Volume 22 (2003), Number 2, yet unknown pages

The State of the Art in Flow Visualization, part 1: Direct, Texture-based, and Geometric Techniques

Helwig Hauser, Robert S. Laramee, Helmut Doleisch,
{Hauser,Laramee,Doleisch}@VRVis.at
VRVis Research Center, Austria
http://www.VRVis.at/

Frits H. Post, Benjamin Vrolijk
{F.H.Post,B.Vrolijk}@its.tudelft.nl
Delft University of Technology, The Netherlands
http://visualisation.tudelft.nl/

Abstract

Flow visualization has been a very attractive part of visualization research for a long time. Usually very large datasets need to be processed, which often consist of multivariate data with a large number of sample locations, often arranged in multiple time steps. Recently, the steadily increasing performance of computers again has become a driving factor for a reemergence in flow visualization, especially in techniques based on feature extraction, vector field clustering, and topology extraction.

In this article and a companion paper [69], the state of the art in flow visualization (FlowVis) is presented. In this paper, direct (or global) flow visualization such as hedgehog plots, texture-based flow visualization such as line integral convolution, and geometric flow visualization such as stream lines is discussed. Part 2 of this report [69] focusses on feature-based flow visualization in detail. In this paper, also an attempt of categorizing FlowVis solutions is presented which is incorporated as a structure for part 1 (this paper). FlowVis fundamentals are shortly addressed as well as conclusions and future prospects.

Categories and Subject Descriptors (according to ACM CCS): I.3 [Computer Graphics]: visualization, flow visualization, computational flow visualization

1. Introduction

Flow visualization (FlowVis) is one of the traditional subfields of visualization, covering a rich variety of applications, ranging from the automotive industry, aerodynamics, turbomachinery design, to weather simulation and meteorology, climate modelling, ground water flow, and medical applications, with different characteristics relating to the data and user goals. Consequently, the spectrum of FlowVis solutions is very rich, spanning multiple dimensions of technical aspects, e.g., 2D vs. 3D solutions, techniques for steady and time-dependent data.

Bringing many of those solutions in linear order (as necessary for a text like this) is neither easy nor intuitive. Several options of subdividing this broad field of literature are possible. Hesselink et al., for example, addressed the problem of how to categorise FlowVis techniques in their 1994 overview of (at that time) recent research issues [24] and mention the dimensionality of flow data to deal with as the first example. In the following, several of those aspects are discussed on a higher level before literature is addressed directly.

1.1. Classification

According to the different needs of the users, there are different approaches to flow visualization (cf. Fig. 1):

- *Direct flow visualization:* Using an as direct as possible translation of the flow data into visualization cues, for example, by drawing arrows or color coding velocity, quite intuitive pictures can be provided, esp. in 2D. FlowVis solutions of this kind allow immediate investigation of the flow data, without requiring a lot of mental interpretation.
- *Texture-based flow visualization:* Similar to direct flow visualization, a texture is computed, which subsequently is used for rendering, to provide a dense visualization of the flow data. Through the integrated relation of texture values along the flow, a notion of where the flow goes is incorporated. In most cases this effect is achieved through filtering of texture values along the flow.
- *Geometric flow visualization:* For a better communication of the long-term behavior induced by flow dynamics, *integration-based approaches* first integrate the flow data and use resulting integral objects as a basis for flow

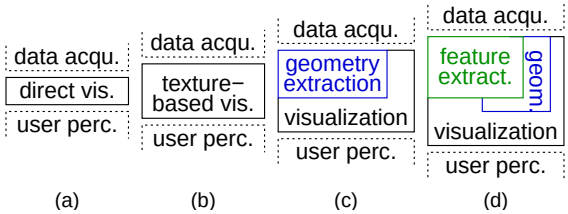

Figure 1: *Computations involved in FlowVis – direct flow visualization (a) vs. texture-based FlowVis (b) vs. FlowVis based on geometric objects (c) vs. feature-based FlowVis (d). This classification relates to the first-level structure of this report.*

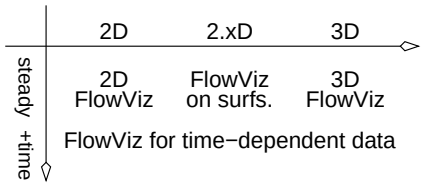

Figure 2: *FlowVis spans different spatial dimensions, and also time (1D & nD omitted here). This categorization corresponds to the second-level structure of this report.*

visualization; examples are stream lines, streak lines, and path lines. In a similar way, also differently computed geometric objects, like iso-surfaces or timelines also are useful for flow visualization. In the following, we will jointly discuss techniques of this kind as *geometric techniques*.

- *Feature-based flow visualization:* Yet another approach makes use of an abstraction and/or extraction step which is performed before visualization. Special features are extracted from the original dataset, such as important phenomena or topological information of the flow. Flow visualization is then based on these flow features (instead of the entire dataset), allowing for compact and efficient flow visualization of even very large and/or time-dependent datasets. Part 2 of this state-of-the-art report [69] covers feature-based flow visualization in detail.

Figure 1 illustrates the difference between the aforementioned classes. Note the increasing amount of computation needed for the visualization step when changing from direct flow visualization (a) to feature-based flow visualization (d). In this overview we use separate chapters for three of these classes: direct flow visualization is discussed in section 3, texture-based flow visualization is reviewed in section 4, and geometric techniques for flow visualization in section 5.

1.2. Dimensions

In flow visualization, available solutions significantly differ with respect to the given dimensionality of the flow data. Techniques which are useful for 2D flow data, like color coding or arrow plots, sometimes lack similar advantages in 3D. Also, the question of whether the flow data is steady or time-dependent has a large impact with respect to the FlowVis solution of choice.

In this state-of-the-art report, we substructure the sections about different classes of FlowVis solutions into subsections with respect to the different spatial dimensions involved. This report focusses on 2D and 3D flow visualization, although there is much interesting work about 1D FlowVis and nD FlowVis (with $n > 3$) as well.

In the following, the top-level sections start with a subsection on *2D FlowVis techniques* (sections *n*.1), i.e., covering solutions which focus on 2D flow data (in planar 2D domains). Since the 2D domain inherently corresponds to the 2D screen, good overviews are possible for these kinds of techniques like with the use of 2D LIC (see below for details). However, the reader should be aware, that real-world flows (at least when talking about fluids or gases) are rarely two-dimensional – datasets therefore are often slices out of a 3D stack of those, or stem from simplifications of the underlying model.

A second subsection (sections *n*.2) discusses *FlowVis solutions for boundary flows or sectional subsets of 3D flows*, for example, flow data on planar cross sections or on 2D manifolds in 3D space. This subsection therefore deals with 2D flow data – at least with respect to the local dimensionality of the data – but which is embedded within 3D space. While boundary flows often are of actual interest to users (like in aerospace design), the visualization of sectional subsets of 3D flow usually needs special care (not at the least because of the usually missing flow component). Especially questionable is the use of integral curves across flow cross-sections as the suggested particle paths do not correspond to actual flow trajectories (which naturally would extend to 3D in this case).

Finally, a third subsection (sections *n*.3) discusses *truly 3D FlowVis solutions*, i.e., visualization techniques, which apply to true 3D flow data. With true 3D flow visualization, rendering becomes a central issue. In many cases compromises are needed, trading visibility for completeness. Solutions range from clipping and opacity modulations to feature-based selections.

In addition to the spatial dimensions as addressed above, also *dimensionality with respect to time* is of great importance in flow visualization. First of all, flow data itself incorporates a notion of time – flows often are interpreted as differential data with respect to time (cf. equ. 1), i.e., when integrating the data, paths of moving entities are obtained (cf. equ. 3). Additionally, the flow itself can change over time (like in turbulent flows), resulting in time-dependent or unsteady data. In this case, two dimensions of time are present and the visualization must carefully distinguish be-

tween both in order to prevent users from being confused. Instanteneous stream lines, for example, are hard to interpret correctly in the context of unsteady flow. This distinction of time dimensions is especially true, when animation is used for flow visualization. Then, even a third temporal dimension shows up in a visualization (like the motion of the camera), requiring special care to avoid confusion with interpretation of the animations.

Although the distinction between steady and unsteady flows could open another dimension when sorting FlowVis literature, in this report solutions for time-dependent data are beside related techniques for steady data.

In many cases, additional to spatial and temporal dimensions, further data attributes are given along with the data, such as temperature, pressure, or vorticity. Flow visualization also takes these values into account, e.g., through coloring or iso-surface extraction.

1.3. Experimental and computational FlowVis

Flow visualization, as discussed in this paper, is considered to be equivalent to what others call *computational flow visualization* – to distinguish it from the large and old fields of experimental and empirical flow visualization.

Although we do not have space to focus on those other variants of flow visualization, it is interesting to recognise that many computational FlowVis solutions more or less mimic the visual appearance of well-accepted techniques in experimental visualization (cf. particle traces, dye injection, or Schlieren techniques).

1.4. Data sources

Computational flow visualization, in general, deals with data that exhibit temporal dynamics such as results from (a) *flow simulation* (e.g., the simulation of fluid flow through a turbine), (b) *flow measurements* (possibly acquired through laser-based technology), or (c) *analytic models* of flows (e.g., dynamical systems [1], given as set of differential equations).

In this report we mainly focus on flow visualization dealing with data from computational flow simulation, i.e., flow data given as a set of samples on a grid. Flow visualization of this kind often is also called *vector field visualization*. In many cases, the velocity information in such a flow dataset (encoded within the set of vectors) represents the prime focus during visualization. Nevertheless, the relation of computational and experimental flow visualization needs to be metioned: experimental flow visualization, as in a wind tunnel, for example, is also used to validate computational flow simulation. In such a case the computational flow visualization needs to be set up in a way such that results can be compared easily.

