Trajectory-based Flow Feature Tracking in Joint Particle/Volume Datasets

Franz Sauer, Hongfeng Yu, *Member, IEEE*, and Kwan-Liu Ma, *Fellow, IEEE*

Abstract— Studying the dynamic evolution of time-varying volumetric data is essential in countless scientific endeavors. The ability to isolate and track features of interest allows domain scientists to better manage large complex datasets both in terms of visual understanding and computational efficiency. This work presents a new trajectory-based feature tracking technique for use in joint particle/volume datasets. While traditional feature tracking approaches generally require a high temporal resolution, this method utilizes the indexed trajectories of corresponding Lagrangian particle data to efficiently track features over large jumps in time. Such a technique is especially useful for situations where the volume dataset is either temporally sparse or too large to efficiently track a feature through all intermediate timesteps. In addition, this paper presents a few other applications of this approach, such as the ability to efficiently track the internal properties of volumetric features using variables from the particle data. We demonstrate the effectiveness of this technique using real world combustion and atmospheric datasets and compare it to existing tracking methods to justify its advantages and accuracy.

Index Terms—Feature extraction and tracking, particle data, volume data, particle trajectories, flow visualization

1 Introduction

Joint particle and volume data have become increasingly popular, especially in the scientific community. Researchers in numerous fields, such as combustion science [26], plasma physics [23], and atmospheric science [11], deploy large scale simulations to model specific physical processes and obtain a deeper understanding of their area of study. These simulations record results simultaneously in two very different ways: as field data on a spatial grid (volume data) and as discrete particles which are able to move spatially (particle data).

The field data, in the form of either vector or scalar fields, represent the *Eulerian* specification of the system and record information at fixed spatial locations throughout the domain. On the other hand, the particle data represent the *Lagrangian* specification of the system and record information as advected tracer particles. In many cases, the field data are analyzed to extract volumetric features of interest, and the particle data are assembled into a set of time series curves or trajectories. However, the fact that these two data types have such inherently different representations makes it difficult to utilize both forms simultaneously in many analysis techniques.

One such technique, feature extraction and tracking, has become a fundamental necessity for today's scientists. The growing size and complexity of simulations are making it increasingly difficult for scientists to study the full extent of their datasets. Extracting subsets of the data in the form of features allows scientists to easily isolate and study portions of interest without becoming overwhelmed by the sheer size of the dataset. In time-varying data, feature tracking can determine a correspondence between the same feature at different points in time. Not only is it important to be able to extract the same subset of information (feature) over the entire temporal domain, but studying how this subset changes over time can also have a number of interesting implications to domain scientists.

Traditionally, feature tracking techniques use only Eulerian specifications of the data (see Section 2). However, in this work, we present a new feature tracking method which utilizes data from both the Eulerian and Lagrangian specifications. It is based on a set of techniques which can determine a correspondence between volume and particle data, and perform joint-analyses involving both data types. We utilize these methods to determine sets of particles that correspond to our features of interest. We then use the trajectories of these particles to relocate these same volumetric features and/or track their evolution in multiple subsequent timesteps.

There are many advantages to this approach. First, this method works well even when there is a very low temporal resolution in the field data. Since particles are easily distinguishable from one another through some indexing or naming scheme, we can identify our features of interest over very large time intervals without having to access (potentially missing) intermediate data. In other words, we can simply jump to any future timestep in our particle data and re-extract our volumetric feature. This is also useful when working with extremely large datasets which make it computationally difficult to efficiently track a feature though all intermediate timesteps. The ability to skip numerous timesteps in the feature tracking process can lead to large speedups. We also address the issue of uncertainty in the predictions made by this approach since the accuracy of extracted features is likely to decrease over extremely large jumps in time.

In addition, we can use this scheme to study more than just feature movement. In many cases, particle data record a number of variables other than spatial location. Through this correspondence between volumetric features and particles, we can more efficiently track internal changes of the feature based on variables found in the particle data. This is because it is much easier to measure data fluctuations by following a set of corresponding particles rather than extracting a volumetric feature at every point in time.

In this paper we present our new trajectory-based feature tracking technique and make the following contributions:

- We introduce a new feature tracking method that tracks volumetric features using corresponding particle trajectory data.
- We develop uncertainty metrics to quantify the discrepancy between the particle and volume data to reduce errors over large jumps in time.
- We extend this method to efficiently track other non-spatial time-varying properties of a feature of interest using variables in the particle data.

We demonstrate the effectiveness of our approach using real-world combustion and atmospheric datasets, and justify its advantages and accuracy by comparing it to existing methods.

- *Franz Sauer and Kwan-Liu Ma are with the University of California, Davis. E-mail: fasauer@ucdavis.edu, ma@cs.ucdavis.edu.*
- *Hongfeng Yu is with University of Nebraska, Lincoln. E-mail: hfyu@unl.edu.*

Manuscript received 31 Mar. 2014; accepted 1 Aug. 2014. Date of publication 11 Aug. 2014; date of current version 9 Nov. 2014.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org.
Digital Object Identifier 10.1109/TVCG.2014.2346423

2 Related Work

Some work has been done on combining volume and particle data for analytical purposes. Our previous work [17] developed a scalable framework for joint-analyses on large scale particle and volume datasets. While the correspondence between the data types is useful, it lacked a more general description to include analyses on time-varying data. Crossno and Angel [4] utilized particle data to extract isosurfaces in volume datasets. In addition, Krueger [8] utilized sets of virtual particles to interact with and extract extra information from volumetric scalar fields and applied it towards volume rendering techniques.

There has been an extensive amount of work done on feature tracking techniques. While originally designed for computer vision, feature tracking techniques were first applied towards 3D volume data analysis by Samtaney et al. [16], who determined a correspondence across timesteps by comparing features with nearby centroids. Reinders et al. [15] conducted a similar procedure by comparing properties like the mass and size of the features.

A different approach was taken by Silver and Wang [18, 19] who organized features into tree-like structures and then identified connected components. This method works by assuming that a feature will overlap itself over two consecutive timesteps. In addition, Ji et al. [7] introduced a way of tracking features using isosurfaces in a higher dimensional geometry. Tzeng and Ma [22] used a machine learning approach capable of learning information from transfer functions to aid in tracking features. Ozer et al. [13] extended tracking to follow groups of features rather than individual ones as it is more cost effective in large datasets. Takle et al. [20] also tracked groups of cosmological features made up of groups of dark matter particle clouds. Contrary to many previous approaches, which involve extracting features in separate timesteps and then identifying a correspondence, Muelder and Ma [12] developed a method which uses a prediction-correction method. A feature from the previous timestep is used as a prediction and is then refined by region growing and refinement. However, all of these methods still require a high temporal resolution in the data, and in many cases, a spatial overlap between features across consecutive timesteps.

Other recent approaches try to address feature tracking in a number of ways. Caban et al. [1] as well as Glatter et al. [5] used a texture-based tracking approach which looks for similarities between textural characteristics to track patterns in time-varying data. This method works well with tracking noisy volumetric features; however, may not work as well when tracking features resembling solid shapes. Theisel and Seidel [21] developed an approach that constructs streamlines to track critical points in vector fields, and as a result, can track features like saddles or vortices. While effective, this method lacks a more general feature tracking description that can track features in scalar fields. In addition Lee and Shen [10] as well as Gu and Wang [6] utilized Time Activity Curves (TAC) in order to quantify the similarity between the time series of voxels and features for tracking purposes.

There has also been work done by Chen et al. [2] in developing feature extraction and tracking methods in distributed environments, where features can span multiple refinement levels and processors. Also, Wang et al. [24] have developed a scalable parallel extraction and tracking method for use with large scale datasets. Other feature tracking approaches can be found in the survey by Post et al. [14].

