Interactive Tracking of Three-dimensional Critical Points in Unsteady, Large-scale CFD Datasets

Andreas Gerndt*
Louisiana Immersive Technologies Enterprise
537 Cajundome Blvd., Lafayette, LA 70506

Torsten Kuhlen†
Virtual Reality Group
Aachen University, Germany

Carolina Cruz-Neira‡
Louisiana Immersive Technologies Enterprise
University of Louisiana at Lafayette

ABSTRACT

Vector field topologies are very important to obtain an insight into the dynamic behavior of flow fields. Their essential features are critical points depicting locations where the velocity vanishes. Detecting critical points is a well-known technique. In discrete unsteady datasets, the assessments of the topology are only carried out on each single simulated or measured time level. The real-time tracking of 3D topology features from one time level to the next is generally not offered by available post-processing toolkits. We present a framework particularly for engineering scientists aiming at the interactive analysis of complex, time-varying flow field topologies in virtual environments. The approach to extract critical point trajectories over time is based on local similarity estimation using several quantities of the velocity gradient tensor. We compare and validate these quantities is this paper as well.

Index Terms: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Virtual reality; G.1.3 [Numerical Analysis]: Numerical Linear Algebra—Eigenvalues and eigenvectors; I.4.8 [Image Processing and Computer Vision]: Scene Analysis—Tracking

1 Introduction

The analysis of local flow patterns is essential for the understanding of global flow motions. Fundamental parts of the flow field topology are critical points defining where the velocity vanishes. A basic classification uses the velocity gradient tensor, also called as the Jacobian of the velocity field, at each critical point. The assessment of the eigenvalues of the Jacobian distinguishes between critical point types of spirals and nodes. These can then be subdivided into further sub-classes depending on the signs of the eigenvalues.

The velocity gradient tensor can also be classified by three matrix invariants. In contrast to the eigenspace, which may also consists of complex eigenvalues and eigenvectors, invariants are always real numbers. They define a further three-dimensional coordinate space for the evaluation of critical points. An in-depth discussion of invariants for the analysis of any smooth three-dimensional vector field is given by Chong et al. in [12] and [2].

Visualizing critical points of 2D flows was introduced by Helman and Hesselink (cf. [8]). In 1991, Globus et al. [7] presented an extraction approach of 3D critical points. Non-linear vector fields were considered by Scheuermann et al. in [15], and the work of Theisel et al. (cf. [18, 22]) aimed at the visualization of the global three-dimensional vector field topology.

Tracking of unsteady features in discrete datasets was investigated first by Deborah Silver and Xin Wang (cf. [16]). Later on, Kopp Reinders presented an integrated framework for feature-based visualization of time-dependent data (cf. [13]). Particle tracing plays an important role particularly for the analysis of unsteady flow fields. Papers about particle integration and visualization over time were published e.g. by Lane et al. (cf. [10]) and Sadarjoen et al. (cf. [14]). In 2002, Trichoche introduced unsteady topology visualization and tracking for 2D vector fields in [19]. An extension to 3D tetrahedral grids was later on published in [4]. A further application for critical point tracking was presented by Klein and Ertl [9] which is based on the general feature tracking work of Theisel and Seidel [17] and shows how to identify related critical points on different levels of multi-resolution datasets.

The idea of this paper is primarily based on the geometrical interpretation of the complex eigenspace that classifies critical points. We present an approach that identifies successors of 3D critical points on discrete time levels of an unsteady dataset mainly by comparing their eigenvalues and eigenvectors as well as the invariants of the characteristic polynomials. Further quantities of the velocity gradient tensor (cf. [3]) like curl and divergence are also available. Based on derived values, similarities are searched and tracking events are defined. Our approach follows the visualization pipeline concept and uses the output result of the critical point extraction filter for the similarity test. Therefore, it is grid type independent. As virtual reality techniques offer the most intuitive way for an exploratory analysis of complex flow fields (cf. [20]), the described algorithm aims at interactive visualization and animation even of a large amount of critical points.

The remaining paper is structured as follows: the next section gives a survey of the eigensystem, invariants, and the α-β space for the classification of critical points. It is followed by a description how critical points can be rendered as icons and how this approach is enhanced for animated critical point tracking. Section 4 presents the quantities and the similarity algorithm to identify successors, before section 5 shows results with several datasets. This paper is closed by a conclusion and a perspective for future work.

2 Critical Points

Critical points are locations within a flow field where the velocity vanishes. These zero velocity locations are the anchor features for the definition of the flow field's topology. Once identified, a critical point can be classified using the eigensystem of the gradient velocity tensor at its position. How to compute and classify first order critical points is described in [6].

Based on the eigenvalues and eigenvectors, further topology structures can be determined (cf. [18]). Also the interpolated animation, presented later in this paper, primarily operates on the geometrical interpretation of the eigensystem.

