

3D flow features visualization via fuzzy clustering

Huaxun Xu · Zhi-Quan Cheng · Ralph R. Martin ·
Sikun Li

Published online: 19 April 2011
© Springer-Verlag 2011

Abstract A key approach to visualizing a flow field is to emphasize regions with significant behavior. However, it is difficult to give concrete criteria for classifying feature regions. In this paper, we use a novel framework in which fuzzy sets are used to determine flow features: Fuzzy relationships assess structural properties of features. A fuzzy *c*-means-like clustering algorithm is used to evaluate the importance of each voxel. Our approach can be readily modified with new fuzzy relationships describing other features of interest to users. We use a multi-resolution approach which displays structural features in greater detail, and represents the background by coarse-grained information. Experiments on synthetic and real datasets show that our framework can highlight significant aspects of the whole flow while avoiding occlusion and clutter. Interactive performance is achieved via a GPU implementation.

Keywords Feature visualization · 3D flow features · Fuzzy clustering · GPU

H. Xu
Army Aviation Institute of PLA, Beijing, China
e-mail: xhxnudt@gmail.com

H. Xu · S. Li
Computer School, NUDT University, Changsha, China

S. Li
e-mail: lisikun@263.net

Z.-Q. Cheng (✉)
PDL Lab, Computer School, NUDT University, Changsha, China
e-mail: cheng.zhiqian@gmail.com

R.R. Martin
School of Computer Science and Informatics, Cardiff University,
Cardiff, UK
e-mail: ralph@cs.cf.ac.uk

1 Introduction

Visualization of 3D flow fields is a large-scale and complex problem. The aim is to highlight features of interest, characterized by the positions, velocities, and other properties of points within them. Feature extraction methods can be based on criteria like critical points, streamlines, and vortex core regions [5, 6]; other properties like pressure and temperature may also be of interest. Once features have been defined and extracted, a further challenge is how to provide simultaneous visualization of multiple features, giving a structural overview yet avoiding occlusion and clutter. One approach is to use a high level of abstraction to represent the data, such as the use of flow topology methods [4] which extract critical points. However, such methods are typically limited to considering singularities of the flow field.

An alternative is to partition the flow field based on user-defined properties, and to only display flow structures in regions that meet certain conditions. Explicitly partitioning the flow field according to flow behavior [14] is a directive, intuitive approach. Such partitioning techniques extract and visualize only a small number of regions of interest, leading them to be popular. They have also been further developed to represent complex vector field structures. However, the assumption of simple criteria which can give explicit, clear feature partitioning is unrealistic, as there are rarely sharp boundaries between different features in flow fields. Various important features such as vortices and singularities (see Fig. 1(left)) contain such complex mixing that explicit fixed criteria cannot readily determine them. In the worst case, some complex feature structures do not have any precise definition at all.

To avoid these difficulties, we give an approach for visualizing 3D flow features guided by fuzzy set theory, rather than explicit partitioning criteria. Our framework allows the

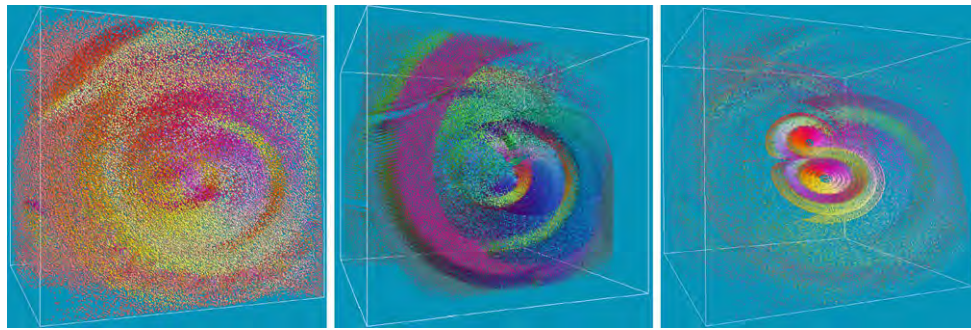


Fig. 1 Visualizing a Lorenz attractor. *Left*: the Lorenz attractor is hard to see when visualized with a particle system, due to its complicated embedding within the data. *Center*: result provided by λ_2 method [5]

for comparison. *Right*: Occlusion and clutter are effectively suppressed by our framework, allowing the Lorenz attractor to be clearly visualized