1.5. Placement and interaction

Many FlowVis solutions build on the use of individual visualization objects such as stream lines. The placement of those visualization cues is an issue within FlowVis literature for at least three reasons: (1) when using geometric objects such as stream lines, an even distribution of seed locations usually does not result in an even distribution of geometric objects in image space – separate algorithms need to be employed; (2) when dealing with 3D flow data, occlusion and/or visualization complexity raises special challenges – dense placement often results in severe cluttering within rendered images; (3) when using feature-based strategies, placement needs to be coupled and aligned with the feature extraction aspects of the visualization technique.

In addition to placement, user interaction plays an important role, especially in case of flow analysis. Users require systems which allow (1) navigation, including zooming and panning, (2) interactive placement of visualization cues, for example, using an interactive rake for stream-line seeding, or even (3) options of directly interacting with the flow data itself, for example, through *steering* [44] of the simulation.

2. FlowVis fundamentals

Before outlining some of the most important FlowVis techniques in the main part of this paper, a short overview about the common mathematical background as well as some general concepts with regard to the computation of FlowVis results are discussed.

2.1. Flow data

An inherent characteristic of flow data is that derivative information is given with respect to time, which is laid out with respect to an n-dimensional domain $\Omega \subseteq R^n$, for example, $n = 3$ for representing 3D fluid flow. In the case of multidimensional flow data ($n > 1$), temporal derivatives $\mathbf{v}$ of nD locations $\mathbf{p}$ within the flow domain Ω are n-dimensional vectors:

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

In analytic models (like dynamical systems [1]), vectors $\mathbf{v}$ often are described as functions of the respective spatial locations $\mathbf{p}$, say like $\mathbf{v} = \mathbf{A}\mathbf{p}$ for steady, linear flow data, $\mathbf{A}$ being a constant $n \times n$-matrix. A more general formulation of (possibly unsteady, i.e., time-dependent) flow data $\mathbf{v}$ would be

$$\mathbf{v}(\mathbf{p}, t) : \Omega \times \Pi \rightarrow R^n \qquad (2)$$

where $\mathbf{p} \in \Omega \subseteq R^n$ represents the spatial reference of the flow data and $t \in \Pi \subseteq R$ represents the system time. For steady flow data, the more simple case of $\mathbf{v}(\mathbf{p}) : \Omega \rightarrow R^n$ is given ($\mathbf{v}$ not dependent on t).

In results from nD flow simulation, like in automotive applications or airplane design, vector data $\mathbf{v}$ usually is not given in analytic form, but needs to be reconstructed from

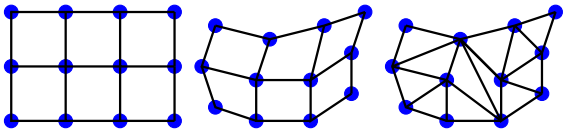

Figure 3: *Grids involved in flow simulation – Cartesian (a) vs. curvilinear (b) vs. unstructured grids (c). Issues are implicit (a) vs. explicit locations (b, c), implicit (a, b) vs. explicit neighborhood (c).*

the discrete simulation output. As usually numerical methods are used to actually do the flow simulation, such as finite element methods, the output of flow simulation usually is a large-sized grid of many sample vectors $\mathbf{v}_i$, which discretely represent the solution of the simulation process. Furthermore, it is assumed that the flow simulation is based on an at least locally continuous model of the flow, thus allowing for continuous reconstruction of the flow data $\mathbf{v}$ during visualization. One option for doing so would be to apply a reconstruction filter $h : R^n \rightarrow R$ to compute $\mathbf{v}(\mathbf{p}) = \sum_i h(\mathbf{p}-\mathbf{p}_i)\,\mathbf{v}_i$. As – for practical reasons – filter h usually has only local extent, efficient procedures for finding those flow samples $\mathbf{v}_i$, which actually are nearest to the query point $\mathbf{p}$, are needed to do proper reconstruction.

2.2. Dealing with Grids

In flow simulation, the vector samples $\mathbf{v}_i$ usually are laid out across the flow domain with respect to a certain type of grid. Grid types range from simple Cartesian grids over curvilinear grids to complex unstructured grids (cf. Fig. 3). Typically, simulation grids also exhibit large variations in cell sizes. This variety of grids stems from the high influence of grid design onto the flow simulation process and the thereby derived need to model the flow grid as optimal as possible with respect to the simulation in mind.

Although the principal theory of function reconstruction from discrete samples does not exhibit too many differences with respect to grid types involved, the practical handling does. While neighbor searching might be trivial in a Cartesian grid, it usually is not in a tetrahedral grid. Similar differences are given for the problems of point location and vector reconstruction. In the following we shortly describe several fundamental operations which form the basis for FlowVis computations on simulation grids.

Starting with *point location*, i.e., the problem of finding the grid cell which a given *n*D-point lies in, usually two cases are distinguished. For the general point location problem, special data structures can be used which subdivide the spatial domain to speed up the search. For iterative point location, which often is needed during integral curve computation, algorithms are used which efficiently exploit spatial coherence during the search. One kind of such algorithms starts with an initial guess for the target cell, checks for point-containment then and refines accordingly afterwards. This process is iterated until the target cell is found. More details can be found in older texts about flow visualization fundamentals [83, 67].

Beside point location, *flow reconstruction* within a cell of the flow dataset is a crucial issue in flow visualization. Often, once the cell which contains the query location is found, only the sample vectors at the cell's vertices are considered for flow reconstruction. The most often used approach is first-order reconstruction by performing linear interpolations within the cell. Within a hexahedral cell in 3D, for example, trilinear flow reconstruction can be used.

Using point location and flow reconstruction, flow visualization can begin: vectors can be represented with glyphs, virtual particles can be injected and traced across the flow domain. Nevertheless, the *computation of derived data* might be necessary to do more sophisticated flow visualization. Usually, the first step is to request (second-order) gradient information for arbitrary points in the flow domain, i.e., $\nabla\mathbf{v}|_{\mathbf{p}}$, which gives information about local properties of the flow (at point $\mathbf{p}$) such as flow convergence and divergence, or flow rotation and shear. For feature extraction, flow vorticity $\omega = \nabla \times \mathbf{v}$ can be of high interest. Further details about local flow properties can be found in previous work [68, 57].

2.3. Flow integration

Recalling that flow data in most cases is derivative information with respect to time, the idea of integrating flow data over time is natural to provide an intuitive notion of (long-term) evolution induced by the flow data. An example would be flow visualization by the use of particle advection. A respective particle path $\mathbf{p}(t)$ – here through unsteady flow – would be defined by

$$\mathbf{p}(t) = \mathbf{p}_0 + \int_{\tau=0}^{t} \mathbf{v}(\mathbf{p}(\tau), \tau + t_0)\,\mathrm{d}\tau \qquad (3)$$

where $\mathbf{p}_0$ represents the seed location of the particle path and t_0 equals the time when the particle was seeded. Note, that equations 1 and 3 are complimentary to each other. For other types of integral curves, such as stream lines and streak lines, refer to later parts of this text or previous works [83, 45].

In addition to the theoretical specification of integral curves, it is important to note, that respective integral equations like equation 3 usually cannot be resolved for the curve function analytically, and thereby *numerical integration methods* need to be employed. The most simple approach is to use a first-order Euler method to compute an approximation $\mathbf{p}_E(t)$ – one iteration of the curve integration is specified by

$$\mathbf{p}_E(t + \Delta t) = \mathbf{p}(t) + \Delta t \cdot \mathbf{v}(\mathbf{p}(t), t) \qquad (4)$$

where Δt usually is a very small step in time and $\mathbf{p}(t)$ denotes the location to start this Euler step from. A more

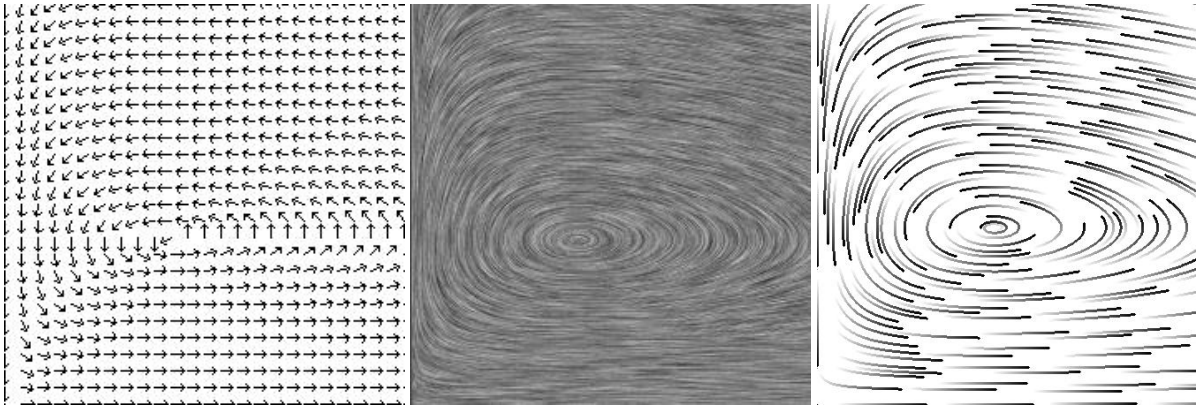

Figure 4: *Example from theoretical biology (a food chain) [53] for comparing FlowVis techniques from Sects. 3, 4, and 5. FlowVis by the use of arrows (left) vs. texture-based FlowVis by the use of LIC (middle) and FlowVis based on geometric objects (right).*

accurate but also more costly technique is the second-order Runge-Kutta method, which uses the Euler approximation $\mathbf{p}_E$ as a look-ahead to compute a better approximation $\mathbf{p}_{RK2}(t)$ of the integral curve:

$$\mathbf{p}_{RK2}(t+\Delta t) = \mathbf{p}(t) + \Delta t \cdot (\mathbf{v}(\mathbf{p}(t),t) + \mathbf{v}(\mathbf{p}_E(t+\Delta t),t))/2 \quad (5)$$

Higher-order methods like the often used fourth-order Runge-Kutta integrator utilise more such steps to better approximate the local behavior of the integral curve. Also, adaptive step sizes are used to make smaller steps in regions where a lot of change takes place in the flow.

