J Vis (2019) 22:1125–1144
https://doi.org/10.1007/s12650-019-00592-3

REGULAR PAPER

Jun Ma · Jun Tao · Chaoli Wang · Can Li · Ching-Kuang Shene · Seung Hyun Kim

Moving with the flow: an automatic tour of unsteady flow fields

Received: 7 July 2019 / Accepted: 19 July 2019 / Published online: 30 August 2019

Abstract We present a novel framework that creates an automatic tour of unsteady flow fields for exploring internal flow features. Our solution first identifies critical flow regions for time steps and computes their temporal correspondence. We then extract skeletons from critical regions for feature orientation and pathline placement. The key part of our algorithm determines which critical region to focus on at each time step and derives the region traversal order over time using a combination of energy minimization and dynamic programming strategies. After that, we create candidate viewpoints based on the construction of a simplified mesh enclosing each focal region and select the best viewpoints using a viewpoint quality measure. Finally, we design a spatiotemporal tour that efficiently traverses focal regions over time. We demonstrate our algorithm with several unsteady flow data sets and perform a user study and an expert evaluation to confirm the benefits of including internal viewpoints in the design.

Keywords Unsteady flow · Critical regions · Feature correspondence · Seed placement · Internal viewpoints · Automatic tour

Electronic supplementary material The online version of this article (https://doi.org/10.1007/s12650-019-00592-3) contains supplementary material, which is available to authorized users.

J. Ma
Qualcomm, Inc., San Diego, USA
E-mail: junm@qit.qualcomm.com

J. Tao (✉)
Sun Yat-sen University, Guangzhou, China
E-mail: taoj23@mail.sysu.edu.cn

C. Wang
University of Notre Dame, Notre Dame, USA
E-mail: chaoli.wang@nd.edu

C. Li
Haven Life Insurance Agency, LLC, New York, USA
E-mail: tonylic1989@gmail.com

C.-K. Shene
Michigan Technological University, Houghton, USA
E-mail: shene@mtu.edu

S. H. Kim
The Ohio State University, Columbus, USA
E-mail: kim.5061@osu.edu

1 Introduction

For large and complex flow fields, it is important for users to view the features or pattern of interest, especially for those hidden or occluded ones which are not clearly visible from outside the volume. We refer to such features as *internal features*. This occlusion problem is similar to that in volume rendering, which could be addressed by tuning the transfer function. For integration-based flow visualization, if we choose to keep flow lines opaque and do not filter out the surrounding lines, considering internal viewpoints becomes necessary. Internal viewpoints give users closeup views for detail observation of what lies inside of the flow field. External viewpoints are commonly placed at the volume's bounding sphere and look at the center of the volume. They are suitable for observing *external features*, i.e., flow features around the volume's boundary. In contrast, internal viewpoints have much more freedom as they can be placed anywhere inside the volume and look at any points of interest. The recent work of FlowTour (Ma et al. 2014) generates an automatic tour for exploring internal flow features. FlowTour places viewpoints around the features of interest, i.e., critical flow regions, to form the view path. However, it only considers 3D steady flow fields. The design of an automatic tour for exploring 3D unsteady flow fields remains an unsolved problem.

Streamlines are steady over time, but pathlets traced over an unsteady flow field move over time. This poses several unique challenges which demand a new solution for automatic tour design. First, for a steady flow field, we place seeds once to generate streamlines. For an unsteady flow field, we should carefully place seeds over time to capture critical flow regions at different time steps. Using pathlet animation, we need to make sure that the *density of pathlet* is appropriate and varies smoothly over space and time. Therefore, new seeds need to be placed in the subsequent time steps to highlight new critical regions. Additional seeds need to be placed to account for disappearing pathlets which go out of the domain boundary or get trapped around the vicinity of a critical point. Second, in the tour animation, since all streamlines remain unchanged, the shifting from one focal region to another region only needs to consider how smooth the transition will be. For an unsteady flow field, we assume that the animation follows the order of time steps of the data. Since all pathlets change along the animation, we need to solve the issue of *dynamic shifting of focus* as the critical regions may merge or split over time. To this end, we need to first identify the correspondence among critical regions, then determine an optimal traversal order of these critical regions. Third, placing cameras along the tour should take into account the size, orientation, and lifespan of each critical region of focus so that the tour would highlight different critical regions *at the right moment* and *for a right duration*. Only by doing so can we produce an informative tour path to capture important time-dependent flow features for a comprehensive understanding.

In this paper, we present a new framework that designs an automatic guide for exploring internal flow features for unsteady flow fields. Our solution encompasses feature identification, pathlet placement, region traversal-order determination, viewpoint selection, and tour generation. In particular, we propose an optimal solution for determining the critical region traversal order that integrates energy minimization and dynamic programming techniques. This important step essentially determines the outline of the tour path, leaving path details to be solved in the subsequent steps. Our work can be considered as an extension of FlowTour (Ma et al. 2014). For a single time step, it shares similar ideas with FlowTour, e.g., critical region detection and region skeleton extraction. However, since our algorithm focuses on unsteady flow fields, the key questions are how to find the correspondence among critical regions and how to determine the best region traversal order. To the best of our knowledge, our work is the first that aims at designing a guided tour for exploring an unsteady flow field.

2 Related work

Many flow visualization techniques were proposed to make flow features and pattern easily and clearly perceivable. Examples include feature-based (Post et al. 2003) and geometry-based (McLoughlin et al. 2010) flow visualization. Among these techniques, streamline visualization has long been a popular method for visualizing steady flow fields. A streamline is a curve everywhere tangent to a steady flow field. Intuitively, it is the path that a particle will follow if released in the field. Seed placement and streamline selection form two major branches of streamline visualization. For seed placement in 3D (Spencer et al. 2009; Xu et al. 2010; Yu et al. 2012; Ma et al. 2014), the goal is to select the best seeds so that the corresponding streamlines can clearly reveal flow features and patterns. As an alternative to seed placement, streamline selection (Marchesin et al. 2010; Lee et al. 2011; Tao et al. 2013; Ma et al. 2013) aims at

selecting the best streamlines from a pool of streamlines to capture important flow features while reducing occlusion and clutter.

Pathlines are the trajectories of particles seeded in an unsteady flow field. Unlike streamline tracing which uses the same vector data throughout the integration, the vector data are updated at each time step when pathlines are advected. McLoughlin et al. (2013) presented an approach that computes the similarity between integral curves (streamlines and pathlines) for interactive seeding. Their similarity-based clustering enables filtering of the integral curves to provide a nonuniform seeding distribution along the seeding object. Different approaches were proposed to visualize unsteady flow fields using pathline advection. Steinman (2000) described a simple method for visualizing 3D periodic velocity data based on the subdivision and sequential display of pathlines. The algorithm provides the flexibility to control the length and spacing of the pathlines. Chandler et al. (2015) presented a method to trace pathlines in scattered, particle-based flow fields. By developing a modified k-d tree representation for highly compressible data sets, they performed pathline computation via identifying, tracking, and updating an enclosing, dynamic particle neighborhood as particles move over time. Besides pathline visualization, other techniques were also developed for unsteady flow field visualization. Lane (1993) built a simple and effective system to visualize unsteady flow fields using streaklines. The system can process multiple time steps without requiring all the data simultaneously, making it suitable for out-of-core processing. Sadlo et al. (2011) visualized Lagrangian coherent structures of unsteady flow fields using an efficient method to compute the finite-time Lyapunov exponent and its height ridges related to the flow structure.

Viewpoint selection is an important issue in volume (Bordoloi and Shen 2005; Takahashi et al. 2005) and flow (Lee et al. 2011; Tao et al. 2013) visualization. Typically after finding the best viewpoints, the subsequent task is to generate a good path that traverses all selected viewpoints. Ji and Shen (2006) developed a solution to select viewpoints for a time-varying volumetric data set using a dynamic programming algorithm and produced a smooth viewpoint animation for viewing the data set. Viola et al. (2006) found characteristic viewpoints based on an information channel between the model in the scene and the viewpoint set. They designed a transformation path between two selected viewpoints by utilizing an intermediate viewpoint so that the camera path changes smoothly by switching the focus from one feature to another. Hsu et al. (2013) leveraged the medial-axis-based roadmap technique and introduced several constraints for computing camera paths in order to create effective animations for volume visualization. Bai et al. (2016) presented a view path design method to display the evolution of volumetric features with their topology for a time-varying data set. Their viewpoint entropy computation includes both visual information (i.e., the information channel between viewpoints and feature groups) and topology information (i.e., the skeleton of features quantified with Kullback–Leibler distance). Meuschke et al. (2017) designed an automatic selection of viewpoints to form a camera path for supporting the exploration of time-dependent cerebral aneurysms. Their camera path reveals the most interesting regions during the cardiac cycle based on user-selected morphological and hemodynamic parameters.

3 Algorithm overview

Our algorithm consists of three stages: critical region identification (Sect. 1), region traversal-order determination (Sect. 4), and viewpoint creation and tour generation (Sect. 5). At the first stage, we detect critical regions at each time step and construct their temporal correspondence. We also extract each region's skeleton for skeleton-based seeding. At the second stage, we design a solution that integrates energy minimization and dynamic programming to obtain the optimal traversal order for critical regions. This stage is the most important one as the region traversal order essentially outlines the underlying tour. Once the order of traversal is determined, we create sample viewpoints along each focal region at the third stage. We then select the best viewpoints based on their quality to generate the actual tour path. In this section, we briefly describe the critical region identification as its relatively straightforward. For the other two stages, we will elaborate their procedures in Sects. 4 and 5, respectively.

