Visual Analysis of Differential Information

Helwig Hauser, Hauser@VRVis.at

VRVis Research Center in Vienna, Austria,
http://www.VRVis.at/

Abstract

In many fields of research as well as in many applications, differential information plays an important role. Often, the differential information under investigation results from the measurement, simulation, or modeling of a real-world phenomenon. As such, differential information may be given in the form of a (usually large) set of vectorial samples or analytically, e.g., in the form of differential equations.

There are different options of how to deal with differential information, and visualization, especially interactive visualization, is one very interesting of them. The visual system of human beings offers a broad-band access to the human mind and visualization exploits this opportunity to enable insight into often large and complex datasets.

Below, a review of visualization approaches to the depiction and analysis of differential information is presented with a special focus on the related work which has been done in Vienna, Austria, during the past decade. Questions of how to depict differential information, of how to do selective visualization, of how to enable interactive visual analysis of differential information, and of how to integrate computational approaches within the visualization are addressed, especially also in the context of data which is given in more than two dimensions.

1 Introduction

Science and applications are continually engaged in the investigation of real-world phenomena. On the one hand, scientists develop mathematical models of real-world phenomena such as equations to represent gravitational forces at astronomic scales or models of electron density around atoms and molecules. On the other hand, measurements and computational simulations are used to depict phenomena of the real world in form of electronic data. Models as well as datasets exist in different forms – in this paper we will concentrate on differential information, i.e., information which represents phenomena of change, to charaterize most generally.

Together with the development of new models as well as in conjunction with the acquisition of datasets through measurements or simulations, the representation and analysis of the resulting information is of great importance. There are different means of how to investigate such models or datasets, including approaches which are more analytical (e.g., mathematical, statistical, and other computational approaches) and such which more rely on a more direct form of representation. Visualization has become quite popular as one very effective opportunity for the investigation of models or datasets. Below, we will review what visualization can do when it comes to representing and/or analyzing differential information. To do so, we will discuss these opportunities as ranging from pure representational methods to such which are more based on analytical procedures.

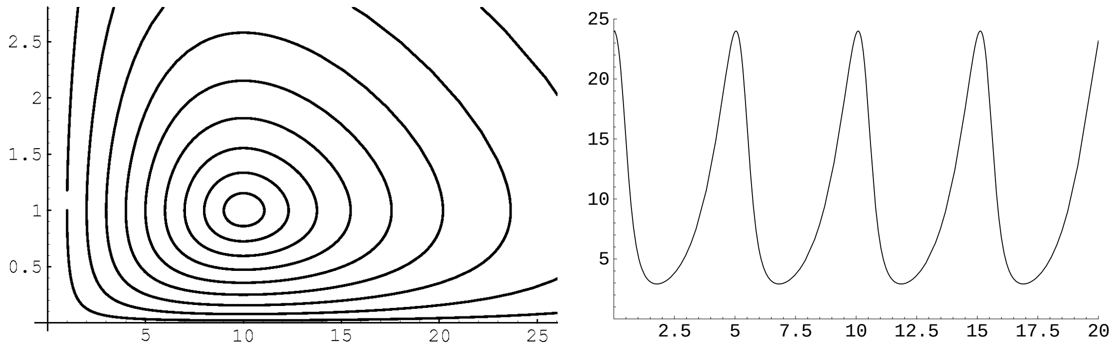

Figure 1: Cycles of (counterclockwise) system evolution in the Lotka-Volterra predator-prey model (left [31]) and the evolution of prey over time for one particular starting scenario (right [31]).

2 Differential Information

In the context of this review, we will use the term *differential information* for any model or data which stands for something varying. Often, this varying "something" will be some sort of physical matter in spatial dimensions, e.g., gaseous or liquid fluid which moves over time. However, to some extent this "varying something" can also be a set of system states, e.g., the population sizes of different species in a joint predator-prey environment.

In the context of our review, we will further assume that this "varying something" does extend over multiple dimensions. Most visualization approaches as addressed further below provide opportuntities for the representation or analysis of two- or three-dimensional models/data – this is due to the natual mapping of models/data to the 2D or 3D visualization space. However, there are also visualization approaches which address the challenge of how to represent/analyze models/data in higher dimensions [64, etc.].

To provide a more formal basis for our review, we introduce the following notation to represent the differential information under investigation:

$$\mathbf{v} = \mathrm{d}\,\mathbf{p}\,/\,\mathrm{d}\,t \qquad \mathbf{p} \in \Omega \subseteq \mathcal{R}^n,\ \mathbf{v} \in \mathcal{R}^n,\ t \in \mathcal{R} \tag{1}$$

According to this definition, $\mathbf{v}$ can be interpreted as (differential) information about how some n-dimensional $\mathbf{p}$ moves or changes over time t. In the context of this review, we will assume that $\mathbf{v}$ is either given as an analytic model $\mathbf{v}(\mathbf{p}, t)$, e.g., as a set of n differential equations, or as a discrete set of n-dimensional vectorial samples $\mathbf{v}[\mathbf{p}_i, t_j]$ such as a vector field which is associated with a certain grid structure. In these (quite general) considerations, $\mathbf{v}$ is assumed to be also dependent on time t – in many cases, however, $\mathbf{v}$ is assumed to be steady with respect to time, resulting in the simpler scenarios of $\mathbf{v}(\mathbf{p})$ or $\mathbf{v}[\mathbf{p}_i]$.

With respect to visualization, there are some important differences according the whether an analytic model or a discrete dataset is given – however, still many of the below outlined approaches can be used to deal with either form of models or data. In fact, there are many cases where a dataset which is given in discrete form (vector field $\mathbf{v}[\mathbf{p}_i]$) actually represents a continuous phenomenon, e.g., some continuous fluid flow. In these cases, appropriate interpolation or approximation schemes can be used to reconstruct an analytic data model $\mathbf{v}(\mathbf{p}, t)$ from the set of samples, e.g., through the use of a linear reconstruction filter h [42]: $\mathbf{v}(\mathbf{p}, t) = \sum_i h(\mathbf{p} - \mathbf{p}_i)\,\mathbf{v}[\mathbf{p}_i]$.

A simple example for an analytic model $\mathbf{v}$ is the predator-prey model of Lotka and Volterra [43] in which the temporal evolution of two populations (prey and predators) in a joint environment is modeled differentially:

$$\mathbf{v}(\mathbf{p}) = \mathrm{d}\,\mathbf{p}\,/\,\mathrm{d}\,t = \mathrm{d}\begin{pmatrix} x \\ y \end{pmatrix}/\,\mathrm{d}\,t = \begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} = \begin{pmatrix} rx - pxy \\ epxy - my \end{pmatrix} \tag{2}$$

In this steady model ($\mathbf{v}$ is not dependent on time), the population amounts vector $\mathbf{p} \in [0,\infty)^2$ and its differential changes $\mathbf{v} \in \mathcal{R}^2$ are two-dimensional entities with x and y representing the amount of prey and predators in the system, respectively, and r, p, and m being system parameters which are assumed constant for one particular instantiation of this model. For any particular configuration $\mathbf{p}$ of the system (a particular amount of prey x and predators y), the above equation yields the differential system change $\mathbf{v}$ with respect to time at this particular configuration. Figure 1 shows a selected set of temporal evolutions in the Lotka-Volterra model (cyclic evolutions in the x-y-plane on the left, and the evolution of prey x over time t on the right). The cyclic behavior of temporal evolutions in this model stems from the fact that $\dot{x}$ is modeled to be (largely) positive when y is (very) small and negative when y is comparably large, whereas $\dot{y}$ is modeled to be large when x is large and small when x is small.

Starting from a particular configuration $\mathbf{p}_0$ of the system at a particular point in time t_0, the temporal evolution of the system can be calculated by integrating equation 2 over time. Thereby, a parameterized path (or trajectory or pathline or . . .) $\mathbf{p}(s)$ is calculated:

$$\mathbf{p}(s) = \mathbf{p}_0 + \int_{\tau=0}^{s} \mathbf{v}(\mathbf{p}(\tau),\, t_0{+}\tau)\;\mathrm{d}\,\tau \qquad 0 \le s \in \mathcal{R} \tag{3}$$

Integration curves $\mathbf{p}(s)$ represent the evolution of such a system over time. Accordingly, many visualization techniques build upon this concept to represent differential information. Unfortunately, it usually is impossible to solve equation 3 analytically. Alternatively, numerical integration schemes are required to approximate the solutions $\mathbf{p}(s)$. In visualization, often relatively simple integration procedures are used. In addition to the most simple Euler intergration scheme – $\mathbf{p}_E(t{+}\Delta t) = \mathbf{p}(t){+}\Delta t\,\mathbf{v}(\mathbf{p}(t))$ –, which only is useful when very small time steps Δt are utilized, more advanced integration schemes such as the 2nd- or 4th-order Runge-Kutta integration [40, 42] are commonly used.

An example for a discrete form of differential information $\mathbf{v}$ is the result from a numerical CFD (computational fluid dynamics) simulation (or a subset thereof) which is vectorial information given on a certain grid, i.e., a vector field. Figure 2 shows (parts of) a computational grid (in this case just a 2D longitudinal cross-section through a scene with a car, composed of several curvilinear blocks) in the upper image – near to the car body the flow domain is finely resolved whereas farther away larger grid cells are used – and a simple visualization of the vector field $\mathbf{v}[\mathbf{p}_i]$ as resulting from the CFD simulation in the lower image – the arrows represent the flow direction at the cell centers $\mathbf{p}_i$, color has been used to represent pressure in this flow.