3 Approach

As previously described, our approach utilizes both particle and volume data to track features. While feature identification/extraction is done in the volume space, the tracking is done in the particle space. This is extremely advantageous because it is easy to re-identify a set of particles across timesteps by either matching indexes or tracing along their trajectories. The challenge with this approach lies in determining a correct correspondence between these two domain spaces. Our feature tracking method consists of the following general steps: 1) Identify/extract feature of interest; 2) Determine corresponding particles; 3) Trace particles along their trajectory; 4) Re-extract feature based on new particle locations. These steps are illustrated in Figure 1 in their respective domain spaces.

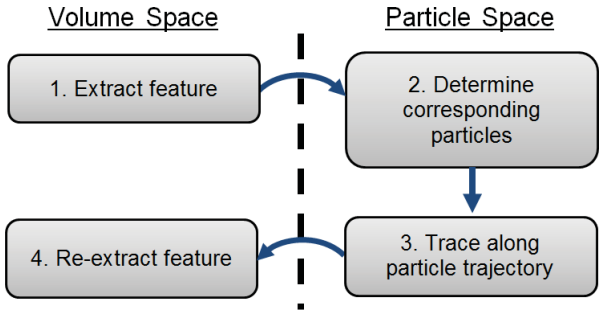

Fig. 1. Trajectory-based feature tracking workflow with each step shown in its respective data space.

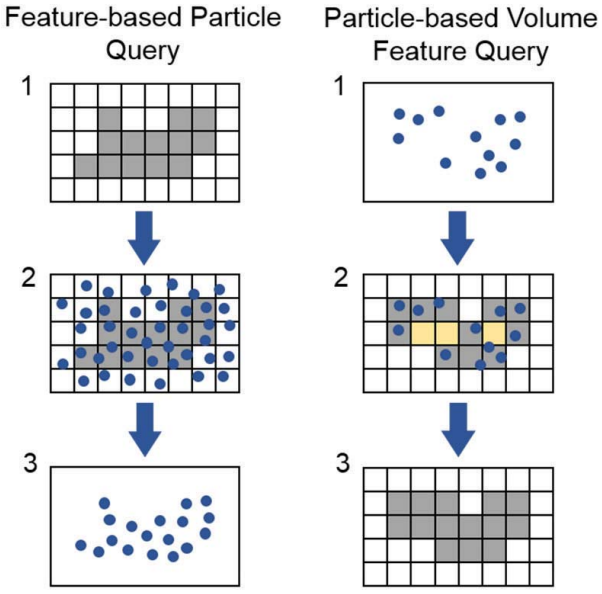

Fig. 2. Feature-based particle query (left): Particles are mapped to the volume space and kept only if that voxel is part of a feature. Particle-based volume feature query (right): Particles are mapped to the volume space signifying that a voxel is part of a region. Region growing then fills in any potential gaps.

3.1 Particle/Volume Data Correspondence

We derive two main tasks to determine a correspondence between particle and volume data. The first is *feature-based particle query*, which allows us to extract a subset of particles corresponding to a volumetric feature. We define the shape of a feature using a 3D bitmask representing voxels in the volume domain. A value of 1 in the bitmask represents a voxel that is part of our feature of interest, while a value of 0 represents a voxel that is not part of our feature. When it is time to query particles, each particle is mapped to a voxel in the volume space based on its location and is checked against the bitmask. If the corresponding voxel is part of our region, the particle is kept; otherwise the particle is discarded. As a result, only a single pass over all particles is required. This allows us to efficiently extract particles within irregular spatial regions as defined by our features of interest.

The second task, *particle-based volume feature query*, represents the opposite task of extracting volumetric features based on a subset of the particle data. Once again, each particle is mapped to a voxel in the volume space based on its location. If one or more particles correspond to a voxel, we set a value of 1 in the bitmask, and set a value of 0 elsewhere. This new bitmask is meant to represent a new volumetric feature. However, gaps may occur in a feature where no particles were found. We therefore use region growing techniques to complete the feature.

We perform seeded region growing in a breadth-first manner in that a queue is maintained for searching and classifying voxels as part of

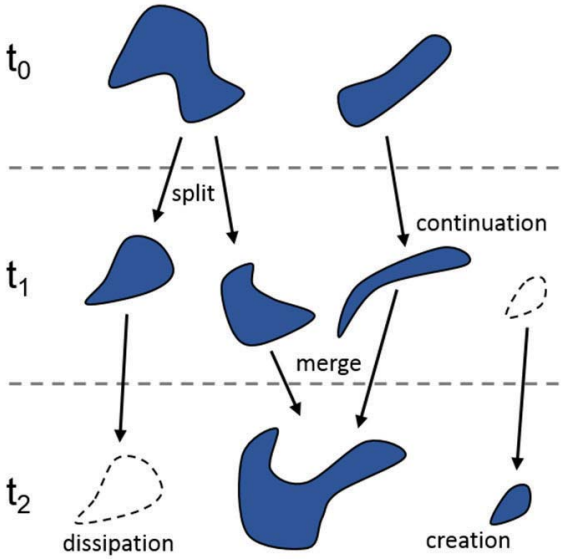

Fig. 3. A visual representation of the different types of feature interactions across three points in time.

our feature. The queue is first initialized using a seed point(s). For each voxel in the queue we check the data values of its neighbors. If its value is within a user defined threshold, the voxel is marked as part of our feature and its neighbors that have not been visited are added to the queue. This process is repeated until the queue is empty. To fill in gaps caused by particle-based volume feature query we simply initialize the queue with voxels that correspond to a value of 1 in our bitmask as seed points. Figure 2 shows a visual representation of feature-based particle query and particle-based volume feature query.

The above description assumes a structured regular grid; however, we can also determine a correspondence between the two data types using unstructured grids as well. If an easy transformation from particle location to volume cell is not available, then a nearest neighbor approach can be used to associate each particle to a particular mesh vertex (or volume cell) in the unstructured scalar field. Mapping particles to the mesh vertex with the smallest Euclidian distance from the particle location will generate the correspondence. In this case, the 3D bitmask would use the same unstructured grid shape that the scalar field uses.

3.2 Trajectory-based Feature Tracking

Traditionally, feature evolution is characterized into a number of interactions: *continuation*, *bifurcation*, *amalgamation*, *creation*, and *dissipation* [16]. Continuation occurs when the same feature exists across multiple timesteps. The feature may change location or shape across this interval but remains as one connected component. Bifurcation (splitting) occurs when a feature separates into two or more sub-features, and amalgamation (merging) occurs when two or more features combine. Lastly, creation and dissipation occur when features appear and disappear between timesteps. These interactions can be seen in Figure 3. In the following subsections, we describe how trajectory-based feature tracking can be used to successfully handle each of these cases.

3.2.1 Feature Continuation

We begin describing trajectory-based feature tracking by example using the simplest case, feature continuation. This allows us to make the assumption that a feature not only exists between two (not necessarily consecutive) timesteps, but also remains as one single connected component. We first identify a feature of interest in the volume data space at an initial timestep t_i. In our implementation, users select features by placing seed points which are used as input to a breadth-first region growing algorithm. Neighboring voxels with values within a certain user defined threshold are then added to the region. Our extracted feature of interest is then defined using a 3D bitmask as described earlier.

Next, we utilize feature-based particle query to extract a unique subset of particle positions p at our original timestep t_i from the set of all particles P. For each particle, $p_k \in p$:

$$p_k(t_i) = (x_k(t_i), y_k(t_i), z_k(t_i)) \quad (1)$$

Because each voxel in the volume space represents a volumetric region, we can map each particle position to a corresponding voxel and then check against the bitmask to determine which particles to extract. Note that this mapping may not be one-to-one as a single voxel could correspond to multiple (or zero) particles. Our extracted subset of particles will then be used to track our feature and relocate it at different timesteps.