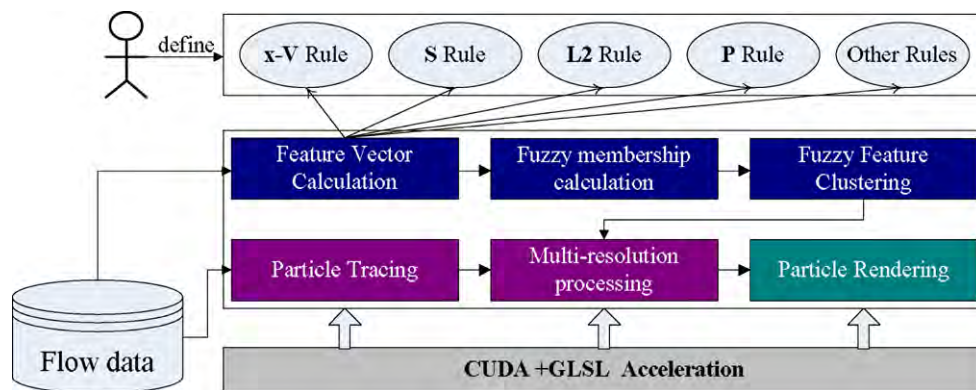


Fig. 2 Overview. Fuzzy feature membership is pre-calculated for each region. Random particles are seeded and traced in the flow fields. Multi-resolution rendering displays particles at different densities based on the fuzzy membership field

user to define and combine multiple feature criteria as fuzzy relationships, and to process and display the desired feature structures using a multi-resolution technique based on a particle system. Our framework can easily be specialised to a particular problem of interest by adding new fuzzy relationships that describe properties of special features, such as critical points, vortices, and typhoons.

The workflow of our approach is illustrated in Fig. 2. We use a particle system [3], in which particles follow the streamlines, to visualize 3D flow features. We first use a fuzzy clustering method to pre-compute fuzzy membership of voxels to features of interest based on user-defined feature properties. Tracking particles not only allows us to visualize flow movements, but also emphasize feature structures by assigning greater opacity to particles in regions with high membership. A multi-resolution technique is used to process the whole flow, allowing greater detail in areas on interest. We use GPU techniques to accelerate the fuzzy membership computation, the particle tracing, and the multi-resolution processing, while the rendering algorithm takes advantage of the GPU shading language. This allows complex 3D flow fields to be explored at interactive rates. Computation of fuzzy membership is time-consuming. We implement it as a

pre-processing step, decoupling it from the rendering stage. This frees up the GPU for rendering during the output stage.

In the following, we concentrate on the novel fuzzy-related aspects of our system. Fuzzy representations of 3D flow features are explained in Sect. 3. Fuzzy relationships for qualitative determination of various features are given in Sect. 4, while Sect. 5 presents our fuzzy membership algorithm which evaluates the importance of each voxel. Other implementation details are briefly explained in Sect. 6. Results and conclusions are given in Sect. 7.

2 Related work

We now briefly review related work concerning feature-based, topology-based, and fuzzy-related flow visualization. There is a large literature on these topics, and we refer the reader to surveys in [2, 13, 14].

Feature-based approaches have been developed to extract various physically meaningful structures in flow fields, particularly vortices and typhoons. A popular partition-based local vortex criterion is the λ_2 method [6], based on

evaluation of the Jacobian of the vector field by matrix decomposition and eigenvalue computation. It has been extended to analyze vortex structures in turbulent flows by addition of a novel vortex core line detection approach [17]. It has also been extended using local statistical complexity analysis to find distinctive structures in time-dependent multi-fields [5]. A novel vortex core line extraction method based on a vortex region criterion was proposed in [16], to help visualize vortex features in 3D flows. The velocity masking method has also been adopted to extract typhoon centers for emphasis [3]. Yet another approach is to extend parallel-vectors operator methods to a coplanar vectors operator, which can extract cores of swirling particle motion in transient flows [21]. A new criterion to characterize hierarchical two-dimensional vortex regions induced by swirling motion was given in [12], which defined vortex regions as closed loops intersecting the flow field at a constant angle; a parameter free algorithm is provided for identification of such regions. In summary, a range of feature-based techniques exist. Different methods rely on different definitions of the same feature, and therefore yield differing results. In contrast, our fuzzy clustering framework defines vortices and typhoon structures using fuzzy relationships, so that feature regions have statistically greater importance than other regions.

Topology-based 3D flow methods generally provide better visualization of critical points via limit sets of streamlines. Flow topology was first introduced to the 3D visualization community by [4], and used to separate a vector field into regions with similar behavior. A more recent approach for visualizing higher order critical points of 3D vector fields is [22]. A dense flow visualization method, which shows the overall flow behavior while accentuating structural information, is given in [10]. Topology-based vortex analysis has also been used to examine a high-dimensional, massive flow data set around an airfoil [20]. Another approach is to first extract singularities and separatrices to decompose the flow field into topological regions, then select a seeding path from a set of streamlines integrated in the orthogonal flow field in each region. This produces evenly-spaced long streamlines while preserving topological features of the flow field [23]. A stream surface algorithm which robustly handles special conditions associated with critical points and periodic orbits, e.g. vanishing velocity, unbounded curvature, and tightly winding spirals, is given in [11]. Ultimately, the connection between topological structures and physical properties of the flow is not always direct. Instead, we use fuzzy relationships to determine critical points in terms of position, velocity, and streamline distance.

Fuzzy-theory has so far been little used for feature visualization. Defining flow features using fuzzy point predicates (Boolean functions over all points of a data set) [15] and logical operators [1] have given good results. A set of

streamlines is computed and then individually accepted or rejected according to whether it matches the predicate. In contrast, we first use fuzzy rules to create a combined subset of features, and then use a fuzzy c -means-like clustering algorithm to classify regions. This saves computation time for particle traces, since we do not need to perform predicate checking on each point on the streamline. It also allows us to highlight natural feature region boundaries, enhancing the visual results.