2.1 Eigensystem

After critical points are detected, the velocity gradient tensor J (also called as the Jacobian of the velocity field) can be determined for each of them (cf. equation 1).

$$J = \nabla v = \begin{pmatrix} \partial v_x/\partial x & \partial v_x/\partial y & \partial v_x/\partial z \\ \partial v_y/\partial x & \partial v_y/\partial y & \partial v_y/\partial z \\ \partial v_z/\partial x & \partial v_z/\partial y & \partial v_z/\partial z \end{pmatrix} \quad (1)$$

*e-mail: andreas@lite3d.com
†e-mail: kuhlen@rz.rwth-aachen.de
‡e-mail: carolina@lite3d.com

This 3-by-3 matrix can be considered as a measurement for the rate of deformation in a flow field at the given position, in our case around a critical point. It can be used to determine important quantities of the flow field such as vorticity, divergence, and shear strain. For the classification of critical points, the eigenvalues λ of the velocity gradient tensor are needed. These can be computed by solving the first-order differential equation given in equation 2.

$$J\,e = \lambda\,e \tag{2}$$

The term e denotes the eigenvector corresponding to the eigenvalue λ. For the considered three-dimensional flows, equation 2 leads to three eigenvalues and three eigenvectors. The eigenvalues can be all real (which yields a so-called node), or one is real and the other both are conjugate-complex (called as a spiral or focus). Using the complex plane, patterns as shown in figure 1 and figure 2 can be distinguished.

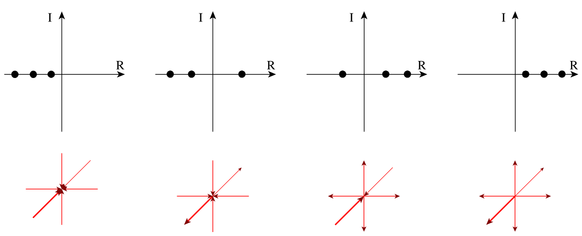

Figure 1: Eigenvalues positions on the complex plane and corresponding flow patterns for nodes.

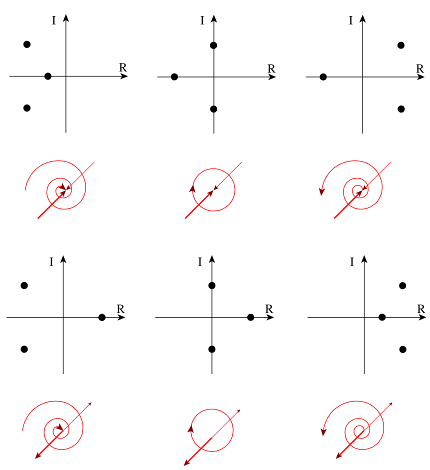

Figure 2: Spiral patterns with corresponding complex plane patterns, complex eigenvalues are moving from left to right, the real eigenvalue changes its side from upper to lower row.

Additionally to the depicted patterns, the transition variants are also possible where one of the real or the one real eigenvalue is zero. In general, all critical points which have eigenvalues with a zero real component are considered unstable and can usually not be found in discrete flow field datasets. This also concerns the two center patterns in figure 2 (mid column).

2.2 Invariants

Solving equation 2, the characteristic polynomial is used:

$$P_J(\lambda) = \lambda^3 + P\lambda^2 + Q\lambda + R = 0 \tag{3}$$

The terms P, Q, and R are the so-called invariants of the Jacobian. Actually, P is the negative trace, Q is the sum of the minors, and R is the negative determinant of J.

It is possible to transform the Jacobian to its canonical version which has the same eigenvalues but the eigenvectors are now orthogonal to each other. The canonical form can be expressed only by the values of the eigenvalues, i.e., the invariants also depends merely on the eigenvalues (cf. [2]). The advantage of this relationship is that one can determine the invariants even if only the eigensystem is known.

Besides the eigenvalues, the invariants may be used to classify critical points too. Considering the P-Q-R space, two thresholds depicted in equation 4 are separating the regions between real and complex solutions.

$$\begin{aligned} R_a &= \tfrac{1}{3}P\left(Q - \tfrac{2}{9}P^2\right) - \tfrac{2}{27}\left(-3Q + P^2\right)^{3/2} \\ R_b &= \tfrac{1}{3}P\left(Q - \tfrac{2}{9}P^2\right) + \tfrac{2}{27}\left(-3Q + P^2\right)^{3/2} \end{aligned} \tag{4}$$

Looking along the P-axis on the Q-R plane, a current critical point below R_a and on the left hand side of R_b is classified as a real solution (cf. figure 3). Otherwise, it is complex. Furthermore, it can be shown that solutions with only complex eigenvalues (the real one is zero) satisfy $PQ - R = 0$. This solution curve is also depicted in figure 3.

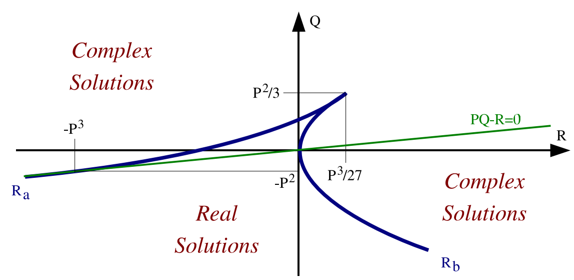

Figure 3: P-Q-R values separating real and complex solutions.

For an in-depth discussion of the P-Q-R space for solutions of first-order differential equations, see [2].

