table 1.

Concurrently with feature tracking, one can also compute *feature-to-feature attributes*. These are the attributes that are computed by comparing a feature to neighbouring features in the same time step. They can be derived from feature attributes or time dependent feature attributes. For example, the mean (or variance) of a feature attribute can be computed over the neighbour features.

In the next step groups are determined (Figure 2c). A *group* is a set of coherent features that are associated together based on some criteria. These criteria can be expressed in terms of any combination of the feature attributes. A list of sample attributes can be found in Table 1. For instance, the grouping criteria can be geometry, distance, shape, rotation or orientation based. For a group, all these criteria can be combined and expressed within a single "similarity" function which is used by a clustering algorithm to determine the groups (See Section 3.1.1). Once groups are determined, spatial group attributes can also be computed in this step. Spatial group attributes are defined similar to the spatial feature attributes. They summarize the spatial properties of the extracted group and its member features.

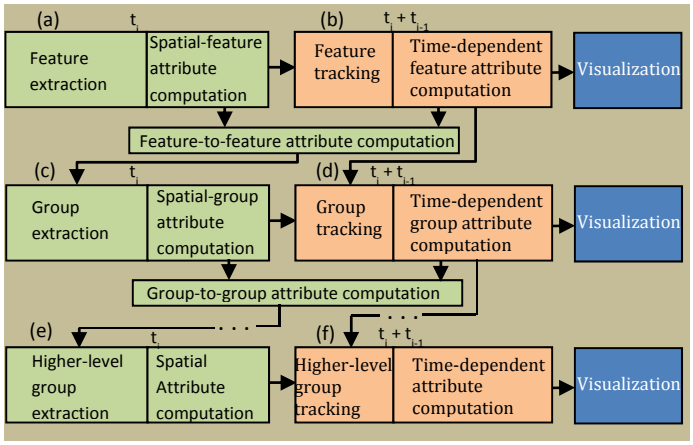

Figure 2: Group tracking framework

Tracking groups is the next step (Figure 2d). Group tracking correlates the extracted groups in time step t_i to the groups in time step t_{i-1}. Once computed, this correlation information is saved as group history. *Time-dependent group attributes* are also computed in this step by using the group history. *Time-dependent group attributes* are defined similar to the *time-dependent feature attributes.* These are the attributes derived by using the current and previous *spatial-group attribute*s.

Once groups are determined and tracked, *Group-to-group attributes* can be computed. These attributes are also defined similar to their feature counterpart, i.e., *feature-to-feature attributes*. They are computed by comparing a group to one another or to a certain number of neighbour (K-Nearest) groups in the data.

The last steps involve creating super-structures of groups (Figure 2e). If the domain has super-structures (groups of groups) that are defined with a different set of criteria, then similar to the group extraction step, super-structures can be extracted by using the related similarity function for the super-structures. At this step, each group is assigned to a super-structure by a clustering algorithm. *Spatial super-structure attributes* are also computed at this step. These attributes are also defined similar to the *spatial-feature attribute*s. They summarize the properties of the super-structure and its member groups. Furthermore, *time-dependent super-structure attributes* can also be defined similar to the *time-dependent feature attributes.* These are the attributes derived by using the current and previous *spatial- super-structure attribute*s.

Once defined, all the higher level structures (e.g. groups of super-structures) can be recursively extracted and tracked in a similar way to the super-structures. And their associated attributes can be computed.

Time-varying visualization uses the tracking results at each structure level. Therefore once the tracking is performed and the related time-varying attributes are computed, the associated level structures can be visualized. In the following sections we will focus on clustering, group tracking and group visualization in detail.

3.1 Grouping via clustering

Once features are extracted and their feature-to-feature attributes are computed, they can be grouped using a clustering algorithm. The focus below is on simulations where features cluster together and then act in groups. Hierarchical and partitioning clustering algorithms are two main types of similarity based clustering methods [33], [34]. Hierarchical clustering seeks to build a hierarchy of clusters. It either starts from each individual data object as a cluster and then merges two most similar clusters recursively until only one cluster is left (agglomerative clustering); or it starts from the whole collection of data as a cluster and split the data set recursively until reaching a pre-specified cluster number. Partitioning clustering, such as the K-means algorithm, divides data objects into a number (often specified by a priori K value) of clusters according to some optimization criterion. Hierarchical clustering yields a hierarchical structure besides the final cluster information. Hierarchical clustering does not require a pre-specified number of clusters as opposed to the partitioning clustering. Moreover, hierarchical clustering does not require any initialization parameters as opposed to K-means algorithm that require initial cluster centroids. However, there is a trade-off between these advantages of hierarchical clustering and its computational efficiency. For more information on the clustering algorithms please refer to the text books [35] and [36]. In this work we use hierarchical clustering to extract groups in our group tracking model. At each time step the clustering algorithm is run and groups are formed.

In a clustering algorithm, the features with the highest similarity are grouped into the same cluster. Notice that within a cluster, all the features should be similar to each other, while each of them should be dissimilar for the inter-cluster features. Similarity is defined in the next sub-section.