We can now immediately identify the evolution (changes in position, shape, etc.) of the feature by relocating it at a later timestep t_{i+n}. Such an operation may be necessary because the volume data may be temporally sparse with all intermediate $n-1$ timesteps unavailable. Another possibility is that the data may be too large and users do not have the computational resources to efficiently track the feature through all intermediate timesteps. Since the particle (trajectory) data are indexed, we can quickly identify the new particle positions $p_k(t_{i+n})$ without accessing any (possibly unavailable) intermediate data.

With our new particle locations, the last step is to reconstruct the feature using the volume data at time t_{i+n} through particle-based volume feature query. We map each particle location $p_k(t_{i+n})$ back to a corresponding voxel to be designated as a seed point for region growing. Any seed points whose data values lie outside of the original threshold used to extract the region at timestep t_i are discarded, because these points represent particles that have fallen outside our region of interest. We also include the option of adding a difference value ($\pm\delta$) to the original threshold. This user defined value is meant to accommodate expected variations in the internal properties of the feature. For example, if we extract a feature based on a specific range of a variable and expect this range to decrease over time in our feature of interest, we can be more lenient about including voxels whose values may be below our original threshold. The remaining seed points are then used in breadth-first region growing as described earlier to fill in any gaps and re-extract the rest of the feature. A visual representation of this procedure can be seen in Figure 4.

Note that this technique also works in reverse, as we can jump to an earlier timestep t_{i-n} in the particle space and then re-locate our feature of interest. This forwards/backwards duality becomes even more important when tracking the other types of feature evolution. However, one challenge of this technique is quantifying and ensuring the accuracy of re-grown features as we need to account for any potential errors or mismatches between the two data spaces, especially over extremely large jumps in time. This is discussed in detail in Section 3.3.

3.2.2 Feature Splitting and Merging

To track splitting and merging in feature evolution, we first observe that these two interactions become identical simply by reversing the direction of time. By traveling backwards in time, a split becomes a merge and vice versa. Therefore, if we are able to identify when splitting occurs, we can use a similar technique when tracking features backwards to identify that a merge has occurred.

To identify a split, we begin by tracking a feature using the same method as the one used for feature continuation. We first identify a feature of interest at timestep t_i, extract corresponding particles, trace them forward in time, and re-extract our feature. The difference is that we end up with two or more separate sub-features after the last region growing phase. We can identify whether our newly extracted feature(s) at timestep t_{i+n} are connected using a standard connected components algorithm [24] on a graph where voxels represent nodes with edges connecting neighboring voxels. If we end up with more than one connected component, then with some likelihood (see Section 3.3), a split must have occurred somewhere between timesteps t_i and t_{i+n}. This can be seen in Figure 5.

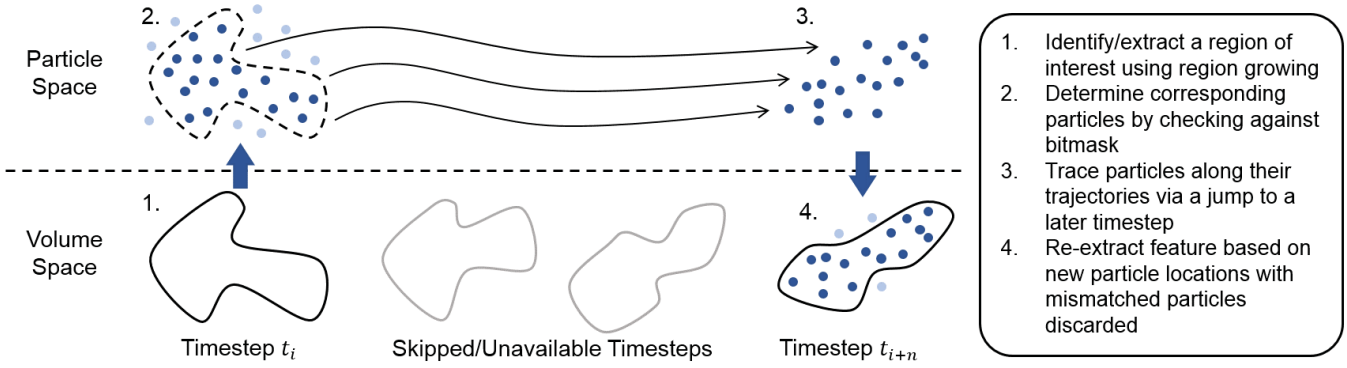

Fig. 4. Steps involved in trajectory-based feature tracking for a feature exhibiting continuation.

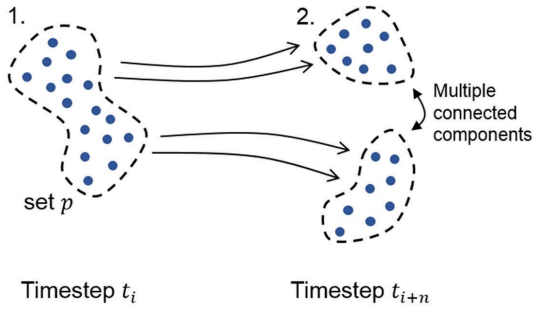

Fig. 5. Identifying a split in feature evolution: 1) Identify feature and extract particles. 2) Trace particles forward and re-extract feature. Then check connectivity to identify and isolate a new set of sub-features.

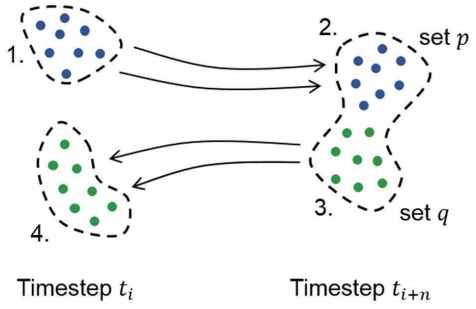

Fig. 6. Identifying a merge in feature evolution: 1) Identify feature and extract particles (p). 2) Trace particles forward and re-extract feature. 3) Identify any additional particles (q) found in the feature. 4) Trace the new set of particles (q) backwards and re-extract feature.

Since users are generally interested in tracking features forward in time, we need an alternate method to determine whether a merge has occurred within our time interval. After re-extracting our feature from the particle locations $p_k(t_{i+n})$ and using region growing to fill in the gaps, we perform an additional feature-based particle query to extract another subset of particle positions q from the set of all particles P, where $q_m(t_{i+n}) \in q$. This excludes the original set p of particles used to track the feature ($p \cap q = 0$, where $p_k, q_m \in P$). This set represents any additional particles that may have entered our feature within the time interval.

Next, we trace the new particle set backwards to our original timestep t_i and identify the original locations of particles $q_m(t_i)$. Through another final particle-based volume feature query and connected components check, we can extract an additional feature(s) at timestep t_i and conclude with some likelihood (see Section 3.3) that these features merged between timesteps t_i and t_{i+n}. This can be seen in Figure 6. We can use the combination of these methods to track cases where both splits and merges occur over our time interval. For example, a feature can split onto one or more sub-features, and one of these sub-features may have undergone a merge. This is achieved by performing a connectivity check as well as a check for additional particles after every feature re-extraction (i.e., particle-based volume feature query).

3.2.3 Feature Creation and Dissipation

In the same way that splitting and merging can be considered the same type of evolution by reversing the direction of time, so can creation and dissipation. Dissipation occurs when the voxels that make up a feature of interest no longer fall within the threshold requirements that define the feature. In our trajectory-based feature tracking approach, we say that a feature has dissipated if we discard all seed points during the feature re-extraction phase. In other words, if particle-based volume feature query results in no available seed points for growing because all corresponding voxel values lie outside of the threshold, then we can say with high likelihood that our feature has dissipated. To identify feature creation, we reverse the direction of time and trace our particle positions to an earlier timestep. If we discard all seed points during the feature re-extraction phase, then we conclude that our feature of interest underwent creation during our time interval.

