Volume 22 (2003), number 4 pp. 775–792 COMPUTER GRAPHICS *forum*

The State of the Art in Flow Visualisation: Feature Extraction and Tracking

Frits H. Post[1], Benjamin Vrolijk[1], Helwig Hauser[2], Robert S. Laramee[2] and Helmut Doleisch[2]

[1]Computer Graphics Group, Delft University of Technology, The Netherlands
[2]VRVis Research Center, Vienna, Austria
F.H.Post@ewi.tudelft.nl, B.Vrolijk@ewi.tudelft.nl, Hauser@VRVis.at, Laramee@VRVis.at, Doleisch@VRVis.at

Abstract

Flow visualisation is an attractive topic in data visualisation, offering great challenges for research. Very large data sets must be processed, consisting of multivariate data at large numbers of grid points, often arranged in many time steps. Recently, the steadily increasing performance of computers again has become a driving force for new advances in flow visualisation, especially in techniques based on texturing, feature extraction, vector field clustering, and topology extraction.

In this article we present the state of the art in feature-based flow visualisation techniques. We will present numerous feature extraction techniques, categorised according to the type of feature. Next, feature tracking and event detection algorithms are discussed, for studying the evolution of features in time-dependent data sets. Finally, various visualisation techniques are demonstrated.

Keywords: Visualisation, flow visualisation, feature-based flow visualisation.

ACM CSS: I.3.8 Computer Graphics—*applications*

1. Introduction

Flow visualisation is one of the traditional subfields of data visualisation, covering a rich variety of applications, ranging from automotive, aerospace, and turbomachinery design, to weather simulation and meteorology, climate modelling, and medical applications, with many different research and engineering goals and user types. Consequently, the spectrum of flow visualisation techniques is very rich, spanning multiple dimensions of technical aspects, such as 2D and 3D techniques, and techniques for steady and time-dependent data.

In this article we present the state of the art in flow visualisation techniques. These techniques can be categorised into four groups:

- *Direct flow visualisation*: The data is directly visualised, without much pre-processing, for example by colour-coding or drawing arrows. These techniques are also called *global techniques*, as they are usually applied to an entire domain, or a large part of it.
- *Texture-based flow visualisation*: Texture-based techniques apply the directional structure of a flow field to random textures. These are mainly used for visualising flow in two dimensions or on surfaces. The results are comparable to the experimental techniques like windtunnel surface oil flows. This group has some characteristics of the previous and approaches.
- *Geometric flow visualisation*: Geometric objects are first extracted from the data, and used for visualisation. Examples are streamlines, stream surfaces, time surfaces, or flow volumes. These geometric objects are directly related to the data. The results of these techniques can be compared to experimental results such as dye advection or smoke injection into the flow.
- *Feature-based flow visualisation*: The last approach lifts the visualisation to a higher level of abstraction, by extracting physically meaningful patterns from the data

sets. The visualisation shows only those parts that are of interest to the researcher, the *features*. Both the definition of what is interesting, and the way these features are extracted and visualised are dependent on the data set, the application, and the research problem.

The approaches are not entirely distinct. For example, the second and third approaches can be combined into *dense flow visualisation*.

In this article, we survey the last approach, feature-based flow visualisation.

Features are phenomena, structures or objects in a data set, that are of interest for a certain research or engineering problem. Examples of features in flow data sets are shock waves, vortices, boundary layers, recirculation zones, and attachment and separation lines.

There are a number of factors motivating the feature-based approach to visualisation. First, by extracting only the interesting parts, and ignoring the rest, we can increase the information content. Furthermore, by abstracting from the original data, the researcher is able to focus more on the relevant physical phenomena, which is better related to his conceptual framework. A large data reduction can be achieved (in the order of 1000 times), but because the reduction is content-based, no (important) information is lost. So far, this is one of the few approaches that is truly scalable to very large time-dependent data sets. Finally, the objects or phenomena extracted can be simplified and described quantitatively. This makes the visualisation easy, using simple geometries or parametric icons. Also, quantification facilitates further research, comparison and time tracking.

The paper is structured as follows: in the next section, we will discuss some fundamentals for flow visualisation, which are necessary for understanding the rest of the paper. In Section 3 an introduction to feature extraction is given, with a categorisation of the general approaches to feature extraction. In Section 4 feature extraction techniques are discussed, for several different types of features. Section 5 discusses feature tracking and event detection, that is, the study of the evolution of features in time-dependent data sets. Section 6 presents different iconic representations of features and the visualisation of features and events. Finally, in Section 7 some conclusions and further prospects are presented.

2. Flow Visualisation Fundamentals

For a proper understanding of the rest of the article, it is necessary to discuss a number of fundamentals for flow visualisation, mainly from vector algebra.

2.1. Gradients

In three dimensions, a scalar p has three partial derivatives. The partial derivative of p with respect to $\mathbf{x}$ is $\frac{\partial p}{\partial \mathbf{x}}$. The gradient of a scalar field is the vector of its partial derivatives:

$$\mathrm{grad}p = \nabla p = \left[\frac{\partial p}{\partial \mathbf{x}} \quad \frac{\partial p}{\partial \mathbf{y}} \quad \frac{\partial p}{\partial \mathbf{z}}\right]. \tag{1}$$

The gradient of a vector field $\mathbf{v}$ is found by applying the gradient operator to each of the components $[u \ \ v \ \ w]$ of the vector field. This results in a 3×3 matrix, called the *Jacobian* of the vector field, or the matrix of its first derivatives:

$$\nabla \mathbf{v} = \begin{bmatrix} \dfrac{\partial u}{\partial \mathbf{x}} & \dfrac{\partial u}{\partial \mathbf{y}} & \dfrac{\partial u}{\partial \mathbf{z}} \\ \dfrac{\partial v}{\partial \mathbf{x}} & \dfrac{\partial v}{\partial \mathbf{y}} & \dfrac{\partial v}{\partial \mathbf{z}} \\ \dfrac{\partial w}{\partial \mathbf{x}} & \dfrac{\partial w}{\partial \mathbf{y}} & \dfrac{\partial w}{\partial \mathbf{z}} \end{bmatrix} \tag{2}$$

This matrix can be used to compute a number of derived fields, such as the divergence, curl, helicity, acceleration, and curvature. The curl of a *velocity* field is called the *vorticity*. This derived vector field indicates how much the flow locally rotates and the axis of rotation. These quantities are all used in different feature extraction techniques, which will be discussed later. The exact definitions can be found elsewhere [1,2]. For the understanding of this article, it is sufficient to know that the Jacobian, or gradient matrix, is an important quantity in flow visualisation in general and in feature extraction in particular.

2.2. Eigenanalysis

Another indispensable mathematical technique is eigenanalysis. An *eigenvalue* of a 3×3 matrix M is a (possibly complex) scalar λ which solves the eigenvector equation: $M\mathbf{x} = \lambda\mathbf{x}$. The corresponding non-zero vector $\mathbf{x}$ is called an *eigenvector* of M. The eigenvectors and eigenvalues of a Jacobian matrix indicate the direction of tangent curves of the flow, which are used, for example to determine the vector field topology, see Section 3.2.

2.3. Attribute Calculation

As a part of the feature extraction process, characteristic attributes of the features have to be calculated. One conceptually simple and space efficient technique, is the computation of an ellipsoid fitting. An ellipsoid can give a first-order estimation of the orientation of an object. The axes can be scaled to give an exact representation of the size or volume of the object. Furthermore, an ellipsoid is a very simple icon to visualise. The computation of an ellipsoid fitting involves eigenanalysis of the *covariance matrix* of the object's grid points. For a detailed description, see Haber and McNabb [3], Silver *et al.* [4] and de Leeuw [1].

Another technique that can be used for attribute calculation of features is *center line extraction*. As an example, a skeleton, or Medial Axis Transform, reduces an object to a single center line, or graph, while preserving the original topology of the object. Using this graph, an icon can be constructed from cylinders and hemispheres, to construct an approximation of the original shape of the object [5]. This is a useful representation, especially when the topology is an important characteristic of the features.

3. Feature Extraction Approaches

Feature-based flow visualisation is an approach for visualising the flow data at a high level of abstraction. The flow data is described by features, which represent the interesting objects or structures in the data. The original data set is then no longer needed. Because often, only a small percentage of the data is of interest, and the features can be described very compactly, an enormous data reduction can be achieved. This makes it possible to visualise even very large data sets interactively.

The first step in feature-based visualisation is *feature extraction*. The goal of feature extraction is determining, quantifying and describing the features in a data set.

A feature can be loosely defined as any object, structure or region that is of relevance to a particular research problem. In each application, in each data set and for each researcher, a different feature definition could be used. Common examples in fluid dynamics are vortices, shock waves, separation and attachment lines, recirculation zones and boundary layers. In the next section a number of feature-specific detection techniques will be discussed. Although most feature detection techniques are specific for a particular type of feature, in general the techniques can be divided into three approaches: based on image processing, on topological analysis, and on physical characteristics.

3.1. Image Processing

Image processing techniques were originally developed for analysis of 2D and 3D image data, usually represented as scalar (greyscale) values on a regular rectangular grid. The problem of analysing a numerical data set, represented on a grid, is similar to analysing an image data set. Therefore, basic image processing techniques can be used for feature extraction from scientific data. A feature may be distinguished by a typical range of data values, just as different tissue types are segmented from medical images. Edges or boundaries of objects are found by detecting sharp changes in the data values, marked by high gradient magnitudes. Thus, basic image segmentation techniques, such as thresholding, region growing, and edge detection can be used for feature detection. Also, objects may be quantitatively described using techniques such as skeletonisation or principal component analysis. However, a problem is, that in computational fluid dynamics simulations, often grid types are used such as structured curvilinear grids, or unstructured tetrahedral grids. Many techniques from image processing cannot be easily adapted for use with such grids. Furthermore, many digital filtering techniques are defined only for scalar data. Adaptation to vector fields is not always straightforward.