Critical region identification Our approach first identifies the critical regions at each individual time step and then connects the regions based on their temporal correspondence. The critical regions are detected as high-entropy regions. We evaluate the entropy at a voxel by considering the variation of vector directions in a 4D local window centered at that voxel. In our experiment, we use a window resolution of $5 \times 5 \times 5 \times 5$. Then, we leverage a parallel region growing algorithm to group the high-entropy voxels into regions. Since a critical region could span several time steps in an unsteady field, we further apply the

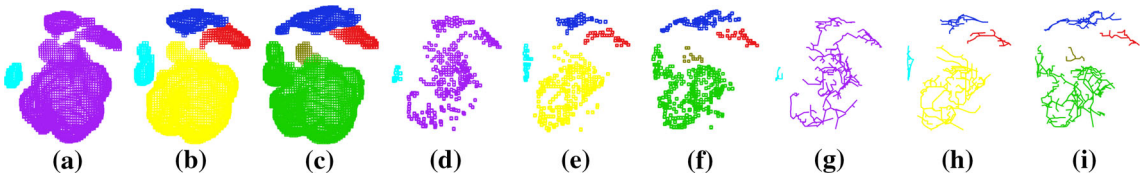

Fig. 1 **a–c** Are the critical regions detected for three consecutive time steps of the supernova data set. Their corresponding volume thinning results are shown in **d–f**, and skeletons extracted are shown in **g–i**

feature tracking algorithm proposed by Silver and Wang (1998) to identify the temporal correspondence among critical regions. Based on the correspondence, we merge regions corresponding to the same feature into a new critical region. In order to identify the shape of a critical region, we compute its skeleton using a volume thinning algorithm (Gagvani and Silver 1999). We place additional seeds along the skeletons together with some random seeds to ensure the critical regions are covered by pathlets. An example of the detected critical regions, their corresponding volume thinning results, and their skeletons is shown in Fig. 1. Please refer to "Critical region identification" section in Appendix for details.

4 Region traversal-order determination

In general, we may have multiple critical regions at a single time step while a critical region may span across multiple time steps. Our goal is to produce a "smooth" traversal of "good" critical regions over time, conveying the most information about the underlying unsteady flow field. Assume that we have a total of n regions and m time steps. Since we select a single critical region to focus on for each time step, the search space for determining the region traversal order is bounded by $O(n^m)$. The worst case happens when each region occupies all the time steps. Obviously, the brute-force method is not practical due to its exponential time complexity. A straightforward greedy algorithm which always picks the "best" region for each time step could be applied. However, the greedy method does not guarantee a globally optimal result. Moreover, temporal correspondence among regions is not considered. In this paper, we present a novel method which integrates *energy minimization* and *dynamic programming* to obtain the optimal traversal order for critical regions.

4.1 Overview of method

We define $I_{r,t}$ in the range of [0, 1] as the score of region r at time step t. A larger (smaller) value of $I_{r,t}$ indicates that r has a higher (lower) chance of being selected as the focus at t. Intuitively, $I_{r,t}$ is the probability of region r to be selected at time t. We consider multiple properties of critical regions and compute $I_{r,t}$ by an optimization algorithm given in Sect. 4.2. We divide the entire time sequence into multiple time windows with an equal size W_s and compute a local traversal order for each time window separately. For a time window w, we minimize an energy function to compute the optimal scores for critical regions in w and then apply a dynamic programming algorithm to determine the region traversal order. In order to incorporate region temporal correspondence into score computation, the resulting traversal order for w will be leveraged as the input for score computation in the next time window $w + 1$. After all the time windows have been processed, we apply the same dynamic programming algorithm over all the region scores to identify the globally optimal traversal order for the entire time sequence.

We point out that there is a significant difference between typical minimization methods and our method. In these minimization methods, energy functions are formulated with a set of initial state values. Then, a linear system is utilized to solve for the optimal solution. In our method, the traversal result from the current time window highly depends on the result from the previous time windows. That is, selecting the current focal region should consider the regions picked before in order to provide a smooth and efficient traversal experience. As such, rather than solving the linear system in a single pass as usual, we adopt a hybrid approach which applies energy minimization and dynamic programming iteratively across different time windows to obtain the final optimization result.

4.2 Optimal score computation

To formulate the energy function for optimal score computation, we define the following two types of constraints: *static constraints* and *dynamic constraints*. Static constraints consider the intrinsic properties of critical regions while dynamic constraints incorporate the influence of the preceding region traversal order on the current region traversal order being determined.

4.2.1 Intrinsic properties

Before we introduce the static constraints, we first define the following intrinsic properties of critical regions:

- *Size* $(S_{r,t})$. This term indicates the volumetric size of critical region r at time step t. The larger the size, the more important the region. Therefore, region score $I_{r,t}$ should be proportional to $S_{r,t}$.
- *Average entropy* $(E_{r,t})$. This term computes the average entropy value over all the grid points inside of region r at time step t. Intuitively, the value of $E_{r,t}$ indicates the amount of information contained in r at t. Therefore, $I_{r,t}$ should be proportional to $E_{r,t}$.
- *Coefficient of variation* $(V_{r,t})$. This term measures the normalized dispersion of entropy value distribution inside of region r at time step t. $V_{r,t}$ is computed as follows

$$V_{r,t} = \frac{\delta E_{r,t}}{E_{r,t}}, \tag{1}$$

  where $\delta E_{r,t}$ is the standard deviation of entropy values inside of r at t. We prefer to focus on the region with higher $V_{r,t}$ since such a region contains richer information about the underlying flow features. Thus, $I_{r,t}$ should be proportional to $V_{r,t}$.
- *Skeleton complexity* $(C_{r,t})$. Since a region skeleton extracted in "Region skeleton extraction" section in Appendix represents the overall shape of the corresponding region, we also consider its complexity when computing the region's score. A high value of $C_{r,t}$ indicates that the corresponding region has a complicated shape pattern and would be interesting to focus on. Region score $I_{r,t}$ should be proportional to $C_{r,t}$. Because our skeleton is a tree structure, we first identify the number of branches in the skeleton and then evaluate the straightness of each branch. We consider two factors for $C_{r,t}$: *shape complexity* and *quantity complexity*. Shape complexity $C_{s,t}$ is defined as follows

$$C_{s,t} = \sum_{\mathbf{b}\in\mathbf{B}(r,t)} \frac{\sum_{i,j\in\mathbf{b}} d_{i,j} - d_{\mathbf{b}\max}}{d_{\mathbf{b}\max}}, \tag{2}$$

  where $\mathbf{B}(r,t)$ is the set of branches in the skeleton of r at t, $\mathbf{b}$ is a branch in $\mathbf{B}(r,t)$, $d_{i,j}$ is the Euclidean distance between two *consecutive* skeleton points i and j, and $d_{\mathbf{b}\max}$ is the largest distance among *all* points in $\mathbf{b}$. Quantity complexity $C_{q,t}$ records the number of branches in a skeleton. As an example, in Fig. 2, *abcde*, *df*, *cg* and *bhi* are branches of this skeleton, and $C_{q,t} = 4$. For both $C_{s,t}$ and $C_{q,t}$, we compute their normalized values $\overline{C_{s,t}}$ and $\overline{C_{q,t}}$. Then $C_{r,t}$ is defined as follows

$$C_{r,t} = \alpha\overline{C_{s,t}} + (1-\alpha)\overline{C_{q,t}}, \tag{3}$$

  where $\alpha \in [0,1]$. Normally, we consider shape complexity to be more important than quantity complexity. Therefore, we set $\alpha = 0.7$ for all data sets.
- *Time span* (T_r). This term indicates the time span in which region r is alive. If T_r is small, r will be alive for only a few time steps. In order to capture this short-lived region, we assign a higher region score $I_{r,t}$ to r so that it gets a chance to be focused on during its time span. So, $I_{r,t}$ should be inversely proportional to T_r. However, if r with a short time span has parents or children and these parents or children are long-

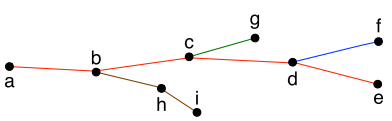

Fig. 2 Four branches of a region's skeleton

lived, then even if we miss r, we can still capture similar flow features by focusing on its parents or children. So, T_r should be applied to only regions with no parents or children.

4.2.2 Static constraints

With the intrinsic properties defined above, we can obtain the following static constraints:
Intrinsic property constraint (O_τ) This constraint enforces that region scores should be as close to the maximum value of 1 as possible according to their intrinsic properties. We first combine all intrinsic properties for a region into a single term $P_{r,t}$

$$P_{r,t} = \begin{cases} \lambda_S S_{r,t} + \lambda_E E_{r,t} + \lambda_V V_{r,t} + \lambda_C C_{r,t} + \lambda_T \frac{1}{T_r}, & \text{r is alive at t} \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

where λ_S, λ_E, λ_V, λ_C, and λ_T are the weights for $S_{r,t}$, $E_{r,t}$, $V_{r,t}$, $C_{r,t}$, and T_r, respectively. In order to better capture short-lived regions, we set λ_T to 2 and the other four weights to 1 for all data sets. With $P_{r,t}$, we define the following energy term

$$O_\tau = \Sigma_{r\in\mathbf{R},t\in\mathbf{W}} P_{r,t} \| I_{r,t} - 1.0 \|^2, \tag{5}$$

where $\mathbf{R}$ is the set of all critical regions and $\mathbf{W}$ is the set of time steps in time window w. Note that if region r is not alive at time step t, its initial score $I_{r,t}$ will be set to 0.