3 Fuzzy representation of 3D flow features

This section presents our mathematical description of 3D flow features based on fuzzy theory. It is based on using *properties* to identify *feature regions*, for example, points in a typhoon feature region have a lower pressure value than surrounding regions—a typhoon has the property of lower pressure.

Definition 1 A **flow feature** in the flow field $U(\mathbf{x})$ is a fuzzy subset F_i , which has property i in the whole region \mathbf{X} . The fuzzy subset can be regarded as an ordered pair:

$$F_i = \{\mathbf{x}, \mu_i(U(\mathbf{x}))\}, \quad \mathbf{x} \in \mathbf{X}; \quad \mu_i(U(\mathbf{x})) \in [0, 1].$$

The mapping $\mu_i(U(\mathbf{x}))$, or $\mu_i(\mathbf{x})$ for short, is a *membership function*, mapping each $\mathbf{x} \in \mathbf{X}$ to a certain value in the range $[0, 1]$: this gives the membership degree to which \mathbf{x} belongs to feature subset F_i . The greater the membership degree, the more likely that the point belongs to the feature region.

Definition 2 If F_i is a flow feature, then the **feature kernel** $\text{Ker } F_i$ is the fuzzy set containing all points within \mathbf{X} whose membership function of F_i is 1.0:

$$\text{Ker } F_i = F_i^{1.0} = \{\mathbf{x} \mid \mu_i(\mathbf{x}) = 1.0\}.$$

Definition 3 If F_i is a flow feature, then the **support region** for F_i is a fuzzy set containing all points within \mathbf{X} satisfying $\mu_i(\mathbf{x}) > 0.0$, denoted by $\text{Supp } F_i$:

$$\text{Supp } F_i = F_i^{0.0+} = \{\mathbf{x} \mid \mu_i(\mathbf{x}) > 0.0\},$$

The above definitions are used to segment the flow field into different feature regions. They represent a flow feature as a fuzzy subset of the flow field having certain physical properties; these are used to control its rendering. The advantage of using fuzzy subsets in these definitions is that it allows qualitative descriptions of flow features. For instance, even though we may not clearly understand vortex features, we may know that the greater negative the λ_2 value a point has, the more likely that it belongs to the vortex region.

4 Fuzzy relationships for typical flow features

We now give several particular fuzzy relationships for various typical flow features, such as critical points, vortices, and a typhoon. Other such relationships can readily be defined for other kinds of features important to flow visualization problems: The examples below demonstrate that such relationships are not particularly complicated. Note that we do not need to specify precise fuzzy membership formulae themselves—fuzzy membership is determined by a computation described in the next section.

4.1 Critical points

Critical points are positions where the velocity is zero. They play an important role in the flow field, lying at the core of features. From the definition of the critical point, we may define the following relationship.

Definition 4 x-V-rule: every point \mathbf{x} in the flow field with velocity zero lies in the critical point kernel:

$$V(\mathbf{x}) = 0 \implies \mu_i(\mathbf{x}) = 1.$$

The **x-V-rule** shows how the basic position and velocity properties can be used to represent the feature property of a critical point.

To decide the feature region around a critical point, we introduce a further variable based on the following observation. In a flow field, physical properties such as heat will propagate along streamlines. The feature region in the flow field should reflect this directional property. Thus, regions closer to the critical point, measured along a streamline, should have a greater membership degree. We use the streamline distance relationship for this purpose. First we define streamline distance:

Definition 5 For any two points $\mathbf{p}_1, \mathbf{p}_2$ in the vector field, the **streamline distance** L is

$$L(\mathbf{p}_1, \mathbf{p}_2) = \begin{cases} \int_{\mathbf{p}_1}^{\mathbf{p}_2} df & \text{iff a streamline } f \text{ joins } \mathbf{p}_1 \text{ and } \mathbf{p}_2 \\ \infty & \text{otherwise} \end{cases}$$

The streamline distance relationship is now:

Definition 6 S-rule: for any two points $\mathbf{p}_1, \mathbf{p}_2$ in the vector field, if \mathbf{p}_1 has a shorter streamline distance to a critical point \mathbf{c} than \mathbf{p}_2 , then \mathbf{p}_1 has a greater degree of membership of the feature associated with \mathbf{c} :

$$L(\mathbf{p}_1, \mathbf{c}) < L(\mathbf{p}_2, \mathbf{c}) \implies \mu_i(U(\mathbf{p}_1)) > \mu_i(U(\mathbf{p}_2)).$$

The **S-rule** associates a point \mathbf{x} in the flow field with its closest critical point, and effectively reflects the location of

the feature region around each critical point. For an isotropic flow field, points with equal streamline distance will constitute an iso-surface consistent with the nature of the isotropic flow.