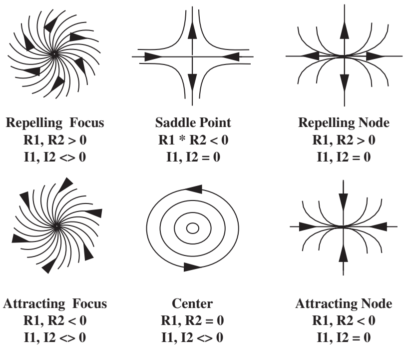

Figure 1: *Vector field topology: critical points classified by the eigenvalues of the Jacobian* [6].

3.2. Vector Field Topology

A second approach to feature extraction is the topological analysis of 2D linear vector fields, as introduced by Helman and Hesselink [6,7], which is based on detection and classification of critical points.

The critical points of a vector field are those points where the vector magnitude is zero. The flow in the neighbourhood of critical points is characterised by eigenanalysis of the velocity gradient tensor, or Jacobian of the vector field. The eigenvalues of the Jacobian can be used to classify the critical points as attracting or repelling node or focus, as saddle point, or center (see Figure 1). The eigenvectors indicate the directions in which the flow approaches or leaves the critical point. These directions can be used to compute tangent curves of the flow near the critical points. Using this information, a schematic visualisation of the vector field can be generated (see Figure 7). Helman and Hesselink have also extended their algorithm to 2D time-dependent and to 3D flows.

Tricoche *et al.* recently presented a topology-based method for visualising time-dependent 2D vector fields [8]. They perform time tracking of critical points and closed streamlines by temporal interpolation. They are able to find and characterise topological events or structural changes (*bifurcations*), such as the pairwise annihilation or creation of a saddle point and an attracting or repelling node.

Scheuermann *et al.* presented an algorithm for visualising non-linear vector field topology [9], because other known algorithms are all based on piecewise linear or bilinear interpolation, which destroys the topology in the case of non-linear behaviour. Their algorithm makes use of Clifford algebra for computing polynomial approximations in areas with non-linear local behaviour, especially higher-order singularities.

De Leeuw and Van Liere presented a technique for visualising flow structures using multilevel flow topology [10]. In high-resolution data sets of turbulent flows, the huge number of critical points can easily clutter a flow topology image. The algorithm presented attempts to solve this problem by removing small-scale structures from the topology. This is achieved by applying a pair distance filter which removes pairs of critical points, that are near each other. This removes small topological structures such as vortices, but does not affect the global topological structure. The threshold distance, which determines which critical points are removed, can be adapted, making it possible to visualise the structure at different levels of detail at different zoom levels.

Tricoche *et al.* also perform topology simplification in 2D vector fields [11]; they simplify not only the topology, but also preserve the underlying vector field, thereby making it possible to use standard flow visualisation methods, such as streamlines or LIC, after the simplification. The basic principle of removing pairs of critical points is similar to the technique of De Leeuw and Van Liere [10], but in this algorithm the vector field surrounding the critical points is slightly modified, in such a way that both critical points disappear.

3.3. Physical Characteristics

The third approach is feature extraction based on physical characteristics. Often, features can be detected by characteristic patterns in, or properties of, physical quantities, for example by low pressure, high temperature, or swirling flow. These properties often follow directly from the feature definitions used. Most of the feature extraction techniques discussed in Section 4 are based on this approach, sometimes in combination with topological analysis or image processing techniques.

3.4. Selective Visualisation

A generic approach to feature extraction is Selective Visualisation, which is described by Van Walsum [12]. The feature extraction process is divided into four steps (see Figure 2).

The first step is the *selection* step. In principle, any selection technique can be used, that results in a binary segmentation of the original data set. A very simple segmentation

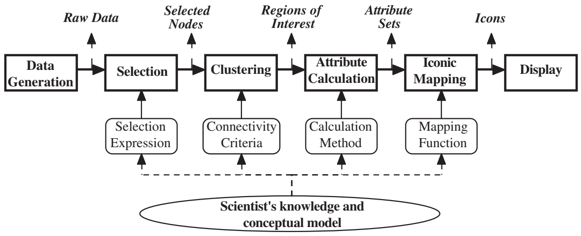

Figure 2: *The feature extraction pipeline* [13].

is obtained by thresholding of the original or derived data values; also, multiple thresholds can be combined. The data set resulting from the selection step is a binary data set with the same dimensions as the original data set. The binary values in this data set denote whether or not the corresponding points in the original data set are selected. The next step in the feature extraction process is the *clustering* step, in which all points that have been selected are clustered into coherent regions. In the next step, the *attribute calculation* step, these regions are quantified. Attributes of the regions are calculated, such as position, volume and orientation. We now speak of objects, or features, with a number of attributes, instead of clusters of points. Once we have determined these quantified objects, we don't need the original data anymore. With this, we may accomplish a data reduction factor of 1000 or more. In the fourth and final step, *iconic mapping*, the calculated attributes are mapped onto the parameters of certain parametric icons, which are easy to visualise, such as ellipsoids.

4. Feature Extraction Techniques

In this section, a number of feature extraction techniques will be discussed that have been specifically designed for certain types of features. These techniques are often based on physical or mathematical (topological) properties of the flow. Features that often occur in flows are vortices, shock waves and separation and attachment lines.

4.1. Vortex Extraction

Features of great importance in flow data sets, both in theoretical and in practical research, are *vortices* (see Figure 3). In some cases, vortices (turbulence) have to be impelled, for example to stimulate mixing of fluids, or to reduce drag. In other cases, vortices have to be prevented, for example around aircraft, where they can reduce lift.

There are many different definitions of vortices and likewise many different vortex detection algorithms. A distinction can be made in algorithms for finding vortex regions and algorithms that only find the vortex cores.

Other overviews of algorithms are given by Roth and Peikert [14] and by Banks and Singer [15].

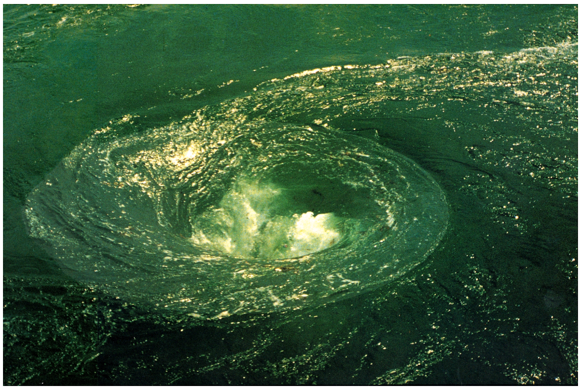

Figure 3: *A vortex in water (WL | Delft Hydraulics).*

There are a number of algorithms for finding *regions with vortices*:

- One idea is to find regions with a high vorticity magnitude. Vorticity is the curl of the velocity, that is, $\nabla \times \mathbf{v}$, and represents the local flow rotation, both in speed and direction. However, although a vortex may have a high vorticity magnitude, the converse is not always true [16]. Villasenor and Vincent present an algorithm for constructing vortex tubes using this idea [17]. They compute the average length of all vorticity vectors contained in small-radius cylinders, and use the cylinder with the maximum average for constructing the vortex tubes.
- Another idea is to make use of helicity instead of vorticity [18,19]. The helicity of a flow is the projection of the vorticity onto the velocity, that is $(\nabla \times \mathbf{v}) \cdot \mathbf{v}$. This way, the component of the vorticity perpendicular to the velocity is eliminated.
- Because swirling flow often swirls around areas of low pressure, this is another criterion that can be used to locate vortex cores [20].
- Jeong and Hussain define a vortex as a region where two eigenvalues of the symmetric matrix $S^2 + \Omega^2$ are negative, where S and Ω are the symmetric and antisymmetric parts of the Jacobian of the vector field, respectively [21]: $S = \frac{1}{2}(V+V^{\mathrm{T}})$, and $\Omega = \frac{1}{2}(V-V^{\mathrm{T}})$. This method is known as the λ_2 method.

The above methods may all work in certain simple flow data sets, but they do not hold, for example, in turbomachinery flows, which can contain strongly curved vortices [14].

There are also some algorithms specifically for finding *vortex core lines*:

- Banks and Singer use streamlines of the vorticity field, with a correction to the pressure minimum in the plane perpendicular to the vortex core [15].
- Roth and Peikert suggest that a vortex core line can be found where vorticity is parallel to velocity [14]. This sometimes results in coherent structures, but in most data sets it does not give the expected features.
- In the same article, Roth and Peikert suggest that, in linear fields, the vortex core line is located where the Jacobian has one real-valued eigenvector, and this eigenvector is parallel to the flow [14]. However, in their own application of turbomachinery flows, the assumption of a linear flow is too simple. The same algorithm is presented by Sujudi and Haimes [22].
- Recently, Jiang *et al.* presented a new algorithm for vortex core region detection [23], which is based on ideas derived from combinatorial topology. The algorithm determines for each cell if it belongs to the vortex core, by examining its neighbouring vectors.

A few of these algorithms will be reviewed in more detail.