Summation constraint (O_σ) This constraint keeps the summation of all the scores $I_{r,t}$ in the range of [0, 1] as much as possible. To achieve this, we set the summation of scores for the regions at the same time step to 1. If the score is less than 0, we will set it to 0 to avoid negative scores. We define this energy term as follows

$$O_\sigma = \Sigma_{t\in\mathbf{W}} \| \Sigma_{r\in\mathbf{R}} I_{r,t} - 1.0 \|^2. \tag{6}$$

Again, if r is not alive at t, its initial score will be set to 0.

4.2.3 Normalized traversal frequency

The dynamic constraints rely on the preceding region traversal order. Therefore, we first introduce how to compute $L_f(r, w)$ which records the frequency that region r has been traversed before the current time window w. We can obtain $L_f(r, w)$ by counting the frequency that r has been visited in the previous time windows. We define L_b as the maximum number of time windows we will backtrack from w. This term is used to control the length of temporal history we would consider for r. Ideally, a good L_b should keep the tracking in a reasonable time span (e.g., not too long or too short). Sometimes, r may last less than L_b during the backtracking. In this case, we will also consider the traversal history of r's ancestors. Using the above two terms, we define the normalized traversal frequency $F(r, w)$ of region r for time window w as

$$F(r, w) = \frac{L_f(r, w)}{L_b}. \tag{7}$$

4.2.4 Dynamic constraints

With $F(r, w)$, we now define several dynamic constraints:

Traversal frequency constraint (O_v) If region r has been traversed for multiple time steps in the previous time windows, its score in the current time window w should be decreased so that other regions could have a chance to be selected as the focus. To achieve this, we attempt to minimize the following energy term

$$O_v = \Sigma_{r\in\mathbf{R},t\in\mathbf{W}} \frac{1}{F(r, w)} \| I_{r,t} - 1.0 \|^2. \tag{8}$$

Temporal correspondence constraint (O_η) This constraint considers the influence of previously focused regions on their temporal corresponding regions. When region r has been visited for multiple time steps, its children which share similar spatial locations with r in the neighboring time steps should have lower scores.

Contrarily, in order to maintain a smooth traversal order, the siblings of r should get higher scores since they are less likely to be similar to r but have temporal relations with r. For example, they will merge in a later time step or they come from the same parent. Considering these two cases, we define the following energy term

$$\begin{aligned} O_\eta = & \Sigma_{r\in\mathbf{R},t\in\mathbf{W}}\Sigma_{c\in\mathbf{C}(r)}\frac{1}{F(r,w)}\|I_{c,t}-1.0\|^2 \\ & +\Sigma_{r\in\mathbf{R},t\in\mathbf{W}}\Sigma_{s\in\mathbf{S}(r)}F(r,w)\|I_{s,t}-1.0\|^2, \end{aligned} \tag{9}$$

where $\mathbf{C}(r)$ and $\mathbf{S}(r)$ are the sets of r's children and siblings, respectively.

Equal opportunity constraint (O_ξ) When region r has been traversed for multiple time steps, its own score will decrease in the subsequent time steps. Meanwhile, we want other regions which have no temporal correspondence with r to have more chance of being selected as the focus. Therefore, we introduce this constraint to increase the scores of such regions based on their spatial distances to r. We formulate the following energy term

$$O_\xi = \Sigma_{r\in\mathbf{R},t\in\mathbf{W}}\Sigma_{k\in\mathbf{K}(r)}\frac{F(r,w)}{D_{r,k}}\|I_{k,t}-1.0\|^2, \tag{10}$$

where $\mathbf{K}(r)$ is the set of regions which have no temporal correspondence with r, $D_{r,k}$ is the distance between the skeletons of r and k at t. In this work, we use the mean of the closest point distances (Moberts et al. 2005) as the distance measure. Intuitively, region k will have a higher score if $D_{r,k}$ is smaller.

4.2.5 Energy function

Based on the above constraints, we formulate the final energy function as follows

$$O = \gamma_\tau O_\tau + \gamma_\sigma O_\sigma + \gamma_v O_v + \gamma_\eta O_\eta + \gamma_\xi O_\xi, \tag{11}$$

where γ_τ, γ_σ, γ_v, γ_η, and γ_ξ are the weights for these constraints, respectively. In our setting, we prefer to emphasize the impact of the temporal corresponding constraint so that the final tour will consider the relationship between neighboring spatiotemporal regions. Therefore, we set γ_η to 2 and the remaining weights to 1 for all data sets. To find the optimal solution, we convert the energy function into a linear system and leverage a GPU implementation of the concurrent number cruncher (CNC) sparse solver (Buatois et al. 2009) to solve the system. After the optimization completes, we have the optimal values for $I_{r,t}$.

4.3 Traversal-order determination

After region scores are computed for a time window w, we utilize a dynamic programming algorithm to find the optimal traversal order $\Omega(w)$ within the time window. $\Omega(w)$ should focus on one region at a time step and the overall region scores along $\Omega(w)$ should be maximized. To achieve this, we define $\breve{I}_{r,t}$ as the maximum traversal score from region r at t to some region at the last time step of w and introduce the following recursive equation

$$\breve{I}_{r,t} = \max_{k\in\mathbf{R}}(I_{r,t}+\breve{I}_{k,t+1}), \tag{12}$$

where $\mathbf{R}$ is the set of all regions. This equation indicates that the maximum traversal score from region r at t to some region at the last time step of w is equal to the sum of the score $I_{r,t}$ and the maximum traversal score from region k at $t+1$ to some region at the last time step of w. In this case, region k will be the focal region at $t+1$ and it will be put in the array $A_{r,t}$, which records the traversal order starting from r at t. For each region, we compute its $\breve{I}_{r,t_0}$ where t_0 is the first time step of w and pick A_{r,t_0} with the largest $\breve{I}_{r,t_0}$ as the optimal traversal order $\Omega(w)$. We apply the same algorithm to find the final traversal order for the entire time sequence when all region scores $I_{r,t}$ are obtained. In Fig. 3, we show snapshots of the region traversal order determined using our optimization method.

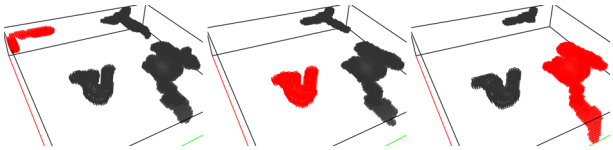

Fig. 3 The shifting of the focal region (shown in red) for three consecutive time steps of the hurricane data set

5 Viewpoint creation and tour generation

For the focal region at each time step, we first create a list of viewpoints based on the isosurface generated from the region. Then we select several best viewpoints to provide users with closeup views of the respective flow pattern for clear observation. If one critical region spans more than one time step, we also consider the location of viewpoints so that the same portion of the region will not be repetitively focused on across multiple time steps. After the best viewpoints are picked for all focal regions, we generate a view path traversing all these viewpoints according to the time order. Our goal is to give a smooth and efficient way of exploring the unsteady flow field.

5.1 Viewpoint creation

We leverage the same strategy described by Ma et al. (2014) to produce candidate viewpoints. For each critical region, we use the marching cube algorithm (Lorensen and Cline 1987) to extract the corresponding isosurface, and simplify the isosurface to avoid oversampled candidates. Given the simplified isosurface, our algorithm creates a list of viewpoints based on its vertices. For each vertex, we first generate a viewpoint v at the vertex's location. The `look-at` center of v is the point on the skeleton of region r_t closest to v. There are two ways to compute the `up` direction: fixing the `up` direction to a predefined direction (such as the positive y direction of the volume) or utilizing the skeleton's major direction as the guidance. Specifically, we define the local skeleton direction $\mathbf{d}$ at the `look-at` center as the vector along the skeleton which starts from the `look-at` center and points toward the skeleton's major direction. We then project $\mathbf{d}$ onto a plane perpendicular to the `look-at` direction $\mathbf{l}$ and the final `up` direction is the projected vector on the plane. In this way, we allow the `up` direction to follow the underlying flow pattern's major direction.

In order to consider the zoom level from the viewpoint v to critical region r_t, we generate a set of five *offset viewpoints* $\mathbf{V}$ associated with v. The position of each offset viewpoint is pushed away from v along the opposite direction of its `look-at` direction $\mathbf{l}$ for some distance δ_l, and it shares the same `look-at` center and the `up` direction of v. In the next step, we will evaluate the quality of each viewpoint in $\mathbf{V}$ and pick one best viewpoint as the representative for $\mathbf{V}$.

5.2 Viewpoint quality evaluation

Given a viewpoint v associated with critical region r_t, we define its quality based on the entropy value of the 2D projection of the pathlets seeded from r_t. We also consider the foreground occlusion and background noise as the penalty to reduce visual clutter and distraction. Specifically, we compute the *viewpoint quality* as follows

$$Q(v) = \lambda_1 S_{\text{focus}} - (\lambda_2 P_{\text{fore}} + \lambda_3 P_{\text{back}}), \tag{13}$$

where S_{focus}, P_{fore}, and P_{back} are the *focal region score*, *foreground occlusion penalty*, and *background noise penalty*, respectively. λ_1, λ_2, and λ_3 are the corresponding coefficients. By setting different values to these coefficients, we ensure that the viewpoint quality $Q(v)$ is always nonnegative. To emphasize the focal region score, we set λ_1 to 1 and give the same weight for the other two penalties ($\lambda_2 = \lambda_3 = 0.2$) for all data sets.

