Visual representation of vector fields: Recent developments and research directions

Visual Representation of Vector Fields

Recent Developments and Research Directions

Frits H. Post[*]
Jarke J. van Wijk[†]

Contribution to the book of the
ONR Workshop on Data Visualization, Darmstadt, Germany, July 1993,
to be published by Academic Press

September 3, 1993

Abstract

The state of the art in the visual representation of vector and tensor fields is described. Fundamentals and basic techniques are covered, and recent advances in the field are reviewed, such as extensions of volume rendering, vector field topology, streamline and stream surface generation, particle-based techniques, tensor field visualization, and the selection and visualization of salient features of the data. Research issues in vector field and CFD visualization are identified, including interaction, validation, and quality assessment. For a fruitful development of the field, it is argued that integration of knowledge and results from several related domains is desirable.

Introduction

The last decade the need for visualization methods has grown exponentially. The development of more and more powerful supercomputers has enabled researchers in disciplines such as Computational Fluid Dynamics (CFD) to perform increasingly complex three-dimensional simulations using fine grids. Visualization of the gigabytes of data that result became a critical step. This need has been recognized, and in the last five years we have seen a tremendous increase in the number of tools, methods, and techniques.

Here we focus on one application domain: the visualization of three-dimensional fluid flow fields. This is one of the most challenging application domains in scientific visualization. One of the main reasons is that no natural visual representation exists for three-dimensional vector fields. We can readily recognize geometric objects, color and texture, but, unfortunately for

[*]Delft University of Technology, Faculty of Technical Mathematics and Informatics, Julianalaan 132, 2628 BL Delft, The Netherlands.

[†]Netherlands Energy Research Foundation ECN, P.O. Box 1, 1755 ZG Petten, The Netherlands.

CFD researchers, vector fields are harder to grasp. Thus, simplified representations must be conceived that can be understood by the human observer.

Our aim is to review the current state of the art in the presentation of fluid flow fields. One should realize that visual representation is just a part (but an important one!) of the problem. To develop a successful visualization system many other issues such as architectural considerations, user interfacing, data handling and data conversion require much attention as well.

We begin with an overview of the fundamentals, next basic techniques are discussed, followed by a discussion of recent advances. We show that significant progress has been made. However, many issues are also still open for further research. We conclude with a discussion of a number of disciplines that can help visualization researchers to find possible answers to the open questions.

Fundamentals

We first present some background material concerning fluid dynamics and the use of visualization in fluid dynamics research.

The CFD environment

Vector field visualization is historically linked to experimental techniques. Fluid dynamics in particular has a rich history of experimentation and visual representation. Van Dyke [1] shows a collection of excellent examples; extensive overviews are given by Merzkirch [2] and Yang [3]. The relation with computer graphics flow visualization is discussed in Post and Van Walsum [4].

In a numerical simulation in Computational Fluid Dynamics (CFD) a discretized model of a physical flow area is used. A grid must be generated that conforms to the boundaries of the flow area and fits the surfaces of objects in the flow. Also, the density of the grid must be variable: density must be higher in areas where high gradients occur or high accuracy is desired. The geometry and topology of the grid also depends on the solution method. Structured curvilinear grids are most common, but also regular orthogonal grids, or unstructured grids are used. Speray and Kennon [5] give an overview of grid types and grid terminology. The resulting data usually are a velocity vector field and scalar quantities such as pressure, density, and temperature. Flow visualization techniques thus must be suitable for the grid types used, and for combinations of vector and scalar fields.

Let us introduce some terminology from fluid dynamics. A flow can be *steady* or *stationary*, which means that the flow field does not change over time. A steady flow can be described by a single vector field. The data of an *unsteady* or *time-dependent* flow is a time-varying vector field, often represented as a sequence of vector fields: one for each time step. One frame from a time-dependent flow thus looks the same as a steady flow field. Many flow visualization techniques are only suitable for steady flows.

Another important distinction is between laminar or turbulent flow. In a *laminar* flow, the fluid flows in layers, and all fluid elements in a small neighborhood have about the same velocity. In *turbulent* flows, velocities of neighboring elements can have random variations.