3.1.1 Similarity Functions

A similarity function is a function $S: R^n x R^n \rightarrow R$ that provides a measure of the similarity between two given vectors in a similarity space [36]. These two given (input) vectors represent two individual features with their n attributes. For any two given vectors $\boldsymbol{F_A}$ and $\boldsymbol{F_B}$, the similarity measure $S(\boldsymbol{F_A}, \boldsymbol{F_B})$ is the same as $S(\boldsymbol{F_B}, \boldsymbol{F_A})$, i.e., $S(\boldsymbol{F_B}, \boldsymbol{F_A}) = S(\boldsymbol{F_A}, \boldsymbol{F_B})$. Moreover, $S(\boldsymbol{F_B}, \boldsymbol{F_A}) \leq S(\boldsymbol{F_A}, \boldsymbol{F_A})$ and $S(\boldsymbol{F_B}, \boldsymbol{F_A}) \geq 0$. Notice that if the vectors $\boldsymbol{F_A}$ and $\boldsymbol{F_B}$ are dissimilar, then the similarity measure is the minimum value, i.e., $S(\boldsymbol{F_B}, \boldsymbol{F_A}) = 0$. With these conditions in mind, a group can be defined in terms of a similarity function. This similarity function will have a nonzero similarity value for any two given features in a given group and will have a zero similarity value for any given two features from different groups. For example, distance based similarity functions yield a higher similarity value as the distance between two input vectors decreases; i.e., as the features get closer to each other, the similarity value increases. Another example can be a shape and distance based similarity function in which the features that are close to each other and look alike would give higher similarity values. A threshold can be set for the similarity function value to define the terms *similar* and *dissimilar*. A higher value of the similarity function implies a greater similarity between the given two features. Once the similarity function is defined or selected from an available list/library, the clustering algorithm can group the features based on this provided similarity function.

4 Group Tracking & Group Events

Identifying the dynamics of structures (e.g., features or groups) requires correlating these structures over time and this correlation process is generally known as the correspondence problem [2]. The tracking step addresses the correspondence problem in the group tracking framework (in Figure 2d). Once we have identified groups and features in the data, we can identify what happens to groups over time. Similar to features, groups can merge, split, appear (birth), disappear (death) or continue (see Section 4.1 for details).

In the framework, group level tracking (matching) from one time step to the next can be performed by combining the feature tracking information of the current time with the extracted group information (as shown in Figure 2d). Clearly, if the features overlap in volume, then their groups also overlap in volume. Therefore, in this work we employ a volume overlapping schema to track groups.

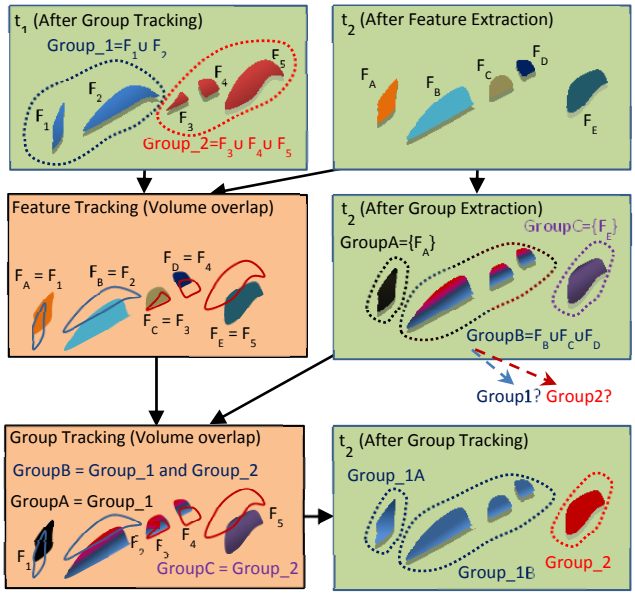

Figure3: An illustration of various steps in group tracking in time t_2. GroupB is formed of two features from Group_2 and one feature from Group_1 in t_1. This is a partial split event. Groups in t_1 and t_2 can be correlated visually after the group tracking process.

Even in the cases where the features do not overlap, their groups can still overlap if a group can be defined by a convex hull surrounding the member features since groups are bigger structures. In our model, each group has a list of its individual features and each feature has its *group_ID*. For reference, each feature is represented by $F_{k,j}^{\ i}$, where k is a unique feature identifier, j is the group identifier that the feature belongs to and i is the time step (t_i) index. Similarly, each group is represented by G_j^i, where j is a unique group identifier in time step t_i. Generally speaking, a particular group is the union of its member features and will assume that a group is a union of its member features, i.e., $G_j^i = \bigcup_{k=1}^{h} F_{k,j}^i$ where h is the number of total features within the group with the identifier j in time step t_i. We use the following definition for volume overlap for groups which is an extension of the one in [19]:

Overlap: If the group G_A^i corresponds (matches) to G_B^{i-1}, then some features from G_A^i overlap with some features from G_B^{i-1}. i.e., $G_B^{i-1} \cap G_A^i \neq \emptyset$. By using this overlap definition, an overlap table can be computed between the groups from one time step to the next. The overlap table can be computed by using one of the following criteria:

(1) Feature overlap criterion, (2) Feature number overlap criterion or (3) Convex hull overlap criterion.