2.3 α-β Space

The eigenvectors of a node span three so-called phase planes defining a local coordinate frame. As two of the eigenvectors of a spiral are complex, in this case, only one phase plane can be determined. This contains the projected rotating flow pattern. Based on Clifford Algebra (cf.[15]), Lavin et al. [11] developed a new space where α and β values, computed by means of the invariants, define the position for each phase plane pattern of a critical point in a regular Euclidean space:

$$\alpha = P; \qquad \beta = sign(P^2 - 4Q)\sqrt{|P^2 - 4Q|} \tag{5}$$

Therefore, for the three-dimensional case, a critical point consists of three α-β pairs. A complete list how to compute these pairs are given in the appendix of [1]. In this space, the distance between two critical points is a metric. Normalizing α and β maps the phase portrait onto a unit circle so that the difference of two patterns can be determined by measuring the angle between them as shown in figure 4.

3 Icon-based Visualization

In order to investigate the topology of flow fields interactively, eigensystem icons can be created which allow the visualization of even complex topologies in real-time in virtual environments. For

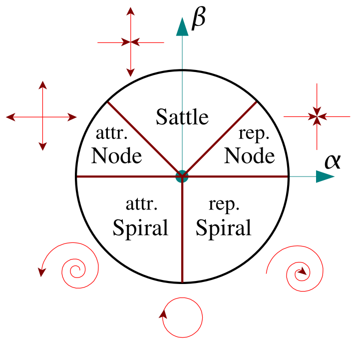

Figure 4: Flow patterns assigned to normalized α-β space.

critical points with real solutions, lines starting from the critical point location are drawn in the direction of the eigenvectors. As the lengths of eigenvectors are undetermined, the corresponding eigenvalues are used to define the length of these lines. Furthermore, coloring the lines blue-to-red depicts the flow directions. Whether a flow particle along an eigenvector runs into a critical point or is emanated from it depends on the sign of the corresponding eigenvalue (cf. [7]).

Computing this phase plane orientation of a spiral, the transposed Jacobian is used to determine a second eigensystem. If the original eigensystem consists of one real eigenvalue (denoted as e_3 in figure 5) and a conjugate-complex pair of eigenvalues, the transposed version shows the same types of eigenvalues. But now, the one real eigenvector f_3 can be interpreted as the normal vector of the sought complex phase plane (cf. [18]).

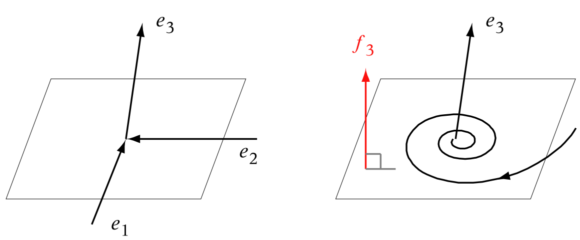

Figure 5: Phase planes of nodes (left) and the complex phase plane of spirals (right).

The spiral icon uses the original real eigenvector e_3 and visualizes it equivalent to the nodal case. For a fast identification, the flow direction is depicted now by a blue-to-green gradient. The complex phase plane, however, is spanned by two arbitrarily projected, orthogonal vectors. The lengths of these vectors are determined by the absolute value of the real and imaginary component of the conjugate-complex eigenvalues. According to the sign of the real component, a texture depicting a sink or a source spiral is mapped on this spanned quadrangle.

This described visualization approach is rather straightforward. On the other hand, it is highly efficient even if a large amount of critical point icons have to be visualized over time. In virtual environments, this gives an intuitive insight into the core features of flow topologies without the problems of occlusion or misinterpretations.

3.1 Interpolated Animation

On their paths over time, several events can happen influencing the animation. A general classification of tracking events was given by Wang (cf. [21]). These are: creation, continuation, bifurcation (splitting), amalgamation (merging), and dissipation. The life time of a feature and possibly occurring events are depicted in figure 6.

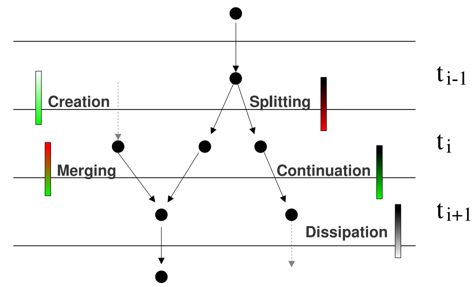

Figure 6: Tracking events, transient edges are colored with depicted gradients.

The visualization algorithm has access to a set of critical points per time level. These sets contain ordered lists of critical point objects with their quantities as location, eigenvalues, eigenvectors, and classification. Furthermore, these objects offer operators facilitating the animation between two critical points.

For the tracking algorithm, additional lists with successor IDs referencing the positions of successive critical points in the list of the next time level are provided. Important for the animation is the assumption that the eigensystem is ordered, i.e., the first eigenvalue and eigenvector of the current critical point correspond to the first of its successor, and so on.

The way how to animate critical points depends on the type of the tracking event. The goal is to find an implementation that guarantees a fluent visualization with a high frame rate even in virtual environments. The most straightforward case is the animation of the continuous event, especially when the critical point classification keeps unchanged. For intermediate critical points, the position is linearly interpolated. The needed interpolation coefficient γ is determined by the animation time frame (cf. figure 7) which returns the current value. Further attributes of the intermediate object are computed by the multiplication of the first critical point by γ and the multiplication of the successor critical point by $1-\gamma$. A final addition of both results yields the interpolated eigenvalues, the orientations of the real eigenvectors, and the orientation of the complex phase plane normal.