In fluid flows, coherent structures can be observed, spatial patterns that exist over a certain time interval. Recognition and description of these structures is an important topic in fluid mechanics. An example of such a structure is a *vortex*, a rotational flow pattern around a central axis, the vortex tube. For identifying vortices, a useful vector quantity is *vorticity*, indicating the rotation of the flow field. Vorticity is defined as the cross product of the velocity vector and its gradient.

The flow visualization pipeline

Data visualization is traditionally conceived as a three-stage pipeline process [6]. It starts with the data generated by a numerical simulation, proceeds through stages of data preparation, visualization mapping, and rendering, and results in a displayed image or animation sequence.

Data preparation consists of a large set of operations, including data input conversions, data selection and enhancement operations, and computation of derived quantities for visualization. We will give only a few examples here.

Filtering and smoothing are sometimes needed to remove irregularities or spurious high frequencies. Data selection is used to focus on interesting areas or details. It can be a simple clipping operation, but also more sophisticated forms of selection can be applied, such as computation of interest criteria or extraction of features or patterns [7]. We will return to this topic later on. Most numerical data fields are defined only at grid points, and therefore interpolation to estimate values in any given position is a common operation. Simple tri-linear interpolation between the eight corner points of a surrounding hexahedral cell can be used for this purpose. But to perform this interpolation, it is necessary to locate the point $\mathbf{x}$ in the field: to find the cell which contains $\mathbf{x}$, and the fractional offsets within the cell. This point location problem is easy to solve in regular rectangular grids, but is less simple in the curvilinear grids often used in CFD.

Transformations between the physical and computational domains, commonly used in CFD simulations, can also be important for visualization, in particular if particle paths are to be generated in computational space.

Other quantities (scalar, vector, or tensor) may be derived from the data to be used for visualization. Examples are the magnitude of velocity or vorticity. Of special importance are the gradients of velocity. The spatial derivatives of velocity define the deformation tensor [8].

Mapping is the crucial step of linking physical quantities to visual primitives and their attributes: shape, size, texture, colour, transparency, etc. It involves choosing a style of visualization, but also setting up transfer functions to define the quantitative relations between physical and visual parameters. Some of the basic visualization mappings for CFD data will be discussed in the next section.

Rendering is the transformation of visual primitives and attributes into displayable images.

If only points, lines, and polygons are used as primitives, rendering is performed by a regular graphics display pipeline, often supported by dedicated hardware. With direct volume rendering [9], or other types of non-standard primitives or attributes, different techniques have to be employed, such as ray casting. For these techniques there usually is little support of graphics hardware, and thus rendering times are much longer. One important problem at this stage is the provision of consistent depth cues for clear representation of three-dimensional shapes and spatial relations. Another problem is the reduction of adverse effects caused by aliasing, in the spatial domain (jaggies, moiré patterns), and especially in the temporal domain (jerky motion, strobing effects).

Basic vector field visualization techniques

The visualization of calculated flow field data already has some history. For a better understanding of the more recent work, we discuss first standard visualization techniques that are nowadays routinely used by CFD-researchers.

Contraction: reduction to scalar quantities

With a first group of visualization techniques the vector data are not visualized directly, but reduced to a scalar field that represents important aspects of it. Examples are vector magnitude, scalar product with a given direction vector, concentration of particles per volume unit, or helicity density (the scalar product of velocity and vorticity). The scalar field can then be visualized using well known volume visualization techniques: iso-surfaces, volume ray casting, or splatting [10].

Arrow plots

Arrows are icons for vectors, and the most obvious way to visualize a velocity field with computer graphics is an arrow plot (sometimes called ’hedgehog’). Base points are given in one or more planes, or throughout the field, and arrows indicate both velocity direction and speed at these points. In this way, a complete view of a vector field can be given. Arrows can give reasonable results in 2D, but there are numerous problems in the 3D case: directional ambiguity, cluttering and a poor spatial effect. As a result, in general arrow plots are not suitable for 3D vector field visualization.

Streamlines, streak lines, path lines, time lines

Field lines are lines that are everywhere tangent to a vector field. In fluid dynamics, the field lines of a velocity vector field are called streamlines. Path lines are trajectories of individual particles released in the flow. Streak lines emerge if particles are released continuously from a single position. For a steady flow, streamlines, path lines, and streak lines are coincident, but in a time-dependent flow they are not [11]. A time line is shown when many particles are released at one instant from points on a straight line.