·*Feature overlap criterion*: This criterion uses the sum of overlapping feature volumes between the groups. Since we assume that a complete feature is a part of a group, a group's volume can be computed by summing up the volumes of each member features. Figure 3 illustrates the tracking process with the feature overlapping criterion. The feature and group extraction steps (with attributes) are completed in the first time step t_1. When the framework starts processing the next time step t_2, the first step is extraction of the features and computing the feature attributes. Then features are matched to the ones in t_1 by using the volume overlapping criterion [2]. Feature tracking relates features from t_1 to the ones in t_2. Assume that grouping is done based on the distance information only. In this case, groups can be extracted by using the spatial-feature attributes only. Group extraction yields GroupA, GroupB and GroupC in t_2. Group tracking step creates the overlap table based on the feature overlapping criteria (instead of using the actual values computed in overlap table for feature tracking). For groups, overlap table is computed by using the entire volume of the joint member features between t_1 and t_2. This is shown in Table 2.

	Group_1	**Group_2**
GroupA	F_A	0
GroupB	F_B	$F_C \cup F_D$
GroupC	0	F_E

Table 2: Overlap table by using the feature overlap criterion.

Based on the overlap table, GroupA is matched to Group_1 and GroupC is matched Group_2 only. However GroupB is matched to both Group_1 and Group_2. The dominant group for Group B is Group_1 since the volume of F_B is greater than the sum of volumes of F_C and F_D.

·*Feature number overlap criterion*: This criterion uses the total number of overlapping features within each group. If the features are relatively homogeneous in volume and shape, then this kind of a simplification could work reasonably faster and accurate enough. However, matching/correspondence is a function of both time and spatial properties. Using only the number of matching features does not summarize the spatial attributes of groups properly since it ignores the volume or shape information and thus may not yield an accurate matching. Especially in cases where a threshold is used, the least matching groups with the least feature numbers can be ignored. However such groups might have bigger volume and might be the actual dominant matching groups.

·*Convex hull overlap criterion*: This criterion first defines a convex hull for each group that comprise all the features within, and then perform a volume overlap test between the convex hulls for group matching. This criterion considers the spaces between the features as well. In this criterion the volume of the containing envelope of all the features in a group is used. Thus a convex hull of an oddly shaped group formed of 2 or 3 small features can contain a bigger volume and can overlap with other neighbour groups.

The only assumption that these above-mentioned overlapping criteria make is that features (and their groups) overlap from one time step to the next. Therefore even if no distance based similarity metric is used in the group definition, as long as the features overlap, one of the abovementioned criteria can still be used for tracking the groups. This is true since we also track the features in advance and have the feature correspondence list available already. In some cases where the data comes from an insufficient sampling rate, an overlap criterion (where available) can be derived by using shape, speed, volume, mass, min/max value, speed or acceleration information to estimate the best match for group tracking (as well as for feature tracking).

In our case study, the features are not homogenous in volume or in shape. Therefore, in this study we use the feature overlap criterion. A tolerance value can be used in an overlap table to ignore smaller overlaps [19]. This process helps eliminating unwanted small amounts of overlaps (matches) or detecting false events. Groups G_A^i and G_B^{i-1} are matched, if:

$$\frac{max(G_A^i - G_B^{i-1}, G_B^{i-1} - G_A^i)}{\max(G_B^{i-1}, G_A^i)} < Tolerance \quad (1)$$

Equation 1 defines the normalized volume difference test as in [19]. *Tolerance* is a domain dependent value.

4.1 Group and cross-level events

Feature tracking can characterize the evolutionary events (dynamics) of features such as merge, split, continue, birth (appear) and death (disappear) [2]. Similar to features, the higher level structures (groups) can split, merge, continue or die. However, these group events are slightly different than the ones defined for features. The difference is due to the fact that the features are unconnected in groups. Therefore we define the following events for groups:

Birth: A new group of features is formed in the current time step and is not correlated to any group in the previous time step. i.e., if the group G_A^i does not overlap any groups in t_{i-1}, then the group G_A^i is considered as a new born group.

Death: An existing group of features in the previous time step disappears in the current time step. If the group G_B^{i-1} does not overlap any groups in t_i, then the group G_B^{i-1} is considered as a disappearing group and the event is called a death event.

Full Split: A single group G_B^{i-1} in t_{i-1} splits into a number of N groups (N>1) in t_i. Each of these groups in t_i overlaps G_B^{i-1}, i.e., $G_B^{i-1} \cap G_j^i \neq \emptyset$ for each $j \epsilon$N.

Full Merge: This is the event where a number of N groups (N>1) in t_{i-1} merge to form a single group G_B^i in t_i. i.e., if a number of N groups in t_{i-1} merge to form a single group G_B^i in t_i, then G_B^i overlaps each of N groups in t_{i-1}, , i.e., $G_B^i \cap G_j^{i-1} \neq \emptyset$ for each $j \epsilon$N.