In the following, three classes of approaches in the field of flow visualization are discussed in detail – direct flow visualization is described in section 3, texture-based flow visualization in section 4, and geometric flow visualization is discussed in section 5. Figure 4 illustrates these three kinds of flow visualization side-by-side. Part 2 of this report [69] covers the fourth class of feature-based flow visualization in detail.

3. Direct flow visualization

Direct (or global) flow visualization attempts to present the complete dataset, or a large subset of it, at a low level of abstraction. The mapping of the data to a visual representation is performed without complex conversion or extraction steps. These techniques are perhaps the most intuitive visualization strategies as they present the data as is. Difficulties arise, when the long-term behavior induced by flow should be investigated – this potentially requires cognitive integration of direct visualization results.

3.1. Direct flow visualization in 2D

In this subsection we shortly address widely distributed, i.e., fairly standard techniques for 2D flow visualization, including coloring and arrow plots.

Color coding in 2D – a common direct flow visualization technique is to map flow attributes such as velocity, pressure, or temperature to color. Since color plots are widely distributed, this approach results in very intuitive depictions. Of course, the color scale which is used for mapping must be chosen carefully with respect to perceptual differentiation. Color coding for 2D flow visualization extends to time-dependent data very well, resulting in moving color plots according to changes of the flow properties over time. Of course, if time steps are too widely spaced, temporal aliasing can occur.

Arrow plots in 2D – a natural vector visualization technique is to map a line, arrow, or glyph to each sample point in the field, oriented according to the flow field, as in figure 4 (left). Usually a regular placement of arrows is used in 2D, for example, on an evenly-spaced Cartesian grid. Two variants of arrow plots are often used: (1) normalised arrows of unit length which visualise the direction of the flow only and (2) arrows of varying length (proportional to the flow velocity). Klassen and Harrington [43] and Schroeder et al. [77] call this technique a *hedgehog visualization* (because of the bristly result). Hedgehog plots in 2D can be extended to time-dependent data. However, without a proper sampling in the time dimension, easily temporal aliasing can occur (jumping arrows), diminishing the quality of such a visualization.

Hybrid direct FlowVis in 2D – Kirby et al. propose the simultaneous visualization of multiple values of 2D flow data by using a layering concept similar to the painting process of artists [41]. Arrow plots are mixed with color coding to provide rich visualization results.

3.2. Direct flow visualization on slices or boundaries

When dealing with 3D flow data, visualization naturally faces additional challenges such as 3D rendering. Acting

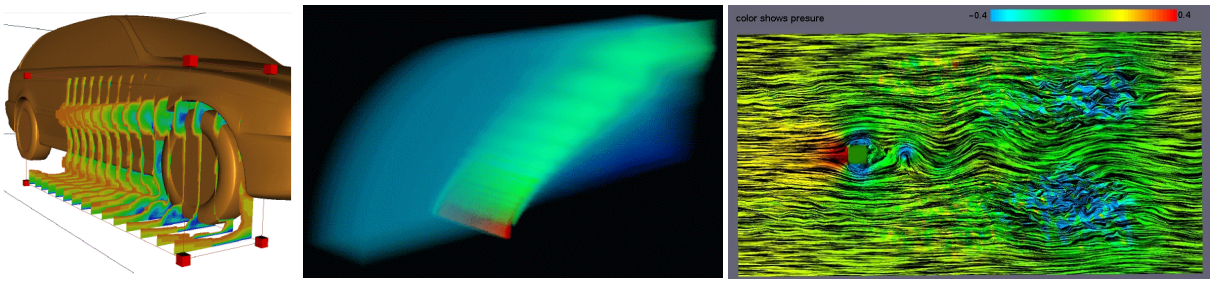

Figure 5: *Examples of direct flow visualization – an interactive slicing probe with colored slices and scalar clipping (left)* [78]*, direct volume rendering based on resampling (middle)* [103]*. Example of texture-based, colored spot noise (right)* [48].

as a middle ground between 2D flow visualization and the visualization of truly 3D flow data, the restriction to subdimensional parts of the 3D domain, e.g., sectional slices or boundary surfaces also is used for flow visualization. Techniques known from 2D flow visualization usually are applicable without major changes (at least from a technical point of view). When working with sectional slices, the treatment of flow components orthogonal to slices requires special care.

Color coding on slices or boundaries – color coding is very effective for visualising boundary flows or sectional subsets of 3D flow data. A good example is NASA's Field Encapsulation Library [62], which allows to easily use both techniques for various flow data. Schulz et al. also use color coding of scalars on 2D slices in 3D automotive simulation data [78] as shown in figure 5 (left). They introduce an interactive slicing probe which maps the vector field data to hue. The use of scalar clipping, i.e., the transparent rendering of slice regions where the corresponding data value does not lie within a specific data range, allows to use multiple (colored) slices with reduced problems due to occlusion.

Arrows on slices or boundary surfaces – using 2D arrows on slices from 3D flow data is also a possible visualization technique [17]. However, results of such a visualization should be interpreted with care, as flow which is orthogonal to the slice usually is not depicted. Hedgehog plots work better for boundary flows. In these cases, rarely cross-boundary flows are given. Therefore the use of arrows spread out over boundary surfaces usually is very effective, as used by Treinish for weather visualization [93].

3.3. Direct flow visualization in 3D

After focussing on direct FlowVis on slices and boundary surfaces, direct flow visualization of real 3D flows is discussed. In contrast to previously mentioned techniques, here rendering becomes the most critical issue. Occlusion and complexity make it difficult (if possible at all) to get an immediate overview of an entire flow dataset in 3D.

Volume rendering for 3D flow visualization – the natural extension of color coding in 2D (or on slices) is color coding in 3D. This, however, poses special rendering requirements (volume rendering) due to occlusion problems and nontrivial complexity. Volume rendering is well-known in the field of medical 3D visualization, i.e., volume visualization. However, those challenges, which closely correspond to flow visualization are briefly addressed here: (1) flow datasets are often significantly smoother than medical data – an absence of sharp and clear "object" boundaries (like organ boundaries) makes mapping to opacities more difficult and less intuitive. (2) flow data is often given on non-Cartesian grids, e.g., on curvilinear grids – the complexity of volume rendering increases with such grids, starting with nontrivial solutions required for visibility sorting and blending. (3) flow data is also time-dependent in many cases, imposing additional loads on the rendering process.

In the early nineties, Crawfis et al. [13], as well as Ebert et al. [16] applied volume rendering techniques to vector fields. Later, Frühauf applied ray casting to vector fields [20]. Recently, Westermann, presented a relatively fast 3D volume rendering method using a resampling technique for vector field data from unstructured to Cartesian grids [103]. A result from this technique is illustrated in figure 5 (middle).

Recently, Clyne and Dennis [12] as well as Glau [22] presented volume rendering for time-varying vector fields using algorithms which make special use of graphics hardware. Ono et al. use direct volume rendering to visualise thermal flows in the passenger compartment of an automobile [66]. Their goal is to attain the ability to predict the thermal characteristics of the automotive cabin through simulation. Swan et al. apply direct volume rendering techniques in flow visualization in a system that supports computational steering [86]. Their visualization results are extended to the CAVE environment.

Recently, Ebert and Rheingans demonstrated the use of nonphotorealistic volume rendering techniques for 3D flow data [15]. They apply silhouette enhancement or tone shading to improve rendering of 3D flows.

Arrow plots in 3D – the use of arrows for direct 3D flow visualization poses at least two problems: (1) the position

and orientation of a vector is often difficult to understand because of its projection onto a 2D screen – using 3D representations of arrows decreases these problems with perception and (2) glyphs occluding one another become a problem – careful seeding is required in contrast to dense distributions.

In actual applications, arrow plots are usually based on selective seeding, for example, all arrows starting from one out of a few sectional slices through the 3D flow. Boring and Pang address the problem of clutter in 3D direct FlowVis by highlighting those parts of a 3D arrow plot, which point in a similar direction compared to a user-defined direction [6]. Their methodology reduces the amount of data being displayed thus results in less clutter. Their methods can be combined with other techniques that use glyph representations and flow geometries such as stream lines for flow visualization. They apply the methods to both analytic and simulation datasets to highlight flow reversals.

3.4. Hybrid direct FlowVis in 3D

Doleisch an Hauser present a system which provides interactive visualization of 3D data from flow simulation with a visual discrimination of foci and context based on a user-driven, interactive feature specification in attribute space [14].

4. Texture-based flow visualization

We make a distinction between geometric flow visualization (see Sect. 5) and dense, texture-based flow visualization, however, these two topics are closely coupled: conceptually, the path from using geometric objects to texture-based visualization is obtained via a dense seeding strategy. That is, densely seeded geometric objects result in an image similar to that obtained by dense, texture-based techniques. Likewise, the path from dense, texture-based visualization to visualization using geometric objects is obtained using something such as a sparse texture for texture advection. Texture-based techniques in flow visualization can provide dense spatial resolution images. Texture-based algorithms are effective, versatile, and applicable to a wide spectrum of applications. Sanna et al. present a summary of this research in a survey paper [74].

4.1. Texture-based flow visualization in 2D

In this subsection, we describe texture-based FlowVis solutions for 2D flow data, i.e., spot noise, line integral convolution (LIC), and related approaches.