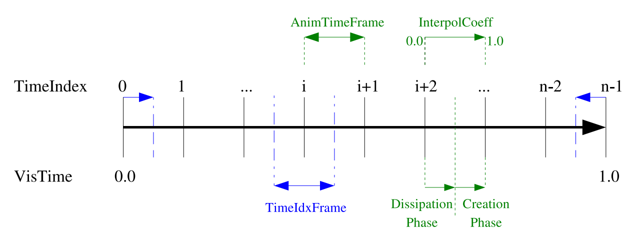

Figure 7: Important time frame definitions for the animation mode.

For creation events, the beginning of the animation time frame is just set in the middle of two time indices as depicted in figure 7. It starts then with a so-called zero critical point object which is then blown up until the full size of the first occurring critical point is reached. The dissipation event is animated vice versa. It starts with the full size and shrinks continuously until the mid-position between two time indices is reached where it finally disappears. If a critical point changes its classification during a continuation event from a node to a spiral or the other way around, the current visualization approach breaks it down to a dissipation event with a subsequent creation event.

The two remaining event types splitting and merging are also possible in dynamic systems and are called bifurcation and annihilation, respectively. They are discussed in more detail in section 4.2. The animation algorithm, however, treats each trace separately and visualizes them as single continuation events.

3.2 Tracking Trajectories

Instead of animated icon motions, the successor relationship can simply be presented by trajectory lines. These can be colored to show more information about the critical point properties. The current implementation maps the tracking event type to such trace lines. The color scheme uses gradients as depicted in figure 6. Start and end points of creation and dissipation events are always identical so that such transitions can be drawn optionally with an user-definable offset.

The lines can be visualized alone or in combination with the icon animation. The benefit of this topology tracking approach is the possibility to survey the over-all movement of current topological features. The trace information is not only useful for the current time frame but also for the presentation of critical point trajectories over the entire visualization time. The single trace lines are connected together which yields a tracking tree as depicted in figure 8.

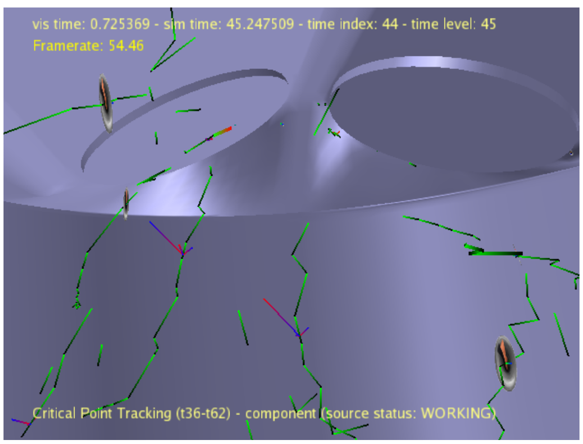

Figure 8: Tracking tree of extracted critical point trajectories.

Here, one can investigate how critical points evolve along the branches of this tree. With color-coded events mapped on single trace lines, it shows the topology development at a glance. Abrupt changes or other exceptional behaviors can be detected immediately and may be set in correlation with further findings.

4 Successive Location

The last aspect in order to complete the tracking approach is to fill up the successor list of each discrete time level. In the used framework, the post-processing of basic datasets is always carried out by Viracocha [5], running on dedicated high-performance computer, exploiting their multi-processor architectures, and relieving the visualization workstation of time-consuming processing tasks. To avoid high communication overhead, parallel algorithms mainly work independently.

In case of the implemented critical point tracking algorithm, the time levels are distributed among available processors. The input data for the searching algorithm consists of two sets of critical points: one for the current and one for the successive time level. Therefore, in a first step, each processor has to compute these two critical point sets. A processor working on its current time level which is already the successive time level assigned to a further processor recomputes these critical points. This seems to be inefficient as critical point computation is a rather time-consuming task. On the other hand, this distributed approach reduces communication considerably and simplifies the implementation. With a large amount of available processors, this strategy can be very efficient.

On the assumption that computing critical points for a certain time level always yields the same ordered critical point list, the IDs of the successor list will always be valid even if they are computed in parallel. Each processor can therefore send its found critical point list and the appropriate successor list back to the visualization system as soon as its current time level is processed. The order of the arrival is not important for the tracking visualization algorithm. It works even with partially transmitted data which will be improved with every newly arriving successor list of a further time level.

4.1 Similarity Quantities

The basic idea to identify successors is based on the consideration that possible candidates are located within a certain vicinity around the current position and consist of similar eigenvalues and eigenvectors. Therefore, in a first step, the amount of possible successors is reduced by a user-defined sphere around the current position. Only critical points of the successive time level within this sphere are considered for the similarity test which provides a considerably reduced list of candidates. Candidates which pass the similarity filter are inserted into the successor list. This list is sorted in descending order, i.e., the candidate with the highest similarity score is the first element. The last element possesses the lowest similarity score. Finding candidates by swapping the input critical point lists results in a predecessor list for the next time level, which can be used for a double-check.