In general, the kind of grids used for the simulation and the kind of result which is delivered can be different. Especially with respect to the grids employed, a large variety of options exist, ranging from simple Cartesian grids to any other forms of orthogonal grids, to curvilinear grids, to unstructured grids, to block-strucutured grids, etc. [40, 39]. In addition to cell-centered vector fields (as demonstrated in figure 2), also vertex-centered vector fields are quite common. Several issues arise in visualization in dependance of whether the one of other kind of grid is used and which kind of results file is delivered, including the challenge of *efficient point location* in large grids, proper *flow reconstruction*, and the *computation of derived data* such as local flow attributes which depend on the local derivation $\nabla\mathbf{v}$ at $\mathbf{p}$ [40, 52].

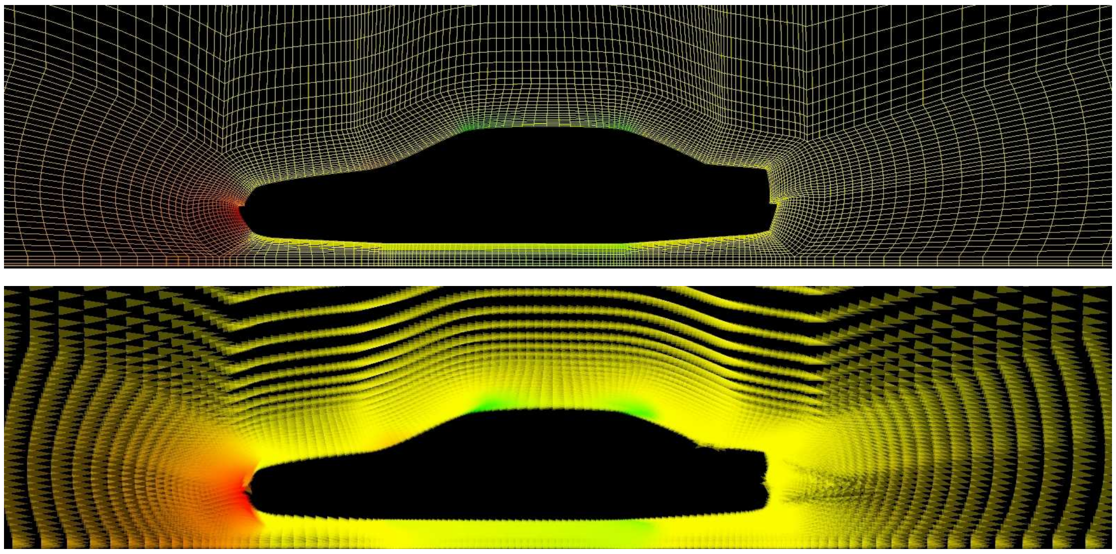

Figure 2: A 2D grid composed of curvilinear blocks (only a small part shown) for a CFD simulation of air flow around a car body (upper image) and a simple arrows-based visualization of the resulting vector field (below). *Data courtesy by AVL List GmbH, Graz, Austria*.

3 Flow Visualization

Visualization (as considered in this review) is the utilization of computer graphics for the purpose of enabling insight into extensive and/or complex amounts of information. Depending on the task of the user, different kinds of visualization prove to be effective when it comes to quickly getting an overview of some large set of information, when it comes to interactively drilling down into interesting subsets of information, when it comes to generating hypotheses about the information, or when it comes to interactively verifying or falsifying these hypotheses. Visualization is an interesting opportunity for the investigation of models/data, because through visualization the extremely efficieny human visual system is exploited to enable a broad-band interface between models/data and the human mind.

In the field visualization research, different subfields can be distinguished, including *volume visualization*, which usually is concerned with the visual representation and analysis of volumetric information such as CT-scans of patients or the like, *flow visualization*, which addresses the class of differential information (as in the focus of this review), and *information visualization*, which encompasses the (usually) interactive visual analysis of abstract information, e.g., databases, census data, etc. In the following, we concentrate on the field of flow visualization and try to review previous work in a structured way. More examples, more references, and more in-depth information about the various approaches can be found in a more extensive review published elsewhere [40].

3.1 Direct Flow Visualization

It is by far not obvious how the field of flow visualization would have to be substructured. The large set of available approaches and techniques can be classified according to a number of different dimensions, including classifications (a) according to the dimensionality of the flow domain (2D, 3D, or different), (b) according to the origin of the differential information (modeling, measurements, or simulation), or (c) according to the amount of computation involved in the visualization (direct vs. integration-based vs. computational methods). In this review we will utilize the last option (classification according to the “distance” between the original information to be visualized and the eventual visualization results; or, formulating a bit different, according to how much processing is required to transform the original data into the visualization).

We start with visualization techniques (here called *direct flow visualization*) which represent a fairly straight-forward mapping of differential information into graphical representations. This class of flow visualization techniques includes (relatively simple) approaches such as mapping vectorial information to a set of arrow glyphs (as also demonstrated in figure 2, lower image), or using color-coding to represent values of change in a dataset [26, 40].

In 2D, just a few interesting questions arise with these kinds of approaches. For color coding, for example, the question of an appropriate color map always is of great importance. In a good solution, for example, perceptual color differences are matched with differences in the original data – a simple, linear interpolation through the RGB color cube, as often seen in poorly designed visualization systems, does not provide this feature [45, 29, 44]! Also, and due to the fact that visualization designers often tend to maximize the information throughput in their visualization solutions, the question of how much information actually should be encoded within color is to be treated. It is possible to map one, two, or even more data dimensions to different aspects of color such as hue and lightness, for example. However, great care is demanded to not overload the visualization – usually it is not useful at all to code more than two dimensions in color [2, 3]!

When using arrow glyphs for the visual representation of differential information, at least two choices have to be made (apart from the choice of an appropriate glyph in terms of shape and color). First, the question of where to instantiate the arrow glyphs has to be answered. In some applications, it might seem appropriate to place an arrow glyph in each and every location where a flow vector is given (case of discrete data) – thereby interpolation is avoided and the data is represented as it is originally given. However, difficulties can arise if the spatial distribution of vectors is very uneven – in such a case it is likely that the resulting arrow glyphs overlap and thereby visual clutter is caused (cf. figure 2, lower image). Of course, it is usually possible to resample the vector field onto a regular grid, e.g., a Cartesian one, and to map the resampled vectors to arrow glyphs instead of the original ones – thereby usually less clutter is generated but details might be lost or artificially generated through the resampling process. A second option along with this approach is to scale the arrow glyphs according to the length of the flow vector, or not. Scaling glyphs will result in visualization which transports more information but which usually is difficult to adjust, especially when flow velocities vary drastically. Normalized arrows are easier to control (with respect to visual clutter) at the cost of communicating flow directions only.

In 3D, a number of further issues arise. Obviously, color coding converts into a significantly more advanced discipline, since in 3D the additional step of volume rendering is required to actually achieve visualization results [7, 11, etc.]. Unfortunately, up to now most of volume rendering research has been focussed on medical applications and the related specifics of volumetric data. Volume rendering of color coded flow data, however, proves to be difficult and often not fully satisfactory. In figure 3, on the left side, an example for a volume rendering of flow data is given [65]. An interesting alternative to volume rendering in 3D is the use of auxiliary 2D geometries, e.g., a stack of planar patches which are parallel to each other, and to color code these auxiliary geometries instead of the entire volume. This technique is especially useful when combined with so-called “scalar clipping” which means that for a certain subset of flow values, no color is drawn, but full transparency is assumed instead. Thereby, the issue of visual clutter can be controled in a reasonable way and still significant insights in the flow data are possible (see figure 3, right, for an example [50]).

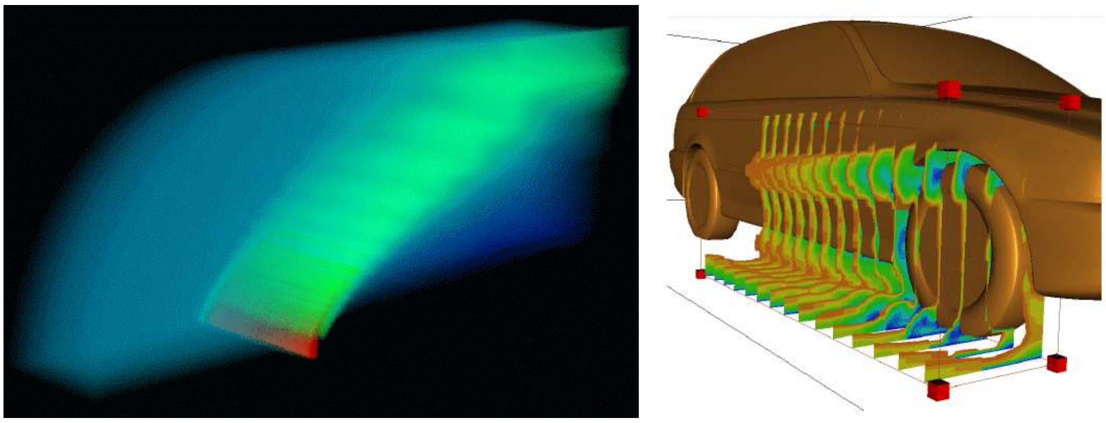
Figure 3: Volume rendering of color-coded flow data (left, image courtesy by R. Westermann [65]) and slice-based color coding of flow data with scalar clipping applied (right, image courtesy by M. Schulz [50]).