- *Focal region score* (S_{focus}). This term indicates the information revealed by the pathlets seeded from the focal region's skeleton. To compute this value, we first perform the viewing frustum culling operation on each pathlet from the focal region and check if it is inside of the viewing frustum. For all the pathlets inside, we compute the entropy value for their 2D projections based on the flow direction and set this value to S_{focus}.
- *Foreground occlusion penalty and background noise penalty* ($P_{\text{fore}}, P_{\text{back}}$). These two terms measure the influence of pathlets seeded from non-focal regions on the current viewpoint. To compute these two

values, we first check whether the pathlets are inside of the viewing frustum. If yes, we then transform the standard OpenGL view projection plane into a predefined plane of 100×100 pixels and check the depth values of these pathlet projections on the plane to determine if they are foreground occlusion or background noise pathlets. P_{fore} and P_{back} are obtained as the entropy values of their 2D projections for these two kinds of pathlets, respectively.

For a critical region which spans m time steps ($m > 1$), we assign the viewpoints to focus on different portions of the region at each occupied time step so that users could observe the flow pattern in a more balanced way. To achieve this, we divide the major axis of the focal region's skeleton into m segments where each segment corresponds to one occupied time step. Then we compute the final viewpoint score according to their distance to the segments. Specifically, for a viewpoint v, we find one segment g which is closest to v and set g as the segment associated with v. Then at each of the m occupied time steps, there will be one *active segment* and all the viewpoints associated with it will get $Q(v)$ as their final score. Other viewpoints get the score of 0. Later on, we will pick the best viewpoints for each time step based on their scores. This guarantees that only viewpoints within the current active segment would be considered.

5.3 Best viewpoint selection

By evaluating viewpoint quality, we first identify the representative viewpoint for each viewpoint set V as the one with the highest score. Then we sort all representatives based on their quality scores and pick the final best viewpoints. In order to avoid selecting similar viewpoints, any two selected best viewpoints should satisfy either one of the following two criteria. First, the angle between their `look-at` directions is greater than a given threshold δ_α. Second, the distance between their `look-at` centers is greater than a given threshold δ_d.

5.4 Tour path generation and animation

After we identify the best viewpoints for the focal region at each time step, we generate a spatiotemporal view path traversing all these viewpoints

To this end, we first order the best viewpoints for a focal region r according to the direction of the skeleton's major axis to obtain a local traversal order D_r. Since the temporal order for all focal regions is already determined by the linear system (Sect. 4), we only need to connect the viewpoints between two temporally adjacent regions to obtain the final traversal order for all the best viewpoints. Specifically, for the focal region r_1 at the first time step, we use its local traversal order D_{r_1} as the final traversal order for this region. Then for the next focal region r_2 along the temporal sequence, we compute the distance between the last viewpoint v_{last} of r_1 and the two end viewpoints of the local order of r_2 and pick the one closer to v_{last} as the starting viewpoint for r_2. So the final order for r_2 will be either D_{r_2} or its reverse $\overline{D_{r_2}}$. We repeat this process until we connect all the focal regions and get the global viewpoint traversal order. Finally, we utilize a cubic B-spline curve to connect all these viewpoints to obtain the spatiotemporal view path. In Figs. 4 and 5, we show examples of the entire path generated, the path segments along different focal regions, and the path segments along the same focal region.

In the tour animation, when flying through each best viewpoint we keep the current viewpoint staying still for a certain duration. This would allow users to better observe pathlet movements at fixed viewpoints. The transition between two best viewpoints is dynamic, i.e., the viewpoint is changing while pathlets are moving. We render pathlines as gray, thin tubes in the background and overlay pathlets as arrows to show

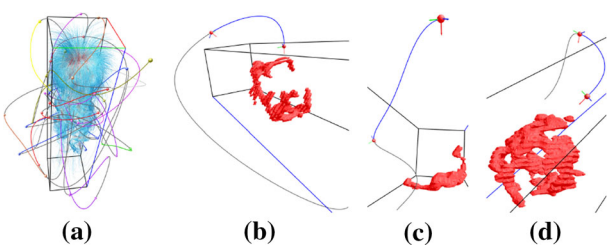

Fig. 4 **a** The entire tour path along the solar plume data set. Different colors denote the path segments along different focal regions. The gray color denotes the transition between two focal regions. **b–d** Three path segments along three different focal regions

their movements along the corresponding pathlines. Pathlets are also rendered as tubes. The tube radius for focal pathlets is larger than that for non-focal pathlets.

6 Results and discussion

In this section, we report the performance, parameters, and case study results generated from three unsteady flow field data sets. For the best evaluation of our approach, please refer to the accompanying videos in the supplemental material for a comparison between our view tour and the baseline tour ("Baseline view tour" section in Appendix). Please also refer to "User study" section in Appendix for a user study.

6.1 Configuration, timing, and parameter setting

We leveraged a hybrid CPU-GPU solution in our computation with the following hardware configuration: Intel Core i7 quad-core CPU running at 3.20 GHz, 24 GB main memory, and an nVidia Geforce GTX 580 graphics card with 1.5 GB graphics memory. Entropy field computation, critical region detection, and viewpoint quality evaluation were implemented using CUDA in the GPU. All other computations were performed in the CPU. For better observation of the changes of the focal region over time, we increase (decrease) the radii of the pathlets from the current (previous) focal region. In order to guarantee smooth update of the changes of pathlets, we utilized the vertex buffer object (VBO) to render pathlets and used the GPU to process their geometry changes on the fly. The timing results for the three flow data sets are reported in Table 1. As we can see, the total processing time for each data set is up to around 15 min. The processing time depends on both the size and complexity of the data set. Entropy computation is mainly related to the size, while region correlation, skeleton extraction, and viewpoint evaluation are highly related to the number and size of critical regions. Therefore, it is possible to further reduce the processing time by considering smaller critical regions (e.g., increasing the entropy threshold for critical region identification).

Our algorithm requires the setting of multiple parameters. Some of them are equation weights, e.g., λ_S in Eq. 4, γ_τ in Eq. 11, and λ_1 in Eq. 13. For such parameters, we empirically set the same parameter values for all data sets, reported in Sects. 4 and 5. Other parameters are thresholds and size values, e.g., entropy value threshold δ_e to determine the critical regions and the overlap threshold δ_o to identify temporal correspondence. In our experiment, we use $\delta_o = 0.1$ for all data sets and Appendix$\delta_e = 4.0, 3.5, 4.0$ for the supernova, hurricane and solar plume data sets, respectively. We refer the readers to "Parameter influence" section in Appendix for the influence of parameters.

6.2 Case studies

Case study 1: supernova The supernova data set comes from a simulation of the explosion and collapse of a supernova. The simulation produced 105 time steps revealing how the dust collapsed back into the center of the star after the supernova explosion. Since the flow near the supernova's core is heavily turbulent, it is difficult for users to observe detail pattern around the core due to the occlusion and clutter among pathlets. Therefore, we designed an automatic view path to explore the data set in a hope that our method can provide users with more information on the internal flow pattern. In Fig. 6a, we can see the overall pattern of the

Table 1 Timing results for the three unsteady flow data sets. ST, VP, TS, and "reg." stand for spatiotemporal, viewpoint, time steps, and regions, respectively

Data set	Dimension	Entropy comp.	Region detect.	# ST reg.	Average duration	Reg. corr.
Supernova	$108 \times 108 \times 108 \times 105$	225.7s*	31.8s	277	3TS	124.8s
Hurricane	$100 \times 100 \times 20 \times 48$	21.7s	4.2s	49	4TS	3.7s
Solar plume	$63 \times 63 \times 256 \times 28$	63.7s	9.6s	17	7TS	37.9s

Data set	Skeleton extract.	Traversal order	VP creat.	Total # lines	# Best VPs/ # total VPs	VP eval.	Tour gen.
Supernova	109.7s	2.6s	0.4s	9989	105/4575	500.1s	0.1s
Hurricane	3.4s	0.7s	0.1s	8244	48/2676	273.8s	0.02s
Solar plume	387.6s	3.7s	0.5s	6368	56/2615	248.7s	0.1s

A * denotes out-of-core processing

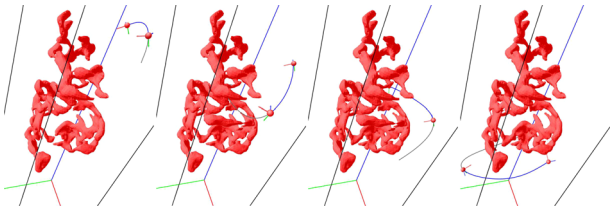

Fig. 5 Left to right: the tour path segments in order along the same focal region of the solar plume data set