Spot noise in 2D – spot noise, introduced by Van Wijk [105], was amongst the first texture-based techniques for vector field visualization. Spot noise generates a texture by distributing a set of intensity functions, or spots, over the domain. Each spot represents a particle moving over a small step in time and results in a streak in the direction of the local flow from where the particle is seeded.

One limitation of the original spot noise algorithm was the lack of velocity magnitude information in the resulting texture. Enhanced spot noise [52], by De Leeuw and Van Wijk, was introduced to address this problem. Spot noise has also been applied to the visualization of turbulent flow [50]. De Leeuw and Van Liere also compare spot noise to LIC [51]. Spot noise in 2D combined with color coding is shown in figure 5 (right).

Line integral convolution in 2D – *line integral convolution* (LIC), first introduced by Cabral and Leedom [10] is a very popular technique for the dense coverage of vector fields with flow visualization cues. The original methodology behind LIC takes as input a vector field on a Cartesian grid and a white noise texture of the same size. The noise texture is locally filtered (smoothed) along the path of stream lines to acquire a dense visualization of the flow field. See figure 4 (middle) for an example.

The research in flow visualization based on LIC described here extends LIC in several ways: (1) adding directional cues, (2) showing velocity magnitudes, (3) added support for non-Cartesian grids, (4) allowing real-time and interactive exploration, (5) extending LIC to 3D, and (6) extending LIC to unsteady vector field visualization with time coherency.

Shen et al. address the problem of directional cues in LIC by combining animation and introducing dye advection into the computation [80]. Kiu and Banks proposed to use a multifrequency noise for LIC [42]. The spatial frequency of the noise is a function of the magnitude of the local velocity.

Khouas et al. synthesise LIC-like images in 2D with fur-like textures [40]. Their technique is able to locally control attributes of the output texture such as orientation, length, density, and color.

Much research has been done to bring LIC computation to interactive rates. Stalling and Hege present significant improvements in LIC performance by exploiting coherence along stream lines [85, 23]. Parallel implementations of LIC are presented by Cabral and Leedom [9], and Zöckler et al. [110].

OLIC for 2D flow visualization – Wegenkittl et al. also address the problem of orientation of flow with their OLIC (Oriented Line Integral Convolution) approach [101]. Conceptually, the OLIC algorithm makes use of a sparse texture consisting of many separated spots which are more or less smeared in the direction of the local vector field through integration. A fast version of OLIC (called FROLIC) is presented by Wegenkittl and Gröller [100] via a trade-off of accuracy for time.

2D Texture Advection – Jobard and Lefer use a motion map data structure for animating 2D, steady-state flow fields [36]. The motion map contains both a dense representation of the flow and the information required to animate the flow. It offers the advantage of saving memory and computation time since only one image of the flow has to be computed and stored in the motion map data structure.

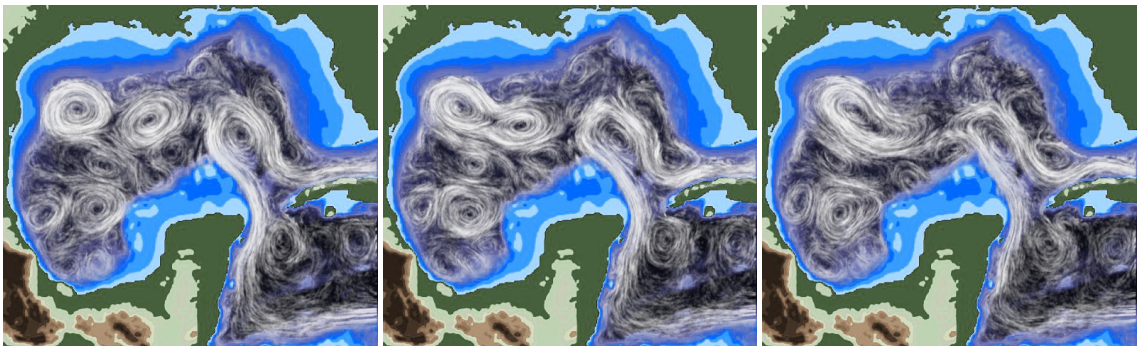

Figure 6: *Three images taken from an animation of an unsteady vector field created with the Lagrangian-Eulerian advection algorithm* [32].

Jobard et al. propose a technique to visualise dense representations of unsteady vector fields based on what they call a Lagrangian-Eulerian Advection scheme [33]. The algorithm combines a dense, time-dependent, integration-based representation of the vector field with interactive frame rates. Some results from the technique are shown in figure 6.

Jobard et al. [30, 31] present additional 2D, unsteady flow visualization techniques. They achieve high performance via the use of graphics hardware. They also detail spatial and temporal coherence techniques, dye advection techniques, and feature extraction.

Unsteady flow visualization techniques may address the problem of interactive performance time through the use of texture mapping supported by the graphics hardware. Becker and Rumpf illustrate hardware-supported texture transport for 2D, unsteady flow data [5].

In 2002, Jarke van Wijk [96] proposed image-based flow visualization which allows to achieve effects similar to LIC, spot noise, particle advection, and, of course, texture advection. By distorting the vertices of an underlying mesh and texturing the mesh with the visualization result from the last time step, interactive rendering frames even for time-dependent flow can be achieved.

4.2. Texture-based flow visualization on slices or boundaries

Texture-based techniques are, in general, better methods for conveying flow information on sectional slices than techniques using geometric objects. This is because the connection along the path of what would be a stream line is lost with dense, texture-based techniques. Thus the depiction of the flow is not misleading in terms of a potential suggestions of particle paths. Let us recall that the vector component orthogonal to the slice is removed when using texture-based and geometric methods for visualization results.

Spot noise on boundaries or slices – De Leeuw et al. extend the spot noise algorithm to surfaces in a study that compares experimental surface flow visualization with oil to that of spot noise on surfaces [49]. A combination of both texture-based FlowVis (on slices) and 3D arrows for 3D flow visualization is employed by Telea and Van Wijk [92] where arrows denote the main characteristics of the 3D flow after clustering and a 2D slice with spot noise or LIC is used to visualise the rest of the vector field on a slice.

LIC for boundary flows – a large body of research literature is dedicated to the extension of LIC onto boundary surfaces, surveyed by Stalling [84].

The extension of LIC to non-Cartesian grids and surfaces is presented by researchers such as Forssell [18]. Forssell and Cohen [19] extend LIC to curvilinear surfaces with animation techniques, add magnitude and direction information, and show how to use LIC to depict time-dependent flows. Their algorithm also utilises texture mapping hardware to improve performance time towards interactive rates.

Teitzel et al. [90] present an approach that works on both 2D unstructured grids and directly on triangulated surfaces in 3D space. Mao et al. [60] present an algorithm for convolving solid white noise on triangle meshes in 3D space, and extend LIC for visualising a vector field on arbitrary surfaces in 3D.

Battke et al. [3] describe an extension of LIC for arbitrary surfaces in 3D. Some approaches are limited to curvilinear surfaces, i.e., surfaces which can be parameterised by using 2D-coordinates. Their method also handles the case of general, multiply connected surfaces.

Scheuermann et al. present a method for visualising 3D vector fields that are defined on a 2D manifold [75]. Their work addresses the normal vector component to the surface that other methods do not.

A problem with many curvilinear grid LIC algorithms is that the resulting LIC textures may be distorted after being mapped onto the geometric surfaces, since a curvilinear grid usually consists of cells of different sizes. Mao et al. propose a solution to the problem by using multigranularity noise as the input image for LIC [59].

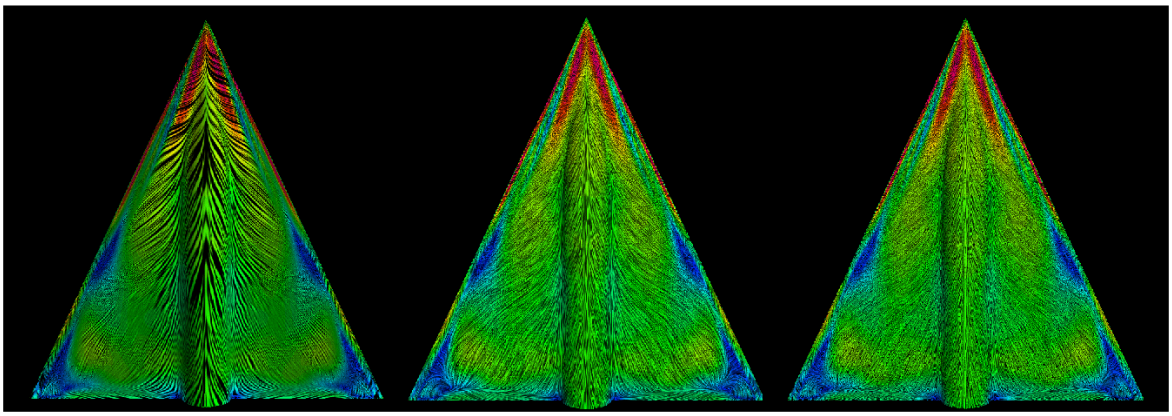

Figure 7: *A comparison of 3 LIC techniques: (left) UFLIC, (middle) ELIC, and (right) PLIC* [97].

UFLIC, PLIC, and others, – Shen and Kao present UFLIC (Unsteady Flow LIC), which incorporates time into the convolution [81, 79]. See figure 7 (left). Their algorithm addresses problems with temporal coherency by successively updating the convolution results over time. They also propose a parallel UFLIC algorithm. Verma et al. present a method for comparative analysis of stream lines and LIC called PLIC [97]. A visual comparison between ELIC (enhanced LIC) [65], PLIC, and UFLIC is shown in figure 7.

4.3. Texture-based flow visualization in 3D

High computational costs, demanding memory requirements, occlusion, and visual complexity can all be inhibitants for texture-based flow visualization in 3D.