4.2 Vortices

Various algorithms have been designed to identify vortices. Among them, the λ_2 method [5] is a popular region-based, local detection approach. We use it to add a metric L_2 relationship.

Definition 7 L_2 -rule: Any point \mathbf{p}_1 with more negative λ_2 value than some other point \mathbf{p}_2 has higher membership degree of the corresponding vortex \mathbf{c} :

$$\lambda_2(\mathbf{p}_1) < \lambda_2(\mathbf{p}_2) \implies \mu_i(U(\mathbf{p}_1)) > \mu_i(U(\mathbf{p}_2)).$$

As demonstrated in [5], λ_2 is able to extract the vortex region from the flow field. As alternatives to using λ_2 , we could also construct alternative L_2 -rule based on other vortex detection methods, such as eigenvectors [18], or the parallel vectors method [21]. This particular L_2 -rule is sufficient to demonstrate the flexibility of our framework, which can integrate any appropriate user-defined relationships.

4.3 A typhoon

A typhoon can to a certain extent be regarded as a special type of vortex. Typhoon cores are simple and the user can locate them interactively. Alternatively, other properties such as temperature, pressure, etc. also serve to identify a typhoon. We use the following relationship concerning the pressure field to help identify a typhoon feature without user input.

Definition 8 P-rule: Given a pressure field, every point \mathbf{p}_1 with lower pressure value than some other point \mathbf{p}_2 has greater membership degree of the feature associated with the sink point:

$$P(\mathbf{p}_1) < P(\mathbf{p}_2) \implies \mu_i(U(\mathbf{p}_1)) > \mu_i(U(\mathbf{p}_2)).$$

The **P-rule** is useful to describe the relationship between features in flows and the pressure property. In a similar way, if the user can qualitatively describe the connection between other properties (e.g. pressure, density) and flow features, then further relationships of this type can be devised and added to our framework.

5 Fuzzy voxel significance algorithm

The above rules describe different feature properties qualitatively. Note that we only visualize *one* kind of feature in

Algorithm 1 Fuzzy c -means-like clustering algorithm for vortices

In: $\mathbf{X}, \mathbf{V}, C$: sample points, velocities, feature kernels

Out: S_k for each point \mathbf{x}_k

```

1: for each point  $\mathbf{x}_k$  in  $\mathbf{X}$  do
2:   Find minimum streamline distance value at  $\mathbf{x}_k$ .
3:   Find the  $\lambda_2$  value at  $\mathbf{x}_k$ .
4:   for each feature kernel  $C_i$  do
5:     if  $\mathbf{x}_k \in C_i$  then
6:        $\mu_{ik} \leftarrow 1$ .
7:     else
8:        $\mathbf{A}(C_i) = \text{NormalizeFeatureVector}(C_i)$ ;
9:        $\mathbf{A}(\mathbf{x}_k) = \text{NormalizeFeatureVector}(\mathbf{x}_k)$ ;
10:       $d_{ik} = \text{DifferenceNorm}(\mathbf{A}(C_i), \mathbf{A}(\mathbf{x}_k))$ ;
11:       $\mu_{ik} = \text{FuzzyMembership}(d_{ik})$ ;
12:     end if
13:   end for
14:    $S_k = \max(\mu_{1k}, \dots, \mu_{Ik})$ 
15: end for

```

any given run. We now discuss how we use these rules to measure the membership degree of each voxel of the flow. For convenience, the sampling resolution is chosen in agreement with the voxelization: sample points lie at the center of each voxel. Our algorithm (Algorithm 1 shows the particular case for vortices) works in a similar way to c -means clustering, except that we do not recalculate the cluster centroids as in the standard c -means method. This is because the input feature kernels are determined by the fuzzy definitions selected by the user and hence represent a concrete idea of parts of the flow that the user expects to definitely lie inside each region. Thus, we treat the input feature kernels as unchanging clustering centroids. Clearly, iteration is no longer useful, and a single clustering step is used; this also makes our method fast. The input data to the algorithm include the space field \mathbf{X} , the velocity field V and the set C of feature kernels (number is I). The set C is approximately pre-computed by some suitable method: for example, we use the λ_2 method [5] to calculate the vortex field kernel. The output of our algorithm is a significance value S for each voxelized region, indicating its importance in the visualization.

To do so, we must use different property variables to evaluate different rules; so, for example, we use position and velocity values for the **x-V-rule**, and pressure for the **P-rule**. The main difficulty lies with the **S-rule**. Since there are m critical points in the field, and m is large, a high dimensional feature vector will be needed if we consider the streamline distance (s_1, \dots, s_m) to every critical point. Fortunately, each sample point \mathbf{x} lies on just one streamline joining just two critical points. Therefore, we may use the minimum s_i in the **S-rule** if we do not want to visually dis-

tinguish between different critical points (our examples use this strategy to simplify the feature vector, for speed).