When using arrows in 3D, the issue of placement becomes even more important as compared to the 2D case. Usually, it is no good idea to densly fill the 3D domain with arrow glyphs, significant visual clutter is very probable and problems with the perception of the visualization arise. Amongst the more popular placement strategies are placements along auxiliary geometries, e.g., arrows which originate from a (relatively small) set of planar patches that are parallel to each other, or the use of clustering ahead to the visualization to derive a significantly smaller set of vectorial cluster representatives which then are mapped to arrow glyphs [56].

3.2 Texture-based Flow Visualization

In direct flow visualization, as briefly discussed above, a direct mapping of differential data to visualization (either color or glyphs) is employed. If the user is interested in the temporal evolution of the data or model, then he or she is required to mentally integrate the visual representation, e.g., to mentally follow arrows or a color gradient (usually difficult and not really satisfactory). In texture-based flow visualization, i.e., in the next class of visualization techniques to be discussed in this review, visualization attempts to directly encode the temporal evolution as induced by the differential information, at least in terms of relatively short evolution pathlets.

In texture-based flow visualization, texture maps (wich subsequently are used to color-code auxiliary geometries in the flow domain through texture mapping) are computed which fulfill the condition that the texture values, i.e., the color values in the texture map, are correlated along the flow whereas they exhibit high frequencies across the flow. Fulfilling this property, these kinds of texture maps convey the visual impression of a dense set of particles which all have been smeared out a bit into the flow direction. In figure 4, left image, an example is shown for a texture-based flow visualization on a slice through a cylinder of a gas engine in a passenger car [28] – it is clearly visible from the texture map in which way the flow goes as well as where the flow is fast and where not.

The two by now classic algorithms to compute such textures are line integral convolution (LIC [6, 53]) and spot noise [62], together with a large number of extensions which have been published lateron [26, 40]. The basic approach to compute a LIC texture is easily explained: Assume you have a scalar texture map $n(\mathbf{x}) : \mathcal{R}^n \rightarrow [0, 1]$ available – n encoding light intensities within the texture – which does not exhibit any correlations amongst its values with respect to whatsoever spatial directions, e.g., a white noise texture. Assume further that you also have numerical flow integration available such that you can compute a (usually short) stream-

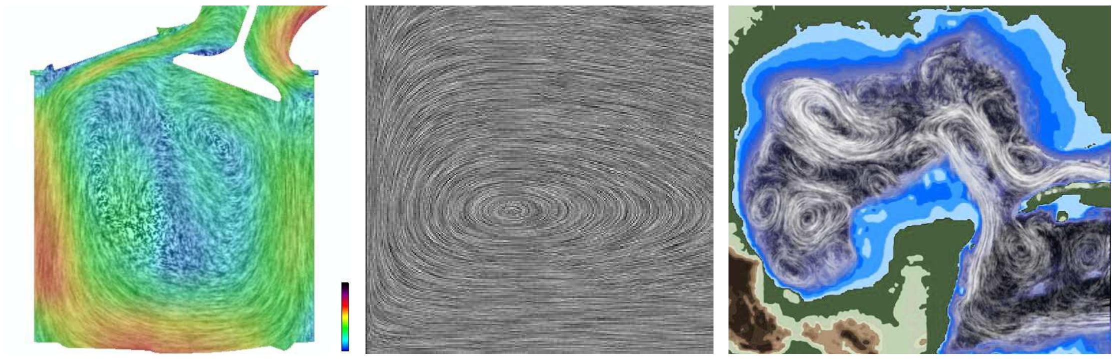

Figure 4: Example for a texture-based flow visualization (left image, on a slice through a cylinder of a gas engine [28]), an LIC texture for the Lotka-Volterra model (middle image [31], cf. equation 2), and a still image from Lagrangian-Eulerian advection (right image, image courtesy of Br. Jobard [22]).

line $\mathbf{p}(s)$ for each and every point $\mathbf{x}$ in your destination texture $lic(\mathbf{x}) : \mathcal{R}^n \rightarrow [0,1]$. For a particular location $\mathbf{x}$ in the LIC texture, $\mathbf{p}(0)$ may equal $\mathbf{x}$ and s may range from $-s_{max}$ to s_{max}. With any kind of smoothing filter $h(s) : [-s_{max}, s_{max}] \rightarrow [0,1]$, e.g., a truncated or windowed Gaussian or a simple box filter, the LIC texture $lic(\mathbf{x})$ can be computed by $lic(\mathbf{x}) = \int_{\tau=-s_{max}}^{s_{max}} n(\mathbf{p}(\tau))\, h(-\tau)\, \mathrm{d}\,\tau$.

The original algorithm for computing LIC [6] did not take flow velocities into account when integrating the short streamlets $\mathbf{p}(s)$ – accordingly, only flow directions were encoded in original LIC images (cf. figure 4, middle image). Newer algorithms such as Lagrangian-Eulerian texture advection [22] or image space based advection of textures [27] can incorporate a notion of flow velocities on demand. They are efficient variants of LIC and Spot Noise which also work for time-dependent data. See figure 4, right image, for a still image from an animation which has been produced with Lagrangian-Eulerian advection [22]. For a more thorough discussion for texture-based flow visualization, see our other review [26, 40].

3.3 Integration-based Flow Visualization

Even though texture-based flow visualization techniques usually are computationally quite expensive, they conceptually are still quite related to direct visualization techniques, only with the difference that a better indication for the flow evolution is given. In the next class of flow visualization techniques, we make another step away from direct flow visualization in terms of how much computation is necessary ahead to the visualization mapping. The visualization procedure now changes to first extracting intermediate geometries through the use of numerical flow integration and to visually represent these derived geometric structures instead of the original data. The left image in figure 1, for example, is composed of nine integral curves which have been computed by numerical integration of the Lotka-Volterra model (equation 2), starting from initial configurations $(i, 1)^T$ with $i \in \{1, 2, \ldots, 9\}$. The integral curves have been represented as polylines in this simple visualization.

In general, there are several more opportunities of how to exploit flow integration for indirect, integration-based flow visualization. First, and especially in the context of time-dependent data, i.e., in the cases of $\mathbf{v}(\mathbf{p}, t)$ or $\mathbf{v}[\mathbf{p}_i, t_j]$, there are different options of which curves actually to compute [57, etc.]: streamlines $\mathbf{m}$, pathlines $\mathbf{p}$, streaklines $\mathbf{k}$, or timelines $\mathbf{t}$. To properly relate them to each other we will use the following more general definition of an integral curve $\mathbf{c}$, given for time s, associated with a seed time t_0 and a seed location $\mathbf{p}_0$, and parameterized in

$t \in [0, t_{max}]$):

$$\mathbf{c}(\mathbf{p}_0, t_0, s, t) = \mathbf{p}_0 + \int_{\tau=0}^{t} \mathbf{v}(\mathbf{c}(\mathbf{p}_0, t_0, s, \tau),\ u(t_0, s, \tau))\ \mathrm{d}\,\tau \tag{4}$$

Integration time t parameterizes curve $\mathbf{c}$, starting from $t = 0$. Function $u(t_0, s, \tau)$ determines the time step from which the flow vector $\mathbf{v}$ is taken during integration in curve points $\mathbf{c}(\mathbf{p}_0, t_0, s, \tau)$.

- **Pathlines** $\mathbf{p}(\mathbf{p}_0, t_0, s, t)$ for time step s, seeded in location $\mathbf{p}_0$ at seed time t_0, and parameterized in $t \in [0, s - t_0]$, are a very intuitive visualization methodology for the investigation of unsteady flow fields. They are defined as the path of massless particles whose movements are solely governed by the differential information under investigation. Pathlines $\mathbf{p}(\mathbf{p}_0, t_0, s, t)$ equal the definition of $\mathbf{c}(\mathbf{p}_0, t_0, s, t)$ from equation 4 with $u(t_0, s, \tau) = t_0 + \tau$.
- **Streaklines** $\mathbf{k}(\mathbf{p}_0, t_0, s, t)$ for time step s, seeded from $\mathbf{p}_0$ since time t_0, and parameterized in $t \in [0, s - t_0]$, are also very popular for the investigation of unsteady flow data, especially due to their origin in traditional experimental flow visualization, where a continuous injection of particles into a moving fluid generates such a streakline. Accordingly, and making use of equation 4 again, a streakline $\mathbf{k}(\mathbf{p}_0, t_0, s, t)$ can be written as $\mathbf{c}(\mathbf{p}_0, s{-}t, s, t)$ with $u(t_0, s, \tau) = s - t + \tau$. For $t = 0$ streakline $\mathbf{k}$ equals $\mathbf{p}_0 = \mathbf{c}(\mathbf{p}_0, s, s, 0)$, for $t = s - t_0$ streakline $\mathbf{k}$ equals $\mathbf{c}(\mathbf{p}_0, t_0, s, t)$, i.e., a pathline $\mathbf{p}$ of temporal length $t = s - t_0$, seeded at time t_0 in location $\mathbf{p}_0$.
- **Timelines** $\mathbf{t}(t_0, s, t)$ for time s, seeded at time t_0, and parameterized in t, are yet another option of how to investigate steady as well as unsteady flow data. Timelines are defined to be a parameterized set of points which have been introduced into the flow at time t_0 and then advected by the flow for $s - t_0$ time. Accordingly, a timeline $\mathbf{t}$ at time s can be written as $\mathbf{c}(\mathbf{p}_0(t), t_0, s, s)$ with $u(t_0, s, \tau) = t_0 + \tau$.
- **Streamlines** $\mathbf{m}(\mathbf{p}_0, s, t)$ for time step s, starting in seed location $\mathbf{p}_0$ and parameterized in $t \in \mathcal{R}^+$, are a classic methodology for the investigation of steady vector fields $\mathbf{v}$ with $\mathbf{v}(\mathbf{p}, t_1) = \mathbf{v}(\mathbf{p}, t_2)$ for any two t_1 and t_2. They are defined as curves which are tangential to the flow vectors $\mathbf{v}(\mathbf{m}(\mathbf{p}_0, s, t))$ in each and every point of the curve. In unsteady flow fields, streamlines do not take the changes of the vector field over time into account, i.e., they can be writte as $\mathbf{m}(\mathbf{p}_0, s, t) = \mathbf{c}(\mathbf{p}_0, s, s, t)$ from equation 4 with $u(t_0, s, \tau)$ being s (or anything else, e.g., t_0 or $t_0 + \tau$ as above for pathlines). Streamlines illustrate the flow characteristics of selected time step only.