Sujudi and Haimes developed an algorithm for finding the centre of swirling flow in 3D vector fields and implemented this algorithm in pV3 [22]. Although pV3 can use many types of grids, the algorithm has been implemented for tetrahedral cells. When using data sets with other types of cells, these first have to be decomposed into tetrahedral cells. This is done for efficiency, because linear interpolation for the velocity can be used in the case of tetrahedral cells. The algorithm is based on critical-point theory and uses the eigenvalues and eigenvectors of the velocity gradient tensor or rate-of-deformation tensor. The algorithm works on each point in the data set separately, making it very suitable for parallel processing. The algorithm searches for points where the velocity gradient tensor has one real and two complex-conjugate eigenvalues and the velocity is in the direction of the eigenvector, corresponding to the real eigenvalue. The algorithm results in large coherent structures when a strong swirling flow is present, and the grid cells are not too large. The algorithm is sensitive to the strength of the swirling flow, resulting in incoherent structures or even no structures at all in weak swirling flows. Also, if the grid cells are large, or irregularly sized, the algorithm has difficulties finding coherent structures or any structures at all.

Kenwright and Haimes also studied the eigenvector method and concluded that it has proven to be effective in many applications [24]. The drawbacks of the algorithm are that it does not produce contiguous lines. Line segments are drawn for each tetrahedral element, but they are not necessarily continuous across element boundaries. Furthermore, when the elements are not tetrahedra, they have to be decomposed into tetrahedra first, introducing a piecewise linear approximation for a non-linear function. Another problem is that flow features are found that are not vortices. Instead, swirling flow is detected, of which vortices are an example. However, swirling flow also occurs in the formation of boundary layers. Finally, the eigenvector method is sensitive to other non-local vector features. For example, if two axes of swirl exist, the algorithm will indicate a rotation that is a combination of the two swirl directions. The eigenvector method has successfully been integrated into a finite element solver for guiding mesh refinement around the vortex core [25].

Roth and Peikert have developed a method for finding core lines using higher-order derivatives, making it possible to find strongly curved or bent vortices [26]. They observe that the eigenvector method is equivalent to finding points where the acceleration **a** is parallel to the velocity **v**, or equivalently, to finding points of zero curvature. The acceleration **a** is defined as:

$$\mathbf{a} = \frac{D\mathbf{v}}{Dt}, \tag{3}$$

where the notation $\frac{Df}{Dt}$ is used for the *derivative following a particle*, which is defined, in a steady flow, as $\nabla f \cdot \mathbf{v}$. Therefore:

$$\mathbf{a} = \frac{D\mathbf{v}}{Dt} = \nabla\mathbf{v} \cdot \mathbf{v} = J \cdot \mathbf{v}, \tag{4}$$

with J the Jacobian of **v**, that is the matrix of its first derivatives.

Roth and Peikert improve the algorithm by defining vortex cores as points where

$$\mathbf{b} = \frac{D\mathbf{a}}{Dt} = \frac{D^2\mathbf{v}}{Dt^2} \tag{5}$$

is parallel to **v**, that is, points of zero torsion. The method involves computing a higher-order derivative, introducing problems with accuracy, but it performs very well. In comparison with the eigenvector method, this algorithm finds strongly curved vortices much more accurately. Roth and Peikert also introduce two attributes for the core lines: the strength of rotation and the quality of the solution. This makes it possible for the user to impose a threshold on the vortices, to eliminate weak or short vortices. Peikert and Roth have also introduced a new operator, the "parallel vectors" operator [27], with which they are able to mathematically describe a number of previously developed methods under one common denominator. Using this operator they can describe methods based on zero curvature, ridge and valley lines, extremum lines and more.

Jiang *et al.* recently presented a new approach for detecting vortex core regions [23]. The algorithm is based on an idea which has been derived from Sperner's lemma in combinatorial topology, which states that it is possible to deduce the properties of a triangulation, based on the information given at the boundary vertices. The algorithm uses this fact to classify points as belonging to a vortex core, based on the vector orientation at the neighbouring points. In 2D, the algorithm is very simple and straightforward, and has only linear complexity. In 3D, the algorithm is somewhat more difficult, because it first involves computing the vortex core direction, and next, the 2D algorithm is applied to the velocity vectors projected onto the plane perpendicular to the vortex core direction. Still, also the 3D algorithm has only linear complexity.

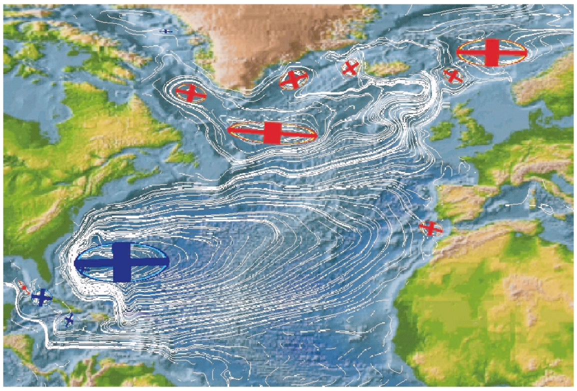

Figure 4: *Flow in the Atlantic Ocean, with streamlines and ellipses indicating vortices. Blue and red ellipses indicate vortices rotating clockwise and counterclockwise, respectively* [30].

The above described methods all use a local criterion for determining on a point-to-point basis where the vortices are located. The next algorithms use global, geometric criteria for determining the location of the vortices. This is a consequence of using another vortex definition.

Sadarjoen and Post present two geometric methods for extracting vortices in 2D fields [28]. The first is the curvature centre method. For each sample point, the algorithm computes the curvature centre. In the case of vortices, this would result in a high density of centre points near the centre of the vortex. The method works but has the same limitations as traditional point-based methods, with some false and some missing centres. The second method is the winding-angle method, which has been inspired by the work of Portela [29]. The method detects vortices by selecting and clustering looping streamlines. The winding angle α_w of a streamline is defined as the sum of the angles between the consecutive streamline segments. Streamlines are selected that have made at least one complete rotation, that is, $\alpha_w \geqslant 2\pi$. A second criterion checks that the distance between the starting and ending points is relatively small. The selected streamlines are used for vortex attribute calculation. The geometric mean is computed of all points of all streamlines belonging to the same vortex. An ellipse fitting is computed for each vortex, resulting in an approximate size and orientation for each vortex. Furthermore, the angular velocity and rotational direction can be computed. All these attributes can be used for visualising the vortices (see Figure 4).

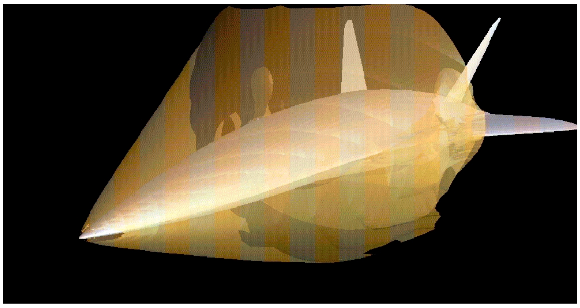

Figure 5: *A shockwave around an aircraft (H.-G. Pagendarm).*

4.2. Shock Wave Extraction

Shock waves are also important features in flow data sets, and can occur, for example, in flows around aircraft (see Figure 5). Shock waves can increase drag and cause structural failure, and therefore, are important phenomena for study. Shock waves are characterised by discontinuities in physical flow quantities such as pressure, density and velocity. Therefore, shock detection is comparable to edge detection, and similar principles could be used as in image processing. However, in numerical simulations, the discontinuities are often smeared over several grid points, due to the limited resolution of the grid.

Ma *et al.* have investigated a number of techniques for detecting and for visualising shock waves [31]. Detecting shocks in two dimensions has been extensively investigated [32–34]. However, these techniques are in general not applicable to shocks in three dimensions. They also describe a number of approaches for visualising shock waves. The approach of Haimes and Darmofal [35] is to create isosurfaces of the Mach number normal to the shock, using a combined density gradient/Mach number computation. Van Rosendale presents a two-dimensional shock-fitting algorithm for unstructured grids [34]. The idea relies on the comparison of density gradients between grid nodes.

Ma *et al.* compare a number of algorithms for shock extraction and also present their own technique [31]:

- The first idea is to create an isosurface of the points where the Mach number is one. However, this results in the sonic surface, which, in general, does not represent a shock.
- Theoretically, a better idea is to create an isosurface of the points where the normal Mach number is equal to one. However, if the surface is unknown, it is impossible to compute the Mach number, normal to the surface.
- This problem can be resolved, by approximating the shock normal with the density gradient, since a shock is also associated with a large gradient of the density. Therefore, $\nabla\rho$ is (roughly) normal to the shock surface. Thus, the algorithm computes the Mach number in the direction of, or projected onto, the density gradient. The shock surface is constructed from the points where this Mach number equals one. This algorithm is also used by Lovely and Haimes [36], but they define the shock region as the region within the isosurface of Mach number one, and use filtering techniques to reconstruct a sharp surface.
- Pagendarm and Seitz presented an algorithm that searches for maxima in the density gradient [37]. The first and second derivatives of the density in the direction of the velocity are computed. Next, zero-level isosurfaces are constructed of the second derivative, to find the extrema in the density gradient. Finally, the first derivative is used to select only the maxima, which correspond to shock waves, and discard the minima, which represent expansion waves. This can be done by selecting only positive values of the first derivative. However, the second derivative can also be zero in smooth regions with few disturbances. In these regions the first derivative will be small, therefore, these regions can be excluded by discarding all points where the first derivative is below a certain threshold ϵ. Of course, this poses the problem of finding the correct ϵ. When the value is too small, erroneous shocks will be found, but if the value is too large, parts of the shocks could disappear. This algorithm can also be used for finding discontinuities in other types of scalar fields, and thus for finding other types of features.
- Ma *et al.* present an adapted version of this algorithm, which uses the normal Mach number to do the selection in the third step [31]. Again, in the first and second step, the zero-level isosurfaces of the second directional derivative of the density are constructed. But for discriminating shock waves from expansion waves and smooth regions, the normal Mach number is used. More precisely, those points are selected where the normal Mach number is close to one. Here also, a suitable neighbourhood of one has to be chosen.