The exact feature vector used will vary according to the application. For example, if we are interested in critical points, we would use the **x-V-rule** and the **S-rule**, which would result in a 7-dimensional vector $(x_1, x_2, x_3, v_1, v_2, v_3, s_i)$, where x_1-x_3 are position coordinates, v_1-v_3 are velocity components, and s_i is the minimum streamline distance. For vortices, we add the λ_2 rule, so the feature vector is $(x_1, x_2, x_3, v_1, v_2, v_3, s_i, \lambda_2)$. For typhoon data, we do not use the λ_2 -rule as there are many other cyclones and anti-cyclones, but instead we use the **P-rule**, giving $(x_1, x_2, x_3, v_1, v_2, v_3, s_i, p)$ as the feature vector. Our framework is quite adaptable.

Having computed the feature vector $\mathbf{A}(\mathbf{x}_k)$ at each position \mathbf{x}_k , we normalize each component to the range $[0,1]$ by considering the range of values for that component over all points. We next compute the absolute difference of the feature vector of each point \mathbf{x}_k and the feature vector for each point of each kernel C_i (each kernel point is treated as a separate clustering center): $d_{ik} = \|\mathbf{A}(\mathbf{x}_k) - \mathbf{A}(C_i)\|$. For the fuzzy membership calculation, we do not employ traditional functions, such as the S -membership algorithm [9], but instead use the fuzzy c -means clustering technique [19]. In our algorithm, each cluster point lies in a feature kernel, so our clustering algorithm does not need to perform the iterative process used in [19] to find cluster centers (these are obviously not given in the dataset). In probabilistic fuzzy clustering, the task is to minimize the objective function

$$J(\mu, \mathbf{d}) = \sum_{i=1}^I \sum_{k=1}^K \mu_{ik}^2 d_{ik}^2.$$

where μ_{ik} is \mathbf{x}_k 's membership of the i th cluster. This may be minimized by simply setting its derivative to zero, with two constraints. Firstly, the μ_{ik} must be non-negative. Secondly, the appropriate fuzzy membership relationships from Sect. 3 must be satisfied. The result is that

$$\mu_{ik} = \begin{cases} \sum_{j=1}^I d_{ik}/d_{jk} & \text{if } I_k = \emptyset \\ 1 - L(\mathbf{x}_k, C_i)/L(\mathbf{x}_k) & \text{if } I_k \neq \emptyset, i \in I_k \text{ and } i \leq I \\ 0 & \text{if } I_k \neq \emptyset, i \in \overline{I_k} \text{ and } i \leq I \end{cases}$$

Here, \emptyset is the empty set, $I_k = \{i \mid 1 \leq i \leq I, d_{ik} = 0\}$, $\overline{I_k} = \{i \mid 1 \leq i \leq I\} - I_k$, $L(\mathbf{x}_k, C_i)$ is the streamline distance between \mathbf{x}_k and the feature kernel C_i , and $L(\mathbf{x}_k)$ is the total length of the streamline through \mathbf{x}_k in the flow field. Finally, the *significance* S_k of each voxel is determined by the maximal membership value μ_{ik} for the center sample \mathbf{x}_k for all features. The significance tells us how important the voxel is when rendering.

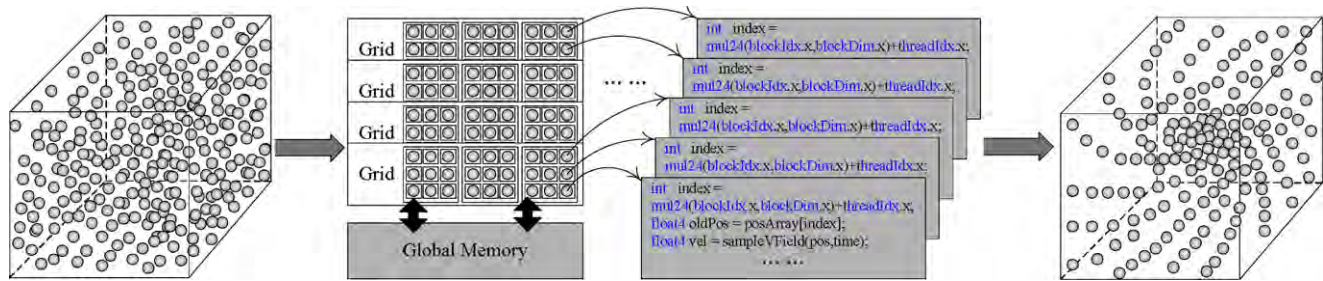


Fig. 3 Particle system pipeline based on CUDA. Particles are seeded randomly in the flow field. Their positions and velocities are sent to CUDA kernels. These calculate the new position for each particle and send them to the rendering pipeline, which draws the particles at their new positions

6 Implementation

Our fuzzy clustering framework is built on a particle tracing platform [3] for visualizing 3D feature structures, but redesigned for GPU acceleration using CUDA [8]; see Fig. 3. Firstly, we allocate CUDA arrays for the particle positions and the velocity field and transfer these variables to the GPU. As particles are independent, we can use CUDA threads to compute particle positions in parallel.

We now further discuss GPU implementation of the fuzzy membership computation, and explain how a multi-resolution technique is used to process and visualize the feature regions with greater detail than the background. The main loop in Algorithm 1 is actually split into several separate phases in the implementation.