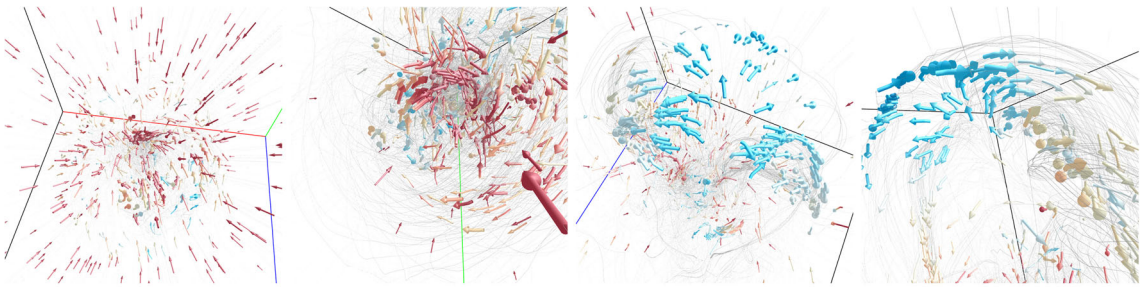

Fig. 6 **a–d** Snapshots of the animation of the supernova data set along our automatic tour path

supernova at an early time step. It is clear that all pathlets go straightly inward and then become turbulent near the core. The thicker pathlets indicate that they are from the current focal region. We also display the corresponding pathlines to help users better follow flow trajectories. In (b), a sink-like region which absorbs all pathlets can be clearly observed. From (c), we can observe that some pathlets are repelled from the core. They then form a semi-spherical surface around the core. For automatic tours, this feature is difficult to catch since it is not always present throughout the time series. Furthermore, it is also hidden inside of the turbulent flow, and therefore is not visible if the viewpoints are placed outside of the volume. Using our method, we can not only clearly observe such an interesting flow pattern in a closeup view but also detect the changes of velocity. In (d), it is clear that the velocity changes from high to low as the flow is repelled from the core.

Case study 2: hurricane The hurricane data set comes from a simulation of Hurricane Isabel, a strong hurricane in the western Atlantic region in September 2003. The simulation produced 48 time steps, demonstrating how the hurricane moved from the Atlantic Ocean to the east coast of Florida. Since the data set is flat ($100 \times 100 \times 20$), users would easily get disoriented if the tour goes across the volume frequently along the z direction. Therefore, we add one more constraint that the final view path should not cross through the volume more than once along the z direction during any given animation time interval. In Fig. 7a, we show the entire tour path. We can see that the most portion of the path is on the two sides of the volume and the path only traverses across the volume a few times. In (b), the global pattern of the hurricane is clearly shown. We can see that the hurricane's center lies in one corner of the volume with the corresponding pathlets moving out. (c) shows the hurricane's center from below. We can observe the spiral pattern and the velocity difference between the center flow (slower) and the surrounding flow (faster). From

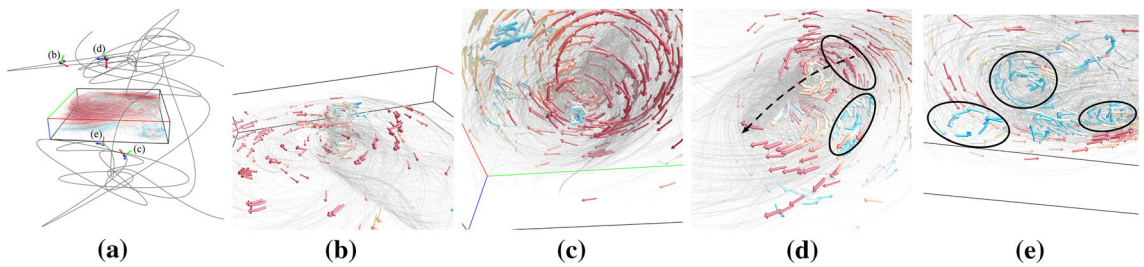

Fig. 7 **a** The automatic tour path for the hurricane data set. **b–e** Snapshots of the animation along the path

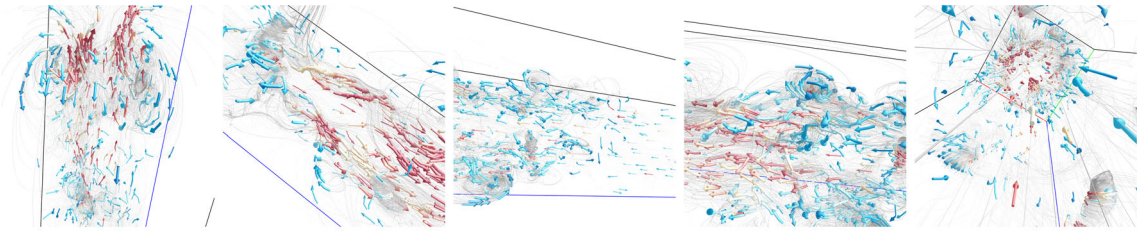

Fig. 8 **a–e** Snapshots of the animation of the solar plume data set along our automatic tour path

(d), we can find out that the surrounding flow around the center bifurcates into two opposite directions (highlighted in the ellipses). The velocity change could also be discerned via pathline color. Besides, the pathlines depict the moving trajectory of the center: from one corner of the volume to another along the diagonal direction (highlighted by the dashed line). In (e), the snapshot shows three small spirals with slower speed (highlighted in the ellipses) at the boundary of the volume which only last for a few time steps. These features could be easily missed if the view path does not focus on them at the right moment.

Case study 3: solar plume The solar plume data set comes from a simulation of the down-flowing solar plume for studying the heat, momentum, and magnetic field of the sun. This data set consists of 28 time steps, demonstrating the heat flow emitting from the sun's surface. Figure 8 shows snapshots of the animation along the automatic tour path. Refer to Fig. 4a for the overall view path generated by our algorithm. Since the tail of the solar plume only contains straight flow lines and most interesting features are around the head, our view path is almost around the head portion so that users can gain a clear observation of these important flow features. In (a), an overview of the flow pattern around the head region is captured. Two big swirls with slower speed and the central flow with higher speed are clearly shown, (b) focuses on the central flow pattern around the head, (c) gives us a glance on how the straight-line flow becomes turbulent in the middle portion of the plume. From (d), we can observe a spiral band pattern with slower speed in the middle portion of the plume. Our view tour also provides users with an expressive traversal experience to observe the internal "kernel" flow pattern by "standing" inside of the volume, which is shown in (e). The straight flow lines moving to the head and the hollow shaft pattern are clearly visible.

7 Expert evaluation

We also conducted an evaluation with a domain expert in fluid mechanics and turbulent combustion to evaluate the effectiveness of our method. The expert's research focuses on the modeling of multiscale and multiphysics problems in relation to energy science and technology. The evaluation consists of two major stages. At the first stage, the expert spent about 90 min learning our framework, playing back tour animation video clips multiple times, and asking questions related to the tour generation algorithm and the actual pathlet animation as shown in the video. At the second stage, the expert completed the evaluation form which includes 18 questions (hurricane: seven, solar plume: five, and supernova: six) and 14 general questions. The 18 questions ask for the identification of flow features, flow speed difference, and additional flow patterns. The 14 questions ask for the overall effectiveness of the solution, animation settings, and suggestions for improvement. This stage took about 120 min. The expert was instructed to show no bias in his evaluation.

The expert provided the following feedback. In general, the tours we generated for the unsteady flow fields are interesting and effective. The expert's answers to the 18 questions are all correct except one which asked him to identify the overall moving trajectory of the hurricane's center. He provided a comment which explains "Due to the camera angle change, the movement of the hurricane's center was not clearly noticed." He also gave a suggestion "It may help to show the same pathlets in another (third) view where the camera is fixed." For effectiveness, he strongly agreed that the tour provides a comfortable way to explore the flow field: "It automatically captures the important flow features and the most appropriate angle of view to visualize the features." Nevertheless, he also pointed out one limitation: although the tour also captures the small-scale features, when important small-scale features are inside large-scale features which are also important, it appears that automatic selection of pathlets may miss these structures. He agreed that parameter

settings for pathlines and pathlets, and camera and animation are appropriate. He strongly agreed that it is appropriate to pause the camera at each selected viewpoint in order to gain clear observation of the moving pathlets. Finally, he strongly agreed that the path view helps users better orient themselves in the tour animation.

The expert also pointed out two suggestions for improvement. First, a better hint for time steps in the path view could help user understanding. Since we actually interpolate between time steps to generate a number of intermediate frames for a smooth animation effect, it would be more appropriate to display time steps including fractions. (We implemented this. Refer to the accompanying video.) Second, the current algorithm is fully automatic which helps users grasp the main features of the flow field with minimal efforts. Once this is done, it would be good to give options for user interaction. For instance, users may want to choose some features and focus on them for all time steps (perhaps, with different viewing angles and zooming levels).

The expert further commented on the application of this automatic tour framework. One of his main areas of interest is the computation and modeling of turbulent combustion. Turbulent combustion simulations usually generate big data. The physicochemical processes are inherently transient and involve complex interactions of fluid flow and chemistry. Exploring such transient data poses a substantial challenge. Traditionally, statistical approaches where time or spatial averages, variances, and correlations of certain key quantities are investigated. To better interpret the statistical data and shed light on relevant physical mechanisms, the exploration of key flow and scalar structures in terms of visualization is critical but is practically challenging due to the size of the data and the complexity of the structures. The present tool could offer an interesting and attractive way of exploring the interaction of flow and scalar structures in turbulent flames. The automatic tour of important flow features would provide insights on the overall flow characteristics, using which the expert can connect their observation with statistics of key scalar quantities, with minimal effort. When a domain expert has an option of setting criteria for important features to be explored, e.g., regions with high scalar gradients or high reaction rates, the method would greatly benefit combustion scientists and engineers by helping them understand the characteristics and underlying physical mechanisms in turbulent reacting flows.