LIC in 3D – occlusion and interactive performance are nontrivial challenges when implementing LIC in 3D. Rezk-Salama et al. tackle the problem of interactive performance using a 3D-texture mapping approach combined with an interactive clipping plane to address the problems of occlusion and interaction [70]. A combined approach of direct volume rendering and LIC is taken by Interrante [29] for extending LIC to 3D. Interrante and Grosch address some perceptual difficulties encountered with dense, 3D visualizations [27, 28, 29]. Techniques for selectively emphasising important regions of interest in the flow, enhancing depth perception, and improving orientation perception of overlapping stream lines are discussed.

Texture advection in 3D – Kao et al. discuss the use of 3D and 4D texture advection for the visualization of 3D fluid flows [38]. Formidable challenges are introduced by the memory requirements involved in using 3D and 4D textures. They also apply a steady-state animation to these 3D and 4D textures.

5. Geometric flow visualization

Geometric flow visualization entails the extraction of geometric objects whose shapes are directly related to the underlying data – like an iso-surface corresponding to a certain velocity magnitude or a stream line which represents the temporal evolution of the flow. In what follows, we therefore discuss geometric flow visualization techniques such as contouring in both 2D and 3D as well as geometric flow visualization using integral objects (such as stream lines).

5.1. Geometric flow visualization in 2D

In this subsection we shortly discuss geometric FlowVis techniques in 2D based on objects such as streamlets, stream lines, and their relatives within unsteady flows. Also, the seeding problem is addressed, which requires a solution in order to realise better distributions of geometric objects.

Streamlets in 2D – if flow vectors are integrated for a very short time, *streamlets* are generated. Even though short, streamlets already communicate temporal evolution along the flow. Figure 4 shows an example, where several streamlets are used to visualise a 2D flow field.

Stream lines in 2D – if longer integration is performed (as compared to streamlets), *stream lines* are gained. They are a natural extension of glyph-based techniques and offer intuitive semantics: users easily understand that flows evolve along integral objects.

Stream lines, timelines, and path lines – when unsteady flow data are investigated, several distinct integral objects are used for flow visualization. A *path line* or *particle trace* is the trajectory that a particle follows in a fluid flow [77]. A *timeline* joins the positions of particles released at the same instant in time from different insertion points, i.e., joins points at a constant time t [64]. A *streak line* is traced by a set of particles that have previously passed through a unique

Figure 8: *Three images from an interactive exploration of a vector field using the MR viewer [34]. A suitable level of resolution can be chosen while maintaining a roughly constant stream-line density.*

point in the domain [77]. Stream lines relate to continuous injection of foreign material into real flow. Sanna et al. present an adaptive visualization method using streak lines where the seeding of streak lines is a function of local vorticity [73].

Stream-line seeding in 2D – one important aspect of stream lines, or integral curves, when used for visualising continuous vector fields is the best choice of initial conditions. Since, in general, evenly distributed seed points do not result in evenly spaced stream lines, special algorithms need to be employed:

1. Turk, Banks [94] and Jobard, Lefer [35] developed techniques for automatically placing seed points to achieve a *uniform distribution* of stream lines on a 2D vector field.
2. Strategies for stream-line seeding in 2D may also be *topology-based*. Verma et al. [98] present a seed placement strategy for stream lines based on flow features in the dataset. Their goal is to capture flow patterns near critical points in the flow field.
3. Building on their previous work, Jobard and Lefer presented a multiresolution (MR) method for visualising large, 2D, steady-state vector fields [34]. The MR hierarchy supports enrichment and zooming. The user is able to interactively set the density of stream lines while zooming in and out of the vector field (Fig. 8). The density of stream lines can also be computed automatically as a function of velocity or vorticity.

Seeding of integral objects becomes a special challenge when dealing with time-dependent data. Jobard and Lefer presented an unsteady FlowVis algorithm by correlating instantaneous visualizations of the vector field at the stream-line level [37]. For each frame, a feed forward algorithm computes a set of evenly spaced stream lines as a function of the stream lines generated for the previous frame. Their method also provides full control of the image density so that smooth animations of arbitrary density can be produced.

Contouring in 2D – *contouring* is a natural extension to color coding in 2D, instead of color changes, contour lines represent the distribution of values. Also, the combination of both approaches is useful.

5.2. Geometric FlowVis on slices or boundaries

After discussing 2D FlowVis based on geometric objects, this subsection shortly addresses similar approaches on subsets of 3D flows such as boundary flows. Interpretation of integral curves on sectional slices requires special care, again.

Integrated tufts – Wegenkittl et al. use *integrated tufts* (similar to streamlets), seeded on specific equilibrium surfaces, for visualizing a complex dynamical system [102], also over variations of that system in a fourth dimension.

Geometric flow visualization on surfaces or boundaries – similar to 2D flow visualization, geometric objects such as stream lines are also used for visualising boundary flows or sectional slices through 3D flow [17]. However, it is important to note that the use of these objects on slices may be misleading, even within steady flow datasets. A stream line on a slice may depict a closed loop, even though no particle would ever traverse the loop. The reason again lies in the fact, that flow components which are orthogonal to the slice are omitted during flow integration.

Stream-line seeding on boundary surfaces – Mao et al. [58] extend the stream-line seeding of Turk and Banks [94] in order to generate evenly distributed stream lines on boundary surfaces within curvilinear grids.

5.3. Geometric flow visualization in 3D

When dealing with 3D flow, a rich variety of geometric objects is available for flow visualization. This subsection addresses a series of objects, from streamlets to flow

volumes, primarily sorted according to their dimensionality, and within equal dimensionality roughly with respect to which technique extends to another.

Streamlets in 3D – streamlets easily extend to 3D, although perceptual problems may arise due to distortions resulting from the rendering projection. Also, seeding becomes more important in 3D, again. Löffelmann and Gröller use a thread of streamlets along characteristic structures of 3D flow to gain selective, but importance-based seeding as well as an enhancement of abstract flow topology through direct visualization cues [54].

Streamlets in 3D – at NASA the Flow Analysis Software Toolkit (FAST) [2] is used to visualise CFD data based on stream lines in 3D. Careful seeding is necessary to obtain useful results, since visual clutter can easily become a problem. If interactive seeding (by the use of a VR-steered rake [8], for examaple) is used, however, less problems with occlusion occure (due to the selective placement).

Zöckler et al. present illuminated stream lines to improve perception of stream lines in 3D by taking advantage of the texture mapping capabilities supported by graphics hardware [109]. Their shading technique increases depth information. By making the stream lines partially transparent, they also address the problem of occlusion, as shown in figure 9 (left). For seeding, the authors propose an interactive seeding probe which can be moved around to start stream lines at specific places of interest. Also, seeding near potential objects of interests is demonstrated.

Particle tracing in 3D – Kenwright and Lane present an efficient, 3D particle tracing algorithm that is also accurate for interactive investigation of large, unsteady, aeronautical simulation [39]. A performance gain is obtained by applying tetrahedral decomposition to speed up point location and velocity interpolation in curvilinear grids.

Teitzel et al. analyse different integration methods in order to evaluate the trade-off between time and accuracy [89, 91]. They present a 3D particle tracing algorithm targeted at sparse grids that is very efficient with respect to storage space and computing time. The authors recommend using sparse grids as a data compression method in order to visualise huge datasets.

Nielson presents efficient and accurate methods for computing tangent curves for 3D flows [63]. The methods work directly with physical coordinates, eliminating the need to switch back and forth to computational coordinates. Efficient particle tracing methodologies are also addressed by Sadarjoen et al. [72].

Since stream lines are usually easily computed in real time, they offer (together with their intuitive semantics) an often chosen tool for interactive flow analysis [8].

Stream ribbons and stream tubes – a first extension of stream lines in 3D are *stream ribbons* and *stream tubes*. A stream ribbon is basically a stream line with a winglike strip added, to also visualise rotational behavior of the 3D flow which is not possible with stream lines alone [95]. A stream tube is a thick stream line that can be extended to show the expansion of the flow [95]. Stream ribbons and stream tubes offer advantages over stream lines in that way, that they can encode more properties, such as divergence and convergence of the vector field, in the geometric properties of the respective integral objects.

Ueng et al. present techniques for efficient construction of stream lines, stream ribbons, and stream tubes on unstructured grids [95]. A specialised Runge-Kutta method is employed to speed up stream-line computation. Explicit solutions are calculated for the angular rotation rates of stream ribbons and the radii of stream tubes. The resulting speedup in overall performance aids in the exploration of large flow fields.

Fuhrmann and Gröller [21] use so-called *dash tubes*, i.e., animated, opacity-mapped stream tubes, as a visualization icon. An algorithm is described which places the dash tubes evenly in 3D space. They also apply a magic lens and magic box as interaction techniques for investigating densely filled areas without filling the image with visual detail and complexity.

Laramee introduces the *stream runner* as an extension of stream tubes – an interactively controlled 3D flow visualization technique that attempts to minimise occlusion, minimise visual complexity, maximise directional cues, and maximise depth cues by letting the user control the length of the stream tubes [46].

Stream polygons – another extension of stream lines are *stream polygons* used by Schroeder et al. [76]. Stream polygons are tools to visualise vectors and tensors using tubes with a polygonal cross section. The properties of the polygons such as the radius, the number of sides, the shape and the rotation reflect properties of the vector field including strain, displacement, and rotation.

Stream balls and streak balls – *Stream balls* are a useful flow visualization technique used by Brill et al. [7], which visualises divergence and *acceleration* in fluid flow. Stream balls split or merge depending on convergence/divergence and acceleration/deceleration, respectively. Teitzel and Ertl introduce *streak balls* when they present and compare two different approaches to accelerate particle tracing on sparse grids and curvilinear sparse grids for unsteady flow data [88].