In the case of steady flow fields, pathlines, streaklines, and streamlines coincide according to $\mathbf{p}(\mathbf{p}_0, t_0, s, t) = \mathbf{k}(\mathbf{p}_0, t_0, s, t) = \mathbf{m}(\mathbf{p}_0, s, t)$. Table 1 summarizes the differences between pathlines, streaklines, timelines, and streamlines in terms of equation 4.

A second choice with respect to integration-based flow visualization – especially in the context of 3D flow data – is from which kind of seed object the integration is started from. Most

pathline $\mathbf{p}(\mathbf{p}_0, t_0, s, t)$:	$\mathbf{c}(\mathbf{p}_0,$	$t_0,$	$s,$	$t)$	&	$u(t_0, s, \tau) = t_0 + \tau$
streakline $\mathbf{k}(\mathbf{p}_0, t_0, s, t)$:	$\mathbf{c}(\mathbf{p}_0,$	$s - t,$	$s,$	$t)$	&	$u(t_0, s, \tau) = s - t + \tau$
timeline $\mathbf{t}(t_0, s, t)$:	$\mathbf{c}(\mathbf{p}_0(t),$	$t_0,$	$s,$	$s)$	&	$u(t_0, s, \tau) = t_0 + \tau$
streamline $\mathbf{m}(\mathbf{p}_0, s, t)$:	$\mathbf{c}(\mathbf{p}_0,$	$s,$	$s,$	$t)$	&	$u(t_0, s, \tau) = s$ (or $\ldots$)

Table 1: Summary of how pathlines, streaklines, timelines, and streamlines can be compared to each other in terms of equation 4.

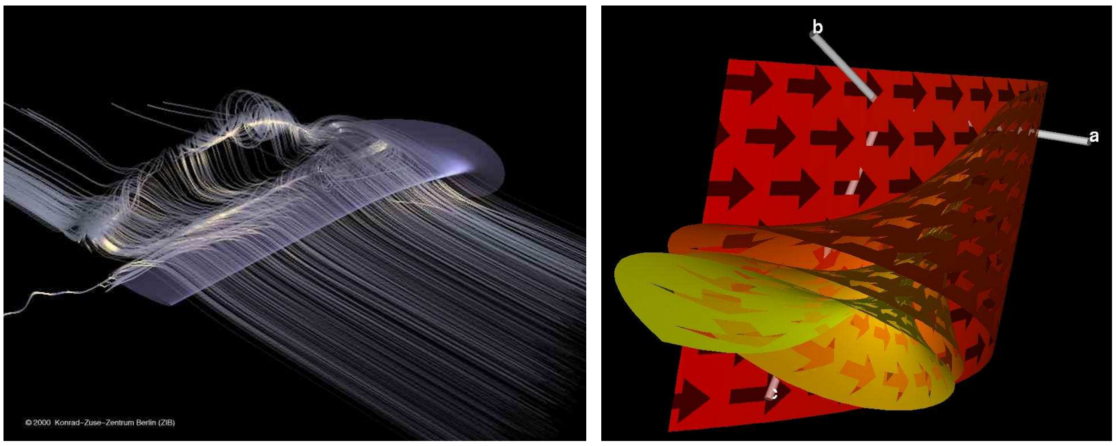

Figure 5: Illuminated streamlines illustrate the flow across a wing of an airplane (left image, image courtesy by Zöckler et al. from the ZIB in Berlin, Germany [68]); a streamsurface with streamarrows is used to selectively depict the flow through a dynamical system (right image [35]).

classic, and conforming to the above discussion, differnt kinds of integral curves (pathlines, etc.) are commonly used. Sometimes, short variants of these integral curves are used, i.e., pathlets, streamlets, aso., especially when more of them are to be integrated within a visualization [33]. If curves of this kind are used for flow visualization, then their representation in the visualization is an important question. In general, special perceptual challenges can be identified when 1D structures are used in 3D visualization (depth perception is significantly challenged for 1D objects in 3D space). One opportunity in such a situation is to employ special shading for the line structures and to generate illuminated streamlines [68, 54]. Figure 5, left image, shows an example of how illuminated streamlines can illustrate the flow across a wing of an airplane [54]. Another option is to represent the 1D curves by 2D geometric objects, e.g., in the form of a streamribbon or streampolygon [61].

In addition to integral curves as integrated from seed points, it is also possible to use integral surfaces – so called streamsurfaces in steady flow fields – which are integrated from seed curves instead. Assume that the seed curve $\mathbf{p}_0(t)$, parameterized in t, is a dense and continuous set of seed points in 3D. Then – conceptually – every single seed point along this curve is integrated over time through the steady flow field: $\mathbf{p}(s,t) = \mathbf{p}_0(t) + \int_{\tau=0}^{s} \mathbf{v}(\mathbf{p}(\tau,t))\,\mathrm{d}\tau$. The resulting two-parameter set $\mathbf{p}(s,t)$ is then a streamsurface [19, 20]. Even though introduced in the early 1990s, streamsurfaces have become more popular only recently. This most probably is due to the computational complexity of their calculation [49, 12]. Streamsurfaces are attractive for flow visualization in 3D because they are conveniently perceived. However, streamsurfaces can also cause visual clutter if too much of a seed curve is used, if integration is continued for too long, or if too much divergence is present in the flow. Options to deal with these challenges are to either reduce the geometry load from the visualization, e.g., by cutting out so-called streamarrows [35], or by modulating the transparency of the stream surface [12]. Figure 5, right image, shows a streamsurface with streamarrows through a 3D dynamical system (a 3D model of chemical reactions [35]).

Of course, it is also possible to take a two-parameter patch as seed set for flow integration: $\mathbf{p}(s,t,u) = \mathbf{p}_0(t,u) + \int_{\tau=0}^{s} \mathbf{v}(\mathbf{p}(\tau,t,u))\,\mathrm{d}\tau$. The three-parameter set $\mathbf{p}(s,t,u)$ is then called a flow volume and volume rendering is to be employed for visualization [37].

As last aspect of integration-based flow visualization, which is discussed in this review, we address the issue of seeding. In the case of integration-based flow visualization, always a selective visualization is generated – a comparably small set of selected integral objects is used for visualization. The more selective this visualization is, e.g., just one streamsurface is used for visualization, the more critical the question of seeding becomes – where to actually start the integral object. Several solutions have been published for this challenge, e.g., approaches to generate evenly-spaced streamlines [23, 60], or such which enable interactive placement of seed locations [4, 50].

3.4 Feature-based Flow Visualization

Up to now, we have discussed direct, texture-based, and integration-based visualization of differential information. Still remaining in this review is the very large class of visualization techniques which build on a significant amount of processing ahead to the actual visualization mapping. Due to the large number of already published approaches, only a rough outline is provided here and the reader is refered to a more extensive overview which recently has been published elsewhere [41, 40].

Obviously, there are many possible ways of preprocessing the data before visualization is applied. In this review, we will only focus on a subset of relatively common approaches.

Topology-based Flow Visualization

One important approach to analyzing differential information is to first do some more mathematical analysis on the data before visualization is used. Analysis is done to identify subsets of the flow domain which exhibit behavior that is similar in terms of long-term evolution, e.g., all stream lines within such a region converge to a joint attractor. It is noted that most of the previous work on topology-based flow visualization has focused on the case of steady data. Only recently, approaches to the topology-based visualization of unsteady flows are being developed [59].