8 Conclusions and future work

Designing an automatic tour for exploring hidden or occluded internal features for unsteady flow fields is a challenging task due to the flexibility of viewpoint locations and orientations, the nature of dynamic spatiotemporal features, and the difficulty to observe both viewpoint movement and flow line movement simultaneously. Yet such a tour is quite useful for gaining a good understanding of the underlying flow field, including both the overview and feature details. As the size and complexity of flow field data keep growing, we expect an increasing need to observe flow features and pattern in an automatic fashion. We have presented such a solution and demonstrated the effectiveness of our work with results gathered from several flow field data sets and through user evaluation. To the best of our knowledge, our work is the first that aims at designing a tour for examining internal flow features for unsteady flow fields.

Our entropy-based feature extraction may not capture certain global flow pattern, e.g., separation. In addition, the computation does not consider the Galilean invariance property of the unsteady flow. There exist seemly featureless flows (e.g., the flow behind a cylinder) which actually consist of rich flow dynamics hidden by some constant flow. For future work, we would like to leverage other techniques to extract such kinds of flow features. For tour exploration, we plan to provide flexibility to users in their viewing and navigation. Instead of a fully automatic tour, users may find it helpful to control the tour in different ways. For example, pausing the tour at selected viewpoints and changing the orientation of viewpoint to look around for a customized exploration. Beside pathline and pathlet rendering, we will also leverage other means of visualization such as particle rendering to enrich the experience of touring an unsteady flow field.

Acknowledgements This research was supported in part by the U.S. National Science Foundation through Grants IIS-1456763, IIS-1455886, and DUE-1833129.

Appendix

Critical region identification

Critical Region Detection

Given the input unsteady flow field, we adopt a 4D moving window ($5 \times 5 \times 5 \times 5$) centered at each voxel and calculate its entropy value by evaluating the variation of vector directions in each local window. Since calculating the entropy value of one voxel is independent of another, we compute the entropy field using CUDA in the GPU.

A critical region at each time step can be defined as a subvolume which contains rich flow features such as the neighborhoods of critical points. Entropy values in a critical region are normally larger than a non-critical region because vector directions located in the critical region have a higher degree of variation. Therefore, we detect critical regions by applying a region growing algorithm. In each time step, we detect all connected voxels with their entropy values larger than a given entropy threshold δ_e as a critical region. Multiple critical regions can be identified for each time step. We discard critical regions whose sizes are smaller than a given threshold. To speed up the computation, we implement critical region detection using CUDA in the GPU. Specifically, each GPU thread takes the responsibility for one voxel in the volume. Each thread checks if the entropy value for the corresponding voxel is larger than δ_e. If yes, the voxel is marked as a critical voxel. At the same time, the boundary critical voxel is also marked if at least one of its neighboring voxels is not critical. Each critical voxel is then assigned a unique ID. Next, for a critical voxel v with ID v_{ID}, we check each of its neighboring critical voxel's ID n_{ID} and set $n_{\mathrm{ID}} = v_{\mathrm{ID}}$ if $n_{\mathrm{ID}} > v_{\mathrm{ID}}$. This is applied to all critical voxels in parallel. After one round, some voxels' IDs are changed to smaller values. We iteratively apply this operation for multiple rounds until there is no ID change for any voxel. At this moment, all voxels with the same ID form a critical region.

Temporal correspondence computation

For an unsteady flow field, a critical region at one time step could span several time steps and overlap other regions in the neighboring time steps. To identify such temporal correspondences, we detect all matching regions between consecutive time steps by computing their overlap rates (Silver and Wang 1998). We point out that for fast moving structures, alternative solutions such as optical flow or mass transport minimization should be used. In our scenario, two regions are matched when their overlap rate is larger than a given region overlap rate threshold δ_o. In this way, we can construct an *overlap table* for every pair of neighboring time steps t and $t+1$ indicating the matching relations (i.e., continuation, bifurcation, amalgamation, dissipation, and creation) among critical regions.

Based on those overlap tables, we further apply the feature tracking algorithm developed by Silver and Wang (1997) to build the temporal correspondence among critical regions. This algorithm considers every two consecutive time steps by examining their overlap table in the order of bifurcation, continuation, amalgamation, dissipation, and creation. When any region has been categorized into one correspondence, we will not consider the same region in the subsequent examination. Finally, based on their temporal correspondence, we merge regions which have the continuation correspondence into a new critical region. We also identify a region's parents or children if there is a bifurcation or amalgamation correspondence. Figure 9 shows an example of the temporal correspondence of critical regions.

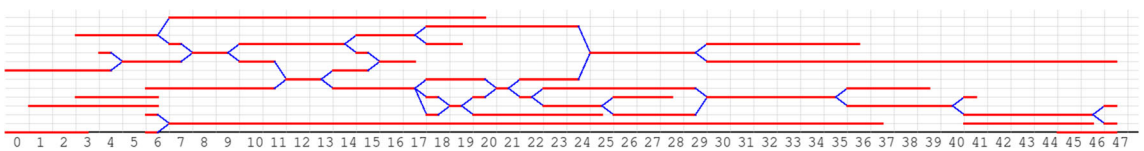

Fig. 9 The temporal correspondence of critical regions extracted from the hurricane data set. The horizontal axis represents time step. Each red line represents the lifespan of one critical region

Region skeleton extraction

In order to identify the shape of each critical region detected, we construct its skeleton by applying the volume thinning algorithm presented by Gagvani and Silver (1999). For each voxel, the algorithm first computes the *distance transform* (DT) value, which is the minimum of the distances from the current voxel to all boundary voxels. We identify a voxel as a skeleton point if its DT value is larger than the summation of the average of its neighboring voxels' DT values and a predefined thinning parameter δ_t. Obtaining all skeleton points for each critical region, we employ a *minimum spanning tree* (MST) algorithm to connect them to form the skeleton. We also define the *major axis* of a skeleton as a line connecting the two endpoints which have the longest Euclidean distance on the skeleton. The major direction of the skeleton is the vector starting from one endpoint of the skeleton with the lower y value to the other one. In practice, for a critical region which spans several time steps, we extract its 3D skeleton at each time step separately.

Skeleton-based seeding

In order to trace pathlines, we apply a skeleton-based seeding strategy by evenly placing seeds along the skeletons extracted from critical regions. Our goal is to make sure that each focal region at every time step has enough pathlets to represent itself for clear highlighting, and the entire flow field has approximately the same number of pathlets for each time step. To achieve this, we place two types of seed: *region seeds* and *random seeds*. First, we place region seeds along the skeleton of each focal region detected at every time step (refer to Sect. 4 in the paper) and ensure that the number of seeds for each region is proportional to its size. Furthermore, to ensure that the traced pathlets will capture each focal region, we actually start to trace region seeds a few time steps before the region is being focused on. For random seeds, we keep track of the number of pathlets in each time step. The number of pathlets could decrease as the time evolves since pathlets may go out of the domain boundary or get absorbed around the vicinity of a critical point (such as a sink). If the number is less than a given threshold N_p, we will add some more seeds randomly. This keeps the overall pathline number relatively stable at each time step. In Fig. 10, we compare skeleton-based seeding against random seeding. It is evident from (a) and (b) that skeleton-based seeding generates more pathlines from interesting flow regions. The same conclusion can be drawn when comparing (c) and (d). From regions highlighted in red boundaries, we can see that the skeleton-based seeding favors regions with more intricate flow patterns while placing fewer streamlines at the bottom region where the flow pattern is almost straight.

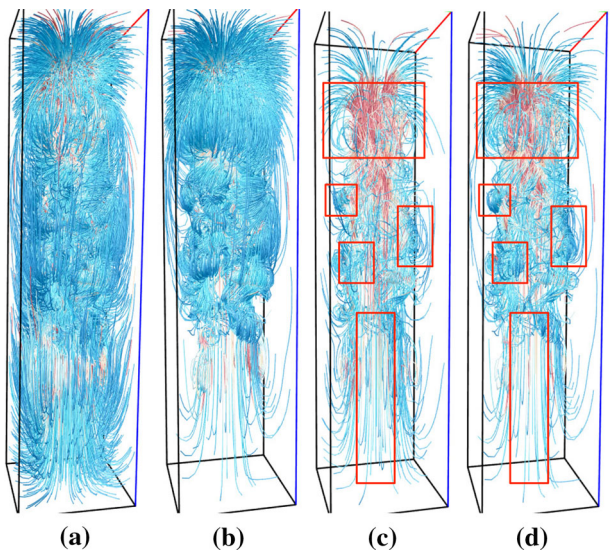

Fig. 10 **a**, **b** Pathlines traced from random and skeleton-based seeding, respectively, over all time steps (each 4000 lines). **c**, **d** Pathlines traced from random and skeleton-based seeding, respectively, from the first three time steps (each 300 lines). Note that skeleton-based seeding only applies to the focal region, while the remaining regions are still randomly seeded. Blue to gray to red are mapped to pathline colors for slow to medium-to-high-velocity magnitudes

Parameter influence

We briefly discuss the effects of the parameters and how we set their values. We categorize the parameters into three sets based on which stage of the algorithm they are applied to:

- *Region extraction parameters.* This set of parameters controls region extraction based on the entropy field and temporal correspondence of these regions. The entropy threshold δ_e allows users to control the extracted region size at different levels. The thinning parameter δ_t is used to control the point density of the region's skeleton. The overlap rate δ_o determines if two regions are overlapped and therefore controls the final region temporal correspondence.
- *Region traversal-order determination parameters.* This set of parameters determines the final scores of spatiotemporal regions in the linear system. They include the factor weights (e.g., λ_S for the intrinsic property region size $S_{r,t}$, and γ_v for the traversal frequency constraint O_v in the linear system), the time window size W_s, and the maximum number of backtracking time windows L_b used in the normalized traversal frequency computation. Changing the parameter values will vary the influence of corresponding factors (e.g., the region size or the region time span) in the linear system and affect the final region scores. Since the traversal order is determined based on the region scores, manipulating these parameters will finally impact the region traversal order.
- *Viewpoint creation parameters.* This set of parameters controls the properties of the final viewpoints (e.g., position, `look-at` direction, and viewpoint score). The mesh decimation factor δ_s determines the number of vertices (viewpoints) for the simplified mesh. The offset threshold δ_l determines the distance from the created viewpoints to the focal region. λ_1, λ_2, and λ_3 control the weights of different factors in the viewpoint score computation. The thresholds δ_α and δ_d are used to control the final quality of the selected viewpoints.