The similarity filter uses similarity quantities to assess successor candidates. In the current implementation, following quantities are available:

- Eigenvalues distance
- Eigenvectors angle
- Invariants distance
- Invariants angle
- Critical point classification
- α-β space angle
- Curl and divergence distance

The classification assessment is the most straightforward one. It assumes that in most cases the successor will show the same class and, therefore, a candidate with a same class would be the most likely successor. The classification distinguishes between the cases given by figure 1 and figure 2, which results in 4 nodal and 6 spiral classes.

Transitions between classes are only allowed if merely one real eigenvalue changes it sign. Transition between types, i.e., between complex and real solutions, is only considered as valid if the signs of all real eigenvalue components keep unchanged. Transition events are marked with increased line widths, which can be configured by the user (cf. fig. 8). Possible transitions and corresponding codes are concluded in figure 9.

The α-β space angle approach can also be used to assess the classification of two critical point. It is more accurate as it estimates differences even if critical points consist of the same class type.

The next similarity quantity is the eigenvalues distance which is just defined as the sum of all distances of the critical point's eigenvalues in the complex plane from corresponding eigenvalues of a candidate. The general idea is that corresponding eigenvalues merely change slightly over time. For the assessment, the largest absolute value of each component of all eigenvalues is searched. Then, all components are divided by this maximum value. Only these normalized eigenvalues are used for the distance computation. This provides a quantity in a range of 0.0 and 8.4853 (3 times

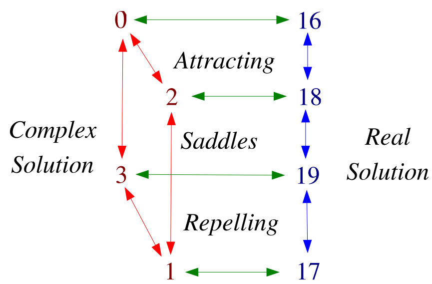

Figure 9: Valid transition events.

the diagonal of a 2x2 quadrangle), which is accurate both for large and for small eigenvalues.

The eigenvectors angle is similar to the eigenvalues distance quantity. It measures the angles between all three eigenvectors and compares this value with the measurement of a candidate. One angle always lies between 0.0 and 90.0 degree. A second version compares each corresponding vector of the current and the candidate critical point. For spirals, only the angle between real eigenvector and the normal of the complex phase plane is computed.

It could be considered that successive critical points which visually belong together can show a certain eigensystem derivation. In comparison to the eigensystem icon approach, figure 10 presents icons which use the invariants P, Q, and R to define the lengths of X-Y-Z aligned lines. This visualization approach facilitates the visual identification of successors.

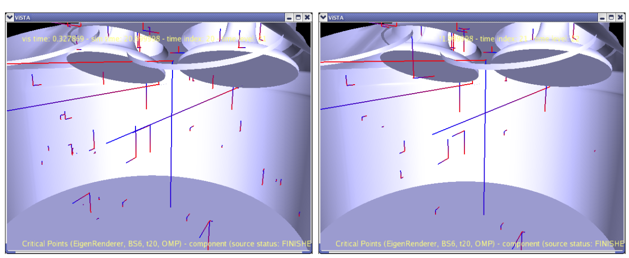

Figure 10: Icons present the invariants at critical point locations on two successive time levels.

Furthermore, for an automatic assessment, it is not necessary to handle complex numbers anymore. This can be exploited for further quantities. The first one is the invariants distance which is defined as the distance of two critical points in the P-Q-R space. Because changes of P-Q-R coordinates farther away from the origin are larger, the invariants are normalized first according to the normalization scheme of the eigenvalues distance quantity. This produces invariants distances in a range of 0.0 and 3.4641 (diagonal of a 2x2x2 cube).

The second invariant quantity computes the angle between lines of the current and a candidate critical point and works sufficiently whenever the lines have a certain length. The smaller the length, the higher is the possibility that a change of P, Q, and R yields a larger angle deviation. Therefore, this quantity is "weak" as it can be used only when other requirements are met. This is the same for a last invariant quantity measuring the relative change of the P-Q-R coordinate distances to the origin. It returns the length in percentage of the shorter of two compared lines.

After the amount of candidates was already reduced by the sphere vicinity criterion, a similarity filter decreases the set even more. This filter applies positive thresholds first. Whenever one of the low level thresholds is met, the candidate will be marked as a possible successor and the filter procedure stops. After some positive tests, negative thresholds are evaluated. If a computed quantity is below such a threshold, the candidate will immediately be rejected.

All thresholds are user-definable. The first positive threshold is a minimum eigenvalues distance followed by a threshold testing the eigenvector angles (second version) of equal critical point classes. The third threshold evaluates a minimal invariants distance. The negative thresholds are maximum eigenvalues distance, maximum invariants distance, maximum eigenvector angle, and maximum α-β space angle. The maximum eigenvector angle threshold is based on the first mentioned version. This is more reliable even if the eigensystem itself rotates, which is often the case.

All other quantities mentioned before are just used to be written into a log file and can be evaluated later on manually. This might result in an improved filter. The log file can also help to adapt the thresholds as well as to analyze the flow field topology in more depth.