To extract the topology of a flow field or a dynamical system, first the critical points are identified, i.e., all configurations $\mathbf{p}^*$ of the flow domain in which there is no flow, i.e., $\mathbf{v}(\mathbf{p}^*) = 0$ [17, 18]. Next, the flow is investigated in the local surrounding of the critical points. This is done through the approach of local linearization. Utilizing the Taylor expansion of $\mathbf{v}$ around $\mathbf{p}^*$, i.e.,

$$\mathbf{v}(\mathbf{p}^*+\Delta\mathbf{p}) \;=\; \textstyle\sum_{i\geq 0} \, (\Delta\mathbf{p}\cdot\nabla)^i \, \mathbf{v}\,|_{\mathbf{p}^*} \;=\; \mathbf{v}(\mathbf{p}^*) + \Delta\mathbf{p}\cdot\nabla\mathbf{v}\,|_{\mathbf{p}^*} + O(\Delta\mathbf{p})^2\,,$$

neglecting the terms of second as well as higher order ($O(\Delta\mathbf{p})^2$), and substituting $\mathbf{v}(\mathbf{p}^*) = 0$ due to the main characteristic of critical points, we identify the Jacobian matrix $\nabla\mathbf{v}\,|_{\mathbf{p}^*}$ of flow $\mathbf{v}$ in critical point $\mathbf{p}^*$ as the locally dominating flow component. Matrix $\nabla\mathbf{v}$ induces linear flow behavior around (non-degenerated) critical points $\mathbf{p}^*$ and its characteristics are described by the eigenvalues and eigenvectors of this matrix. Negative/positive eigenvalues correspond to attracting/repelling behavior along directions which are given by the associated eigenvectors. Eigenvalue pairs with a non-zero imaginary component correspond to flow which rotates around the critical point. Accordingly, critical points are classified into the non-degenerated cases of nodes (either attractors or repellors), foci (rotating behavior), and saddles. Once, the critical points are identified and classified, it is often interesting to investigate characteristic streamlines as emanating from $\mathbf{p}^*$ into the direction of the eigenvectors of $\nabla\mathbf{v}\,|_{\mathbf{p}^*}$. Thereby, the relation between critical points is investigated and so-called separatrices are computed (flow exhibits significantly different long-term behavior on either side of a separatrix, e.g., converging to different attractors). There are many examples in which the topological skeleton of a flow field was utilized for visualization [18, 32]. In figure 6, left image, an example for a topology-based visualization of a 3D dynamical system is shown.

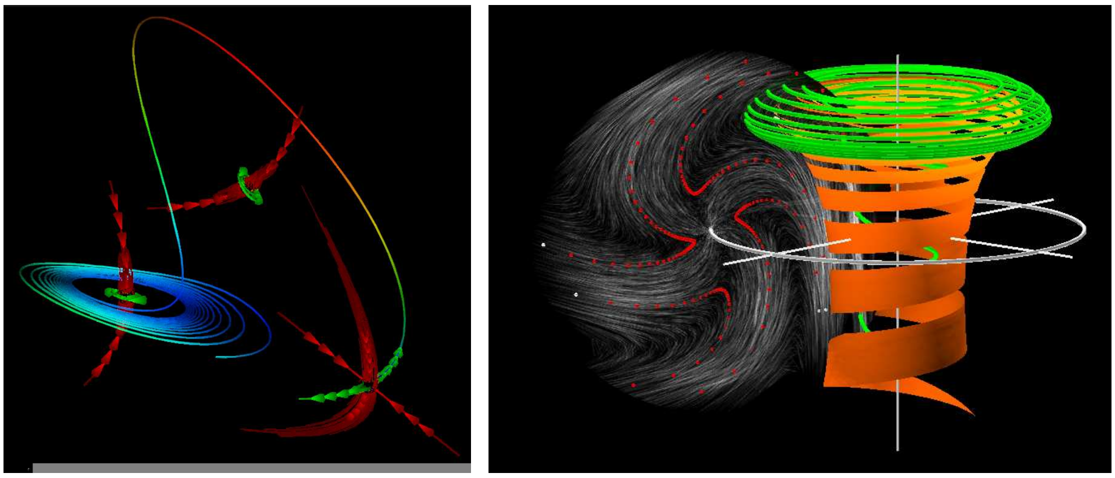
Figure 6: Topology-based visualization of a 3D dynamical system (left side [32]). Visualization of a periodic dynamical system in 3D based on a Poincaré map (right side [34])

In addition to critical points, also critical structures of higher order can be investigated. Most important next to critical points are closed trajectories, i.e., cycles. They are defined as all points $\mathbf{p}^{**}$ for which $\mathbf{c}(\mathbf{p}^{**}, s-T, s, T) = \mathbf{p}^{**}$ and $T > 0$ holds (using equation 4 again with $u(t_0, s, t) = s$ for steady flows). Typically, cycles are more difficult to identify, but useful algorithms have been proposed recently [67, 66, 58]. Once detected, also closed streamlines can be classified similar to critical points. In general, a k-dimensional critical structure in an n-dimensional flow ($k < n$) can always exhibit attracting/repelling behavior as associated with selected directions. Saddle behavior as well as rotating behavior can only be found for critical structures of dimension $k \leq n-2$. An non-degenerated invariant torus, for example, in a 3D vector field, only has codimension $n-k = 1$ and thus may only be either attracting or repelling.

Cycles are not only more difficult to extract but also to visualize in an abstracted form (as compared to the topology-based visualization based on critical points). One interesting approach is to utilize Poincaré maps for this purpose. Poincaré maps are an $(n–1)$-dimensional abstraction of nD flow near closed streamlines. A Poincaré map $\mathbf{P}$ is related to an $(n-1)$-dimensional Poincaré section Π in such a way that any $\mathbf{p} \in \Pi$ is related to a $\mathbf{P}(\mathbf{p}) \in \Pi$ if it is possible to find a $\mathbf{P}(\mathbf{p}) = \mathbf{c}(\mathbf{p}, s-T, s, T)$ for a minimal $T > 0$ such that $\mathbf{P}(\mathbf{p}) \in \Pi$. According to the definition of closed streamlines (see above), points $\mathbf{p}^{**} \in \Pi$ are related to themselves through $\mathbf{P}$: $\mathbf{P}(\mathbf{p}^{**}) = \mathbf{p}^{**}$ (points $\mathbf{p}^{**} \in \Pi$ are critical points of map $\mathbf{P}$). This abstraction of flow nearby closed streamlines can be used for visualization [34]. Figure 6, right image, shows an example of a Poincaré map based visualization of a periodic dynamical system in 3D.

Feature Extraction for Flow Visualization

In addition to topology-based flow visualization, also flow visualization which builds on the extraction of other kinds of flow features such as vortices (see below), shock waves [36, etc.], attachment/detachment points [24, etc.], or recirculation zones [14, etc.] are very popular. Especially in the domain of vortex extraction, a lot of research has been done [40, 41].

Different approaches have been proposed to detect those subsets of a flow which are characterized by a signififcant amount of local rotation. One subset of algorithms identifies vortical regions, e.g., by searching for regions of high amounts of flow vorticity $\nabla \times \mathbf{v}$ [63], or by investigating helicity $(\nabla \times \mathbf{v}) \cdot \mathbf{v}$, i.e., vorticity projected onto the flow [30]. Other algorithms focus on the identification of vortex core lines (instead of vortical regions). One approach is to follow

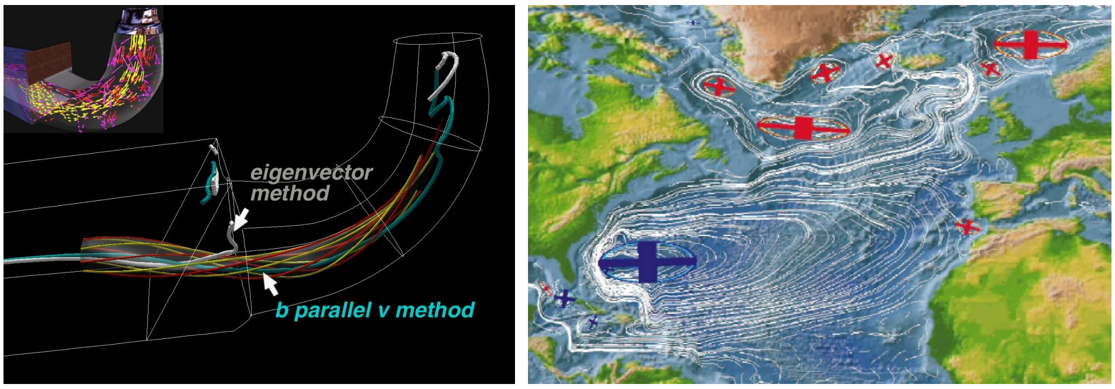

Figure 7: The parallel vector technique [38] applied to flow through a draft tube in turbomachinery design (left image, image courtesy by M. Roth et al., ETH Zürich [46]); Sample result from a geometric vortex extraction technique (right image, image courtesy by I. Sadarjoen et al., Delft University [48]).

streamlines from critical points [13]. Another is to consider streamlines of the vorticity field [1], also taking flow pressure into account (flow vortices are characterized by locally low pressure). A very popular algorithm is to search for locations in which the vorticity field is parallel to the velocity field [47, 38, 46]. In figure 7, left side, the application of the parallel vector technique is illustrated in the context of flow through a draft tube in turbomachinery design [38, 46]. Another very popular algorithm is to search for locations in the flow where the gradient of the flow ($\nabla \mathbf{v}$) exhibits two eigenvalues with non-zero imaginary components, i.e., rotating flow behavior, and where the eigenvector which is associated with the real-values eigenvalue is aligned with the flow [55, 25].

In addition to algorithms which are based on the analysis of local flow characteristics, also algorithms which utilize information from a larger neighborhood are very interesting. One algorithm utilizes combinatorial topology (Sperma's theorem) for the detection of vortex core lines [21]. Another paper introduces two geometric techniques for the detection of rotating flow substructures [48]. A sample result from their application is shown in figure 7, right side, indicating vortices in ocean flow in the northern Atlantic.

Interactive Feature-based Flow Visualization

Alternatively to approaches which aim at the as automatic as possible extraction and visualization of flow features, the approach of interactive feature extraction and visualization (as recently developed at the VRVis Research Center, www.SimVis.at) is worth mentioning.