Stream surfaces – Yet another extension to stream lines are *stream surfaces*, which are surfaces that are everywhere tangent to a vector field. A stream surface can be approximated by connecting a set of stream lines along the corresponding timelines of the instanteneous flow (and varying the number of stream lines used according to convergence or divergence

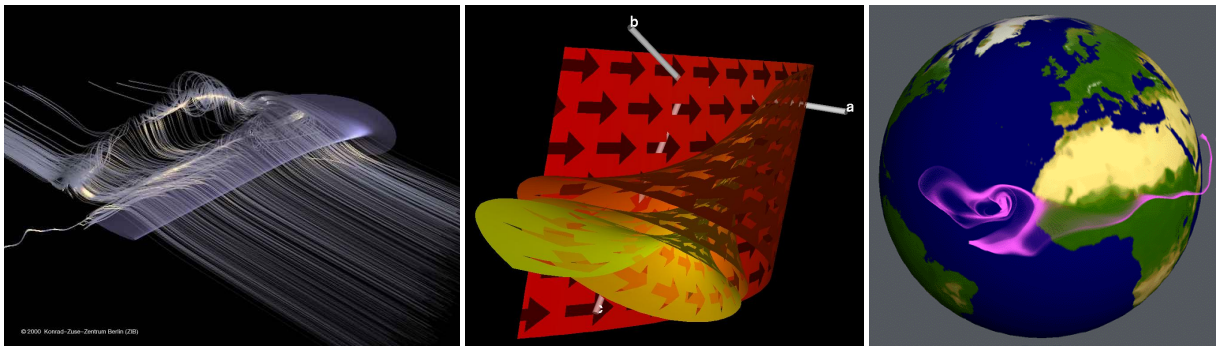

Figure 9: *Examples of flow visualization using geometric objects – illuminated stream lines (left)* [109]*, stream arrows (middle)* [56]*, and flow volumes (right)* [61].

of the flow) [26]. Stream surfaces are very good for texture-based visualization techniques such as Spot Noise and LIC, because there is no cross-flow component normal to the surfaces, i.e. the vector field is not projected like it is for 2D slices through a 3D domain. Stream surfaces present challenges related to occlusion, visual complexity, and interpretation.

Hultquist presents an interactive flow visualization technique using stream surfaces [25]. Van Wijk presents two follow-up techniques for generating implicit stream surfaces [107]. Cai and Heng [11] address the issues associated with the placement and orientation of stream surfaces in 3D.

Löffelmann et al. present *stream arrows* (see Fig. 9, middle) as an enhancement of stream surfaces by separating arrow-shaped portions from a stream surface [56, 55]. Stream arrows address the problem of occlusion associated with 3D flow visualization, but especially with stream surfaces. Stream arrows also provide additional information about the flow, usually not seen with stream surfaces, such as flow direction and flow convergence/divergence.

Van Wijk simulates stream surfaces by a large set of so-called surface particles [106]. Surface particles exhibit less occlusion when compared to stream surfaces. Interestingly, Van Wijk's approach in a way anticipated recent advances in pixel-based rendering techniques.

Flow volumes – The last (direct) extension of a stream line into 3D described here are *flow volumes* (see Fig. 9, right). A flow volume is a specific subset of a 3D flow domain, which is traced out by a particular initial 2D patch over time as described by Max et al. [61]. The resulting volume is divided up into a set of semitransparent tetrahedra, which are volume rendered in hardware in a way derived from the method of Shirley and Tuchmann [82].

Becker et al. extend flow volumes to unsteady flow [4]. The resulting unsteady flow volumes are the 3D analogue of streak lines. Considerations are made when extending the visualization technique to unsteady flows since particle paths may become convoluted in time. The authors present some solutions to the problems which occur in subdivision, rendering, and system design. The resulting algorithms are applied to a variety of flow types including curvilinear grids.

Time surfaces in 3D – a natural extension of timelines (in 2D or 3D) are *time surfaces*, when constant-time instants of moving particles are assumed, which previously have been released from a two-dimensional patch. An example of an application of this principle, are level-set surfaces used by Westermann et al. [104].

Isosurfaces for 3D flow visualization – extending contouring from 2D to 3D, results in the use of isosurfaces for 3D flow visualization. Special care needs to be taken with isovalue selection, mostly because of the usually smooth nature of flow data – in cases of no sharp transitions within the data, any isovalue lacks (at least partially) intuitive interpretation. Nevertheless there are useful applications of isosurfaces to flow data, e.g., in the visualization of shock waves [87] or burning fronts in simulated combustion data. Furthermore, when scalar clipping is used together with color coding of slices, this naturally combines with isosurfaces as long as isovalue and clipping value coincide.

Röttger et al. present a hardware accelerated volume rendering technique which allows to use multiple (semitransparent) isosurfaces for visualization [71]. Treinish applies isosurfacing to visualise (unsteady) weather data [93]. Weber et al. [99] present crack-free isosurface extraction for adaptive (multiresolution) grids. Laramee and Bergeron provide isosurfaces for super adaptive resolution grids [47].

6. Conclusions and future prospects

A state-of-the-art report must end with an assessment: what has been achieved in flow visualization during the last 15 years? Have the problems been solved? Are the results applied in practice? What are the remaining challenges?

A large number of techniques has been developed and refined. In general, which techniques are the best, depends

strongly on the purpose of the visualization: the research problems addressed, the methods and approaches used, and the personal interest of the researcher or engineer. Users may also have different purposes, such as exploration, detailed analysis, or presentation. Therefore, we believe that a large variety of techniques must be available to allow researchers to choose the most suitable technique for their purpose. In this sense, good progress has been made.

A very successful group is the texture-based techniques (see Sect. 4), mainly used for 2D flows and surface flow fields. They are very suitable for animation, both of stationary and time-dependent flows. Performance limitations seem to be overcome [108], and interactive use with unsteady flows is now feasible. However, the generalization to 3D flow fields as well as to flows on boundaries still is problematic. Techniques based on integration for generating geometries and particle animation (see Sect. 5) are also very successful, and generalise better to 3D fields.

One of the original key problems in flow visualization was the direct visualization of directional structures in a 3D field, possibly varying in time. Despite some heroic attempts, this problem has not been solved, as perceiving three spatial and three data dimensions directly seems a very tough job for the human eye and brain. At the same time, the scale of numerical flow simulations, and thus the size of the resulting datasets, continues to grow rapidly. For these reasons, simplification strategies have to be conceived, such as spatial selection (slicing, regions of interest), data dimension reduction, geometry simplification, and feature extraction.

Slicing in a 3D field reduces the problem to 2D, allowing use of good 2D techniques, but care must be taken with interpretation, as the loss of the third dimension may lead to physically irrelevant results and wrong interpretation. Taking a single 3D time slice from a 3D time-dependent dataset has similar dangers. Other spatial selections such as 3D region-of-interest selection are less risky, but may lead to loss of context. Reduction of data dimension, such as reducing vector quantities to scalars will give more freedom of choice in visualization techniques (such as using volume rendering), but will not lead to much data reduction. Geometry simplification techniques such as polygon mesh decimation, levels-of-detail, or multiresolution techniques will be effective in managing very large datasets and interactive exploration, enabling users to trade accuracy with response time.

Some areas that need additional work are: (a) comparative visualization and multisource comparisons, (b) data analysis visualization of multivariate flow fields with scalar, vector, and tensor data, (c) handling and exploring huge time-dependent flow datasets, (d) the use of virtual environments for visual data exploration and computational steering: problems of performance and 3D interaction, (e) user studies for evaluation, validation, and field testing of flow visualization techniques, and (f) visualization of inaccuracy and uncertainty.

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

Acknowledgements

The authors thank all those who have contributed to this research including AVL (`www.avl.com`), the Austrian governmental research program called K plus (`www.kplus.at`), and the VRVis Research Center (`www.VRVis.at`). Furthermore, the authors thank all the colleagues from FlowVis research who permitted to use the images as shown in the paper.

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”.

References

1. D. Arrowsmith and C. Place. *An Introduction to Dynamical Systems.* Cambridge Univ. Press, 1990. 3
2. G. V. Bancroft, F. J. Merritt, T. C. Plessel, P. Kelaita, R. K. McCabe, and A. Globus. FAST: A multiprocessed environment for visualization of computational fluid dynamics. In *Proc. IEEE Vis. ’90*, pages 14–27, 1990. 11
3. H. Battke, D. Stalling, and H. Hege. Fast line integral convolution for arbitrary surfaces in 3D. In *Visualization and Mathematics*, pages 181–195, 1997. 8
4. B. G. Becker, D. A. Lane, and N. L. Max. Unsteady flow volumes. In *Proc. IEEE Vis. ’95*, 1995. 12
5. J. Becker and M. Rumpf. Visualization of time-dependent velocity fields by texture transport. In *Visualization in Sci. Computing ’98*, pages 91–102. 8
6. E. Boring and A. Pang. Directional flow visualization of vector fields. In *Proc. IEEE Vis. ’96*, pages 389–392, 1996. 7
7. M. Brill, H. Hagen, H.-C. Rodrian, W. Djatschin, and S. V. Klimenko. Streamball techniques for flow visualization. In *Proc. IEEE Vis. ’94*, pages 225–231, 1994. 11
8. S. Bryson and C. Levit. The virtual wind tunnel. *IEEE CGA*, 12(4):25–34, July 1992. 11
9. B. Cabral and C. Leedom. Highly parallel vector visual environments using line integral convolution. In *Proc. of the 7th SIAM Conf. on Parallel Proc. for Sci. Computing*, pages 802–807, Feb. 1995. 7
10. B. Cabral and L. C. Leedom. Imaging vector fields using line integral convolution. In *Computer Graphics (SIGGRAPH ’93 Proceedings)*, volume 27, pages 263–272, Aug. 1993. 7