If all thresholds are passed, a candidate is also marked as a successor. Should more than one candidate be left after the similarity filter was applied, one of the assessment quantities is used to identify the one with the best value. The current implementation uses the eigenvalue distance to determine the similarity score.

4.2 Bifurcation

In general, the computed velocity gradient at a 3-dimensional critical point has full rank, i.e., its determinant is unequal zero. Such a critical point is said to be first order. If all eigenvalues of first order critical points have real parts unequal zero, it is also called a hyperbolic critical point. Bifurcation is defined as the event when one critical point splits into two (or more) or vice versa. This primarily happens when a non-hyperbolic, unstable critical point appears. For the 2-dimensional case, Tricoche describes several types in [19].

In general, bifurcation events cannot be detected directly on discrete time levels. The reason is that non-synthetic, highly complex, and dynamic simulations come with a certain noise, and a bifurcation is just an unstable, discrete event. Therefore, one can only estimate the occurrence of bifurcations between two discrete time levels by considering critical points actually identified.

Our approach is based again on similarities. Nearest neighbors are not only searched on the successive but on the same time level as well. Furthermore, for each critical point on the successor time level, all similar neighbors and all predecessors are also stored. This results in four lists which are computed in advance and can then be used for a bifurcation analysis on the current time level.

Splitting events are detected whenever multiple successor candidates are available. Inverted bifurcations indicate merges and are also called annihilations. In order to identify such merging events, a critical point on the successive time level has to consist of multiple predecessors. Two further events not covered by types presented in figure 6 are bifurcation without predecessors or successors, respectively. The creation + splitting event shows similar neighbors without predecessors on the next time level. The merging + dissipation event can be detected if critical points on the current level are similar neighbors but do not point to any successors. The added trajectory color coding is depicted in figure 11.

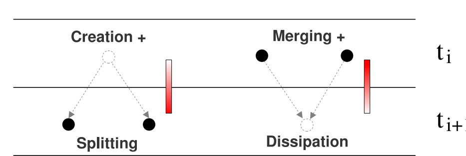

Figure 11: Two further event types for bifurcation.

5 Results

In order to evaluate the presented critical point tracking approach, three different datasets were applied denoted as Engine, Propfan, and Shock. The engine dataset consists of 62 time levels but only the compression phase was investigated. The dataset itself was normalized, i.e., the maximum expansion of the domain fits into an 1-by-1 cube and the velocities are adapted to a delta time of 1.0 between two time levels. However, the simulation time between two discrete time levels is rather large so that the eigensystem of critical points may show considerable changes even when they are obviously related.

The simulation of the counter-clockwise rotating propulsion turbine is stored in 50 time levels and comes with a considerably finer resolution. Critical points can primarily be detected at its blades and stay there rather constantly (cf. figure 12). The velocities are again normalized. As the flow fields around rotated blades are identical, found critical points show the same eigensystem and invariants at corresponding locations.

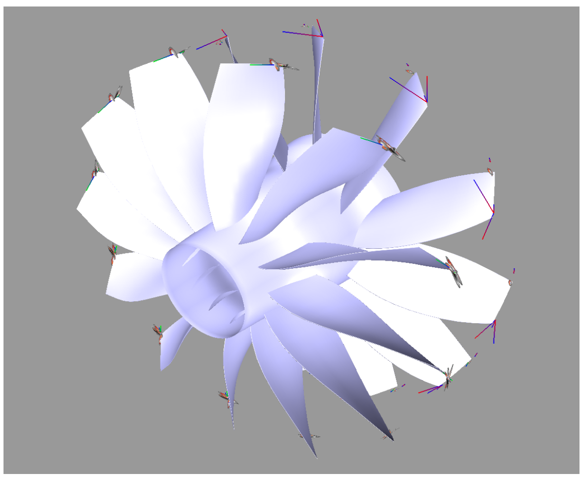

Figure 12: Critical points of the propfan dataset mainly appear at blade tips and are rather stable over time.

The last evaluated dataset shows a shock induction over an edge and consists of 919 time levels. A snapshot of 10 successive time levels was considered. The turbulence over time increases significantly. A very large amount of critical points can be extracted primarily in the affected region (cf. figure 13). More details about the data sizes and amount of critical points of the used datasets is presented in [6].

The tracking algorithm sets the positive threshold for the normalized eigenvalue distance to a default value of 0.1. The upper default eigenvalue distance is defined to 1.0. For the eigenvalue angle, these thresholds are set to 5 degree and 45 degree, respectively. The same values are used for the invariant thresholds. Only the upper invariant distance is set to a little higher value of 1.75. 15 degree was used for the α-β space angle.

For the Engine and the Propfan dataset, a search radius of 0.1 is sufficient whereas for the Shock this value has to be set to 0.9. This reduces the candidate set up to a maximum of 4 critical points. As eigenvalues and invariants are quite small, their resulting quantities are often around the thresholds.

This is different for the Propfan dataset. Eigenvalues rises up to +/- 20.0 and invariants sometimes show values above 1000.0 which yield clearly distinguishable measurement quantities. In general, matching successors are very similar so that positive thresholds can already be applied successfully.