In this approach, the goal is (a) to put forth the rich variety of contents in a large and multi-dimensional flow simulation dataset by visualizing selected subsets of the data in an intuitive way, (b) to provide a flexible set of interaction mechanisms which enables the user to conveniently formulate his or her interests in the data in the form of implict feature characteristics (example: slow but nevertheless hot flow which also is near to the boundary surface of a cooling jacket), and (c) to concurrently provide a 3D focus+context visualization of the simulation data which relates the focus–context discrimination to the current user interest. To realize this goal, the following technological components have been developed and integrated (more information is provided by a related paper [16]):

- Selected visualization views such as histograms, scatterplots, and parallel coordinates, are used to put whatsoever data attributes in a visual relation to each other. In figure 8, for example, a CFD dataset is visualized by the means of a histogram ((d), velocity values along x), a 2D scatterplot ((b), velocities on x, turbulence values on y), a 3D scatterplot

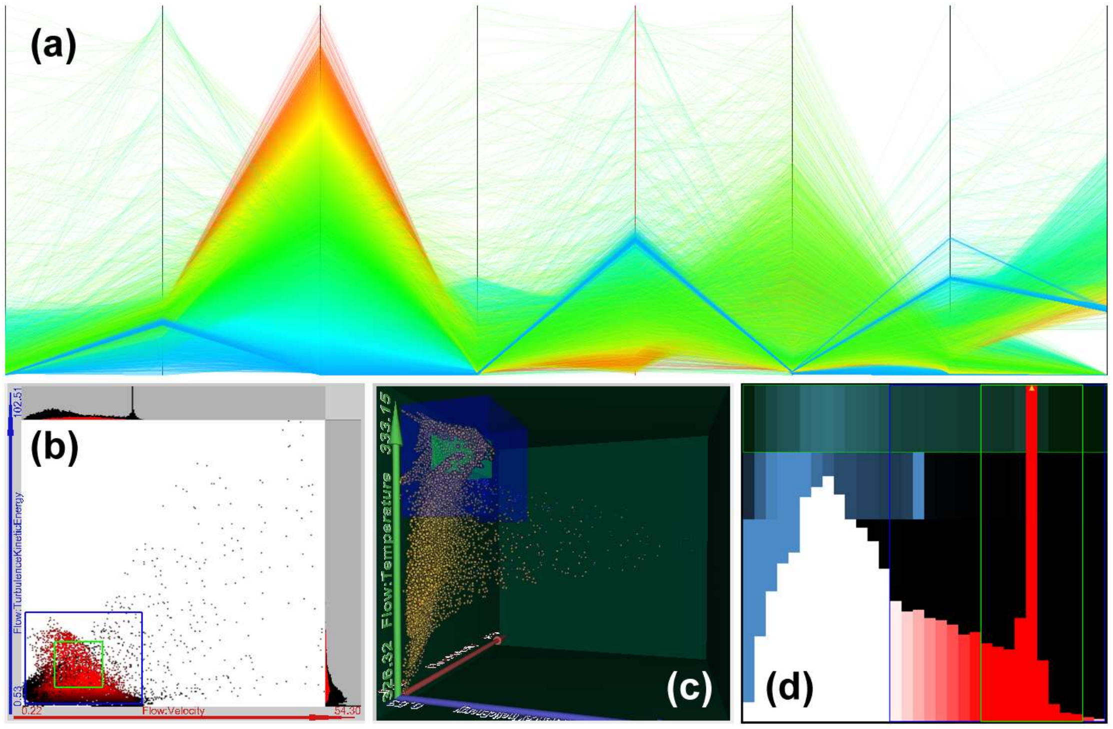

Figure 8: A CFD dataset is visualized by the means of a histogram of velocity values (d), a scatterplot "velocities vs. turbolence values" (b), a 3D scatterplot "velocities, turbolence, temperatures" (c), and parallel coordinates (a).

((c), velocities, turbulence and temperature values), and a plot of parallel coordinates ((a), eight different flow attributes shown). In the two scatterplots and in the histogram brushes are used to mark subsets of special interest. In figure 8(a), one axis (showing velocity values) is used to color the polylines – this eases to relate data items across multiple dimensions.

- An approach called *linking & brushing* in visualization [5, etc.] is used in SimVis to allow for interactive visual analysis of high-dimensional and complex datasets [8]. Multiple views are used to all jointly display one dataset under investigation. Brushing enables the user to interactively mark certain subsets of the data as being of special interest – the user directly brushes the displayed data in one of the views or uses off-screen sliders and direct numerical input widgets to tell the system what he or she (currently) focuses on. Linking between views ensures that the entire visualization is visually consistent, i.e., in all the views the same data subsets are consistently visually enhanced (or deemphasized), for example, by everywhere coloring the data in focus red.

  In figure 9, three scatterplots and three histograms have been used to specify the current focus in the visualization of a simulation of hurricane Isabel (cloud structures and land). The scatterplots (ca), (cv), and (av), and the histogram (h) are used to highlight all those parts of the clouds in the hurricane which exhibit a significant amount of wind speed (cv), which do not raise up too high into the atmosphere (h), and which either move noth or south (two 3D brushes in ca and av). The distinction between north and south winds enables to detect an interesting flow behaviour in the north of the hurricane. For the purpose of additional orientation, a (dimmed) visualization of the east cost of America has been added through (partly) selecting the respective cells in two histograms (h' & h"). The 3D view (3D) provides a spatial and also time-dependent 3D visualization (color visualizes wind directions, green: north, purple: south).

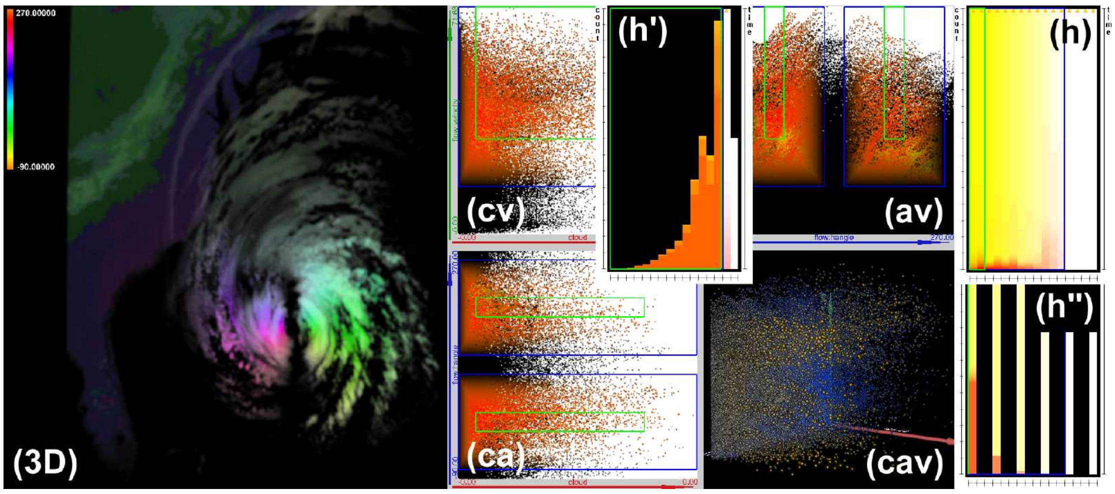

Figure 9: The simulated hurricane Isabel is visualized. Through the use of three scatterplots and a histogram, relatively fast clouds are emphasized which do not raise too high up into the atmosphere – only winds which are (at least to some extent) aligned north-south are highlighted to reveal an interesting flow structure in the north of the hurricane. Additionally, the land is shown for geographical reference.

- The interactive process of semantically annotating the data (through brushing) to differentiate between more and less interesting parts of the data consequently causes the views to show a (so called) focus+context visualization of the data [15, & refs. therein]. Data subsets in focus are visually enhanced as compared to their context, i.e., the rest of the data, which is just shown in reduced form, e.g., less opaque and not colored, to provide a visual reference for the current focus and to ease orientation and navigation. In SimVis, the approach of focus+context visualization also has been extended to the more traditional views, i.e., the 3D rendering of the data. Data in focus is visually discriminated from the context by selective coloring as well as by variations of opacity and style.

  Figure 10, left side, demonstrates how focussing allows to reveal interesting structures in a 3D dataset. In (F+C) the interface between water and air has been selected in a two-phase simulation of the flood after the breaking of a dam and its effect on an obstacle in the flow (color: flow velocities). In (C) no feature has been selected whereas in (F) all the cells are shown in the same style – whereas in (C) too little is shown, in (F) too much is visualized.

  In figure 10, right side, all those cells in a diesel particulate filter (DPF) for passenger cars have been highlighted which are characterized by the presence of lots of CO and CO_2 and which also exhibit relatively high temperatures (color shows flow velocities) – thereby the front of an oxidation process is shown in the DPF which is used for the periodic regeneration of the filter [10].

  In figure 11, all those cells in a $45°$ subset of the combustion chamber (in one cylinder of a diesel engine) are highlighted (at four different time steps of the simulation) where the combustion already is well progressed while still a significant amount of diesel is left (an undesired situation) – color visualizes the amount of O_2 in the cells. The respective analysis [9] affirms that combustion stops too early because too little oxygen is collocated with the fuel mixture to be burnt in certain locations.