6.1 CUDA-accelerated membership computation

The streamline distance calculations are time-consuming, because we need to know the minimum value to a critical point for every position in the flow field. We first locate the critical points, and transfer this position array to the CUDA kernels. Then for each cell of the flow field, we integrate along the streamline until its streamline distance to any critical point is smaller than half the voxel size, or we leave the domain. The least streamline distance to any critical point gives the property value for the *S-rule* (∞ is used if we leave the domain). These integrations are executed in parallel on multiple CUDA threads for speed.

It is also time-consuming to compute the λ_2 feature value. However, this local vortex detection method can be independently computed at each position in parallel using the GPU. Other properties such as velocity and pressure can be obtained directly.

6.2 Multi-resolution processing and rendering

After calculating the significance of each voxel in the flow field, we use a multi-resolution strategy to choose a filter factor M_k for rendering particles. Let M_k be the maximum number of particles to be rendered in voxel k in the flow

field. We set $M_k = Rg(S_k)$ where $R > 0$ is a user specified constant, S_k is the significance of the voxel, and g is any monotonically increasing function; we simply use $g(x) = x$ here.

To accelerate particle rendering, we visualize each particle at its position at each time step with the aid of traditional vertex and fragment programs [7]. This approach can readily handle lighting using state variables available in the OpenGL rendering pipeline. It is also simple to add further fuzzy operators, such as a tone operator, to modify the depiction of the flow field. For example, for a vector field U , a tone operator $H_\alpha : F_U \mapsto F_U$ can be defined as follows:

$$H_\alpha(\mu(x)) = (\mu(x))^\alpha, \quad \alpha > 0$$

For $\alpha > 1$, H_α acts as a centralising filter, focusing in on the area of interest, while $\alpha < 1$ gives a diffusing filter with a broader view. In our experiments, we mapped such a range of user-friendly terms such as ‘extremely focused’, ‘normal’, etc. to the parameter values of $\alpha = 4, 2, 1, 0.5$, and 0.25 , which helped users to control the visualization.

Overall, our approach gives more detail in important regions and less information elsewhere. As a result, the output shows a general picture of the entire data field, while clearly presenting flow features in enhanced detail.

7 Results

Our framework has been tested on many synthetic datasets, and datasets from numerical simulations. The results demonstrate that our approach can effectively solve problems of occlusion and clutter. Here, we demonstrate visualization results for a synthetic flow representing a classical Lorenz attractor

$$\mathbf{v} = (10(y - x), 28x - y - xz, xy - (8/3)z).$$

We also give results for numerical data from CFD simulation of the Haitang typhoon, and flow around a spacecraft, both from real engineering projects.

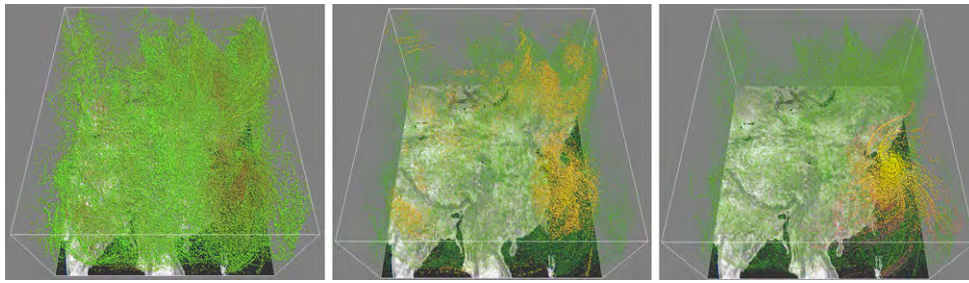


Fig. 4 Visualization of the Haitang typhoon. *Left*: original data, without feature extraction—the typhoon structure cannot be clearly seen. *Center*: some typhoon regions (yellow) are wrongly detected and visu-

alized if velocity masking [3] is used. *Right*: the typhoon is accurately located and clearly visualized by our method

For the Lorenz attractor shown in Fig. 1, we can see the two nearly planar substructures of the Lorenz attractor near its two focal critical points; nevertheless, the surrounding flow movements can still also be seen; see Fig. 1(right), which set the emphasis value to $\alpha = 0.5$. It is easy to see how the particle density is proportional to the membership field. In Fig. 1(center), a comparison is made with a visualization using the traditional λ_2 method [5] to extract the feature region, and our method is clearly superior.

For the typhoon dataset, the field is more complicated. Figure 4(left) shows an overall view of the wind field. There are many vortices including cyclones and anticyclones, there is significant occlusion, and the scene is so complex that it is difficult for the user to analyze the typhoon. The typhoon region detected by the velocity masking method [3] is rendered with yellow particles in Fig. 4(center). Although there is only one typhoon in this flow, unfortunately, several regions are highlighted. Figure 4(right) demonstrates the result of our method: both the location of the typhoon and the overall flow are correctly visualized.