Table 2 shows all parameters for each data set. We point out that although multiple parameters exist in our algorithm, users are not required to adjust them for each data set. Actually, the motivation for providing such parameters is to offer advanced users the capability to explore the flow field based on their own requirements. In practice, most of the parameters in our case studies remain the same for all data sets. We refer to these parameters as data-independent parameters. Contrarily, the data-dependent parameters are the ones which should be adjusted based on the given data set. There are only a few of data-dependent parameters, e.g., W_s (depending on the total time steps) and δ_d (depending on the dimension of the data set).

Data-independent parameters are specifically designed for advanced users to control the final exploration. In Fig. 11, we give an example on how the size weight λ_S in the linear system controls the final region-order selection for the hurricane data set. The three images in the first row show the focal regions (highlighted in red) of three continuous time steps based on the original value ($\lambda_S = 1$) used in our experiment and the three images in the second row show the focal regions for the same time steps using a larger value ($\lambda_S = 5$). Therefore, when emphasizing the size factor in the linear system, our algorithm prefers to select regions with larger sizes. Figure 12 demonstrates the influence of the temporal correspondence weight γ_η to the final region traversal order for the solar plume data set. Setting a larger value for the parameter will allow the linear system to prefer a region which is not the child of the previously selected region. The first row highlights the focal regions in red for three consecutive time steps. With a small value

Table 2 Parameter setting for all data sets

Data set	Entropy δ_e	Overlap δ_o	Thinning δ_t	#Lines N_p	Win size W_s	# Win L_b	Decimation δ_s	Offset δ_l	Angle δ_α	Dist δ_d
Supernova	4.0	0.1	1.99	500	5	5	0.033	$V_d/10$	$\pi/6$	20.0
Hurricane	3.5	0.1	1.90	450	4	4	0.033	$V_d/15$	$\pi/6$	15.0
Solar plume	4.0	0.1	1.95	450	4	4	0.033	$V_d/12$	$\pi/6$	20.0

Data set	SC α	IP λ_S	IP λ_E	IP λ_V	IP λ_C	IP λ_T	EF γ_τ	EF γ_σ	EF γ_v	EF γ_η	EF γ_ξ	VPQ λ_1	VPQ λ_2	VPQ λ_2
Supernova	0.7	1	1	1	1	2	1	1	1	2	1	1	0.2	0.2
Hurricane	0.7	1	1	1	1	2	1	1	1	2	1	1	0.2	0.2
Solar plume	0.7	1	1	1	1	2	1	1	1	2	1	1	0.2	0.2

V_d, SC, IP, EF, and VPQ stand for the volume diameter, skeleton complexity, intrinsic property, energy function, and viewpoint quality, respectively

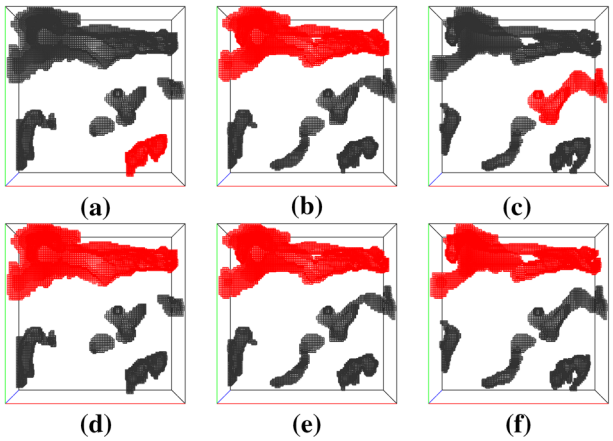

Fig. 11 Two sequences of focal region ordering based on two different size weight λ_S values. The first row **a–c** shows the focal region (highlighted in red) sequence in three continuous time steps with the default λ_S value. The second row **d–f** shows a different sequence with a larger λ_S value favoring regions of larger size

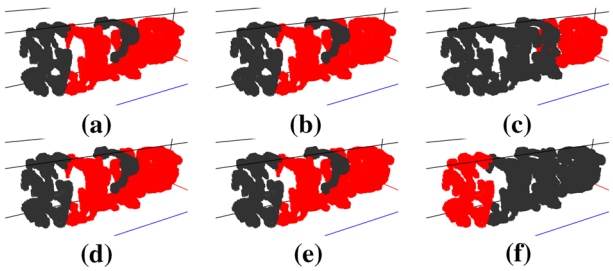

Fig. 12 Two sequences of focal region ordering based on two different temporal correspondence weight γ_η values. The first row **a–c** highlights the focal region (highlighted in red) sequence in three consecutive time steps with the default γ_η value. The focal region in the last time step **c** is the child of the selected region in the previous time step (**b**). The second row **d–f** shows a different sequence with a larger γ_η value which prefers a region that is not the child of the previously selected region

($\gamma_\eta = 2$), the system picks the child of the previously selected region in the last time step. On the contrary, a different region is selected at the same time step if a larger value ($\gamma_\eta = 5$) is set, as shown in the second row.

User study

We conducted a between-subjects user study to evaluate the effectiveness of our method. We used a design of 2 conditions (our view tour and baseline view tour) $\times 2$ tasks (answering questions and identifying critical regions). We recruited 14 users and separated them into two groups: one group of seven users evaluated our view tours and another group of seven users evaluated baseline view tours. Each user was paid $10 for participating in the study. All users are graduate students from different departments (computer science, mechanical engineering, physics) of a university.

Baseline view tour

For the baseline view tour, we first generate a set of viewpoints whose positions are randomly picked both inside and outside of the volume. For external viewpoints, we also keep them not too far away from the volume's center to ensure clear observation of the flow field. The `look-at` center of each viewpoint is also randomly created. However, we constrain their positions to be inside of the volume so that the viewpoints could still focus on the flow field rather than an empty space outside. Next, we connect all these viewpoints in a way such that both the Euclidean distance between viewpoints and the angle change along the path could be minimized. Finally, we interpolate a B-spline curve passing all the viewpoints to generate the tour path. If the total length or the angle change of the baseline view tour is much different from our method, we

will regenerate the path by replacing some viewpoints. Since the viewpoints for the baseline view tour are not selected intentionally for observing any particular flow pattern, we expect that the baseline view tour is unable to guarantee a comprehensive exploration of internal flow features.

Experimental procedure

Before the experiment started, users had been briefly introduced the basic characteristics of unsteady flow fields, the types of critical regions they need to recognize, and the tasks they would be expected to perform in the experiment. After the briefing, users could ask questions about the flow fields or the experiment itself to avoid possible misunderstanding. Then the main study commenced, consisting of tasks for three data sets. For the first data set, a short demonstration of how to use the program was given. During the experiment, users were only allowed to ask questions for clarification about the meaning of the specific questions or how to use the program.

The procedure for each data set was as follows. First, users were shown an animation of the complete tour of the data set without stop or pause. The speed of the animation could be adjusted if desired. Second, users were given a limited time to answer several multiple-choice questions about the flow field and to identify critical regions. They could revisit any part of the tour using a slider or replay the animation. This function was helpful for answering questions and was required for identifying critical regions. Finally, users were asked to answer several post-experiment questions about subjective feedback and suggestions for improvement. They should complete all three data sets in one sitting. The entire experiment took approximately an hour for each user, including the initial paperwork, briefing, and post-experiment questionnaire.

Results and discussion

We present the results of this study in the following aspects: user correctness on multiple-choice questions, ratings of subjective questions, the proportion of critical regions correctly identified, and the post-experiment feedback. Because user performance varied widely by data set, each data set was analyzed individually, comparing user performance between our tour and the baseline tour. We used Student's t-test to analyze statistical significance between the conditions with a significance level $\alpha = 0.05$. Although there are objections to using the t-test on small samples (seven in our case), several articles (Campbell et al. 1995; Janusonis 2009; Fritz et al. 2012) argued that there are no principle objections to use a t-test for a small sample size even though the statistical power may not be high.

Multiple-choice questions Each data set was analyzed individually by comparing users' average proportion of correct answers by the two methods. The hypothesis is that there is a significant difference of the answer correctness rate between our method and the baseline method. The average correctness rates of all users for the two methods are given in Fig. 13a. A t-test shows that our method performed better than the baseline one on the hurricane and solar plume data sets and the differences are statistically significant with $p \ll 0.001$ and $p = 0.045$, respectively. For the supernova data set, although our method also received higher average correctness rate than the baseline method (0.63 vs. 0.54), the difference is not statistically significant.