As expected, the candidate set for the turbulent Shock dataset

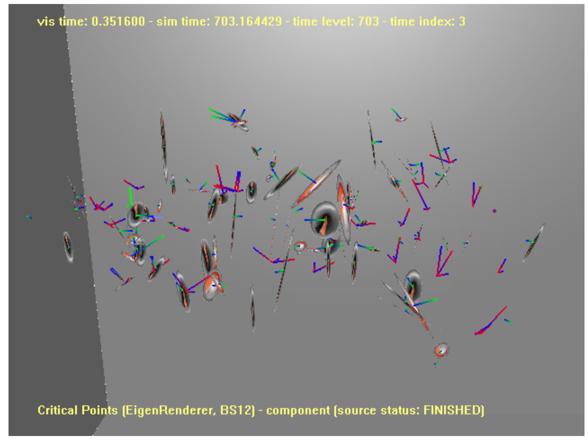

Figure 13: In the shock dataset, a large amount of critical points is closely clustered.

consists of a considerably larger amount of critical points. Eigenvalues and eigenvectors are again rather small. Nevertheless, especially the upper eigenvalue distance threshold finds potential successors reliably.

In all datasets, transition events could be detected within complex and real solutions as well as between them. For the Engine and the Shock, only a few transition events are registered whereas this kind is the frequent event type for the Propfan. Invalid transitions occurred in none of the considered datasets. Also bifurcations (splits) and annihilations (merges) occur in all datasets. Critical points generally split into two successors or merge from two predecessors and keep their solution types. Only few splits and merges including three successors or predecessors could be detected. The Engine also shows one split with different types.

Searching neighbors on the same time level also indicates merging-dissipation as well as creation-splitting events in all datasets. In general, these events involve critical points of same solution types. But on two time levels of the Propfan, a mixing could be detected. More than two participating critical points is a rare event. Only the Shock consists of such event types: one merging-dissipation event and one creation-splitting event.

Comparing the normalized eigenvalue scheme with the α-β space angle approach, the first offers a more stable assessment. Very close eigenvalues can show a variation in α-β space angle up to 10 degrees. Clearly dissimilar critical points, however, shows sometimes angles below 10 degrees. But in most cases, the angle for detected successors was below 1 degree.

The computation was carried out on an SGI Altix 4700 with 160 Itanium-2 processor cores. This was sufficient to determine successors for each time step at once in parallel. All considered time levels of the Engine could be extracted within 40 seconds. On an Opteron visualization workstation with an NVidia Quadro 4500, critical point animation along an activated tracking tree could be rendered with a framerate of 54 Hz. The waiting time and the framerate are both acceptable for virtual environments. Post-processing time for the Propfan and the Shock is considerably larger. Fortunately, the Propfan is a multi-block data structure so that the computation for one time level can also be parallelized. Visualizing critical points with the tracking tree drops here below the interactive framerate. But if only trajectories of the current time level are visualized, a framerate around 40 Hz could be measured.

The Shock needs up to 70 seconds to load the data. For that, prefetching is a solution which is already a strategy offered by the data management of Viracocha. Most of the time, however, is needed to determine critical points (approx. 2:40 min. per time level). To speed-up this local, cell-based computation, the integrated OpenMP parallelization schemes can help (cf. [6]). The tracking algorithm itself only needs 6 seconds. Around 70 critical points per time steps can be rendered with over 100 Hz (without trajectories).

6 Conclusion and Future Work

Tracking of topological features in flow fields can be very helpful to analyze simulation results. It fills the gap between the discrete simulation data and the actually simulated transient flow behavior. However, the combination of discrete context geometries and continuous flow features can sometimes lead to misinterpretations. For instance, the critical points shown at the blade tips of the propfan move always slightly off. This problem can be fixed by animating the geometry as well, which is possible for the Propfan but more complicated for complex moving geometries like the Engine dataset.

The presented framework aims at supporting engineering scientists during an exploratory analysis of complex flow field topologies in virtual environments. This goal could be reached. In a manual time switching mode, parameters to determine successors for the current time level can be modified individually which emphasizes the exploratory character of our approach even more. The animated tracking of critical points is fast and instructive. Also the straightforward visualization scheme when their classifications change over time is sufficient but can be improved even more in future versions. Instead of the disappearing and re-appearing scheme, an eigenvalue interpolation on the complex plane (cf. figure 1 and figure 2) can lead to more accurate results.

There are some ambiguities with the animation approach. As the direction of eigenvectors is undetermined, it can happen that the visualized icon lines change their direction by around 180 degrees. Even the complex phase plane normal can flip suddenly. However, as the computation of the eigensystem and the creation of icons are deterministic, this behavior is valid and could be wanted to show certain changes of the underlying quantities. Nevertheless, a future version will offer the opportunity to handle these ambiguities for an improved animation.

The bifurcation approach is ongoing work but offers already all strategies for an in-depth analysis of special events over time. The primary goal of this first implementation was to detect special events which include more than one critical point per time level. Critical point classification can be integrated into bifurcation analysis which allows a more in-depth bifurcation evaluation. A future version will therefore implement this information into visual concepts.

Acknowledgements

The authors wish to thank the Institute of Aerodynamics, RWTH Aachen University, and the German Aerospace Center (DLR), Institute of Propulsion Technology, Cologne, who kindly made available datasets for evaluation purposes.