Particle paths (also called integral curves) are generated by stepwise integration of the velocity field. From a given position $\mathbf{x}(t)$ and velocity $\mathbf{v}(\mathbf{x}(t))$ of a particle at time t, the next position after a time step Δt can be found from:

$$\mathbf{x}(t+\Delta t) = \mathbf{x}(t) + \int_{t}^{t+\Delta t} \mathbf{v}(\mathbf{x}(t))\mathrm{dt}$$

Note that the location of $\mathbf{x}(t)$ in the grid and interpolation of velocities are repeatedly needed. The numerical integration is usually done with a second or fourth order Runge-Kutta method. A path is computed by choosing an initial point $\mathbf{x}(0)$, and applying this procedure repeatedly, using a constant or variable time step. A series of positions is found, and a curve can be fitted through these. The curves can be rendered as a sequence of short line segments. This gives good results in 2D, but in 3D the curves are hard to localize without further depth cues. Also, only a small number of curves can be displayed without confusion.

Stream ribbons and stream surfaces

The concept of a streamline can be readily extended to a surface. If two adjacent stream lines are connected by a maze of polygons, a stream ribbon results [12]. Alternatively, a ribbon can reflect local rotation (vorticity) of the flow by using the vorticity vector to orient the ribbon at each step.

A stream surface will result from interpolation of streamlines originating from a set of adjacent initial points on a line segment (called a rake [13]), or other base curve. The surface is traced out by the base curve as it moves with the flow. If the base curve is a circle, a tubular surface (called a stream tube) is produced. Stream surfaces and ribbons can be rendered with good spatial effects using standard surface rendering techniques with directed light reflection.

Particle motion animation

Particle motion can be effectively visualized using animation. This is perhaps the most realistic visualization of fluid flow. Velocity of particles is shown directly, provided that consecutive particle positions are given at constant time steps, and display update time is constant.

Particle animations can be produced in real time, but this sets severe limits to the number of particles and the choice of rendering techniques. Also, a constant update rate is difficult to achieve, but the advantage is full interactive control of the animation. An alternative is playback animation, where a series of pre-computed frames is shown on a screen. The update rate is not affected by the complexity of the image, but interactive control is impossible. With a steady flow, cyclic animation can be used [14], showing smooth motion by continuously repeated display of a small number of frames. With time-dependent flow, only open-loop animation is possible, and frames are computed from different velocity fields for each time step. Particles can be simply rendered as points or small line segments, but in that case particles are hard to localize in space, and aliasing artefacts will occur. Therefore, it is desirable to add extra depth cues and deal with aliasing; some recent developments in particle rendering will be described in the next section.

Recent developments

The last few years we have seen an explosion of new visualization techniques. Here we give an overview. We consider the following areas: volume rendering, vector field topology, streamlines and stream surfaces, tensor fields, particles, and feature visualization.

Extensions of volume rendering

Although volume rendering [9] is most suitable for visualization of scalar fields, it has been applied in several ways to visualize flow. In flow data, scalar quantities such as pressure or temperature are often associated with velocity vector fields. Also, many relevant scalar quantities can be calculated from vector fields. Examples are velocity magnitude, the scalar product of the local velocity vector and a given direction vector, or concentration of particles. These scalar fields can be visualized using well-known volume rendering techniques such as ray casting with opacity mapping, iso-surfaces, and gradient shading. But the directional information inherent in a velocity vector field is not directly presented in a static image.

One obvious way to improve this is animation of volume rendering: the directional effect is implied by changing optical densities, shown as moving or fluctuating semi-transparent effects, suggesting smoke, fog, or clouds. We will give three examples here.

Sakas [15] has animated turbulent gaseous motion in this way. He used spectral synthesis to generate stochastic, time and space variant density fields or volume density textures [16]. These fields were rendered using volume ray casting. Realistic turbulent visual effects were achieved by light propagation modeling, and using careful sampling and filtering techniques.