For the spacecraft dataset, we compared our method with both λ_2 and velocity masking methods; see Fig. 5. The vortex region extracted by the λ_2 method (Fig. 5(top)) is correct in the sense that it differentiates the vortex region from the frontal region. However, the vortex region extracted is too small for the user to fully analyze the vortex. The velocity masking method provides better results, filtering the high speed region, and extracting the vortex region behind the spacecraft very well (see Fig. 5(next to top)). However, the velocities in front of the spacecraft are too low to distinguish this region from the vortex region. This limitation is effectively overcome by our method (Fig. 5(next to bottom)) with $\alpha = 1.0$. Because we take the backward streamline distance into account, the regions of origin of particles in vortices are also emphasized. This ability to analyze flow within feature structures is very helpful to the user. In Fig. 5(bottom), background information is also visualized.

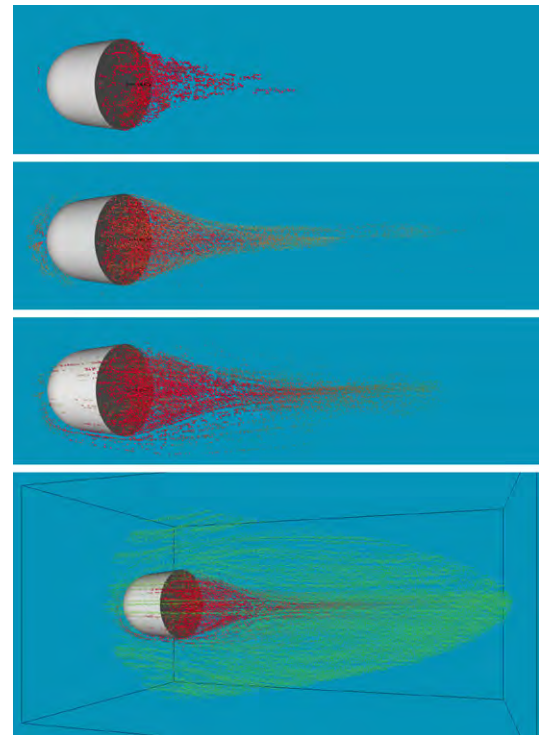


Fig. 5 Top three images: visualization of spacecraft flows by λ_2 method, velocity masking method, and our method, respectively. Our framework extracts a more useful vortex region. Bottom: whole flow visualized by our framework: vortex region particles are colored red and background green

These experiments were carried out on a PC with a 1.6 GHz Pentium Dual CPU with 2 GB RAM and an nVidia GeForce 9800 graphics card with 512 MB memory. Table 1 gives the calculation times for these datasets. The critical kernel points of the flow field are first located, then we calculate region membership using the fuzzy rules. These steps are time-consuming when the number of feature regions is relatively large. Fortunately, our rendering efficiency is unaffected (see the last line), since the membership fields are pre-computed.

Table 1 Performance on experimental datasets. Note that the rendering performance is fast enough for interactive analysis

Dataset	Lorenz	Typhoon			Spacecraft	
Particles	101 × 101 × 81	131 × 89 × 23			128 × 64 × 64	
Feature kernel computation	0.15 s	4.93 s			0.26 s	
Selected feature type	Saddle	Cyclone	Anticyclone	Saddle	Typhoon	Vortex
Number of features	3	209	256	442	1	15
Feature analysis	3.86 s	10.84 s	13.15 s	29.27 s	0.22 s	2.98 s
Rendering	15 fps	38 fps	38 fps	39 fps	35 fps	12 fps

8 Conclusions

We have demonstrated an interactive fuzzy clustering framework for visualizing 3D vector flow fields, which is capable of emphasizing features, showing them without occlusion or clutter, yet still giving a view of the whole flow. Our framework regards the features as fuzzy sets, with simple rules of fuzzy membership defining user-specified feature properties, such as critical points, vortices, and a typhoon. A *c*-means-like clustering algorithm is then used to identify the importance of each region in the flow. A multi-resolution approach determines and displays feature structures with fine-grained information, while the background is shown using coarse-grained information. GPU technology enables the whole process to be accelerated, allowing interactive exploration of complex 3D flow fields.