- In addition to the above mentioned points (visualizing attribute space, establishing a linking&brushing framework for interactive data analysis, and using focus+context visualization to incorporate a notion of user interest), the importance of interaction should be

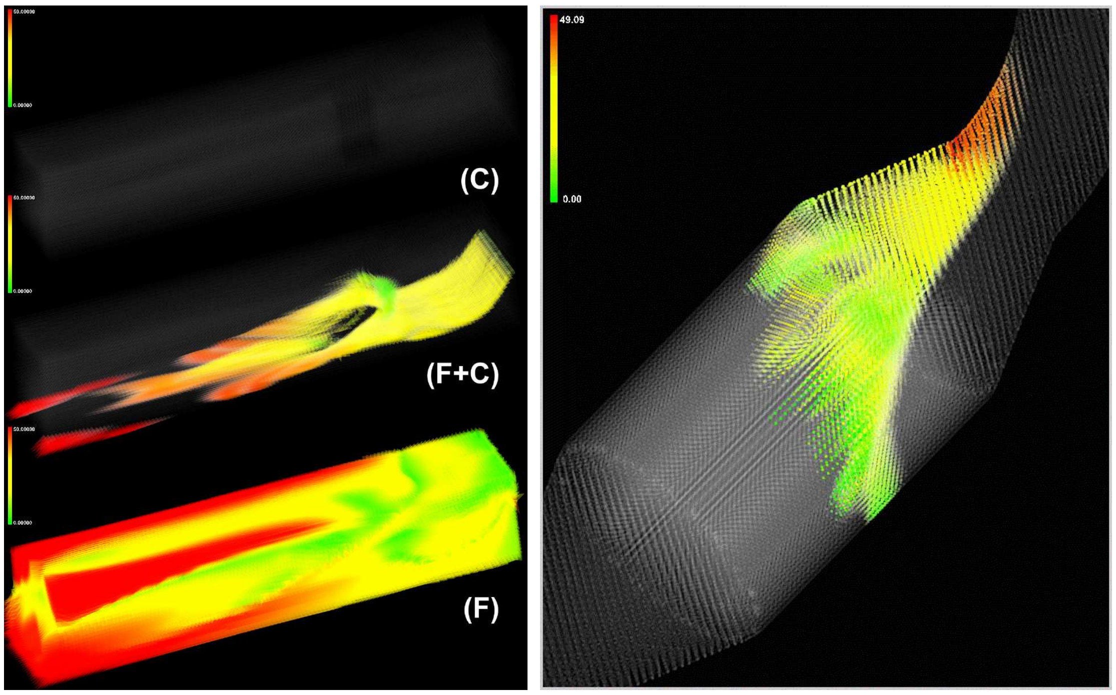

Figure 10: Example for the visual discrimination of data subsets in focus from the respective context (left side). The flood as resulting from a breaking dam is shown together with its effect on an obstacle. Only through the (interactive) focussing on specific subsets of the data (in this example the interface between water and air in "F+C"), meaningful images can be produced. — The front of an oxidation process in a diesel particulate filter (DPF) is shown (right side [10]), which is used for the periodic regeneration of the filter – all those cells are highlighted which exhibit large amounts of CO and CO_2 while at the same time being relatively hot. Color shows flow velocity.

> emphasized. Ben Shneiderman's visual information seeking mantra [51] well describes the main character of working with SimVis: First, some sort of overview visualization is needed – the user wants to get oriented, understand the main features of the data, etc. Then, as soon as specific questions about the data arise, the user wants to perform an information drill-down which allows to gradually refine a feature specification, before at a certain point of the detailed analysis the user might want to export the results of the analysis, for example, to compare a result with another analysis of another dataset.

More information about SimVis and related technology can be acquired from the SimVis home page (www.SimVis.at) or from related papers [8, 9, 10, 16, etc.]

4 Summary and Conclusions

In this paper, we have reviewed a subset of visualization approaches to the challenge of how to enable visual exploration, analysis, and presentation of differential information. An overview has been given of several useful directions, including direct flow visualization (section 3.1), texture-based flow visualization (section 3.2), integration-based flow visualization (section 3.3), and feature-based flow visualization (section 3.4). It is worth mentioning, however, that the current state of the art still is much richer than outlined in this review – more approaches and more references to related literature can be found in other, more extensive reviews [40, etc.].

Concluding, it can be stated that flow visualization has been researched extensively in the past decades. There are solutions for a spectrum of application questions. The large variety of

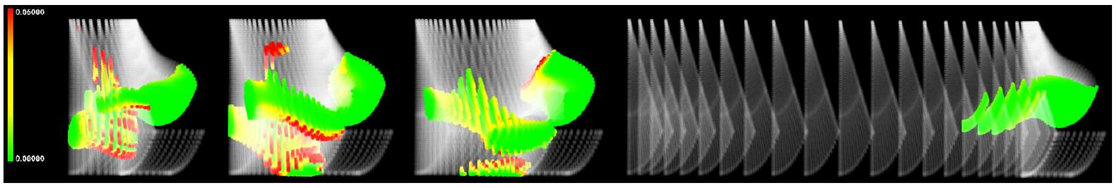

Figure 11: Four time steps of a combustion process are shown [9], focussing on cells where the combustion is well progressed but where still a significant amount of diesel is left (undesired) – color shows the amount of oxygen at this cells (in cells with too little oxygen the combustion terminates too early).

reasonable solutions results from the fact that flow visualization solutions always are dependent on the user task at hand. Different techniques are to be used for the purpose of flow exploration vs. flow presentation, for example. Also, different solutions apply dependent on whether 2D or 3D flow is to be shown. Finally, even in cases where one flow visualization methodology applies to several different application cases, differences might become evident with respect to the implementation of the respective techniques. A flow visualization algorithm for analytically specified dynamical systems, for example, can be significantly different to the same algorithm when applied to simulation data, given on an unstructured grid. This altogether might be responsible for the fact that (by now) there isn't any all-in-one premade visualization system which can be used for whatsoever application. However, still many solutions exist already ready to use.

Acknowledgements

Parts of this work have been done in the scope of strategic research at the VRVis Research Center in Vienna, Austria, http://www.VRVis.at/, which is funded by an Austria governmental research program called K*plus*. The author also thanks his colleagues which participated in related research, especially Robert S. Laramee and Helmut Doleisch.

References

[1] D. C. Banks and B. A. Singer. A predictor-corrector technique for visualizing unsteady flow. *IEEE Transactions on Visualization and Computer Graphics*, 1(2):151–163, 1995.

[2] C. A. Brewer. Color use guidelines for mapping and visualization. In MacEachren and Taylor, editors, *Visualization in Modern Cartography*, pages 123–147. Elsevier Science, 1994.

[3] C. A. Brewer and M. Harrower. ColorBrewer. On-line color brewing tool available from URL http://www.colorbrewer.org/, 2002.

[4] St. Bryson and C. Levit. The virtual wind tunnel. *IEEE Computer Graphics and Applications*, 12(4):25–34, July 1992.

[5] A. Buja, J. McDonald, J. Michalak, and W. Stuetzle. Interactive data visualization using focusing and linking. In *Proc. of IEEE Visualization '91*, pages 156–163, 1991.

[6] B. Cabral and L. C. Leedom. Imaging vector fields using line integral convolution. In *Proc. of ACM SIGGRAPH '93*, pages 263–272, 1993.

[7] R. Crawfis, N. Max, B. Becker, and B. Cabral. Volume rendering of 3D scalar and vector fields at LLNL. In *Proc. of Supercomputing '93*, pages 570–576.

[8] H. Doleisch, M. Gasser, and H. Hauser. Interactive feature specification for focus+context visualization of complex simulation data. In *5th Joint IEEE TCVG - EG Symp. on Visualization (VisSym 2003)*, pages 239–248, 2003.

[9] H. Doleisch, M. Mayer, M. Gasser, P. Priesching, and H. Hauser. Interactive feature specification for simulation data on time-varying grids. In *Proc. Simulation and Visualization Conf. (SimVis '05)*, pages 291–304, 2005.

[10] H. Doleisch, M. Mayer, M. Gasser, R. Wanker, and H. Hauser. Case study: Visual analysis of complex, time-dependent simulation results of a diesel exhaust system. In *6th Joint IEEE TCVG - EG Symp. on Visualization (VisSym 2004)*, pages 91–96.

[11] D. S. Ebert, R. Yagel, J. Scott, and Y. Kurzion. Volume rendering methods for computational fluid dynamics visualization. In *Proc. of IEEE Visualization '94*, pages 232–239, 1994.

[12] Chr. Garth, X. Tricoche, T. Salzbrunn, T. Bobach, and G. Scheuermann. Surface techniques for vortex visualization. In *Proc. Joint EG – IEEE TCVG Symp. on Vis.*, pages 155–164.

[13] A. Globus, C. Levit, and T. Lasinski. A tool for visualizing the topology of three-dimensional vector fields. In *Proc. of IEEE Visualization '91*, pages 33–40, 1991.

[14] R. Haimes. Using residence time for the extraction of recirculation regions. Technical Report AIAA Paper 99–3291, American Institute of Aeronautics and Astronautics, 1999.

[15] H. Hauser. Generalizing focus+context visualization. In G.-P. Bonneau, Th. Ertl, and Gr. Nielson, editors, *Scientific Visualization: The Visual Extraction of Knowledge from Data*, pages 305–327. Springer, 2005.

[16] H. Hauser and H. Doleisch. About SimVis and the related state of the art. Technical Report TR-VRVis-2004-028, VRVis Research Center, 2004.

[17] J. L. Helman and L. Hesselink. Representation and display of vector field topology in fluid flow data sets. *IEEE Computer*, 22(8):27–36, August 1989.