4.3. Separation and Attachment Line Extraction

Other features in flow data sets are separation and attachment lines on the boundaries of bodies in the flow. These are the lines where the flow abruptly moves away from or returns to the surface of the body (see Figure 6). These are important features in aerodynamic design because they can cause increased drag and reduced lift [2], and therefore, their occurrence should be prevented or at least minimised.

Helman and Hesselink use vector field topology to visualise flow fields [7]. In addition to the critical points, the attachment and detachment nodes on the surfaces of

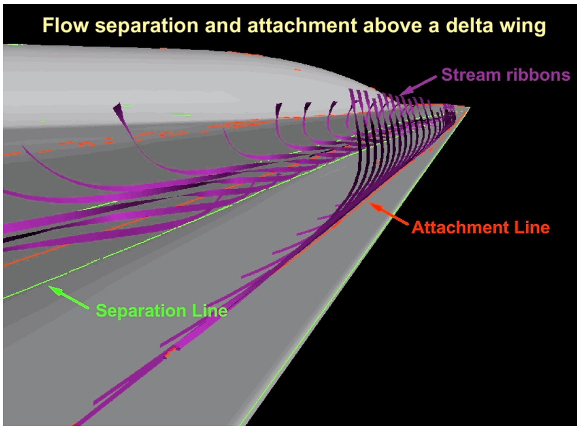

Figure 6: *Separation and attachment lines on a delta wing (D. Kenwright).*

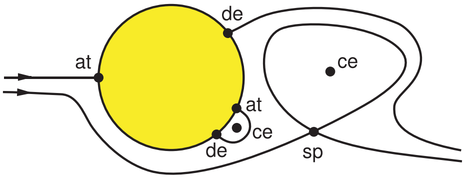

Figure 7: *Vector field topology: a topological skeleton of a flow around a cylinder [7].*

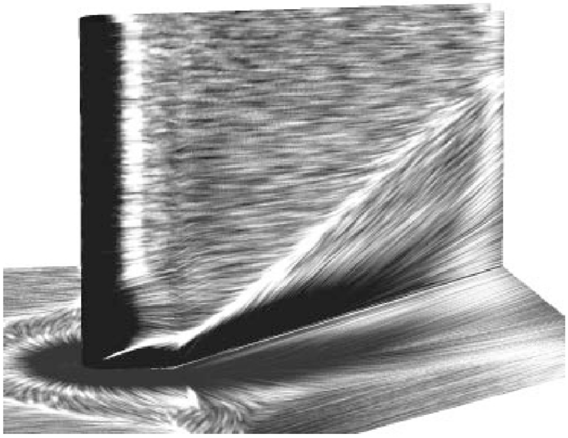

Figure 8: *Skin-friction on a blunt fin from a flow simulation at Mach 5, visualised with spot noise* [40].

bodies determine the topology of the flow (see Figure 7). The attachment and detachment nodes are not characterised by a zero velocity, because they only occur in flows with a no-slip condition, that is, all points on the boundaries of objects are constrained to have zero velocity. Instead, they are characterised by a zero tangential velocity. Therefore, streamlines impinging on the surface terminate at the attachment or detachment node, instead of being deflected along the surface.

Globus *et al.* designed and implemented a system for analysing and visualising the topology of a flow field with icons for the critical points and integral curves starting close to the critical points [38]. The system is also able to visualise attachment and detachment surfaces and vortex cores.

Pagendarm and Walter [39] and De Leeuw *et al.* [40] used skin-friction lines for visualising attachment and detachment lines in the blunt fin data set. For visualising these lines, the wall shear τ_w is computed, which is the flow velocity gradient perpendicular to the wall. Next, a standard streamline algorithm is used to integrate the skin-friction lines from the shear vector field. These skin-friction lines show the location of separation and attachment of the flow at the wall (see Figure 8).

Kenwright gives an overview of existing techniques for visualising separation and attachment lines and presents a new automatic feature detection technique for locating these lines, based on concepts from 2D phase plane analysis [41]. Some common approaches are:

- Particle seeding and computation of integral curves, such as streamlines and streaklines, which are constrained to the surface of the body. These curves merge along separation lines.
- Skin-friction lines can be used, analogous to surface oil flow techniques from wind tunnel experiments [39].
- Texture synthesis techniques can be used to create continuous flow patterns rather than discrete lines [40].
- Helman and Hesselink can generate separation and attachment lines from their vector field topology [7]. These lines are generated by integrating curves from the saddle and node type critical points on the surface in the direction of the real eigenvector. However, only closed separations are found, that is, curves that start and end at critical points.

Open separation does not require separation lines to start or end at critical points, and is therefore not detected using flow topology. Open separation has been observed in experiments, but had not previously been studied in flow simulations. However, the algorithm presented by Kenwright does detect both closed and open separation lines. The theory for this algorithm is based on concepts from linear phase plane analysis. It is assumed that the computational domain on the surface can be subdivided into triangles and the vector components are given at the vertices. The algorithm is executed for each triangle, making it suitable for parallelisation. For each triangle, a linear

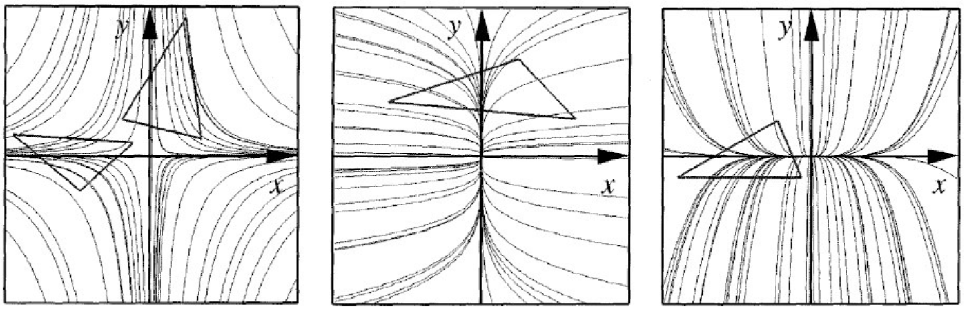

Figure 9: *Three phase portraits, for a saddle, repelling node and attracting node. The intersections of the triangles with the axes contribute line segments to attachment or separation lines [41].* © *2003 IEEE.*

vector field is constructed satisfying the vectors at the vertices. If the determinant of the Jacobian matrix is non-zero, the algorithm continues by calculating the eigenvalues and eigenvectors of the Jacobian. Every triangle has a critical point somewhere in its vector field. The linear vector field is translated to this critical point and the coordinate system is changed so that the eigenvectors are orthogonal. This (x, y) plane is also referred to as the Poincaré phase plane (see Figure 9). By computing tangent curves in the phase plane, we obtain the phase portrait of the system. For a saddle, the tangent curves or streamlines converge along the x and y axes. For a repelling node, they converge along the y axis and for an attracting node, they converge along the x axis. If the phase portrait is a saddle or a repelling node, the intersection of the y axis with the triangle is computed. If it intersects, the line segment will form part of an attachment line. If the phase portrait is a saddle or an attracting node, the intersection of the x axis with the triangle is computed, and if it does intersect, the line segment will form part of a separation line.

A problem with this algorithm is that disjointed line segments are computed instead of continuous attachment and separation lines. Other problems occur when the flow separation or attachment is relatively weak, or when the assumption of locally linear flow is not correct.

Kenwright *et al.* present two algorithms for detecting separation and attachment lines [42]. The first is the algorithm discussed above, the second is the parallel vector algorithm. Both algorithms use eigenvector analysis of the velocity gradient tensor. However, the first is element-based and results in disjointed line segments, while the second is point-based and will result in continuous lines.

In the parallel vector algorithm, points are located where one of the eigenvectors $\mathbf{e}_i$ of the gradient $\nabla\mathbf{v}$ is parallel to the vector field $\mathbf{v}$, that is, points where the streamline curvature is zero, or in formula:

$$\mathbf{e}_i \times \mathbf{v} = \mathbf{0}. \tag{6}$$

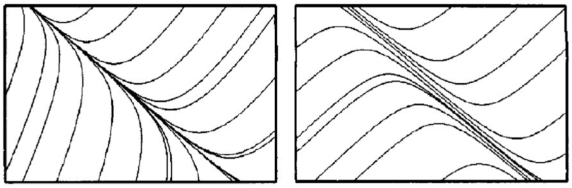

Figure 10: *The vector field in the left figure contains a separation line; the field in the right figure contains an inflection line [42].* © *2003 IEEE.*

The velocity vectors and the eigenvectors can be determined at the vertices of the grid and interpolated within the elements. At the vertices, $\mathbf{e}_i \times \mathbf{v}$ is calculated for both eigenvectors, but only if both eigenvectors are real, that is, the classification of $\nabla\mathbf{v}$ at the vertex is either a saddle or a node. If the cross product $\mathbf{e}_i \times \mathbf{v}$ changes sign across an edge, that means an attachment or separation line intersects the edge. The intersection point can then be found by interpolation along the edge. The attachment and separation lines can be constructed by connecting the intersection points in each element. The distinction between attachment and separation can be made easily, because attachment will occur where $\mathbf{v}$ is parallel to the smallest $\mathbf{e}_i$ and separation where $\mathbf{v}$ is parallel to the largest $\mathbf{e}_i$. Another set of lines is detected with this algorithm, the inflection lines, where one of the eigenvectors is locally parallel to the velocity vector, but the line itself is not an asymptote of neighbouring streamlines (see Figure 10). These inflection lines can easily be filtered out by checking if:

$$\nabla(\mathbf{e}_i \times \mathbf{v}) \cdot \mathbf{v} = \mathbf{0}. \tag{7}$$

This will not be true for inflection lines.