References

1. Botchen, R.P., Lauser, A., Weiskopf, D., Ertl, T.: Flow feature visualization using logical operators on multivariate fields. In: International Symposium on Flow Visualization (2008)
2. Garth, C., Tricoche, X.: Topology- and feature-based flow visualization: methods and applications. In: SIAM Conference on Geometric Design and Computing (2005)
3. Helgeland, A., Elboth, T.: High-quality and interactive animations of 3D time-varying vector fields. *IEEE Comput. Graph. Appl.* **12**(6), 1535–1546 (2006)
4. Helman, J.L., Hesselink, L.: Visualizing vector field topology in fluid flows. *IEEE Comput. Graph. Appl.* **11**(3), 36–46 (1991)
5. Jänicke, H., Wiebel, A., Scheuermann, G., Kollmann, W.: Multi-field visualization using local statistical complexity. *IEEE Trans. Vis. Comput. Graph.* **13**(6), 1384–1391 (2007)
6. JEONG, J., HUSSAIN, F.: On the identification of a vortex. *J. Fluid Mech.* **28**(5), 69–95 (1995)
7. Kruger, J., Kipfer, P., Kondratieva, P., Westermann, R.: A particle system for interactive visualization of 3D flows. *IEEE Trans. Vis. Comput. Graph.* **11**(6), 744–756 (2005)
8. NVIDIA: Nvidia CUDA programming guide (version 2.2.1) (2010)
9. Pal, K., King, R.A.: On edge detection of x-ray images using fuzzy set. *IEEE Trans. Pattern Anal. Mach. Intell.* **1**(5), 69–77 (1983)
10. Park, S.W., Yu, H., Hotz, I., Kreylos, O., Linsen, L., Hamann, B.: Structure-accentuating dense flow visualization. In: Eurographics /IEEE VGTC Symposium on Visualization, pp. 131–138 (2006)
11. Peikert, R., Sadlo, F.: Topologically relevant stream surfaces for flow visualization. In: Spring Conference on Computer Graphics, pp. 43–50 (2009)
12. Petz, C., Kasten, J., Prohaska, S., Hege, H.C.: Hierarchical vortex regions in swirling flow. *Comput. Graph. Forum* **28**(3), 863–870 (2009)
13. Post, F.H., Vrolijk, B., Hauser, H., Laramée, R.S., Doleisch, H.: The state of the art in flow visualization: feature extraction and tracking. *Comput. Graph. Forum* **22**(4), 775–792 (2003)
14. Salzbrunn, T., Jänicke, H., Wischgoll, T., Scheuermann, G.: The state of the art in flow visualization: partition-based techniques. In: Simulation and Visualization, pp. 75–92 (2008)
15. Salzbrunn, T., Scheuermann, G.: Streamline predicates. *IEEE Trans. Vis. Comput. Graph.* **12**(6), 1601–1612 (2006)
16. Schafhitzel, T., Vollrath, J.E., Gois, J.P., Weiskopf, D., Castelo, A., Ertl, T.: Topology-preserving λ_2 -based vortex core line detection for flow visualization. *Comput. Graph. Forum* **27**(3), 1023–1030 (2008)
17. Stegmaier, S., Rist, U., Ertl, T.: Opening the can of worms: An exploration tool for vortical flows. In: IEEE Visualization, pp. 463–470 (2005)
18. Sujudi, D., Haimes, R.: Identification of swirling flow in 3d vector fields. In: Proceedings of AIAA 12th Computational Fluid Dynamics Conference, pp. 1695–1715 (1995)
19. Timm, H., Borgelt, C., Doring, C., Kruse, R.: An extension to possibilistic fuzzy cluster analysis. *Fuzzy Sets Syst.* **147**(1), 3–16 (2004)
20. Weinkauff, T., Sahner, J., Günther, B., Theisel, H., Hege, H.C., Thiele, F.: Feature-based analysis of a multi-parameter flow simulation. In: Simulation and Visualization, pp. 237–252 (2008)
21. Weinkauff, T., Sahner, J., Theisel, H., Hege, H.C.: Cores of swirling particle motion in unsteady flows. *IEEE Trans. Vis. Comput. Graph.* **13**(6), 1759–1766 (2007)
22. Weinkauff, T., Theisel, H., Shi, K., Hege, H.C., Seidel, H.P.: Extracting higher order critical points and topological simplification of 3D vector fields. In: IEEE Visualization, pp. 559–556 (2005)
23. Wu, K., Liu, Z., Zhang, S., Moorhead, R.J. II: Topology-aware evenly spaced streamline placement. *IEEE Trans. Visual. Comput. Graph.* **99**(3) (2009)



Huaxun Xu is a researcher in Army Aviation Institute of PLA, China, and a Ph.D. candidate in national university of defense technology. His current research interests include scientific visualization and virtual reality technologies.



Zhi-Quan Cheng received a B.Sc., M.Sc., and Ph.D. degree from Computer School at National University of Defense Technology in 2000, 2002 and 2008, respectively. He is lecturer at the PDL Laboratory, Computer School, National University of Defense Technology. His research interests include computer vision and graphics, digital geometry processing. He has completed many research projects and published more than 30 papers.



Ralph R. Martin received the Ph.D. from Cambridge University in 1983, with a dissertation on Principal Patches, and since then, has worked his way up from a lecturer to a professor at Cardiff University. He has been working in the field of CAD/CAM since 1979. He has published more than 170 papers and 10 books covering such topics as solid modelling, surface modelling, intelligent sketch input, vision based geometric inspection, geometric reasoning and reverse engineering. He is a fellow of the Institute of Mathematics and Its Applications, and a member of the British Computer Society. He is on the editorial boards of Computer Aided Design, Computer Aided Geometric Design, the International Journal of Shape Modelling, the International Journal of CAD/CAM, and Computer-Aided Design and Applications. He has also been active in the organization of many conferences.



Sikun Li A professor in national university of defense technology. His research interests include scientific visualization, virtual reality, very large scale integration and the design of SoC.