Partial Merge: This is the event where portions (or all) of N groups in t_{i-1} join other groups resulting in a lesser number (M) of total groups in t_i. Different portions of a group in t_{i-1} can merge to several other groups in t_i. Also note that different portions of different groups in t_{i-1} can merge to the same group in t_i. Therefore the total number of N groups in t_{i-1} merge and form a number of M new groups (1<M<N) in t_i, where each of these N groups in t_{i-1} overlaps each of the M groups in in t_i. At the end, the total number of groups M in t_i will be less than the total number of groups N in t_{i-1}.

Partial Split: This is the event where the portions of a number of N groups split from their groups to form M new groups, i.e., only the portions of N groups (N>1) in t_{i-1} form a total number of M new groups in t_i. In this case, each of these N groups in t_{i-1}, overlaps with each of the M groups in in t_i. At the end, the total number of groups M in t_i will be higher than the total number of groups N in t_{i-1}. For example, the event where two groups form three groups is a partial split event in Figure 3.

Continuation: If none of the above-mentioned events occurs, than this is a continuation event for the group.

In addition to these events, when hierarchical structures are considered, the cross-level events can also be defined and detected. Cross-level events are the events that happen between a structure and a higher level structure. For example, group tracking allows detecting a feature leaving its group to join another one (cross-group event).

Cross-group: This is the event where a feature leaves its group to join another group. In this event, while at the group level, groups can remain the same (groups continue); at the feature level, a feature can move from one group to another and therefore, this event is different than the partial merge or split events. An illustration of this event is shown in Figure 4a. In the illustration, when we use the first level (feature) tracking, we only detect the split event that occurs within the Group_S (shown with solid blue arrows). However, when the group tracking is used, we can detect that a feature moves from Group_R in t_{i-1} to Group_S in t_i (shown with the dashed arrow).

5 Visualization

The group tracking framework can visualize structures at a user specified level within a hierarchical structure. Specifically, the tracking results (group history) generated by the group tracking algorithm can be combined with common visualization techniques (such as isosurface or volume rendering) to visualize groups and their evolution in time varying simulations. For example, Figure 1-(IV) shows an isosurface visualization of the evolution of a selected group (packet) over the first five time steps in the data. Alternatively, user can select to visualize the evolution of a selected feature over the time as in Figure 1-(III). In isosurface visualization, each group is given a unique colour. Group history is used to assign the same color to the same group over time. When the groups merge, the newly formed group can have the color of the dominating group. In this case, the dominance can be decided by volume, mass, extents or local extrema value similar to the method of [19]. Alternatively, a new color can be derived as a linear combination of the merged group colors. In this case, some distinctive attribute value can be used to weight the colors in the linear combination (such as the volume). Splitting groups can have the parent group's color. Certain groups (or features) performing a specific event can be highlighted by decoloring all other groups (or features). This can be done by assigning a dull color (such as grey) to all other groups (or features) or by changing the opacity. In our applications, new born groups are assigned a randomly generated unique color, merging groups are assigned the color of the group with the maximum volume. All the children groups are assigned the parent groups's color in the split events. An evolution of a selected group can be visualized over time by only showing the groups that interact with the selected group.

Similarly, various visualization schemas can be used for volume rendering. For example, merging, splitting or continuing groups can be assigned to specific transfer functions to highlight these events only.

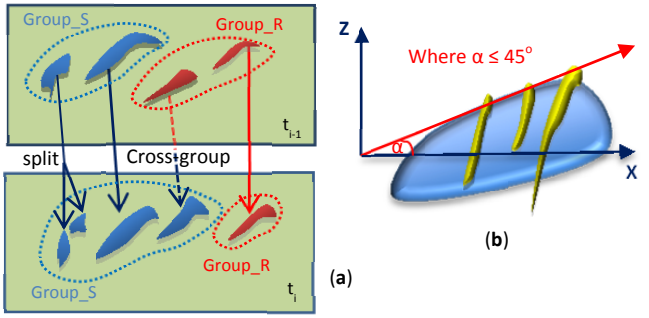

Figure 4: An illustration of cross-group event and a packet (group) in wall bounded turbulence simulations. (**a**) A cross-group event where a feature moves from Group_R to Group_S while the number of groups remains the same. (**b**) An illustration of a packet in wall bounded turbulence simulations. In a packet, yellow hairpins elongate to form a certain angle that is smaller than 45°.

6 Case Study

We apply our group tracking model on wall bounded turbulent Direct Numerical Simulations (DNS). In wall bounded turbulent flow simulation, scientists are interested in searching for the existence of groups (called packets) formed by coherent but unconnected turbulent hairpin vortices.