3.3 Uncertainty Metrics

In this section we discuss uncertainty metrics that can be used to gauge the accuracy of predictions made by trajectory-based feature tracking. While the aforementioned techniques tend be very accurate for reasonable jumps in time, jumping too many timesteps can lead to a build-up in discrepancies between the particle and volume data correspondence (see Section 4.3), and as a result, inaccuracies in feature evolution predictions. There are a number of potential causes for these discrepancies. The first major cause is due to the fact that parts of features (voxels) can pop in and out of existence based on whether its value meets threshold standards, whereas particles often cannot. For example, if parts of our original feature dissipate during our time interval, the particles will continue to evolve and may enter nearby features. This can trigger a false positive, because it can result in the identification of a split, even if the two features have not interacted with one another. Another potential cause is that simulation particles are often massless, whereas our features (e.g. an ash cloud) may not be, resulting in differences in their physical movement. Lastly, computational interpolation errors and inaccuracies in lower order advection schemes can also increase this discrepancy.

The goal of our uncertainty metrics is to quantify the discrepancies between the particle and volume data in an attempt to reduce errors that may arise over large jumps in time. It is based on a few main assumptions: the prediction made by the majority of the particles is most likely the correct one, and if particles truly evolve with the features, then the number of particles that fall outside the feature during re-extraction should be small. However, with the case of dissipation (or features that simply decrease in size), we have to expect that a

certain number of particles may leave the feature.

We can measure the volume V of features in number of voxels and can compute it by summing up all the values in our bitmask,

$$V = \sum bitmask(j), \tag{2}$$

where the index j spans our 3D domain. In addition, we can determine the number of particles originally extracted from our feature at timestep t_i by measuring the size of set p. We define the subset of particles that fall outside our feature at timestep t_{i+n} (i.e., the particles which result in discarded seed points) as p', where $p' \subset p$. We then estimate the discrepancy D between the particle and volume data by checking how much the fraction of discarded particles exceeds the amount of reduction in feature volume:

$$D = \max(\frac{|p'|}{|p|} - \max(1 - \frac{V_{i+n}}{V_i}, 0), 0) \tag{3}$$

This results in a discrepancy value that lies between 0 and 1 with higher values only occurring when both $|p'|$ and V_{i+n} are large (relative to $|p|$ and V_i respectively). By taking the change in size of the feature into consideration, we can handle the case of dissipation ($V_{i+n} = 0$), where we expect all particles to be discarded ($|p'| = |p|$), as well as features that simply reduce in size.

Because we use the particle positions as seed points into region growing, only one particle must remain inside our feature in order to re-extract it in its entirety. Moreover, in the case where splitting occurs, we only require one particle (seed point) per sub-feature. Therefore, even if the discrepancy between the two data types is high, we can fully re-extract our feature with high likelihood. The errors that can arise however, lie in the form of false positives. In other words, when the discrepancy is high, there is a greater chance that a particle wanders into a nearby feature, falsely extracts it, and identifies it as something that has split off of our original feature.

We utilize the discrepancy measure in an attempt to mitigate the possibility of false positives. If a split is identified, we assume that the sub-feature that contains the majority of the particles (the *main sub-feature*) is correct and evaluate the possibility that other sub-features may be false positives. We compute the ratio of the number of particles in each sub-feature to the number of particles in the main sub-feature, and then compare this value to the calculated discrepancy. If this ratio is smaller than the discrepancy we ignore this sub-feature in our final extraction. For example, if the computed discrepancy is 0.33, then we require that each sub-feature has at least 1/3 of the number of particles that the main sub-feature has, otherwise it is discarded. This allows us to potentially ignore extracted sub-features generated from small groups of stray particles. Note that this comparison can be tweaked to form a stricter or more lenient extraction criteria.

4 Results

We demonstrate the effectiveness of this feature tracking technique using two real world datasets. The first is a large-scale combustion dataset that comes from S3D, a massively parallel simulation code developed by scientists at Sandia National Labs [26]. This peta-scale simulation records information as both particle and volume data simultaneously. The second is an atmospheric dataset that comes from two sources. The "volume data", in this case 2D satellite detections, comes from the Atmospheric Infrared Sounder developed by NASA's Jet Propulsion Laboratory [9]. The particle trajectory data representing atmospheric flow comes from Chemical Lagrangian Model of the Stratosphere, a simulation developed by Research Center Jülich [11].

4.1 Combustion Dataset

The combustion dataset represents a 3D lifted ethylene jet flame. Its highly-connected turbulent nature makes feature tracking difficult because inter-feature evolution events (e.g. splitting and merging) are more likely to occur. This particular dataset has sufficient spatial and temporal resolution to compare the results of our trajectory-based feature tracking approach with traditional approaches. The volume dataset ($\sim 10^8$ voxels per timestep) represents a series of 27 different flow classifications determined by computing a local rate-of-deformation tensor from the underlying vector field [3]. A depiction of the two most prominent classifications, FC/U (focusing compressing unstable) and FS/S (focusing stretching stable) can be seen in Figure 7. Scientists are especially interested in these features as they are presumed to have a strong influence on flamelet deformation. The particle dataset ($\sim 10^6$ particles per timestep) consists of massless particles which, in addition to spatial location, records other parameters such as temperature and the mass fractions of molecules that make up the jet. A representative subset of the particle dataset can be seen in Figure 8. The volume dataset in this case is temporally sparser than the particle dataset with one timestep of volume data for every five timesteps of particle data.

Since the features we aim to track are highly connected, simply

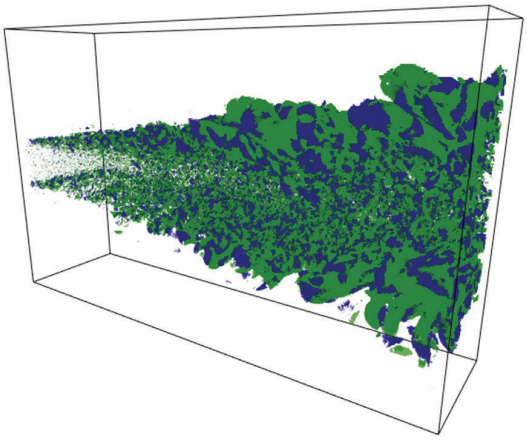

Fig. 7. A depiction of the FC/U (shown in blue) and FS/S (shown in green) flow classifications.

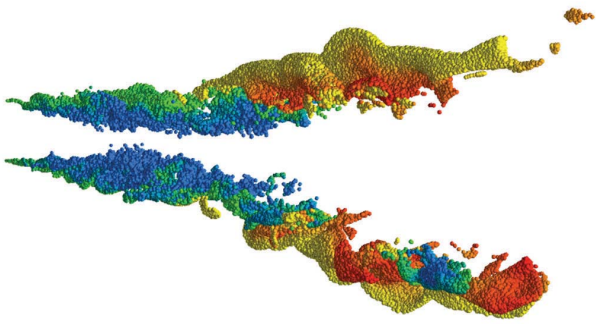

Fig. 8. A subset of the corresponding particle data colored according to temperature. Red indicates hotter portions of the jet while blue indicates colder portions.

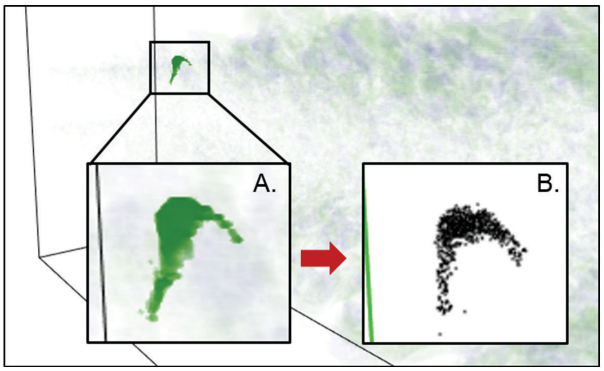

Fig. 9. An extracted volumetric feature of interest. A zoomed-in view (A) as well as the corresponding extracted particles (B) are shown.