11. W. Cai and P. A. Heng. Principal stream surfaces. In *Proc. IEEE Vis. '97*, pages 75–80, 1997. 12

12. J. Clyne and J. Dennis. Interactive direct volume rendering of time-varying data. In *Data Visualization '99*, Eurographics, pages 109–120. May 1999. 6

13. R. Crawfis, N. Max, B. Becker, and B. Cabral. Volume rendering of 3D scalar and vector fields at LLNL. In *Proceedings, Supercomputing '93: Portland, Oregon, November 15–19, 1993*, pages 570–576, 1993. 6

14. H. Doleisch and H. Hauser. Smooth Brushing for Focus and Context Visualization of Simulation Data in 3D. In *WSCG 2002, Int'l Conf. in Central Europe on CG, Vis. and Digital Interactive Media*, pages 147–151. 7

15. D. S. Ebert and P. Rheingans. Volume illustration: Nonphotorealistic rendering of volume models. In *IEEE TVCG*, volume 7(3), pages 253–264, 2001. 6

16. D. S. Ebert, R. Yagel, J. Scott, and Y. Kurzion. Volume rendering methods for computational fluid dynamics visualization. In *Proc. IEEE Vis. '94*, pages 232–239, 1994. 6

17. A. J. Fenlon and T. David. An integrated visualization and design toolkit for flexible prosthetic heart valves. In *Proc. IEEE Vis. '00*, pages 453–456, 2000. 6, 10

18. L. K. Forssell. Visualizing flow over curvilinear grid surfaces using line integral convolution. In *Proc. IEEE Vis. '94*, pages 240–247, 1994. 8

19. L. K. Forssell and S. D. Cohen. Using line integral convolution for flow visualization: Curvilinear grids, variable-speed animation, and unsteady flows. *IEEE TVCG*, 1(2):133–141, June 1995. 8

20. T. Frühauf. Raycasting vector fields. In *Proc. IEEE Vis. '96*, pages 115–120, 1996. 6

21. A. L. Fuhrmann and E. Gröller. Real-time techniques for 3D flow visualization. In *Proc. IEEE Vis. '98*, pages 305–312, 1998. 11

22. T. Glau. Exploring instationary fluid flows by interactive volume movies. In *Data Visualization '99*, Eurographics, pages 277–283. May 1999. 6

23. H. Hege and D. Stalling. Fast LIC with piecewise polynomial filter kernels. In *Mathematical Visualization*, pages 295–314. 1998. 7

24. L. Hesselink, F. H. Post, and J. van Wijk. Research issues in vector and tensor field visualization. *IEEE CGA*, 14(2):76–79, Mar. 1994. 1

25. J. P. M. Hultquist. Interactive numerical flow visualization using Stream Surfaces. *Computing Systems in Engineering*, 1(2-4):349–353, 1990. 12

26. J. P. M. Hultquist. Constructing Stream Surfaces in Steady 3D Vector Fields. In *Proc. IEEE Vis. '92*, pages 171–178, 1992. 12

27. V. Interrante and C. Grosch. Strategies for effectively visualizing 3D flow with volume LIC. In *Proc. IEEE Vis. '97*, pages 421–424, 1997. 9

28. V. Interrante and C. Grosch. Visualizing 3D flow. *IEEE CGA*, 18(4):49–53, 1998. 9

29. V. L. Interrante. Illustrating surface shape in volume data via principal direction-driven 3D line integral convolution. In *SIGGRAPH '97*, pages 109–116, 1997. 9

30. B. Jobard, G. Erlebacher, and M. Y. Hussaini. Hardware-accelerated texture advection. In *Proc. IEEE Vis. '00*, pages 155–162, 2000. 8

31. B. Jobard, G. Erlebacher, and M. Y. Hussaini. Tiled hardware accelerated texture advection for unsteady flow visualization. In *Proceedings of Graphicon 2000*, pages 189–196, August 2000. 8

32. B. Jobard, G. Erlebacher, and M. Y. Hussaini. Lagrangian-eulerian advection for unsteady flow visualization. In *IEEE Visualization*, 2001. 8

33. B. Jobard, G. Erlebacher, and M. Y. Hussaini. Lagrangian-eulerian advection of noise and dye textures for unsteady flow visualization. *IEEE TVCG*, 8(3):211–222, 2002. 8

34. B. Jobard and W. Lefer. Multiresolution flow visualization. In *WSCG 2001, Int'l Conf. in Central Europe on CG, Vis. and Digital Interactive Media*. 10

35. B. Jobard and W. Lefer. Creating evenly–spaced streamlines of arbitrary density. In *Proc. 8th EG ViSC Workshop*, volume 7, April 28-30 1997. 10

36. B. Jobard and W. Lefer. The motion map: Efficient computation of steady flow animations. In *Proc. IEEE Vis. '97*, pages 323–328, 1997. 7

37. B. Jobard and W. Lefer. Unsteady flow visualization by animating evenly-spaced streamlines. In *Computer Graphics Proceedings (Eurographics 2000)*, volume 19(3), 2000. 10

38. D. Kao, B. Zhang, K. Kim, and A. Pang. 3d flow visualization using texture advection. In *International Conference on Computer Graphics and Imaging '01*, August 2001. 9

39. D. N. Kenwright and D. A. Lane. Interactive Time-Dependent Particle Tracing Using Tetrahedral Decomposition. *IEEE TVCG*, 2(2):120–129, June 1996. 11

40. L. Khouas, C. Odet, and D. Friboulet. 2D vector field visualization using furlike texture. In *Data Visualization '99. Proc. VisSym'99*, Eurographics, pages 35–44. 1999. 7

41. R. M. Kirby, H. Marmanis, and D. H. Laidlaw. Visualizing multivalued data from 2D incompressible flows using concepts from painting. In *Proc. IEEE Vis. '99*, pages 333–340, 1999. 5

42. M. Kiu and D. C. Banks. Multi-frequency noise for LIC. In *Proc. IEEE Vis. '96*, pages 121–126, 1996. 7

43. R. V. Klassen and S. J. Harrington. Shadowed hedgehogs: A technique for visualizing 2D slices of 3D vector fields. In *Proc. IEEE Vis. '91*, pages 148–153, 1991. 5

44. O. Kreylos, A. Tesdall, B. Hamann, J. Hunter, and K. Joy. Interactive visualization and steering of cfd simulations. In *Proc. of the Joint EG - IEEE TCVG Symp. on Vis. (VisSym '02)*, pages 25–34, May 27–29 2002. 3

45. D. A. Lane. *Scientific Visualization of Large-Scale Unsteady Fluid Flows*, chapter 5, pages 125–145. Scientific Visualization: Overviews, Methodologies, and Techniques. 1997. 4

46. R. S. Laramee. Interactive 3D Flow Visualization Using a Streamrunner. In *CHI 2002, Conference on Human Factors in Computing Systems, Extended Abstracts*, pages 804–805, April 20-25 2002. 11

47. R. S. Laramee and R. D. Bergeron. An Isosurface Continuity Algorithm for Super Adaptive Resolution Data. In *CGI 2002, Computer Graphics International*, July 1-5 2002. (to appear). 12

48. W. C. de Leeuw. Divide and conquer spot noise. In *Proc. of Supercomputing'97 (CD-ROM)*, Nov. 1997. 6

49. W. C. de Leeuw, H.-G. Pagendarm, F. H. Post, and B. Walter. Visual simulation of experimental oil-flow visualization by spot noise images from numerical flow simulation. In *Proc. 6th EG ViSC Workshop*, pages 135–148, 1995. 8

50. W. C. de Leeuw, F. H. Post, and R. W. Vaatstra. Visualization of turbulent flow by spot noise. In *Virtual Environments and Scientific Visualization '96*, pages 286–295. Apr. 1996. 7

51. W. C. de Leeuw and R. van Liere. Comparing LIC and spot noise,. In *Proc. IEEE Vis. '98*, pages 359–366, 1998. 7

52. W. C. de Leeuw and J. van Wijk. Enhanced spot noise for vector field visualization. In *Proc. IEEE Vis. '95*, pages 233–239, Oct. 1995. 7

53. H. Löffelmann. *Visualizing Local Properties and Characteristic Structures of Dynamical Systems.* PhD thesis, Technical University of Vienna, 1998. 5

54. H. Löffelmann and E. Gröller. Enhancing the visualization of characteristic structures in dynamical systems. In *Vis. in Sci. Computing '98*, pages 59–68, 1998. 11

55. H. Löffelmann, L. Mroz, and E. Gröller. Hierarchical streamarrows for the visualization of dynamical systems. In *Vis. in Sci. Comp. '97*, pages 155–164. 12

56. H. Löffelmann, L. Mroz, E. Gröller, and W. Purgathofer. Stream arrows: Enhancing the use of streamsurfaces for the visualization of dynamical systems. *TVC*, 13:359–369, 1997. 12

57. H. Löffelmann, Z. Szalavári, and M. E. Gröller. Local analysis of dynamical systems – concepts and interpretation. In *Proc. of The 4th Int'l Conf. in Central Europe on CG and Vis.*, pages 170–180, 1996. 4

58. X. Mao, Y. Hatanaka, H. Higashida, and A. Imamiya. Image-guided streamline placement on curvilinear grid surfaces. In *Proc. IEEE Vis. '98*, pages 135–142, 1998. 10

59. X. Mao, L. Hong, A. Kaufman, N. Fujita, and M. Kikukawa. Multi-granularity noise for curvilinear grid LIC. In *Graphics Interface '98*, pages 193–200. 8

60. X. Mao, M. Kikukawa, N. Fujita, and A. Imamiya. Line integral convolution for 3D surfaces. In *Vis. in Sci. Computing '97*, pages 57–70, 1997. 8