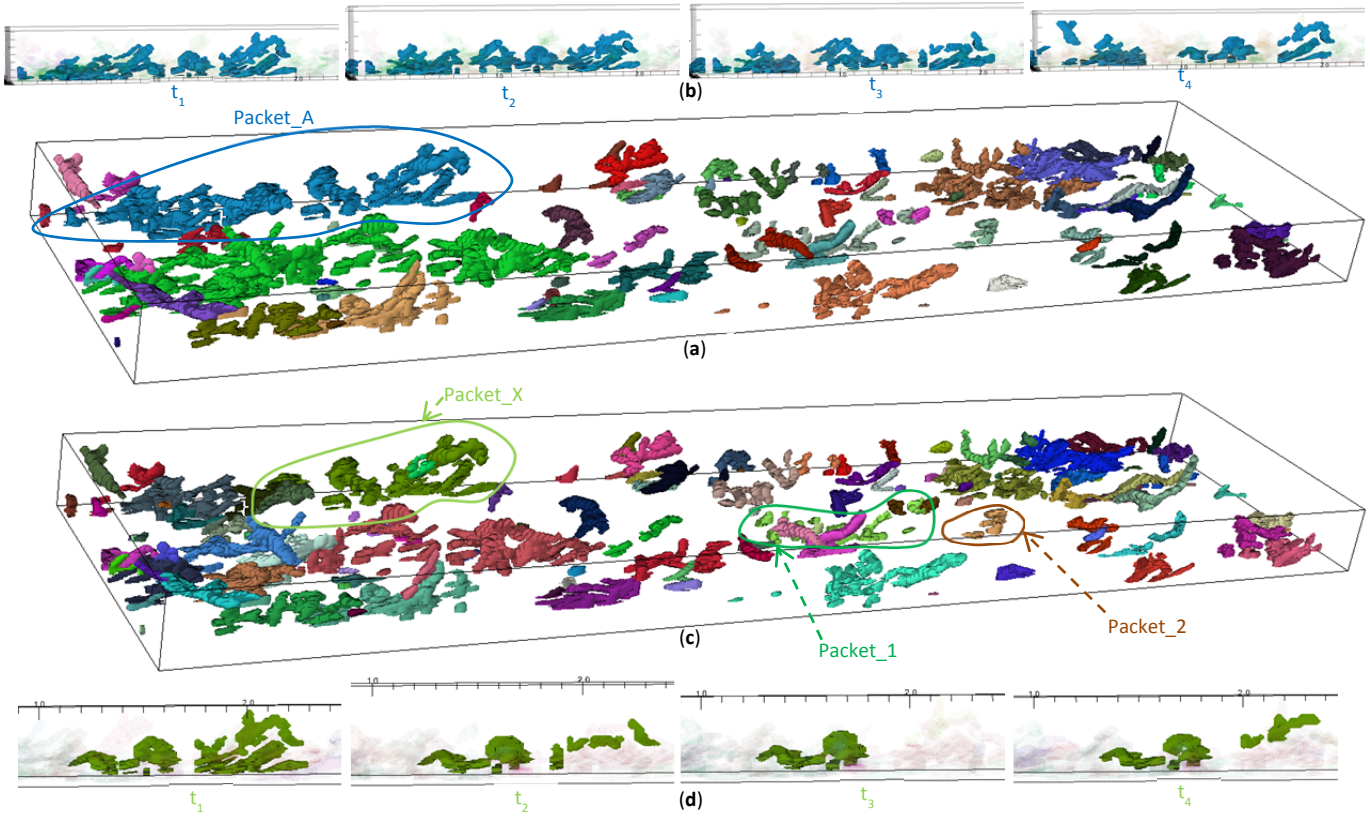

Figure 5: Group tracking in wall bounded turbulent flow simulation. (**a**) visualizes all the determined groups in t_1. Each group has a unique and randomly generated color. (**b**) visualizes the selected Packet_A from (a) and tracks it in the first four time steps. (**c**) visualizes all the determined groups with a smaller distance criteria in t_1. (**d**) visualizes the selected Packet_X from (c) and tracks it in the first four time steps.

6.1 Creating a domain specific similarity function

A packet is defined geometrically based on both the distance and the angle (orientation) between the member features. An illustration of such a packet is shown in Figure 4b. This illustration is similar to the one in [37]. Commonly used similarity functions do not consider the orientation of features within a packet [34], [36]. Therefore in this section, we demonstrate an example to derive a domain specific similarity function for packet identification. The similarity function should provide a positive value for a pair of features in the same packet and a zero value for a pair of features from different packets. We define a packet as a set of features meeting all of the following three criteria:

I. The distance between a given set of two features should be less than a predefined distance (threshold).

II. The angle between two given features should be equal or less than 45^o.

III. The packet should be elongated along the X axis such that the cross section (in y-z plane) increases along the X axis.

Various distance measures can be computed and used for the first criterion, such as the nearest neighbour distance (on the feature surface), the centroid distance or the distance between the local extrema points. We notice that the local maximum point of each feature localizes around the top of the feature. Therefore we use the distance between the local maximum points to simplify the computations. We used different thresholds parallel to each axis for the distance criterion.

In the second criterion, the angle between the two features can be computed in different ways. We can either use the moments [19], line fitting or representative points such as centroids or local extrema. Depending on the shape, the centroid may not be within the boundaries of a feature. In our case study, we observed that using the angle between the local maximum points was sufficient to define a packet. Therefore we use the local maximum points to describe each feature and to simplify the computations.