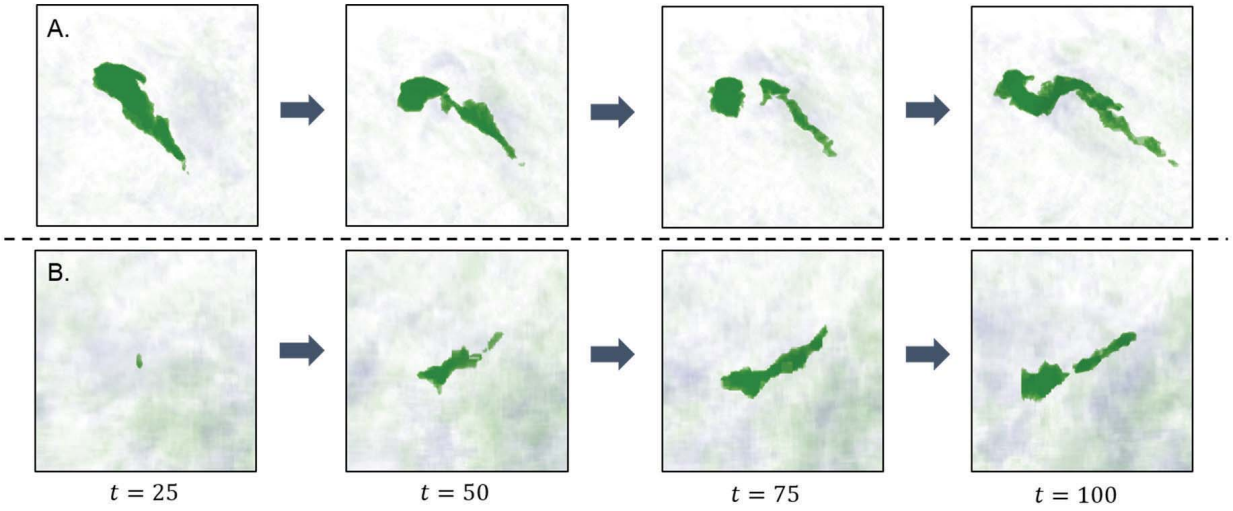

Fig. 10. Tracking a feature that exhibits splitting and merging (A) and a feature that exhibits creation and splitting (B).

growing based on connectivity will result in a large single feature that spans the entire domain. Instead, we use a modified version of region growing to "pinch off" weakly connected portions [17]. This is done by measuring the connectivity strength of voxels by counting the number of similar neighbors. Voxels are then only added to the region if their connectivity strength exceeds a threshold. This modified region growing is used both for the initial feature extraction and re-extraction phases in this particular dataset. An example of an extracted feature of interest as well as the corresponding particles can be seen in Figure 9.

We demonstrate the ability of this technique to track different types of feature evolution in Figure 10. Features and corresponding particles were initially extracted at time $t = 25$. Features were then re-extracted at time $t = 50$, 75, and 100 by jumping 25, 50, and 75 timesteps from the initial extraction timestep respectively. Feature A shows an example of splitting and merging as it splits into two sub-features between $t = 50$ and 75 which merge back together between $t = 75$ and 100. Feature B shows an example of creation and splitting. This feature was created between $t = 0$ and 25 since tracing the particles backwards to $t = 0$ results in no available seed points for growing. The splitting then occurs between $t = 75$ and 100. This coarse method of re-extracting the feature at incremental timesteps is a useful way for users to quickly examine the general evolution of a feature. Users can then examine the evolution in more detail by inspecting the intermediate timesteps (if available).

We also demonstrate the ability of using the particle data to track the internal properties of a feature. In this case we examine the average temperature of a feature of interest by averaging the temperature values of all corresponding particles. We discard any particles that fall outside the feature when computing the average. The results can be seen in Figure 11, which displays the average temperature for the two features shown in Figure 10. From the graph we can see that the temperature of Feature A remains relatively constant, while the temperature of Feature B steadily increases. This indicates that Feature B is entering a portion of the jet where burning is occurring, whereas Feature A is not.

In addition, we compare this method to an existing feature tracking approach to justify its accuracy. In other words, we show that the predictions made by our trajectory-based approach can extract features similar to those generated by an approach that tracks features through all intermediate timesteps. In this comparison, we implement a feature tracking approach similar to the one discussed by Muelder et al. [12]. This is a predictor-corrector method which uses region growing and refinement to track features that physically overlap between consecutive timesteps. The identified feature from a previous timestep is used as a prediction. It is then corrected using region growing and refinement to identify the feature in the current timestep. We choose this method as a comparison as it is one of the more efficient available options for feature tracking.

Figure 12 shows a comparison between the resulting features that are extracted using the trajectory-based method and the predictor-corrector method. In this example, the trajectory-based method extracts particles at $t = 0$ and then jumps directly to $t = 150$, while

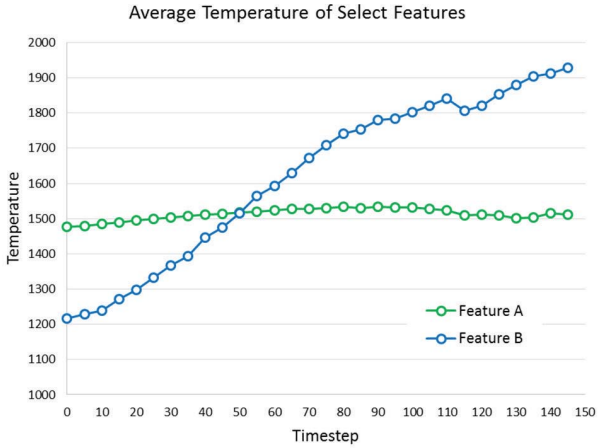

Fig. 11. A graph showing the average temperature of the two features depicted in Figure 10.

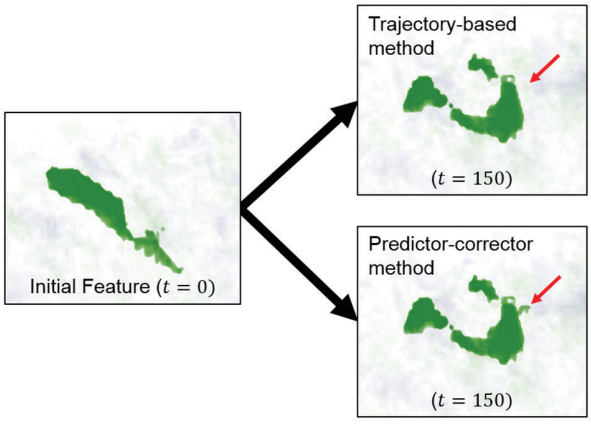

Fig. 12. A comparison between the trajectory-based approach and the predictor-corrector approach. Only subtle differences (approximately 1% of the feature) can be seen at the location of the red arrow.

the predictor-corrector method steps through all available intermediate timesteps. The figure shows that the end results for each method are very similar, although some minor differences can be seen at the location of the red arrow. This difference can be attributed to the lack of available particles (seed points) to extract that sub-feature. A direct voxel comparison between these results via a Jaccard Index indicates that the trajectory-based method extracts a feature that is 98.9% similar to the feature determined by the predictor-corrector method.

In addition, timing results on a desktop computer (using a 3.20 GHz Intel Core i7 processor) show that the trajectory-based method took a total of 59 ms, while the predictor-corrector method took a total of 541 ms, almost a 10x speedup. Admittedly, this comparison can be considered slightly biased as the trajectory-based approach has access to extra information (the particle data). However, it demonstrates the value of our approach in being able to greatly reduce the amount of computation that is generally required to track features. Also, note that the speedup will be heavily dependent on the number of timesteps that are skipped. Of the 59 ms, 20 ms are used to build the correspondence between the feature and the particle data. This time scales with the number of particles in the dataset as it takes a single pass over all particles to build the correspondence.