Both algorithms discussed by Kenwright *et al.* correctly identify many separation and attachment lines, but may fail in identifying curved separation lines [42]. The parallel vector algorithm will result in continuous lines, whereas the phase plane algorithm results in discontinuous line segments. Both algorithms do detect open separation lines, which do not start or end at critical points.

5. Feature Tracking and Event Detection

In time-dependent data sets, features are objects that evolve in time. Determining the correspondence between features in successive time steps, that actually represent the same object at different times, is called the *correspondence problem*. Feature tracking is involved with solving this correspondence problem. The goal of feature tracking is to be able to describe the evolution of features through time. During the evolution, certain *events* can occur, such as the interaction of two or more features, or significant shape changes of features. Event detection is the process of detecting such events, in order to describe the evolution of the features even more accurately.

There are a number of approaches to solving the correspondence problem. Features can be extracted directly from the spatio-temporal domain, thereby implicitly solving the correspondence problem. Or, when feature extraction is done in separate time steps, the correspondence can be solved based on region correspondence, or based on attribute correspondence.

5.1. Feature Extraction from the Spatio-temporal Domain

It is possible to perform feature extraction in 3D or 4D space-time. Tricoche *et al.* present an algorithm for tracking of 2D vector field topologies by interpolation in 3D space-time [8]. Bajaj *et al.* present a general technique for hypervolume visualisation [43]. They describe an algorithm to visualise arbitrary n-dimensional scalar fields, possibly with one or more time dimensions. Weigle and Banks extract features by isosurfacing 4D space-time [44]. This is conceptually similar to finding overlapping features in successive time steps. See also the next section (5.2), about region correspondence. Bauer and Peikert perform tracking of features in (4D or 5D) scale-space [45]. The idea is that the original data is smoothed using a Gaussian kernel. The standard deviation σ of this kernel can be any positive number, and is represented on the scale axis. Together with the normal 3D spatial axes, and possibly one time axis, this scale axis spans the scale-space. In the article, the focus is on line-type features, and specifically vortex cores, but that is just their main application, and not inherent to the algorithm. In 5D scale-space, it is possible to track features not only along the time axis, but also along the scale axis.

5.2. Region Correspondence

Region correspondence involves comparing the regions of interest obtained by feature extraction. Basically, the binary images from successive time steps, containing the features found in these time steps, are compared on a cell-to-cell basis. Correspondence can be found using a minimum distance or a maximum cross-correlation criterion [46] or by minimising an affine transformation matrix [47]. It is also possible to extract isosurfaces from the 4D time-dependent data set [44], where time is the fourth dimension. The correspondence is then implicitly determined by spatial overlap between successive time steps. This criterion is simple, but not always correct, as objects can overlap but not correspond, or correspond but not overlap. Silver and Wang explicitly use the criterion of spatial overlap instead of creating isosurfaces in four dimensions [48,49]. They prevent correspondence by accidental overlap, by checking the volume of the corresponding features and taking the best match. This is also the idea of attribute correspondence, which is discussed next. By using spatial overlap, certain events are implicitly detected, such as a *bifurcation* when a feature in one time step overlaps with two features in the next time step. Event detection is also discussed more extensively later, in Section 5.4.

5.3. Attribute Correspondence

With attribute correspondence, the comparison of features from successive frames is performed on the basis of the attributes of the features, such as the position, size, volume, and orientation. These attributes can be computed in the feature extraction phase (see Section 3.4) and can be used for description and for visualisation of the features, and also for feature tracking, as described here. The original grid data is not needed anymore. Samtaney *et al.* use the attribute values together with user-provided tolerances to create correspondence criteria [50]. For example, for position the following criterion could be used:

$$dist(pos(O_{i+1}), pos(O_i)) \leqslant T_{dist}, \tag{8}$$

where $pos(O_i)$ and $pos(O_{i+1})$ are the positions of the objects in time steps i and $i+1$, respectively, and T_{dist} is the user-provided tolerance. For scalar attributes, the difference or the relative difference could be used. For example, to test the relative difference of the volume, the following formula can be used:

$$\frac{vol(O_{i+1}) - vol(O_i)}{\max(vol(O_{i+1}), vol(O_i))} \leqslant T_{vol}, \tag{9}$$

where $vol(O_i)$ and $vol(O_{i+1})$ are the volumes of the features in the two time steps, and T_{vol} is the tolerance given by the user. Events such as a bifurcation can also be tested. If a feature in time step i splits into two features in time step $i+1$, the total volume after the event has to be approximately the same as before the event. The same formula can be used as for the normal volume test, except that $vol(O_{i+1})$ in this case equals the sum of the volumes of the separate features. The position criterion in case of a bifurcation event could

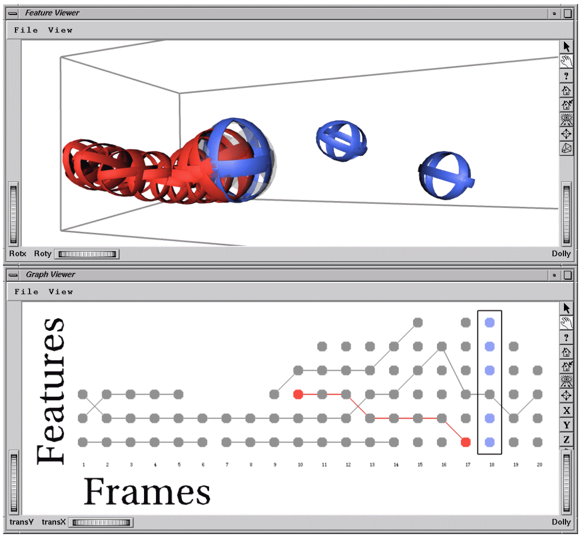

Figure 11: *One step during feature tracking. A path is shown with its prediction, and three candidates in the next time step* [52].

involve the weighted average of the individual positions after the event, where the positions are weighed with the volume:

$$dist(pos(O_i), \frac{\sum(vol(O_{i+1}) \cdot pos(O_{i+1}))}{\sum(vol(O_{i+1}))}) \leqslant T_{dist}, \tag{10}$$

where O_{i+1} now represents all objects in time step $i+1$ that are involved in the event.

Reinders *et al.* describe an algorithm for feature tracking, that is based on prediction and verification [51,52]. This algorithm is based on the assumption that features evolve predictably. That means, if a part of the evolution of a feature (*path*) has been found, a prediction can be made into the next time step (*frame*). Then, in that next time step, a feature is sought, that corresponds to the prediction (see Figure 11). If a feature is found that matches the prediction within certain user-provided tolerances, the feature is added to the evolution and the search is continued to the next time step. When no more features can be added to the path, a new path is started. In this manner, all frames are searched for starting points, both in forward and backward time direction, until no more paths can be created. A path is started by trying all possible combinations of features from two consecutive frames and computing the prediction to the next frame. Then, the prediction is compared to the candidate features in that frame. If there is a match between the prediction and the candidate, a path is started. To avoid any erroneous or coincidental paths, there is a parameter for the minimal path length, which is usually set to 4 or 5 frames. A candidate feature can be defined in two ways. All features in the frame can be used as candidates, or only unmatched features can be used, that is, those features that have not yet been assigned to any path. The first definition ensures that all possible combinations are tested and that the best correspondence is chosen. However, it could also result in features being added to more than one path. This has to be resolved afterwards. Using the second definition is much more efficient, because the more paths are found, the fewer unmatched features require testing. However, in this case, the results depend on the order in which the features are tested. This problem can be solved by starting the tracking process with strict tolerances and relaxing the tolerances in subsequent passes.

The prediction of a feature is constructed by linear extrapolation of the attributes of the features from the last two frames. Other prediction schemes could also be used, for example, if a priori knowledge of the flow is available.

The prediction is matched against real features using correspondence criteria, similar to the ones used by Samtaney *et al.* as discussed above [50]. For each attribute of the features, a correspondence function can be created, which returns a positive value for a correspondence within the given tolerance, with a value of 1 for an exact match, and a negative value for no correspondence. Each correspondence function is assigned a weight, besides the tolerance. Using this weight, a weighted average is calculated of all correspondence functions, resulting in the correspondence factor between the two features. For this correspondence factor, the same applies as for the separate correspondence functions, that is, a positive value indicates a correspondence, with 1 indicating a perfect match. A negative correspondence factor means no match.

5.4. Event Detection

After feature tracking has been performed, event detection is the next step. Events are the temporal counterparts of spatial features in the evolution of features. For example, if the path or evolution of a feature ends, it can be interesting to determine why that happens. It could be that the feature shrinks and vanishes, or that the feature moves to the boundary of the data set and disappears, or that the feature merges with another feature and the two continue as one. Samtaney *et al.* introduced the following events: continuation, creation, dissipation, bifurcation, amalgamation [50] (see Figure 12). Reinders *et al.* developed a feature tracking system that is able to detect these and other events [52]. The terminology they use is *birth* and *death* instead of creation and dissipation, and *split* and *merge* for bifurcation and amalgamation. Furthermore, they can detect *entry* and *exit* events, where a feature moves beyond the boundary of the data set. Finally, for a specific, graph-type feature, the system is able to detect changes in topology. It discriminates *loop* and *junction* events (see Figure 13). Many other types of events can be envisioned, but for each type specific detection criteria have to be provided.