Thus by considering the above-mentioned three criteria, we can set three distinct functions namely A, B and C. Then, we can define the similarity function as the multiplication of these three functions: $S(\boldsymbol{F_1}, \boldsymbol{F_2}) = ABC$ where $\boldsymbol{F_1}$ and $\boldsymbol{F_2}$ represents two individual features. In this equation, A checks for the distance criterion (I); B checks for the angle criterion (II) and C checks for the elongation criterion (III). If any of these conditions is not satisfied by the given two vectors, the final similarity value will become zero. This is the similarity function we derived and used in our application.

In a group not all the pairs of features have to be similar. For example feature $\boldsymbol{F_1}$ can be similar to the feature $\boldsymbol{F_2}$ but not to the feature $\boldsymbol{F_3}$. However, since feature $\boldsymbol{F_2}$ is similar to feature $\boldsymbol{F_3}$, all the features $\boldsymbol{F_1}$, $\boldsymbol{F_2}$ and $\boldsymbol{F_3}$ are considered the members of the same group.

6.2 Data set and results

In this example, we show results from the group tracking framework applied to a subset of a larger wall bounded turbulent DNS. The data set that we used in this study has 46 time steps with a resolution 384x256x69. The variable being visualized is swirl magnitude. A threshold of 0.0005 was used to extract the features in all the figures. The features with a volume smaller than 25 are filtered and are not shown in the figures. The preliminary results of our group tracking algorithm are shown in Figure 1-(II). Figure 1-(IV) shows packet tracking of the circled Packet_A in the first 5 time steps. The individual hairpins were also tracked in subsequent time steps and an example hairpin tracking is shown in Figure 1-(III), while Figure 1-(I) shows all the individual features in t_1 where each feature has a different colour. In all figures, isolated groups are shown by

lowering the opacity value for the background features from other groups.

We visualized the dataset by changing the grouping criteria values (such as the distance threshold or the angle threshold). All the thresholds were given by the domain scientist to reflect physically meaningful groups. Clearly, extracted groups get bigger as the distance threshold used in the distance criterion increases. This is shown in Figure 5. Figure 5a shows all the extracted groups with hierarchical clustering in t_1 with a "large" distance threshold. There are 262 features detected and these features form a total of 79 groups. The total number of features in a group varies between 1 and 41. In both Figure 5a and 5c, each group is assigned a randomly generated unique color. Figure 5b focuses on a selected packet (Packet_A) from Figure 5a and visualizes it in the first four time steps. Figure 5c visualizes the same time step t_1 as Figure 5a, but uses a smaller distance threshold in the similarity function (in the criterion I in Section 6.1). Notice that the total number of groups increases in the data as the distance threshold is decreased. In Figure 5c, the total number of groups increases to 112, while the total number of features remains the same as in Figure 5a (262 features). The total number of features in each group varies between 1 and 26 in Figure 5c. Packet_X in Figure 5c becomes a part of Packet_A when the distance threshold is increased in the distance criterion. Figure 5d focuses on the selected group (Group_X) and visualizes it in the first four time steps. Notice that the feature extents change from time step to the next. Since we visualize only so much of a volume as needed, the size of the volume appears to be changed from time step to time step in both Figures 5b and 5d.

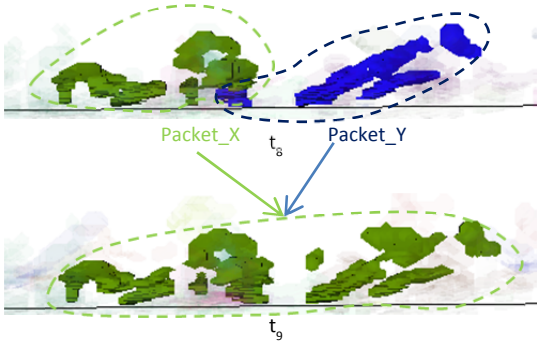

Figure 6: A full merge event. Two packets (Packet_X and Packet_Y) from time step t_8 are merging to form a single Packet in time step t_9. This is a full merge event since both groups form a single new packet. The visualization routine chooses the dominant group as being the Packet_X in this case based on the maximum feature volume criteria and assigns the color of Packet_X to the packet in t_9.

Figure 6 shows an example of a full merge event of a selected packet (Packet_X shown in Figure 5c and Figure 5d) that occurs between the steps t_8 and t_9. A full merge is an event where multiple groups merge to form a single group. In Figure 6, all the members of Packet_X and all the members of Packet_Y from time step t_8 are found in a single packet in t_9, signifying a full merge event.

Figure 7 shows a cross-group event. Consider the side view of Packet_1 and Packet_2 in t_1 in Figure 7a. Packet_1 and Packet_2 in t_1 overlap with the Packet_C and Packet_D in t_2. There are three obvious possibilities that can be considered:

I) both packets (Packet_C and Packet_D) are the result of a split event originating from Packet_1 while Packet_2 disappears,

II) both packets continue where Packet_C = Packet_2 and Packet_D = Packet_1,

III) A portion of Packet_1 merges to Packet_2 to form Packet_C in t_2 and the remaining portion of Packet_1 continues as Packet_D.