4.2 Atmospheric Dataset

The atmospheric dataset represents a 2D global coverage of volcanic ash detections using infrared sensing. Unlike the combustion dataset, the atmospheric dataset is extremely temporally and spatially sparse, and is much smaller in size. This sparseness is inherent in the acquisition method used to collect the data because a satellite can only take measurements from one location at one point in time. Moreover, the location of each detection is limited by the orbital path of the satellite. An image of such satellite detections can be seen in Figure 13. Gray areas in the image represent gaps in the data where no measurements were taken. Datasets like these make feature tracking extremely difficult (and in many cases impossible) using traditional techniques since features can disappear entirely when traversing these missing regions. Our trajectory-based approach overcomes this difficulty by utilizing corresponding particle data to track features. In this case, we can utilize particle trajectories from a corresponding atmospheric simulation to identify the evolution of our features of interest. The particle dataset also records additional information, such as the temperature and pressure at that particular location. An example of the corresponding particle data can be seen in Figure 13. Just like with the combustion dataset, the satellite detections are sparser than the particle data with approximately one timestep of image data for every twelve timesteps of particle data.

In this example, we track an ash cloud produced by the Puyehue-Cordón Caulle Volcano Complex in Chile, which erupted in June 2011. We identify and extract a feature of interest located near the eruption site and track its evolution for four consecutive days using the particles extracted at the first timestep. This can be seen in Figure 14. From the images, we can see that we can track our feature of interest (shown in blue) even though it traverses several patches where no data is available (shown in gray). This particular ash cloud travels east over the South Atlantic Ocean and begins to dissipate just before reaching the coast of Africa. Its trajectory can be seen in the top right portion of the figure.

We can also use the additional data from the simulation to estimate the internal properties of our feature. In this example, we track the temperature and pressure of our feature of interest over the four day period by averaging the values of each of the corresponding particles.

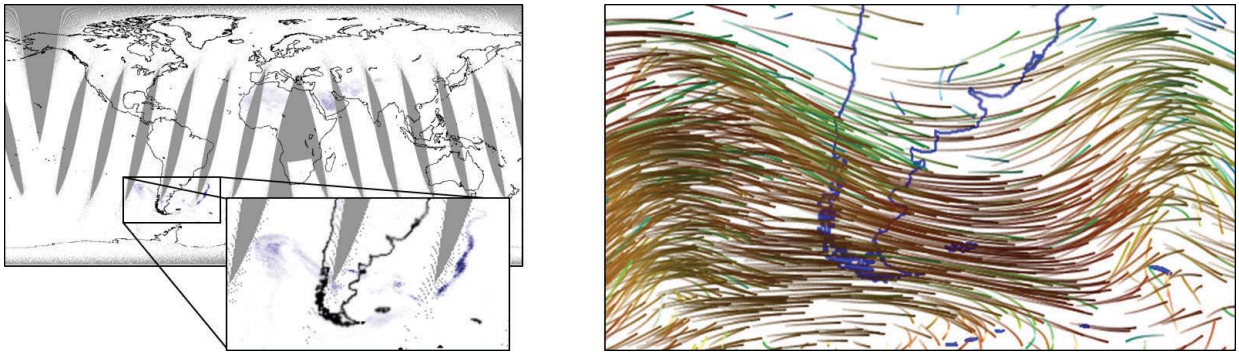

Fig. 13. Left: An image of satellite detections showing a sparse coverage over a twelve hour time window. Gray areas represent gaps in the data. Volcanic ash detections can be seen in blue. Right: An image of the particle data drawn as trajectories. The color corresponds to temperature.

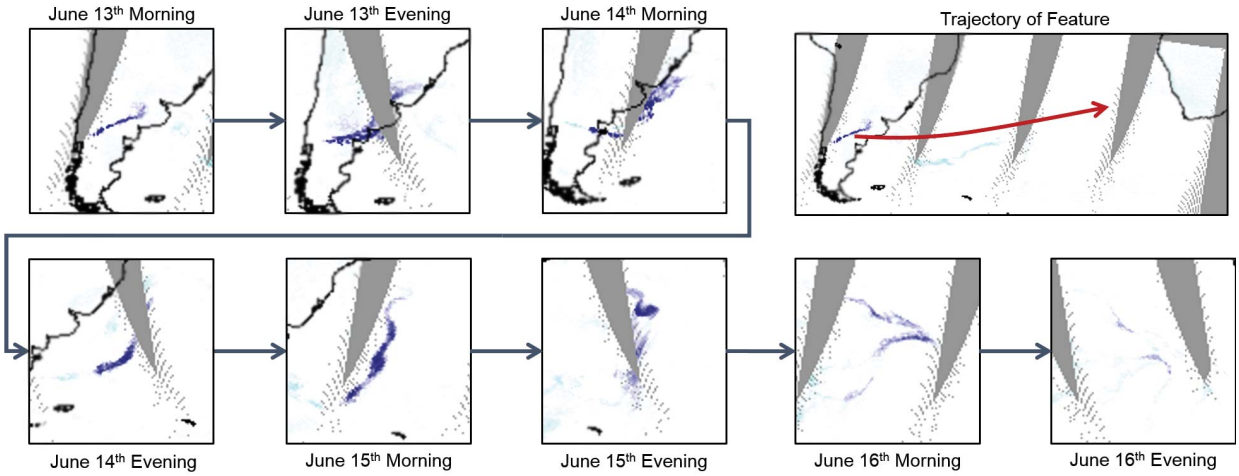

Fig. 14. Tracking an ash cloud from a Chilean volcano over the course of four days. The feature travels east over the South Atlantic Ocean and begins to dissipate just before reaching the coast of Africa. Its trajectory can be seen at the top-right.

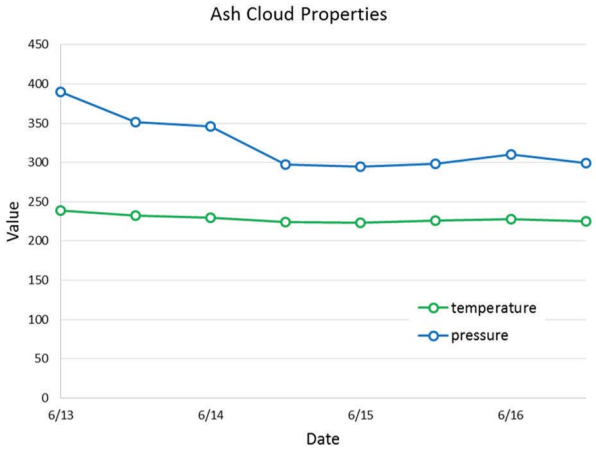

Fig. 15. A graph showing the average temperature and pressure of the ash cloud depicted in Figure 14.

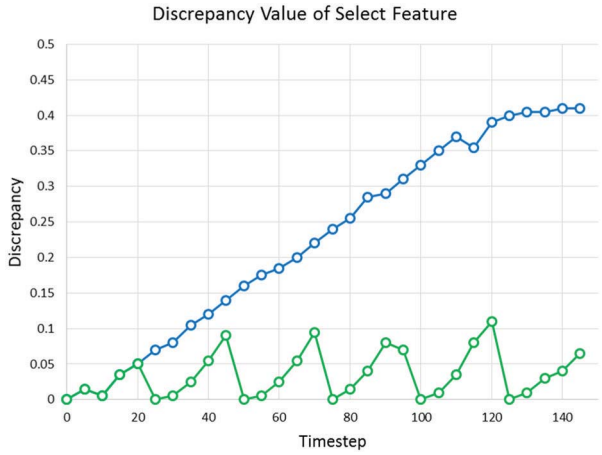

Fig. 16. A graph showing the discrepancy value of an example feature in the combustion dataset. Blue: particles are initially identified at timestep 0 and used to extract the feature at all subsequent timesteps. Green: particles are re-identified every 25 timesteps to keep the discrepancy value low.