61. N. Max, B. Becker, and R. Crawfis. Flow volumes for interactive vector field visualization. In *Proc. IEEE Vis. '93*, pages 19–24, 1993. 12

62. P. Moran, C. Henze, D. Ellsworth, S. Bryson, and D. Kenwright. The Field Encapsulation Library (FEL). www.nas.nasa.gov/Groups/VisTech/projects/fel. 6

63. G. M. Nielson. Tools for computing tangent curves for linearly varying vector fields over tetrahedral domains. *IEEE TVCG*, 5(4):360–372, Oct. – Dec. 1999. 11

64. G. M. Nielson, H. Hagen, and H. Müller. *Scientific Visualization: Overviews, Methodologies, and Techniques.* 1997. 9

65. A. Okada and D. L. Kao. Enhanced line integral convolution with flow feature detection. In *SPIE Vol. 3017 Visual Data Exploration and Analysis IV*, pages 206–217, Feb. 1997. 9

66. K. Ono, H. Matsumoto, and R. Himeno. Visualization of thermal flows in an automotive cabin with volume rendering method. In *Data Visualization 2001. Proc. VisSym'01*, pages 301–308, 2001. 6

67. F. Post and T. van Walsum. Fluid flow visualization. In *Focus on Scientific Visualization*, pages 1–40. 1993. 4

68. F. H. Post and J. J. van Wijk. Visual representation of vector fields: Recent developments and research direction. In *Scientific Visualization: Advances and Challenges*, chapter 23, pages 367–390. 1994. 4

69. F. H. Post, B. Vrolijk, H. Hauser, R. S. Laramee, and H. Doleisch. The state of the art in flow visualization. Part 2: Feature extraction and tracking. *Computer Graphics Forum*, 22(2), 2003. 1, 2, 5

70. C. Rezk-Salama, P. Hastreiter, C. Teitzel, and T. Ertl. Interactive exploration of volume line integral convolution based on 3D-texture mapping. In *Proc. IEEE Vis. '99*, pages 233–240, 1999. 9

71. S. Röttger, M. Kraus, and T. Ertl. Hardware-accelerated volume and isosurface rendering based on cell-projection. In *Proc. IEEE Vis. '00*, pages 109–116, 2000. 12

72. I. A. Sadarjoen, A. J. de Boer, F. H. Post, and A. E. Mynett. Particle tracing in σ-transformed grids using tetrahedral 6-decomposition. In *Vis. in Sci. Computing '98*, pages 71–80. 11

73. A. Sanna, B. Montrucchio, and R. Arinaz. Visualizing unsteady flows by adaptive streaklines. In *WSCG 2000 Conference Proceedings*, 2000. 10

74. A. Sanna, B. Montrucchio, and P. Montuschi. A survey on visualization of vector fields by texture-based methods. *Research Developments in Pattern Recognition*, 1(1), 2000. 7

75. G. Scheuermann, H. Burbach, and H. Hagen. Visualizing planar vector fields with normal component using line integral convolution. In *Proc. IEEE Vis. '99*, pages 255–262, 1999. 8

76. W. Schroeder, C. Volpe, and W. Lorensen. The stream polygon: A technique for 3D vector field visualization. In *Proc. IEEE Vis. '91*, pages 126–132, 1991. 11

77. W. J. Schroeder, K. M. Martin, and W. E. Lorensen. *The Visualization Toolkit.* 2nd edition, 1998. 5, 9, 10

78. M. Schulz, F. Reck, W. Bartelheimer, and T. Ertl. Interactive visualization of fluid dynamics simulations in locally refined cartesian grids. In *Proc. IEEE Vis. '99*, pages 413–416, 1999. 6

79. H. Shen and D. L. Kao. A new line integral convolution algorithm for visualizing time-varying flow fields. *IEEE TVCG*, 4(2), Apr. – June 1998. 9

80. H. W. Shen, C. R. Johnson, and K. L. Ma. Visualizing vector fields using line integral convolution and dye advection. In *Proc. Symposium on Volume Visualization*, pages 63–70, 1996. 7

81. H. W. Shen and D. L. Kao. UFLIC: A line integral convolution algorithm for visualizing unsteady flows. In *Proc. IEEE Vis. '97*, pages 317–323, 1997. 9

82. P. Shirley and A. Tuchman. A polygonal approximation to direct scalar volume rendering. In *Computer Graphics (San Diego Workshop on Volume Visualization)*, volume 24, pages 63–70, Nov. 1990. 12

83. D. Silver, F. Post, and I. Sadarjoen. *Flow Visualization*, volume 7 of *Wiley Encyclopedia of Electrical and Electronics Engineering*, pages 640–652. 1999. 4

84. D. Stalling. LIC on surfaces. In *Texture Synthesis with Line Integral Convolution*, pages 51–64, 1997. 8

85. D. Stalling and H. Hege. Fast and resolution independent line integral convolution. In *SIGGRAPH 95 Conference Proceedings*, Annual Conference Series, pages 249–256, August 1995. 7

86. J. E. Swan, M. Lanzagorta, D. Maxwell, E. Kou, J. Uhlmann, W. Anderson, H. Shyu, and W. Smith. A computational steering system for studying microwave interactions with missile bodies. In *Proc. IEEE Vis. '00*, pages 441–444, 2000. 6

87. C. K. Tang and G. G. Medioni. Extremal feature extraction from 3D vector and noisy scalar fields. In *Proc. IEEE Vis. '98*, pages 95–102, 1998. 12

88. C. Teitzel and T. Ertl. New approaches for particle tracing on sparse grids. In *Data Visualization '99*, Eurographics, pages 73–84. May 1999. 11

89. C. Teitzel, R. Grosso, and T. Ertl. Efficient and reliable integration methods for particle tracing in unsteady flows on discrete meshes. In *Visualization in Scientific Computing '97*, Eurographics, pages 31–42, 1997. 11

90. C. Teitzel, R. Grosso, and T. Ertl. Line integral convolution on triangulated surfaces. In *Proc. of the 5th Int'l Conf. in Central Europe on CG and Vis. '97*, number 8, pages 572–581, 1997. 8

91. C. Teitzel, R. Grosso, and T. Ertl. Particle tracing on sparse grids. In *Visualization in Sientific Computing '98*, Eurographics, pages 81–90, 1998. 11

92. A. Telea and J. van Wijk. Simplified representation of vector fields. In *Proc. IEEE Vis. '99*, pages 35–42, 1999. 8

93. L. A. Treinish. Multi-resolution visualization techniques for nested weather models. In *Proc. IEEE Vis. '00*, pages 513–516, 2000. 6, 12

94. G. Turk and D. Banks. Image-guided streamline placement. In *SIGGRAPH 96 Conference Proceedings*, Annual Conference Series, pages 453–460, Aug. 1996. 10

95. S. K. Ueng, C. Sikorski, and K. L. Ma. Efficient Streamline, Streamribbon, and Streamtube Constructions on Unstructured Grids. *IEEE TVCG*, 2(2):100–110, June 1996. 11

96. J. J. van Wijk. Image based flow visualization. In J. Hughes, editor, *SIGGRAPH 2002*, pages 745–754. ACM Press/ACM SIGGRAPH, 2002. 8

97. V. Verma, D. Kao, and A. Pang. PLIC: Bridging the gap between streamlines and LIC. In *Proc. IEEE Vis. '99*, pages 341–348, 1999. 9

98. V. Verma, D. Kao, and A. Pang. A flow-guided streamline seeding strategy. In *Proc. IEEE Vis. '00*, pages 163–170, 2000. 10

99. G. H. Weber, O. Kreylos, T. J. Ligocki, J. M. Shalf, H. Hagen, B. Hamann, and K. I. Joy. Extraction of crack-free isosurfaces from adaptive mesh refinement data. In *Data Visualization 2001. Proc. VisSym'01*, pages 25–34, 2001. 12

100. R. Wegenkittl and E. Gröller. Fast oriented line integral convolution for vector field visualization via the Internet. In *Proc. IEEE Vis. '97*, pages 309–316, 1997. 7

101. R. Wegenkittl, E. Gröller, and W. Purgathofer. Animating flow fields: Rendering of oriented line integral convolution. In *Computer Animation '97 Proceedings*, pages 15–21, June 1997. 7

102. R. Wegenkittl, E. Gröller, and W. Purgathofer. Visualizing the dynamical behavior of Wonderland. *IEEE CGA*, 17(6):71–79, Nov./Dec. 1997. 10

103. R. Westermann. The rendering of unstructured grids revisited. In *Data Visualization 2001. Proc. VisSym'01*, pages 65–74, 2001. 6

104. R. Westermann, C. Johnson, and T. Ertl. Topology-preserving smoothing of vector fields. In *IEEE TVCG*, volume 7(3), pages 222–229, 2001. 12

105. J. J. van Wijk. Spot noise-texture synthesis for data visualization. In *Computer Graphics. Proc. SIGGRAPH 91*, pages 309–318, 1991. 7

106. J. J. van Wijk. Flow visualization with surface particles. *IEEE CGA*, 13(4):18–24, July 1993. 12

107. J. J. van Wijk. Implicit Stream Surfaces. In *Proc. IEEE Vis. '93*, pages 245–252, 1993. 12

108. J. J. van Wijk. Image based flow visualization. In *Proceedings SIGGRAPH 2002*, 2002. 13

109. M. Zöckler, D. Stalling, and H. Hege. Interactive visualization of 3D-vector fields using illuminated streamlines. In *Proc. IEEE Vis. '96*, pages 107–113, 1996. 11, 12

110. M. Zöckler, D. Stalling, and H. Hege. Parallel Line Integral Convolution. *Parallel Computing*, 23(7):975–989, July 1997. 7