Clearly, there is confusion here. This is neither a split nor a merge event of any group but a cross-group event. This is continuation of groups. None of the split or merge conditions is satisfied by the groups and the total number of overlapping groups in each time step does not change (there are two groups in each time step). This means that features from one group have transferred to another group (circled in black), indicating a cross-group event.

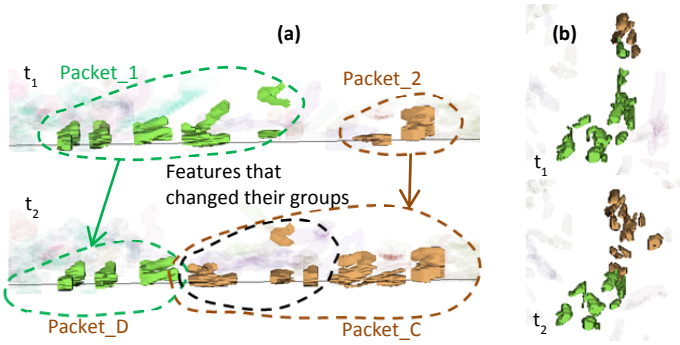

Figure 7: Visualization of a cross-group event. **(a)** The side views (in x-z plane) of Packet_1, Packet_2, Packet_C and Packet_D are shown for time step t1 and t2. Features changing groups are highlighted within a black circle. **(b)** Top view (in y-x plane) of these same packets in step t_1 and t_2.

7 Conclusion and Discussion

In this work we present a framework that can track groups and groups of groups as well as the features. The framework uses the feature tracking model as in [19] and extends it for groups of features acting together. We apply this group tracking framework to a simulation of wall bounded turbulent flow and show how it can be used to identify, track and visualize packets (groups) and packet interactions.

In the framework, domain specific groups can be defined by first specifying the group criteria in terms of the computed attributes. These criteria help to define a similarity function. Then this similarity function is used by a clustering algorithm to assign features to groups. To track these determined groups, we reuse feature tracking results to avoid repeated computation of the same quantity such as volume overlap.

Our group tracking framework helps to answer a new set of questions regarding groups and their events. We define new types of events such as partial merge, partial split and cross-group events. We can also consider groups of groups using the same framework. Currently we are applying this approach to a larger simulation to define and find super-structures.

Group tracking serves as the basis for a more general approach to categorizing complex interactions and events in large scientific simulations. One such approach is activity detection where complex interactions of features or groups that span more than two time steps can be modelled and then detected in time varying simulations [38]. Both individual feature tracking and group dynamics are necessary to understand the complex dynamics that can occur in scientific simulations.

Acknowledgements

This work has been supported in part by the U.S. National Science Foundation through grants OCI 0749227, OCI 0850566, and CCF-0811422, and by the U.S. Department of Energy through the SciDAC program with Agreement No. DE-FG02-09ER25977, DE-FC02-06ER25777 and DE-FC02-12ER26072, program manager Lucy Nowell.

REFERENCES

[1] R. Samtaney, D. Silver, N. Zabusky, and J. Cao, "Visualizing features and tracking their evolution", IEEE Computer, 27(7):20–27, July 1994.

[2] D. Silver and X. Wang, "Volume tracking", In Proceedings of Visualization 1996, 157–164.

[3] G. Ji, H-W. Shen, and R. Wenger, "Volume tracking using higher dimensional isosurfacing", In Proceedings of Visualization 2003, pages 209–216, 2003.

[4] G. Weber, P.-T. Bremer, J. Bell, M. Day, and V. Pascucci, "Feature tracking using Reeb graphs", In Proceedings TopoInVis Workshop, 2009.

[5] C. Muelder and K.-L. Ma, "Interactive feature extraction and tracking by utilizing region coherency", In Proceedings of IEEE Pacific Visualization Symposium, April 2009.

[6] G. Weber, P.-T. Bremer, J. Bell, M. Day, and V. Pascucci, "Feature tracking using Reeb graphs", In Proceedings TopoInVis Workshop, 2009.

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

[8] E. Palsson, H.G. Othmer, "A model for individual and collective cell movement in Dictyostelium discoideum", Proc. Natl. Acad. Sci. USA 97, 10448–10453, 2000.

[9] J. Chen, D. Silver, and L. Jiang, "The Feature Tree: Visualizing Feature Tracking in Distributed AMR Datasets", In Proceedings of the 2003 IEEE Symposium on Parallel and Large-Data Visualization and Graphics (PVG '03), Washington, DC, USA, 2003.

[10] E. LaMar, and V. Pascucci, "A Multi-Layered Image Cache for Scientific Visualization", IEEE Symposium on Parallel and Large-DataVisualization and Graphics (PVG), 2003, pages 61-68.

[11] S. Bhattacharya, S. Habib, K. Heitmann, "Dark Matter Halo profiles of massive clusters: theory vs. observations", Astrophysical Journal, 2011.

[12] M. J. Ringuette, M. Wu and M. P. Martin, "Coherent structures in direct numerical simulation of turbulent boundary layers at Mach 3", J. Fluid Mech. 594, pp. 59-69, 2008.