The results can be seen in Figure 15. We can see that the temperature of the ash cloud remains relatively constant over time, which indicates that the ash has likely already cooled from the initial eruption event. However, the pressure tends to decrease over our time frame as the ash cloud diffuses into the atmosphere and dissipates. This is a useful example of how this feature tracking approach can incorporate data from different sources into a combined analytical result.

4.3 Discussion

The above results demonstrate the effectiveness of this trajectory-based method in being able to track the evolution of features. There are two major advantages to this approach as reflected in each of the tested datasets. With high resolution datasets like the combustion dataset we can achieve feature tracking at a fraction of the computation normally required. The trajectory-based approach can skip intermediate timesteps because the tracking itself is done using the indexed particle data. This is especially useful towards big data applications where computational resources and I/O play a major role. The trajectory-based approach allows users to identify the evolution of a feature of interest while only accessing two timesteps of the data: the start and end step of a particular time window. This drastically reduces the amount of computation and I/O normally required. The above results show that comparing the trajectory-based approach to a traditional feature tracking method results in the extraction of extremely similar features, even over a large jump in time. This approach allows users to choose their desired balance of accuracy and performance by adjusting the number of skipped timesteps.

With low resolution datasets like the atmospheric dataset we can not only easily track features without a physical overlap across timesteps, but can also track features that traverse regions with missing data. By using the particle data for tracking, users can jump to a later timestep after features have emerged from the missing regions and re-extract the feature. Lastly, this trajectory based approach can be applied towards the fusion of information from multiple data sources. In many cases, the particle data contain additional information that is not present in the volume data. By extracting particles that correspond to a feature of interest, we can measure additional internal properties over time, such as the temperature or pressure of the feature.

4.3.1 Discrepancy Measure

With these numerous advantages comes the cost of accuracy. Since this approach uses region growing to re-extract the feature, we simply need at least one particle present in the feature (or one per sub-feature) in order to re-extract it. This makes the likelihood of not being able to re-extract the feature small. However, a build-up of discrepancies between the particle and volume data can potentially trigger false positives in which stray particles wander into neighboring features. We mitigate this using the discrepancy metric as described in Section 3.3.

In addition, users can choose to keep the discrepancy low by continually re-extracting particles every few timesteps rather than using the initial set of extracted particles for all subsequent tracking. Figure 16 shows a graph of the computed discrepancy value for an example feature in the combustion dataset. The points in blue show the case where particles are only identified at timestep 0 and then used to re-extract the feature at all subsequent timesteps. The points in green show the case where particles are re-identified every 25 timesteps in order to "re-synchronize" the particles to our newly identified feature. In other words, we discard any particles that have wandered outside of the feature and include any particles that have wandered into the feature in any subsequent tracking. We can see that the discrepancy value remains low in this case.

Note that the computed discrepancy does not directly reflect the error of the resulting feature. Instead it is a measure of the possibility of false positives. In the case where splitting occurs, we utilize this value to reduce the possibility of falsely extracting sub-features as described in Section 3.3. Even if the discrepancy value is high (e.g. 0.4 after a jump of 150 timesteps as seen in Figure 16), we still can extract features that are very similar to those extracted by traditional feature tracking methods. In the example shown in Figure 12, the extracted feature is 98.9% similar to the feature identified by the traditional approach even though the discrepancy value is above 0.5. While keeping the discrepancy low by continually re-identifying particles might lead to more accurate results, in many cases this is not necessary.

As the discrepancy value can be thought of as a measure of the possibility of a false positive, it can be used to give direct feedback to users choosing an appropriate balance between computation time (number of skipped timesteps) and accuracy. Each feature being tracked has a discrete discrepancy value which can be visually encoded into the visualization (by using labels, coloring the feature, etc.). In other words, the uncertainty of the extraction result can be displayed with each tracked feature. If this discrepancy value becomes too high for select features of interest, users can then choose to skip fewer timesteps and obtain a more accurate result.

4.3.2 Particle Density

This method is also dependent on the density of available particles. Since this technique uses region growing to re-extract features, we only require one particle to remain in each feature of interest (or sub-feature if splitting occurred). However, extreme drops in particle density could prevent this requirement from being met. For example, this method would not be able to detect a split in feature evolution if only one corresponding particle is associated with a feature of interest. While this extreme case is highly unlikely for larger features, it

can pose a problem for smaller ones.

We study how varying the particle density in the combustion dataset affects the accuracy of the feature tracking result for two select features: a large features containing ~ 500 voxels and a small feature containing ~ 10 voxels. We artificially reduce the density of the dataset by randomly removing an increasing number of particles. Each feature was tracked by jumping 150 timesteps using the trajectory-based approach for a number of different densities. Each feature was then compared to the result from the predictor-corrector method (which disregards the particle data) and a similarity measure was computed via the Jaccard Index. The results can be seen in Figure 17. Both the large feature (shown in blue) and small feature (shown in green) retain a high accuracy (similarity) when the particle density is high. We see a sharp decline in the accuracy of the large feature when the particle density reaches 1/16th of maximum available density. This is because there are no longer any particles associated with a major sub-feature that has split off during the evolution. At 1/256th of maximum available density, the large feature is lost entirely. As predicted, the small feature is more sensitive to decreases in density and is lost entirely at 1/4th of the maximum available density.

In some cases, simulations insert/remove particles to accommodate changes in density in certain portions of the domain. Since this approach compares the particle locations at two distinct timesteps, any particles that do not explicitly exist at both of these points in time (any particles that were inserted or removed during skipped timesteps) need to be ignored. This can be done by simply implementing an extra step which removes such particles from the extracted subset whenever this approach jumps forwards or backwards in time. Not doing so could potentially alter the discrepancy measure and/or lead to false predictions.

4.3.3 Applicable Data and Feature Types

While the approach as described in this paper focuses on tracking the evolution of a single starting feature, tracking more complex targets such as multiple features as once is certainly viable. In fact, this is almost identical to tracking a group of sub-features after splitting has occurred. Many of the example features depicted in this paper split into several sub-components throughout their evolution and provide evidence that tracking multiple features simultaneously is possible. Simply assigning a distinct label or index to the group of particles associated with each feature can allow the system to re-identify each feature after jumping to a later timestep. However, as seen in the previous section, this approach is still limited in being able to track very small features as they can easily be lost when not enough particles can be associated with them.

The disadvantage of this method mainly lies in the limitation of applicable dataset types. The dataset must have some form of corresponding particle (trajectory) data, and the movement of the features we wish to track must be governed in a way similar to that of the particles (i.e., advection by a flow). While these two criteria are often met in many scientific endeavors, this may not always be the case, requiring the use of traditional feature tracking approaches. However, it is important to note that if vector flow field data are present, we can choose to implement our own advection scheme to generate artificial trajectories. These trajectories can then be used as input into our feature tracking approach exactly as described in this paper.

5 Conclusion and Future Work

This work presents a new trajectory-based feature tracking approach which uses corresponding particle data to track volumetric features. By determining a correspondence between features and sets of particles, this approach can utilize indexed particle data to jump to later timesteps and re-identify features of interest. Eliminating the need to track features through intermediate timesteps is extremely advantageous. First, this drastically reduces the amount of computation and I/O time required for feature tracking, which plays a major role when analyzing large datasets. Second, this allows our approach to be able to track features in datasets which are both spatially and temporally sparse. Moreover, since the tracking itself is done using the particle data, this method can also track features across regions where volume data may be missing. Our technique also compares favorably with traditional feature tracking methods in terms of accuracy. Results show that we can extract features that are similar to those obtained by a traditional method, and at a fraction of the computation cost. Our approach provides the groundwork for accurate and more efficient feature tracking in a wide variety of datasets containing both particle and volume data.