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

[19] J. P. M. Hultquist. Interactive numerical flow visualization using stream surfaces. *Computing Systems in Engineering*, 1(2-4):349–353, 1990.

[20] J. P. M. Hultquist. Constructing stream surfaces in steady 3D vector fields. In *Proc. of IEEE Visualization '92*, pages 171–178, 1992.

[21] M. Jiang, R. Machiraju, and D. Thompson. A novel approach to vortex core region detection. In *Data visualisation 2002*, pages 217–225. Eurographics Association, 2002.

[22] Br. Jobard, G. Erlebacher, and Y. Hussaini. Lagrangian-Eulerian advection of noise and dye textures for unsteady flow visualization. *IEEE Trans. on Visualization and Computer Graphics*, 8(3):211–222, 2002.

[23] Br. Jobard and W. Lefer. Creating evenly-spaced streamlines of arbitrary density. In *Proc. of the EG Workshop on Visualization in Sci. Computing '97*. Springer, 1997.

[24] D. N. Kenwright. Automatic detection of open and closed separation and attachment lines. In *Proc. of IEEE Visualization '98*, pages 151–158, 1998.

[25] D. N. Kenwright and R. Haimes. Automatic vortex core detection. *IEEE Computer Graphics and Applications*, 18(4):70–74, 1998.

[26] R. S. Laramee, H. Hauser, H. Doleisch, Fr. H. Post, B. Vrolijk, and D. Weiskopf. The state of the art in flow visualization: Dense and texture-based techniques. *Computer Graphics Forum*, 23(2):203–221.

[27] R. S. Laramee, Br. Jobard, and H. Hauser. Image space based visualization of unsteady flow on surfaces. In *Proc. of IEEE Visualization 2003*, pages 131–138.

[28] R. S. Laramee, D. Weiskopf, J. Schneider, and H. Hauser. Investigating swirl and tumble flow with a comparison of visuaization techniques. In *Proc. of IEEE Visualization 2004*, pages 51–58.

[29] H. Lefkowitz and G. T. Herman. Color scales for image data. *IEEE Computer Graphics & Applications*, 12(1):72–80, January 1992.

[30] Y. Levy, D. Degani, and A. Seginer. Graphical visualization of vortical flows by means of helicity. *AIAA Journal*, 28:1347–1352, 1990.

[31] H. Löffelmann. *Visualizing Local Properties and Characteristic Structures of Dynamical Systems*. PhD thesis, Vienna Univ. of Techn., 1998. www.VRVis.at/vis/resources/diss-HL.

[32] H. Löffelmann, H. Doleisch, and E. Gröller. Visualizing dynamical systems near critical points. In *14th Spring Conference on Computer Graphics*, pages 175–184, April 1998.

[33] H. Löffelmann and E. Gröller. Enhancing the visualization of characteristic structures in dynamical systems. In *Proc. EG Workshop on Vis. in Sci. Computing*, pages 35–46, 1998.

[34] H. Löffelmann, Th. Kučera, and E. Gröller. Visualizing Poincaré maps together with the underlying flow. In *Mathematical Visualization*, pages 315–328. Springer Verlag, 1998.

[35] H. Löffelmann, L. Mroz, E. Gröller, and W. Purgathofer. Stream arrows: Enhancing the use of streamsurfaces for the visualization of dynamical systems. *The Visual Computer*, 13:359–369, 1997.

[36] Kw.-L. Ma, J. van Rosendale, and W. Vermeer. 3D shock wave visualization on unstructured grids. In *Proc. Symp. Volume Visualization*, pages 87–94, 1996.

[37] N. Max, B. Becker, and R. Crawfis. Flow volumes for interactive vector field visualization. In *Proc. of IEEE Visualization '93*, pages 19–24.

[38] R. Peikert and M. Roth. The parallel vectors operator – a vector field visualization primitive. In *Proc. of IEEE Visualization '99*, pages 263–270, 1999.

[39] Fr. H. Post and Th. van Walsum. Fluid flow visualization. In *Focus on Scientific Visualization*, pages 1–40. Springer, 1993.

[40] Fr. H. Post, B. Vrolijk, H. Hauser, R. S. Laramee, and H. Doleisch. Feature extraction and visualization of flow fields. In *EG 2002 State-of-the-Art Reports*, pages 69–100, 2002.

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

[42] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling. *Numerical Recipies*. Cambridge University Press, 1986.

[43] S. Rinaldi. Nonlinear dynamical systems: Bifurcation and chaos with applications in ecology. Notes to a lecture held at the Inst. of Econometrics, Vienna Univ. of Technology, May 1995.

[44] B. E. Rogowitz and Ll. A. Treinish. Why should engineers and scientists be worried about color? See URL http://www.research.ibm.com/people/l/lloydt/color/color.HTM.

[45] B. E. Rogowitz and Ll. A. Treinish. Data visualization: the end of the rainbow. *IEEE Spectrum*, 35(12):52–59, December 1998.

[46] M. Roth. *Automatic Extraction of Vortex Core Lines and Other Line-Type Features for Scientific Visualization*. PhD thesis, ETH Zürich, Switzerland, 2000. On-line materials available from martin.nobilitas.com/turbo/phdthesis.

[47] M. Roth and R. Peikert. Flow visualization for turbomachinery design. In *Proc. of IEEE Visualization '96*, pages 381–384, 1996.

[48] I. A. Sadarjoen and Fr. H. Post. Geometric methods for vortex extraction. In *Data Visualization '99*, Eurographics, pages 53–62. Springer-Verlag Wien, 1999.

[49] G. Scheuermann, T. Bobach, H. Hagen, K. Mahrous, B. Hamann, K. Joy, and W. Kollmann. A tetrahedral-based stream surface algorithm. In *Proc. of IEEE Visualization 2001*, pages 151–157.

[50] M. Schulz, F. Reck, W. Bartelheimer, and Th. Ertl. Interactive visualization of fluid dynamics simulations in locally refined cartesian grids. In *Proc. IEEE Visualization '99*, pages 413–416.

[51] B. Shneiderman. The eyes have it: A task by data type taxonomy for information visualization. In *Proc. IEEE Symp/ on Visual Languages*, pages 336–343, 1996.

[52] D. Silver, Fr. H. Post, and I. Sadarjoen. *Flow Visualization*, volume 7 of *Wiley Encyclopedia of Electrical and Electronics Engineering*, pages 640–652. John Wiley & Sons, 1999.

[53] D. Stalling and H.-Chr. Hege. Fast and resolution independent line integral convolution. In *Proc. of ACM SIGGRAPH '95*, pages 249–256.

[54] D. Stalling, M. Zöckler, and H.-Chr. Hege. Fast display of illuminated field lines. *IEEE Trans. on Visualization and Computer Graphics*, 3(2), 1997.

[55] D. Sujudi and R. Haimes. Identification of swirling flow in 3D vector fields. Technical Report AIAA Paper 95–1715, American Institute of Aeronautics and Astronautics, 1995.

[56] A. Telea and J. J. van Wijk. Simplified representation of vector fields. In *Proc. of IEEE Visualization '99*, pages 35–42, 1999.

[57] H. Theisel. Visualizing the curvature of unsteady 2D flow fields. In *Proc. of the 9th EG Workshop on Visualization in Sci. Computing*, pages 47–56, 1998.

[58] H. Theisel, T. Weinkauf, H.-Chr. Hege, and H.-P. Seidel. Grid-independent detection of closed stream lines in 2D vector fields. In *Proc. of the Vision Modeling and Visualization Conference 2004 (VMV 04)*, pages 421–428, 2004.

[59] H. Theisel, T. Weinkauf, H.-Chr. Hege, and H.-P. Seidel. Topological methods for 2D time-dependent vector fields based on stream lines and path lines. *IEEE Trans. on Visualization and Computer Graphics*, 11(4):383–394, 2005.

[60] Gr. Turk and D. Banks. Image-guided streamline placement. In *Proc. of ACM SIGGRAPH '96*, pages 453–460, 1996.

[61] S. K. Ueng, C. Sikorski, and Kw.-L. Ma. Efficient streamline, streamribbon, and streamtube constructions on unstructured grids. *IEEE Trans. on Visualization and Computer Graphics*, 2(2):100–110, June 1996.

[62] J. J. van Wijk. Spot noise – texture synthesis for data visualization. In *Proc. of ACM SIGGRAPH '91*, volume 25, pages 309–318, 1991.

[63] 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.

[64] R. Wegenkittl, H. Löffelmann, and E. Gröller. Visualizing the behavior of higher dimensional dynamical systems. In *Proc. of IEEE Visualization '97*, pages 119–125, 1997.

[65] R. Westermann. The rendering of unstructured grids revisited. In *Proc. Joint EG – IEEE TCVG Symp. on Visualizatation (VisSym 2001)*, pages 65–74. Springer-Verlag, May 28–30.

[66] Th. Wischgoll and G. Scheuermann. Locating closed streamlines in 3D vector fields. In *Proc. Joint EG – IEEE TCVG Symp. on Visualizatation (VisSym 2002)*, pages 227–280.

[67] Th. Wischgoll, G. Scheuermann, and H. Hagen. Tracking closed streamlines in time dependent planar flows. In *Proc. of the Vision Modeling and Vis. Conf. 2001 (VMV 01)*, pages 447–454.

[68] M. Zöckler, D. Stalling, and H.-Chr. Hege. Interactive visualization of 3D vector fields using illuminated streamlines. In *Proc. of IEEE Visualization '96*, pages 107–113, October 1996.