References

[1] R. Batra and L. Hesselink. Feature comparisons of 3-d vector fields using earth mover's distance. In *Proceeding, Visualization 1999, San Francisco, CA*, pages 105–114, Los Alamitos, CA, USA, 1999. IEEE Computer Society Press.

[2] M. S. Chong, A. E. Perry, and B. J. Cantwell. A general classification of three-dimensional flow fields. *Physics of Fluids A*, 2(5):765–777, 1990.

[3] W. de Leeuw. *Presentation and Exploration of Flow Data*. PhD thesis, Technische Universiteit Delft, Netherlande, 1997.

[4] C. Garth, X. Tricoche, and G. Scheuermann. Tracking of vector field singularities in unstructured 3d time-dependent datasets. In *Proceedings, IEEE Visualization 2004*, pages 329–336, Washington, DC, USA, 2004. IEEE Computer Society.

[5] A. Gerndt, B. Hentschel, M. Wolter, T. Kuhlen, and C. Bischof. Viracocha: An efficient parallelization framework for large-scale CFD post-processing in virtual environments. In *The International Conference for High Performance Computing and Communications, SC04*, November 2004.

[6] A. Gerndt, S. Sarholz, M. Wolter, D. an Mey, C. Bischof, and T. Kuhlen. Nested OpenMP for efficient computation od 3D critical points in multi-block CFD datasets. In *The International Conference for High Performance Computing, Networking, Storage, and Analysis, SC06*, November 2006.

[7] A. Globus, C. Levit, and T. Lasinki. A tool for visualization the topology of three-dimensional vector fields. In G. M. Nielson and L. Rosenblum, editors, *Proceedings, IEEE Visualization, San Jose, CA*, pages 33–40, October 22 - 25 1991.

[8] J. L. Helman and L. Hesselink. Visualizing vector field topology in fluid flows. *IEEE Computer Graphics and Applications*, 11(3):36–46, 1991.

[9] T. Klein and T. Ertl. Scale-space tracking of critical points in 3d vector fields. In *Proceedings, Workshop, Topology-based Methods in Visualization (TopoInVis) 2005, Budmerice, Slovakia*, Sept. 29-30 2005.

[10] D. A. Lane. Scientific visualization of large-scale unsteady fluid flows. In G. M. Nielsen, H. Hagen, and H. Müller, editors, *Scientific Visualization - Overview, Methodologies, Techniques*, chapter 5, pages 125 – 145. IEEE Computer Society Press, 1997.

[11] Y.-M. Lavin, R. Batra, and L. Hesselink. Feature comparisons of vector fields using earth mover's distance. In *VIS '98: Proceedings of the conference on Visualization '98*, pages 103–109, Los Alamitos, CA, USA, 1998. IEEE Computer Society Press.

[12] A. E. Perry and M. S. Chong. A description of eddying motions and flow patterns using critical-point concepts. *Ann. Rev. Fluid Mech*, 19:125–155, 1987.

[13] K. F. Reinders. *Feature-Based Visualization of Time-Dependent Data*. PhD thesis, Technical University Delft, Netherlands, 2001.

[14] I. A. Sadarjoen, T. van Walsum, A. J. S. Hin, and F. H. Post. Particle tracing algorithms for 3D curvilinear grids. In G. M. Nielson, H. Hagen, and H. Müller, editors, *Scientific Visualization - Overview, Methodologies, Techniques*, chapter 14, pages 311 – 335. IEEE Computer Society Press, 1997.

[15] G. Scheuermann, H. Krüger, M. Menzel, and A. P. Rockwood. Visualizing nonlinear vector field topology. *IEEE Transaction on Visualization and Computer Graphics*, 4(2):109–116, April-June 1998.

[16] D. Silver and X. Wang. Tracking and visualization turbulent 3d features. *IEEE Transaction on Visualization and Computer Graphics*, 3(2):129–142, April-June 1998.

[17] H. Theisel and H.-P. Seidel. Feature flow fields. In *Proceedings, Symposium on Data Visualisation*, pages 141 – 148, 2003.

[18] H. Theisel, T. Weinkauf, H.-C. Hege, and H.-P. Seidel. Saddle connectors - an approach to visualizing the topological skeleton of complex 3D vector fields. In *Proceedings of IEEE Visualization, Seattle, WA*, pages 225–232, October 19-24 2003.

[19] X. Tricoche. *Vector and Tensor Field Topology Simplification, Tracking, and Visualization*. PhD thesis, University Kaiserslautern, Germany, 2002.

[20] A. van Dam, A. Forsberg, D. H. Laidlaw, J. LaViola, and R. M. Simpson. Immersive virtual reality for scientific visualization: A progress report. *IEEE Computer Graphics and Applications*, 20(6):26–52, November/December 2000.

[21] X. Wang. *Visualization and Tracking Features in 3D Time-Varying Datasets*. PhD thesis, Rutgers State University of New Jersey, New Brunswick, 1999.

[22] T. Weinkauf, H. Theisel, H.-C. Hege, and H.-P. Seidel. Topological construction and visualization of higher order 3D vector fields. In *Proceedings, Eurographics, Grenoble, Frankreich*, volume 23, 2004.