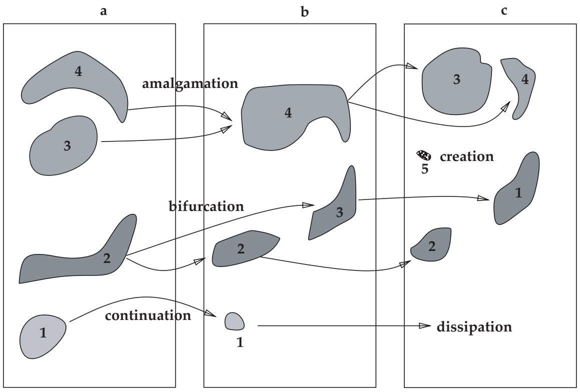

Figure 12: *The different types of events as introduced by Samtaney et al.* [50].

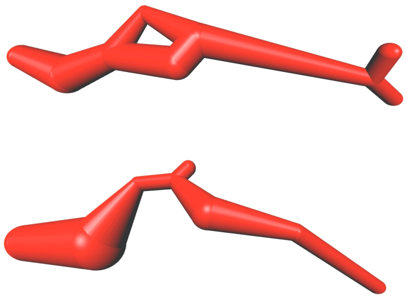

Figure 13: *A loop event has occurred. In the top figure, the feature contains a loop, in the bottom figure, the next frame, the loop has disappeared* [53].

For event detection, just as for feature tracking, only the feature attributes are used. Analogous to the correspondence functions, for event detection, event functions are computed. For example, to detect a death event, two conditions must hold. First, the volume of the feature must decrease. And second, the volume of the prediction must be very small or negative. The event function for this event returns a positive value if the volume of the prediction is within the user-provided tolerance, and is equal to one if the volume of the prediction is negative. If the volume is not within the tolerance, the returned value will be negative. The event functions for the separate attributes are combined into a single factor, which determines if the event is a death event. A birth event can be detected by doing the same tests in the backward time direction.

Similarly, the tests for split and merge events, and for entry and exit events are each other's reverse in time.

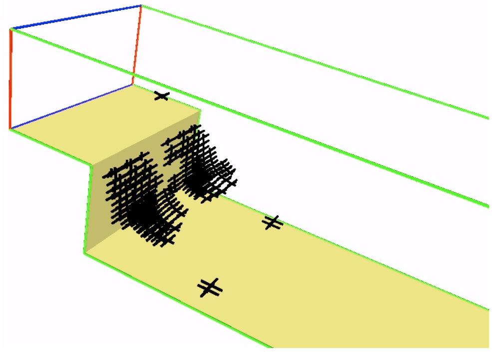

Figure 14: *Visualisation of the selected points in the backward-facing-step data set* [54].

6. Visualisation of Features and Events

The final step in the feature extraction pipeline is, of course, the visualisation of the features. A number of techniques will be covered in this section. The most straightforward visualisation is to show the nodes in the data set, that have been selected in the first step of the feature extraction pipeline. This step results in a binary data set, with each value indicating whether the corresponding node has been selected or not. This binary data set can be visualised, for example, with crosses at the selected nodes. In Figure 14, such a visualisation is shown. The visualisation is of a simulation of the flow behind a backward-facing step. The feature that is visualised here is a recirculation zone, behind the step. The points were selected with the criterion: normalised helicity $H > 0.6$.

Another simple visualisation technique is to use isosurfaces. This can be done on the binary data set, resulting from the selection step, or, if the selection expression is a simple threshold, directly on the original data set. This results in isosurfaces enclosing the selected regions.

Also, other standard visualisation techniques can be used in combination with the Boolean data set resulting from the selection step. For example, in a 3D flow data set, using the standard methods for seeding streamlines or streamtubes, will not provide much information about the features and will possibly result in visual clutter. However, if the selected points are used to seed streamlines, both backward and forward in time, this can provide useful information about the features and their origination. See Figure 15, for an example, where two streamtubes are shown in the backward-facing-step data set. The radius of the tubes is inversely proportional to the square root of the local velocity magnitude, and the colour of the tubes corresponds to the pressure.

If, instead of the separate selected points, the attributes

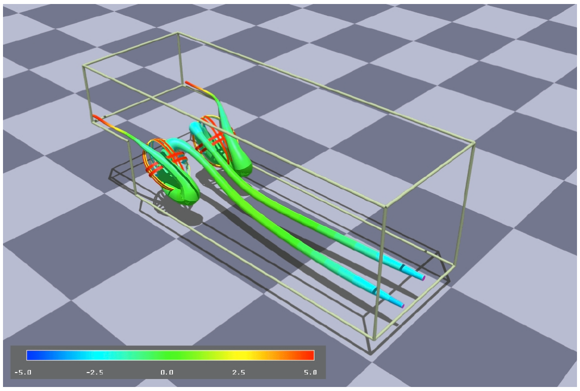

Figure 15: *Visualisation with streamtubes of the recirculation in the backward-facing-step data set* [55].

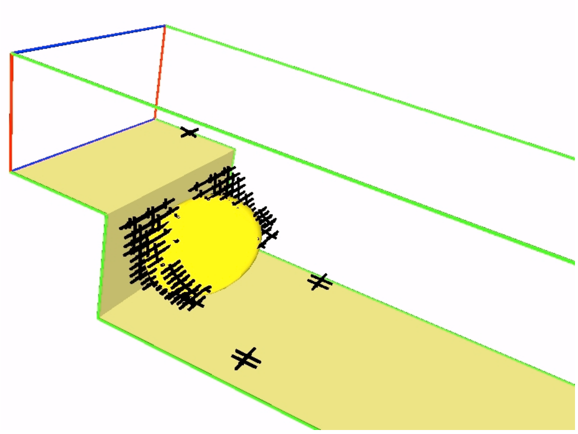

Figure 16: *An ellipsoid fitting computed from the selected points in the backward-facing-step data set* [54].

are used, that have been computed in the feature extraction process, then parametric icons can be used for visualising the features.

If an ellipsoid fitting of the selected clusters has been computed, there are three attribute vectors: the centre position, the axis lengths, and the axis orientations, which can be mapped onto the parameters of an ellipsoid icon. This is a simple icon, but very efficient and accurate. It can be represented with 9 floating-point values, and is therefore space-efficient. Furthermore, it can be very quickly visualised, and although it is simple, it gives an accurate indication of the position and volume of a feature. In Figure 16, an ellipsoid fitting is computed from the selected points in Figure 14. In Figure 17, vortices are shown from a CFD simulation with turbulent vortex structures. The features have been selected by a threshold on vorticity

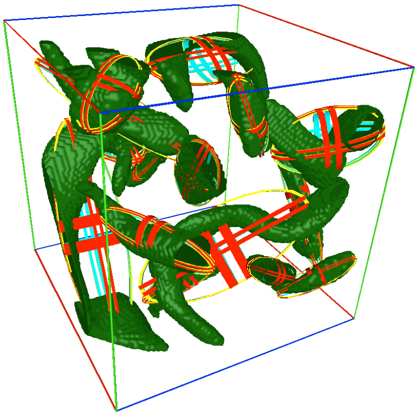

Figure 17: *Vortices in a data set with turbulent vortex structures, visualised using isosurfaces and ellipsoids* [53].

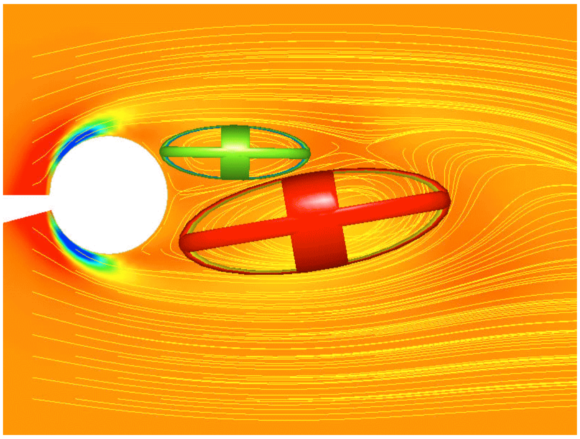

Figure 18: *Vortices behind a tapered cylinder. The colour of the ellipsoids represents the rotational direction* [28].

magnitude. They are being visualised with isosurfaces and ellipsoids. It is clearly visible that, in this application, with the strongly curved features, the ellipsoids do not give a good indication of the shape of the features. But, as mentioned above, the position and volume attributes of the ellipsoids will be accurate, and can be used for feature tracking.

In Figure 18, the flow past a tapered cylinder is shown. Streamlines indicate the flow direction, and rotating streamlines indicate vortices. The vortices are selected by locating these rotating streamlines, using the winding-angle method [28]. Ellipses are used to visualise the vortices, with the colour indicating the rotational direction. Green means clockwise rotation, red means counterclockwise rotation. The slice is coloured with λ_2, which is the second-largest eigenvalue of the tensor $S^2 + \Omega^2$ (see Section 4.1). The tapered cylinder data set consists of a number of horizontal

Figure 19: *3D Vortex structures behind a tapered cylinder* [56]. *The number and curvature of the spokes indicate the rotational speed and direction, respectively.*

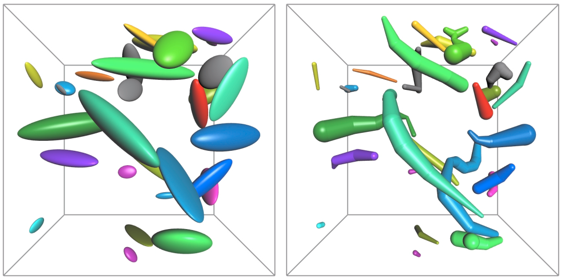

Figure 20: *Turbulent vortex structures represented by ellipsoid icons (left) and skeleton icons (right)* [53].