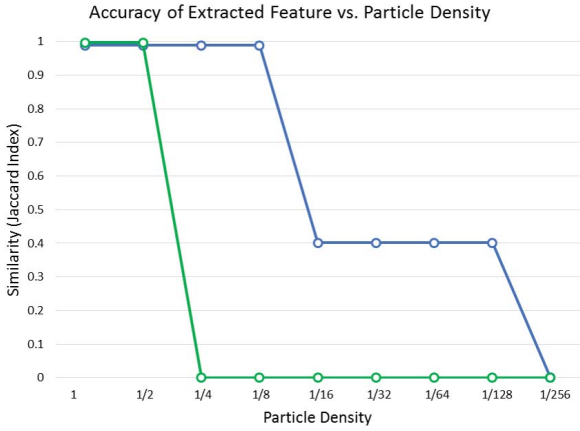

Fig. 17. A graph showing the accuracy of tracking a large feature (blue) and small feature (green) as a function of varying particle density. The accuracy is computed by comparing the result to the predictor-corrector method.

In the future, we plan to extend this technique to be able to interpolate features between consecutive timesteps or within missing regions by using the higher resolution particle data. This can be achieved by using the characteristics of the particle based point-cloud to estimate the size, shape, and location of the feature in these unknown regions. In addition, we plan to integrate this tracking approach with trajectory clustering techniques [25]. By clustering the trajectories of corresponding particles, we can also easily cluster the individual features into groups based on the similarity of their evolution throughout the dataset. This can also be extended to allow users to query features based on specific trajectories.

Acknowledgments

This research has been sponsored in part by the National Science Foundation through grants IIS-1320229, CCF-0938114, CCF-1025269, and IIS-1255237, and Department of Energy through grants DE-FC02-06ER25777, DE-CS0005334, and DE-FC02-12ER26072.

References

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

[2] J. Chen, D. Silver, and L. Jiang. The feature tree: visualizing feature tracking in distributed amr datasets. In *Parallel and Large-Data Visualization and Graphics, 2003. PVG 2003. IEEE Symposium on*, pages 103–110, Oct 2003.

[3] M. S. Chong, A. E. Perry, and B. J. Cantwell. A general classification of threedimensional flow fields. *Physics of Fluids A: Fluid Dynamics (1989-1993)*, 2(5), 1990.

[4] P. Crossno and E. Angel. Isosurface extraction using particle systems. In *Visualization '97., Proceedings*, pages 495–498, Oct 1997.

[5] M. Glatter, J. Huang, S. Ahern, J. Daniel, and A. Lu. Visualizing temporal patterns in large multivariate data using modified globbing. *Visualization and Computer Graphics, IEEE Transactions on*, 14(6):1467–1474, Nov 2008.

[6] Y. Gu and C. Wang. itree: Exploring time-varying data using indexable tree. In *Visualization Symposium (PacificVis), 2013 IEEE Pacific*, pages 137–144, Feb 2013.

[7] G. Ji, H.-W. Shen, and R. Wenger. Volume tracking using higher dimensional isosurfacing. In *Visualization, 2003. VIS 2003. IEEE*, pages 209–216, Oct 2003.

[8] W. Krueger. The application of transport theory to visualization of 3d scalar data fields. In *Visualization, 1990. Visualization '90., Proceedings of the First IEEE Conference on*, pages 273–280, 481–2, Oct 1990.

[9] N. J. P. Laboratory. Atmospheric infrared sounder. http://airs.jpl.nasa.gov/.

[10] T.-Y. Lee and H.-W. Shen. Visualizing time-varying features with tac-based distance fields. In *Visualization Symposium, 2009. PacificVis '09. IEEE Pacific*, pages 1–8, April 2009.

[11] D. S. McKenna, P. Konopka, J.-U. Grooß, G. Günther, R. Müller, R. Spang, D. Offermann, and Y. Orsolini. A new chemical lagrangian model of the stratosphere (clams) 1. formulation of advection and mixing. *Journal of Geophysical Research: Atmospheres*, 107(D16):ACH 15–1–ACH 15–15, 2002.

[12] C. Muelder and K.-L. Ma. Interactive feature extraction and tracking by utilizing region coherency. In *Visualization Symposium, 2009. PacificVis '09. IEEE Pacific*, pages 17–24, April 2009.

[13] S. Ozer, J. Wei, D. Silver, K.-L. Ma, and P. Martin. Group dynamics in scientific visualization. In *Large Data Analysis and Visualization (LDAV), 2012 IEEE Symposium on*, pages 97–104, Oct 2012.

[14] F. H. Post, B. Vrolijk, H. Hauser, R. S. Laramee, and H. Doleisch. The state of the art in flow visualisation: Feature extraction and tracking. *Computer Graphics Forum*, 22(4):775–792, 2003.

[15] F. Reinders, F. Post, and H. Spoelder. Attribute-based feature tracking. In *Data Visualization'99*, Eurographics, pages 63–72. Springer Vienna, 1999.

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

[17] F. Sauer, H. Yu, and K.-L. Ma. An analytical framework for particle and volume data of large-scale combustion simulations. In *Proceedings of the 8th International Workshop on Ultrascale Visualization*, UltraVis '13, pages 1:1–1:8, New York, NY, USA, 2013. ACM.

[18] D. Silver and X. Wang. Volume tracking. In *In Proceedings of the Visualization 96 Conference*, pages 157–164. Computer Society Press, 1996.

[19] D. Silver and X. Wang. Tracking scalar features in unstructured data sets. In *Visualization '98. Proceedings*, pages 79–86, Oct 1998.

[20] J. Takle, D. Silver, and K. Heitmann. A case study: Tracking and visualizing the evolution of dark matter halos and groups of satellite halos in cosmology simulations. In *Visual Analytics Science and Technology (VAST), 2012 IEEE Conference on*, pages 243–244, Oct 2012.

[21] H. Theisel and H.-P. Seidel. Feature flow fields. In *Proceedings of the Symposium on Data Visualisation 2003*, VISSYM '03, pages 141–148, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association.

[22] F.-Y. Tzeng and K.-L. Ma. Intelligent feature extraction and tracking for visualizing large-scale 4d flow simulations. In *Supercomputing, 2005. Proceedings of the ACM/IEEE SC 2005 Conference*, pages 6–6, Nov 2005.

[23] W. X. Wang, Z. Lin, W. M. Tang, W. W. Lee, S. Ethier, J. L. V. Lewandowski, G. Rewoldt, T. S. Hahm, and J. Manickam. Gyro-kinetic simulation of global turbulent transport properties in tokamak experiments. *Physics of Plasmas (1994-present)*, 13(9), 2006.

[24] Y. Wang, H. Yu, and K.-L. Ma. Scalable parallel feature extraction and tracking for large time-varying 3d volume data. In *Proceedings of the 13th Eurographics Symposium on Parallel Graphics and Visualization*, EGPGV '13, pages 17–24, Aire-la-Ville, Switzerland, Switzerland, 2013. Eurographics Association.

[25] J. Wei, H. Yu, J. H. Chen, and K.-L. Ma. Parallel clustering for visualizing large scientific line data. In *In Proceedings of IEEE Symposium on Large Data Analysis and Visualization*, October 2011.

[26] C. S. Yoo, E. S. Richardson, R. Sankaran, and J. H. Chen. A DNS study on the stabilization mechanism of a turbulent lifted ethylene jet flame in highly-heated coflow. *Proceedings of the Combustion Institute*, 33(1):1619 – 1627, 2011.