Subjective questions There are two subjective questions for each data set asking the effectiveness of finding critical regions and identifying the global flow pattern. The subjective questions were also analyzed

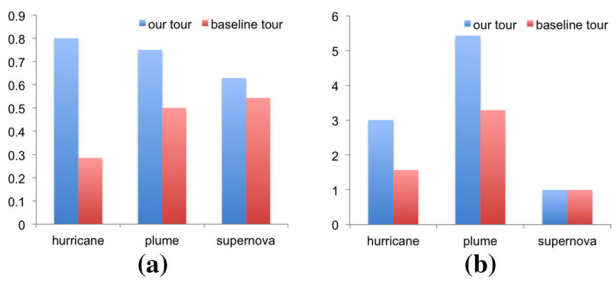

Fig. 13 **a** The average proportion of correct answers of multiple-choice questions. **b** The average number of critical regions detected

individually for each data set. We quantized the answers by setting 1.0 for "Strongly Agree", 0.75 for "Agree", 0.5 for "Neutral", 0.25 for "Disagree", and 0.0 for "Strongly Disagree". The hypothesis is that there is a significant difference of the rating score between our method and the baseline method. For the hurricane data set, our method gets much higher average ratings than the baseline one for both questions (0.93 vs. 0.64 and 0.79 vs. 0.61). But only the first question has significant difference with $p = 0.039$. Our method also receives a higher rating than the baseline one for the solar plume data set with average ratings 0.75 vs. 0.57 and 0.68 vs. 0.61 though no significant difference is shown. For the supernova data set, both methods received similar scores.

Critical region identification Similarly to the multiple-choice questions, each data set was analyzed individually for critical region identification. The analysis was done by comparing how many critical regions the users correctly identified. The average number of critical regions detected for both methods is given in Fig. 13b. The hypothesis is that there is a significant difference in the average number of identified critical regions between our method and the baseline method. The supernova data set gets the same result for both groups since it contains one enormous critical region that is easily identifiable by all users. For the other two data sets, users performed better with our method. The average number of detected critical regions is 3.00 vs. 1.57 for the hurricane data set and 5.43 vs. 3.29 for the solar plume data set. Furthermore, the performance difference for the solar plume is significant with $p = 0.04$ where the same conclusion cannot be made for the hurricane because its p value is only slightly above the significance level.

Post-experiment feedback We received the following major user comments from the post-experiment questionnaire. First, for both our and baseline methods, most users suggested that the camera may stay longer at each of the selected viewpoints to allow better observation and less visual jumping. Second, the background pathlines should be thinner for reducing distraction. Third, some users also suggested providing a global view of the data set before the experiment since the tour path may focus on the internal pattern rather than the global shape of the flow field. Users also had some different opinions on the two methods. For the baseline method, users gave the neutral rating for detecting flow features. One of them claimed that the `look-at` direction sometimes provided an unreasonable view of sight for observing the flow field. By contrast, most users agreed that the tour generated by our method could easily help them identify critical regions. For other questions, such as animation speed and pathlet size, both groups were satisfied with the current configurations.

Discussion In summary, for the multiple-choice questions, users in general performed better with our method than the baseline method. This indicates that our view tour indeed provides users with more information about the underlying flow field. For subjective questions, most users agreed that our method better help them detect critical regions and identify the global flow pattern than the baseline method for the hurricane and solar plume data sets, though the latter one did not have a significant difference. The supernova data set received almost the same rating for both tours due to the simple flow feature which could be easily observed with either method. In terms of identifying critical regions, users performed much better with our method over the baseline method except for the supernova data set which only contains a single obvious sink-like region at the center. From the post-experiment feedback, except for suggestions on animation and interface, most users gave positive feedback to our method over the baseline one.

We conclude that our view tour indeed helps users better identify and observe the internal flow pattern in unsteady flow fields, especially for hidden or occluded features that only exist for a short period of time. Therefore, our view tour could complement the traditional overview tour by providing users with a more comprehensive exploration experience for large and complex flow fields.

References

Bai Z, Yang R, Zhou Z, Tao Y, Lin H (2016) Topology aware view path design for time-varying volume data. J Vis 19(4):797–809

Bordoloi UD, Shen HW (2005) View selection for volume rendering. In: Proceedings of IEEE visualization conference, pp 487–494

Buatois L, Caumon G, Lévy B (2009) Concurrent number cruncher—a GPU implementation of a general sparse linear solver. Int J Parallel Emergent Distrib Syst 24(3):205–223

Campbell MJ, Julious SA, Altman DG (1995) Estimating sample sizes for binary, ordered categorical, and continuous outcomes in two group comparisons. Br Med J 311(7031):1145–1148

Chandler J, Obermaier H, Joy KI (2015) Interpolation-based pathline tracing in particle-based flow visualization. IEEE Trans Vis Comput Graph 21(1):68–80
Fritz CO, Morris PE, Richler JJ (2012) Effect size estimates: current use, calculations, and interpretation. J Exp Psychol Gen 141(1):2–18
Gagvani N, Silver D (1999) Parameter-controlled volume thinning. Graph Models Image Process 61(3):149–164
Hsu WH, Zhang Y, Ma KL (2013) A multi-criteria approach to camera motion design for volume data animation. IEEE Trans Vis Comput Graph 19(12):2792–2801
Janusonis S (2009) Comparing two small samples with an unstable, treatment-independent baseline. J Neurosci Methods 179(2):173–178
Ji G, Shen HW (2006) Dynamic view selection for time-varying volumes. IEEE Trans Vis Comput Graph 12(5):1109–1116
Lane DA (1993) Visualization of time-dependent flow fields. In: Proceedings of IEEE conference on visualization, pp 32–38
Lee TY, Mishchenko O, Shen HW, Crawfis R (2011) View point evaluation and streamline filtering for flow visualization. In: Proceedings of IEEE pacific visualization symposium, pp 83–90
Lorensen WE, Cline HE (1987) Marching cubes: a high resolution 3D surface construction algorithm. In: Proceedings of ACM SIGGRAPH conference, pp 163–169
Ma J, Wang C, Shene CK (2013) Coherent view-dependent streamline selection for importance-driven flow visualization. In: Proceedings of IS&T/SPIE conference on visualization and data analysis, pp 865407:1–865407:15
Ma J, Walker J, Wang C, Kuhl SA, Shene CK (2014) FlowTour: an automatic guide for exploring internal flow features. In: Proceedings of IEEE pacific visualization symposium, pp 25–32
Marchesin S, Chen CK, Ho C, Ma KL (2010) View-dependent streamlines for 3D vector fields. IEEE Trans Vis Comput Graph 16(6):1578–1586
McLoughlin T, Laramee RS, Peikert R, Post FH, Chen M (2010) Over two decades of integration-based, geometric flow visualization. Comput Graph Forum 29(6):1807–1829
McLoughlin T, Jones MW, Laramee RS, Malki R, Masters I, Hansen CD (2013) Similarity measures for enhancing interactive streamline seeding. IEEE Trans Vis Comput Graph 19(8):1342–1353
Meuschke M, Engelke W, Beuing O, Preim B, Lawonn K (2017) Automatic viewpoint selection for exploration of time-dependent cerebral aneurysm data. In: Bildverarbeitung für die Medizin. Springer, Berlin, Heidelberg, pp 352–357
Moberts B, Vilanova A, van Wijk JJ (2005) Evaluation of fiber clustering methods for diffusion tensor imaging. In: Proceedings of IEEE visualization conference, pp 65–72
Post FH, Vrolijk B, Hauser H, Laramee RS, Doleisch H (2003) The state of the art in flow visualisation: feature extraction and tracking. Comput Graph Forum 22(4):775–792
Sadlo F, Rigazzi A, Peikert R (2011) Time-dependent visualization of Lagrangian coherent structures by grid advection. In: Topological methods in data analysis and visualization. Springer, Berlin, Heidelberg, pp 151–165
Silver D, Wang X (1997) Tracking and visualizing turbulent 3D features. IEEE Trans Vis Comput Graph 3(2):129–141
Silver D, Wang X (1998) Tracking scalar features in unstructured data sets. In: Proceedings of IEEE visualization conference, pp 79–86
Spencer B, Laramee RS, Chen G, Zhang E (2009) Evenly spaced streamlines for surfaces. Comput Graph Forum 28(6):1618–1631
Steinman DA (2000) Simulated pathline visualization of computed periodic blood flow patterns. J Biomech 33(5):623–628
Takahashi S, Fujishiro I, Takeshima Y, Nishita T (2005) A feature-driven approach to locating optimal viewpoints for volume visualization. In: Proceedings of IEEE visualization conference, pp 495–502
Tao J, Ma J, Wang C, Shene CK (2013) A unified approach to streamline selection and viewpoint selection for 3D flow visualization. IEEE Trans Vis Comput Graph 19(3):393–406
Viola I, Feixas M, Sbert M, Gröller ME (2006) Importance-driven focus of attention. IEEE Trans Vis Comput Graph 12(5):933–940
Xu L, Lee TY, Shen HW (2010) An information-theoretic framework for flow visualization. IEEE Trans Vis Comput Graph 16(6):1216–1224
Yu H, Wang C, Shene CK, Chen JH (2012) Hierarchical streamline bundles. IEEE Trans Vis Comput Graph 18(8):1353–1367