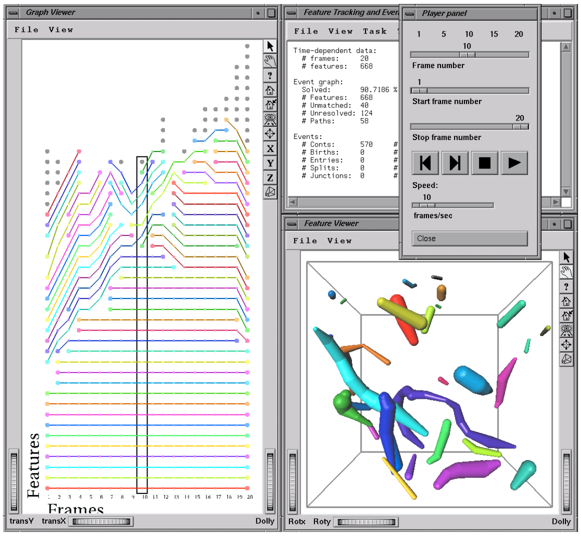

Figure 21: *Playing through the turbulent vortex data set.*

slices, such as the one in Figure 18. Figure 19 shows an image of the 3D vortices, which have been constructed from the ellipses extracted in each slice [56].

For the 3D vortices in Figure 17, an other type of icon has to be used, if we want to visualise the strongly curved shape of the features. Reinders *et al.* present the use of skeleton graph descriptions for features, with which they can create icons that accurately describe the topology of the features, and approximately describe the shape of the features [5]. Compare the use of ellipsoid icons with the use of skeleton icons in Figure 20.

For visualising the results of feature tracking, it is of course essential to visualise the time dimension. The most obvious way is to animate the features, and to give the user the opportunity to browse through the time steps, both backward and forward in time. Figure 21 shows the player from the feature tracking program, developed by Reinders *et al.* [52]. On the left of the figure, the graph viewer is shown, which gives an abstract overview of the entire data set, with the time steps on the horizontal axis, and the features represented by nodes, on the vertical axis. The correspondences between features from consecutive frames are represented by edges in the graph, and therefore, the evolution of a feature in time, is represented by a path in the graph. On the right of the figure, the feature viewer is shown, in which the feature icons from the current frame are displayed. Also, a control panel is visible, with which the animation can be started, paused, and played forward and backward.

The graph viewer can also be used for visualising events [53]. For each event, a specific icon has been created, which is mapped onto the nodes of the graph, so that the user can quickly see which events occur where, and how often they occur. In Figure 22, the graph viewer is shown, with a part of the graph, containing a number of events. Each event is clearly recognisable by its icon. In Figure 23, two frames are shown, between which a split event has occurred. In both frames, the features are shown with both ellipsoid and skeleton icons. The advantage of the use of skeleton icons in this application is obvious. Because the shape of the features is much more accurately represented by the skeleton icons, changes in shape and events such as these are much more easily detected.

7. Conclusions and Future Prospects

Feature extraction is selection and simplification based on content: extracting relevant high-level information from a data set, visualising the data from a problem-oriented point

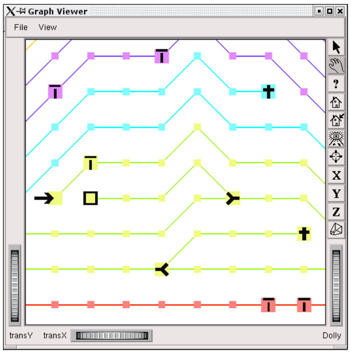

Figure 22: *Events are visualised in the graph viewer with special, characteristic icons.*

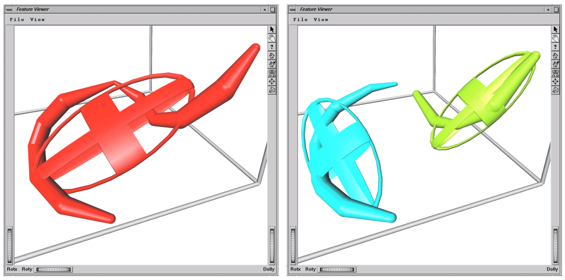

Figure 23: *A split event, before (left) and after (right). The features are visualised with both an ellipsoid and a skeleton icon* [53].

of view. This leads to a large reduction of the data size, and to fully or semi-automatic generation of simple and clear images. The techniques are generally very specific for a certain type of problem (such as vortex detection), the relation with the original raw data is indirect, and the reduction is achieved at the cost of loss of other information, which is considered not relevant for the purpose. But the techniques generalise well to analysis of time-dependent data sets, leading to condensed episodic visual summaries.

A good possibility is combining feature extraction techniques with direct or geometric techniques. For example, selective visualisation has been used effectively with streamline generation (Figure 15), to place seed points in selected areas, and show important structures with only a small number of streamlines. Combining simple advection-based techniques with iconic feature visualisation can also clarify the relation between the raw data and the derived information used in feature detection (Figure 18). The work of visualisation and simulation experts will become inseparable in the future: the distinction between simulation and visualisation will be increasingly blurred. A good example is the tracking of phase fronts (separation between two different fluids in multifluid flows) using level set methods [57], where the feature extraction is a part of both simulation and visualisation.

How about practical application? Feature-based techniques have been incorporated in commercial visualisation systems (`http://www.ensight.com/products/flow-feature.html`). The practical use of flow visualisation is most effective when visualisation experts closely cooperate with fluid dynamics experts. This is especially true in feature-based visualisation, where developing detection criteria is closely connected to the physical phenomena studied. But also other disciplines can contribute to this effort: mathematicians, artists and designers, experimental scientists, image processing specialists, and also perceptual and cognitive scientists [58].

In feature-based visualisation, the following areas need additional work:

- interactive techniques to support extraction and tracking of features [59];
- detection and tracking of new types of features, such as recirculation zones, boundary layers, phase fronts, and mixing zones, and detection of new types of events;
- comparative visualisation based on quantitative feature comparison;
- topological analysis: extension to finding separation surfaces in 3D and to time-dependent flows;
- image processing: adaptation of image segmentation and filtering techniques to irregular grids and use with vector fields;
- online steering of large simulations based on feature extraction and event detection.

Overlooking the whole landscape of flow visualisation techniques, we can say that visualisation of 2D flows has reached a high level of perfection, and for visualisation of 3D flows a rich set of techniques is available. In the future, we will concentrate on techniques that scale well with ever increasing data set sizes, and therefore simplification, selection, and abstraction techniques will get more attention.

Acknowledgements

This project was partly supported by the Netherlands Organization for Scientific Research (NWO) on the NWO-EW Computational Science Project “Direct Numerical Simulation of Oil/Water Mixtures Using Front Capturing Techniques”.

Part of this work has been done at the VRVis Research Center, Vienna, Austria, which is funded by the Austrian governmental research program K plus (`http://www.kplus.at`).

References

1. W. C. de Leeuw. Presentation and Exploration of Flow Data, Delft University of Technology, 1997.

2. M. Roth. Automatic Extraction of Vortex Core Lines and Other Line-Type Features for Scientific Visualization, Diss. ETH No. 13673, Swiss Federal Institute of Technology, ETH Zürich, 2000.

3. R. B. Haber and D. A. McNabb. Visualization Idioms: A Conceptual Model for Scientific Visualization Systems. In G. M. Nielson, B. D. Shriver and L. Rosenblum (eds), *Visualization in Scientific Computing*, IEEE Computer Society Press, pp. 75–93. 1990.

4. D. Silver, N. J. Zabusky, V. Fernandez, M. Gao and R. Samtaney. Ellipsoidal Quantification of Evolving Phenomena. In N. M. Patrikalakis (ed), *Scientific Visualization of Natural Phenomena*, Springer, pp. 573–588. 1991.

5. F. Reinders, M. E. D. Jacobson and F. H. Post. SkeletonGraph Generation for Feature Shape Description. In *DataVisualization 2000. Proc. VisSym'00*, pp. 73–82. 2000.

6. J. L. Helman and L. Hesselink. Representation and Display of Vector Field Topology in Fluid Flow Data Sets. *IEEE Computer*, 22(8):27–36, 1989.

7. J. L. Helman and L. Hesselink. Visualizing vector field topology in fluid flows. *IEEE CG&A*, 11(3):36–46, 1991.

8. X. Tricoche, T. Wischgoll, G. Scheuermann and H. Hagen. Topology Tracking for the Visualization of Time-Dependent Two-Dimensional Flows. *Computers & Graphics*, 26(2):249–257, 2002.

9. G. Scheuermann, H. Kruger, M. Menzel and A. P. Rockwood. Visualizing non-linear vector field topology. *IEEE TVCG*, 4(2):109–116, 1998.

10. W. C. de Leeuw and R. van Liere. Visualization of Global Flow Structures Using Multiple Levels of Topology. In *Data Visualization '99. Proc. VisSym'99*, pp. 45–52.