[13] M. J. Waxman and O. E. Drummond, "A bibliography of cluster (group)tracking," in Proc. SPIE Conf. Signal and Data Processing of Small Targets, vol. 5428, 2004, pp. 551–560.

[14] S. McKenna, S. Jabri, Z. Duric, A. Rosenfeld, and H. Wechsler, "Tracking groups of people.", Comput. Vis. Image Understanding, vol.80, no. 1, pp. 42–56, 2000.

[15] G. Gennari and G. D. Hager, "Probabilistic Data Association Methods in Visual Tracking of Groups.", IEEE CVPR, 2004.

[16] M. Mucientes and W. Burgard, "Multiple hypothesis tracking of clusters of people.", in IEEE/RSJ international conference on intelligent robots and systems, October 2006, pp 692–697.

[17] S-W. Joo and R. Chellappa, "A multiple-hypothesis approach for multiobject visual tracking.", IEEE Trans Image Process 16(11):2849–2854.

[18] B. Lau, K. O. Arras and W. Burgard, "Tracking groups of people with a multi-model hypothesis tracker.", In: International conference on robotics and automation (ICRA), Kobe, Japan.

[19] D. Silver and X. Wang, "Tracking and visualizing turbulent 3d features", IEEE Transactions on Visualization and Computer Graphics 3,2, 129–141, 1997.

[20] D. Silver and X. Wang, "Tracking scalar features in unstructured datasets", In Proceedings of Visualization 1998, pages 79–86, 1998.

[21] J. Chen, D. Silver, and L. Jiang, "The feature tree: Visualizing feature tracking in distributed AMR datasets", In Proceedings of IEEE symposium on Parallel and Large-Data Visualization and Graphics 2003, pages 103–110, 2003.

[22] P. A. Rona, K.G. Bemis, D. Kenchammana-Hosekote, and D. Silver, "Acoustic imaging and visualization of plumes discharging from black smoker vents on the deep seafloor", IEEE Visualization 1998 Proceedings, 1998, pp. 475-478.

[23] F.-Y. Tzeng and K.-L. Ma, "Intelligent feature extraction and tracking for large-scale 4d flow simulations", In Proceedings of Supercomputing 2005 Conference, 2005.

[24] B.S. Sohn and C. Bajaj, "Time-varying contour topology", IEEE Transactions on Visualization and Computer Graphics, 12(1):14–25, 2006.

[25] D. Laney, P.-T. Bremer, A. Mascarenhas, P. Miller, and V. Pascucci, "Understanding the Structure of the Turbulent Mixing Layer in Hydrodynamic Instabilities.", IEEE Transactions on Visualization and Computer Graphics Vol. 12, No. 5, pp. 1053-1060, 2006.

[26] G. Ji and H.-W. Shen, "Feature Tracking Using Earth Mover's Distance and Global Optimization", Pacific Graphics 2006.

[27] J. Caban, A. Joshi, and P. Rheingans, "Texture-based feature tracking for effective time-varying data visualization", IEEE Transactions on Visualization and Computer Graphics, 13(6):1472–1479, 2007.

[28] A. Gezahegne and C. Kamath, "Tracking non-rigid structures in computer simulations", in Proceedings of the IEEE ICIP 2008.

[29] J. Wei, H. Yu, R. W. Grout, J. H. Chen, K. -L. Ma, "Visual Analysis of Particle Behaviors to Understand Combustion Simulations", IEEE Computer Graphics and Applications, 32(1): 22-33, 2012.

[30] J. Wei, H. Yu, J. H. Chen, K. -L. Ma, "Parallel Clustering for Visualizing Large Scientific Line Data", IEEE Symposium on Large Data Analysis and Visualization, 2011.

[31] H. Janicke, M. Bottinger, and G. Scheuermann, "Brushing of Attribute Clouds for the Visualization of Multivariate Data", IEEE Transactions on Visualization and Computer Graphics, 14(6):1459–1466, 2008.

[32] J. Daninels, E. Anderson, L. Nonato, and C. Silva, "Interactive Vector Field Feature Identification", IEEE Transactions on Visualization and Computer Graphics, 16(6):1560–1568, 2010.

[33] A. Jain, M. Murty, and P. Flynn, "Data clustering: A review," ACM Comput. Surv., vol. 31, no. 3, pp. 264–323, 1999.

[34] R. Xu and D.I.I. Wunsch, "Survey of clustering algorithms", IEEE Trans. Neural Networks 16 (3) (2005) 645–678.

[35] Jiawei Han. Data Mining, "Concepts and Techniques", Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2005.

[36] E. Alpaydin, "Introduction to Machine Learning", MIT Press, Cambridge, MA, 2004.

[37] R. Adrian, C. Meinhart, and C. Tomkins, "Vortex organization in the outer region of the turbulent boundary layer", J. Fluid Mech. 422, pp. 1–54, 2000.

[38] S. Ozer, D. Silver, K. Bemis, P. Martin. J. Takle, "Activity Detection for Scientific Visualization", Large Data Analysis and Visualization, IEEE Symposium on, 117-118, 2011 (Posters).