11. X. Tricoche, G. Scheuermann and H. Hagen. Continuous Topology Simplification of Planar Vector Fields. In *Proc. Visualization '01*, pp. 159–166. 2001.

12. T. van Walsum. Selective Visualization on Curvilinear Grids, Delft University of Technology, The Netherlands, 1995.

13. F. Reinders, H. J. W. Spoelder and F. H. Post. Experiments on the Accuracy of Feature Extraction. In *Proc. 9th EG Workshop on Visualization in Scientific Computing*, pp. 49–58. 1998.

14. M. Roth and R. Peikert. Flow Visualization for Turbomachinery Design. In *Proc. Visualization '96*, pp. 381–384. 1996.

15. D. C. Banks and B. A. Singer. A Predictor-Corrector Technique for Visualizing Unsteady Flow. *IEEE TVCG*, 1(2):151–163, 1995.

16. N. J. Zabusky, O. N. Boratav, R. B. Pelz, M. Gao, D. Silver and S. P. Cooper. Emergence of Coherent Patterns of Vortex Stretching during Reconnection: a Scattering Paradigm. *Physical Review Letters*, 67(18):2469–2472, 1991.

17. J. Villasenor and A. Vincent. An Algorithm for Space Recognition and Time Tracking of Vorticity Tubes in Turbulence. *Computer Vision, Graphics, and Image Processing: Image Understanding*, 55(1):27–35, 1992.

18. Y. Levy, D. Degani and A. Seginer. Graphical Visualization of Vortical Flows by Means of Helicity. *AIAA Journal*, 28(8):1347–1352, 1990.

19. L. A. Yates and G. T. Chapman. Streamlines, Vorticity Lines, and Vortices. AIAA-91-0731, American Institute of Aeronautics and Astronautics, 1991.

20. S. K. Robinson. Coherent Motions in the Turbulent Boundary Layer. *Ann. Rev. Fluid Mech.*, 23:601–639, 1991.

21. J. Jeong and F. Hussain. On the Identification of a Vortex. *J. Fluid Mechanics*, 285:69–94, 1995.

22. D. Sujudi and R. Haimes. Identification of swirling flow in 3D vector fields. AIAA Paper 95-1715, 1995.

23. M. Jiang, R. Machiraju and D. Thompson. A Novel Approach To Vortex Core Region Detection. In *Data Visualization 2002. Proc. VisSym'02*, pp. 217–225. 2002.

24. D. N. Kenwright and R. Haimes. Automatic Vortex Core Detection. *IEEE CG&A*, 18(4):70–74, 1998.

25. M. Dindar, A. Lemnios, M. S. Shephard, K. Jansen and D. Kenwright. Effect of Tip Vortex Resolution on UH-60A Rotor-Blade Hover Performance Calculations. In *54th AHS Annual Forum and Technology Display*. American Helicopter Society, Alexandria, VA, 1998.

26. M. Roth and R. Peikert. A Higher-Order Method For Finding Vortex Core Lines. In *Proc. Visualization '98*, pp. 143–150. 1998.

27. R. Peikert and M. Roth. The Parallel Vectors Operator — A Vector Field Visualization Primitive. In *Proc. Visualization '99*, pp. 263–270. 1999.

28. I. A. Sadarjoen and F. H. Post. Geometric Methods for Vortex Detection. In *Data Visualization '99. Proc. VisSym'99*, pp. 53–62. 1999.

29. L. M. Portela. On the Identification and Classification of Vortices, Stanford University, School of Mechanical Engineering, 1997.

30. I. A. Sadarjoen and F. H. Post. Detection, Quantification, and Tracking of Vortices using Streamline Geometry. *Computers & Graphics*, 24(3):333–341, 2000.

31. K.-L. Ma, John van Rosendale and Willem Vermeer. 3D Shock Wave Visualization on Unstructured Grids. In *Proc. Symposium on Volume Visualization*, pp. 87–94,104. 1996.

32. B. van Leer. The computation of steady solutions to the Euler equations: a perspective. University of Michigan, 1986.

33. K. W. Morton and M. A. Rudgyard. Shock recovery and the cell vertex scheme for the steady Euler equations. In D. L. Dowyer, M. Y. Hussaini and R. G. Voigt (eds), *Lecture notes in physics*, Springer, pp. 424–428. 1989.

34. J. van Rosendale. Floating shock fitting via lagrangian adaptive meshes. 94-89, Institute for Computer Applications in Science and Engineering, 1994.

35. R. Haimes and D. Darmofal. Visualization in computational fluid dynamics: a case study. In *Proc. Visualization '91*, pp. 392–397. 1991.

36. D. Lovely and R. Haimes. Shock Detection from Computational Fluid Dynamics Results. In *Proceedings of the 14th AIAA Computational Fluid Dynamics Conference*. AIAA paper 99-3285. American Institute of Aeronautics and Astronautics, 1999.

37. H.-G. Pagendarm and B. Seitz. An Algorithm for Detection and Visualization of Discontinuities in Scientific Data Fields Applied to Flow Data with Shock Waves. In P. Palamidese (ed), *Scientific Visualization: Advanced Software Techniques*, Ellis Horwood, pp. 161–177. 1993.

38. A. Globus, C. Levit and T. Lasinski. A Tool for Visualizing the Topology of 3D Vector Fields. In *Proc. Visualization '91*, pp. 33–39. 1991.

39. H.-G. Pagendarm and B. Walter. Feature detection from vector quantities in a numerically simulated hypersonic flow field in combination with experimental flow visualization. In *Proc. Visualization '94*, pp. 117–123. 1994.

40. W. C. de Leeuw, H.-G. Pagendarm, F. H. Post and B. Walter. Visual Simulation of Experimental Oil-FlowVisualization by Spot Noise Images from Numerical Flow Simulation. In *Proc. 6th EG Workshop on Visualization in Scientific Computing*, pp. 135–148. 1995.

41. D. N. Kenwright. Automatic detection of open and closed separation and attachment lines. In *Proc. Visualization '98*, pp. 151–158. 1998.

42. D. N. Kenwright, C. Henze and C. Levit. Feature Extraction of Separation and Attachment Lines. *IEEE TVCG*, 5(2):1999.

43. C. L. Bajaj, V. Pascucci, G. Rabbiolo and D. R. Schikore. Hypervolume Visualization: a Challenge in Simplicity. In *Proc. Symposium on Volume Visualization*, pp. 95–102. 1998.

44. C. Weigle and D. C. Banks. Extracting Iso-valued Features in 4-dimensional Scalar Fields. In *Proc. Symposium on Volume Visualization*, pp. 103–110. 1998.

45. D. Bauer and R. Peikert. Vortex Tracking in Scale-Space. In *Data Visualization 2002. Proc. VisSym'02*, pp. 233–240. 2002.

46. M. D. in den Haak, H. J. W. Spoelder and F. C. A. Groen. Matching of Images by Using Automatically Selected Regions of Interest. In *Computing Science in the Netherlands '92*, pp. 27–40. 1992.

47. D. S. Kalivas and A. A. Sawchuk. A Region Matching Motion Estimation Algorithm. *Computer Vision, Graphics, and Image Processing: Image Understanding*, 54(2):275–288, 1991.

48. D. Silver and X. Wang. Tracking and Visualizing Turbulent 3D Features. *IEEE TVCG*, 3(2):1997.

49. D. Silver and X. Wang. Visualizing Evolving Scalar Phenomena. *Future Generation Computer Systems*, 15(1):99–108, 1999.

50. R. Samtaney, D. Silver, N. Zabusky and J. Cao. Visualizing Features and Tracking Their Evolution. *IEEE Computer*, 27(7):20–27, 1994.

51. F. Reinders, F. H. Post and H. J. W. Spoelder. Attribute-Based Feature Tracking. In *Data Visualization '99. Proc. VisSym'99*, pp. 63–72. 1999.

52. F. Reinders, F. H. Post and H. J. W. Spoelder. Visualization of Time-Dependent Data using Feature Tracking and Event Detection. *The Visual Computer*, 17(1):55–71, 2001.

53. F. Reinders. Feature-Based Visualization of Time-Dependent Data, Delft University of Technology, The Netherlands, 2001.

54. I. A. Sadarjoen and F. H. Post. Techniques and Applications of Deformable Surfaces. In H. Hagen, G. M.

Nielson and F. H. Post (eds), *Scientific Visualization, Proceedings Dagstuhl '97*, IEEE Computer Society, pp. 277–286. 2000.

55. T. van Walsum, F. H. Post, D. Silver and F. J. Post. Feature Extraction and Iconic Visualization. *IEEE TVCG*, 2(2):111–119, 1996.

56. F. Reinders, I. A. Sadarjoen, B. Vrolijk and F. H. Post. Vortex Tracking and Visualisation in a Flow Past a Tapered Cylinder. *Computer Graphics Forum*, 21(4):675–682, 2002.

57. J. A. Sethian. Level Set Methods and Fast Marching Methods. *Cambridge Monographs on Applied and Computational Mathematics*. Cambridge University Press, 2nd edition, 1999.

58. F. H. Post and J. J. van Wijk. Visual Representation of Vector Fields: Recent Developments and Research Direction. In L. Rosenblum (ed), *Scientific Visualization: Advances and Challenges, chapter 23*, Academic Press, London, pp. 367–390. 1994.

59. H. Doleisch, M. Gasser and H. Hauser. Interactive Feature Specification for Focus+Context Visualization of Complex Simulation Data. In *Data Visualization 2003. Proc. VisSym'03*, pp. 239–248. 2003.