Another example of animated volume rendering for flow visualization is virtual smoke [17]. This method works for steady or instantaneous 3D vector fields. An interactively specified seed point in the field is used to construct an initial control volume around it. The control volume consists of volume cells, and develops in space by examining its neighboring cells, and propagating depending on direction and magnitude of the local velocity field. The control volume is rendered semi-transparently using volume ray casting; in animations this has the effect of simulating the propagation of a smoke tracer in a flow field.

A third example of animated volume rendering is the use of 3D textures [18]. In a visualization of a global climate model, iso-surfaces of cloud density are shown, and wind velocities are shown on these surfaces using solid texture mapping. The texture coordinates are advected with the wind field. The texture function is sampled at the grid points, which are moved forward along streamlines of the field. The texture can move coherently along with the deforming clouds, or can move across the clouds indicating local wind direction.

Another method to extend volume rendering for vector fields is the use of anisotropic volume textures [19]. A directional effect is achieved in direct volume rendering by using a splatting technique [10] with a vector kernel filter, which is moved across the image. The filter displays small line segments with varying color and opacity, and with the orientation determined by the vector field. This results in a fine-grained directional texture, showing the structure of a vector field in a still image.

A clear advantage of these extended volume visualization methods is that they can be easily used for combined visualization of scalar and vector fields in the same image. A problem, in particular with animation of scalar fields is that any moving field strongly suggests a flow field, even when the actual motion of the fluid medium is not shown. For instance, a time dependent pressure field can be mistaken for a cloud of moving fluid particles.

Vector field topology

Analysis and visualization of vector field and flow topology have been investigated since 1987. In a number of papers, Helman and Hesselink [20, 21, 22] describe the development of a system to visualize flow topology. A general classification of flow fields is given by Chong et al. [23]. More recently, similar techniques to those of Helman and Hesselink were described. Dickinson [24] addresses the interactive aspect of flow topology visualization, and Globus et al. [25] give detailed information on the implementation of a flow topology visualization module.

Flow topology analysis is based on critical point theory, which has been used widely to examine solution trajectories of ordinary differential equations. The topology of a vector field consists of critical points (where the velocity vector is zero) and integral curves and surfaces connecting these critical points. Images of a vector field topology display the topological structure of a vector field, without displaying much redundant information. Analysis and visualization of vector field topology proceeds in three steps. First, the locations of the critical points are calculated; second, these critical points are classified; third, integral curves and surfaces are calculated.

It should be noted that, although the techniques are described for velocity fields, they can be applied to any other vector field.

The locations of the critical points can be found by searching all cells in the flow field. Critical points can only occur in cells where all components of the vector pass through zero. The exact position of a critical point can be calculated by interpolation in case of a rectangular grid. In case of a curvilinear grid, the position of a critical point can be calculated by recursively subdividing the cell, and using a numerical method such as Newton iteration [25].

Once the critical points have been found, they are classified. This is done by approximating the velocity field in the neighborhood of the critical point with a first order Taylor expansion. Because the velocity in a critical point is zero, the velocity field in the neighborhood of a critical point is fully determined by the partial derivatives $\nabla\mathbf{u}$. The critical points can be classified according to the eigenvalues and eigenvectors of $\nabla\mathbf{u}$. Positive eigenvalues correspond to velocities away from the critical point (repelling nodes), and negative eigenvalues correspond to velocities towards the critical point (attracting nodes). Complex eigenvalues result in a focus; if the real part is non-zero, a spiral occurs, and if the real part is zero, concentric ellipses occur. If both negative and positive real values exist at a critical point, the critical point is a saddle.

The classified critical points can be used as starting points for integral curves (*particle paths*), and the eigenvectors can be used as starting directions. This means that the starting point is a point on an eigenvector, very close to the critical point. The end points of the integral curves are again critical points, or points on the boundary of the flow domain. Because of numerical errors, some integral curves might ’miss’ a critical point, or might enter an object in the flow. The first problem can be solved by attaching an integral curve to a critical point when it comes very close. To solve the second problem, integral curves that enter an object can be restricted to follow the surface of the object.

In two dimensions, critical points and integral curves completely describe the topology of the flow field. In three dimensions, integral surfaces must also be calculated. Helman and Hesselink [22] describe a way to create some of these integral surfaces. When visualizing the flow around a hemisphere cylinder, they first calculate the topology of the vector field on the surface of the cylinder. This is, in fact, a two-dimensional topology. Points on the (two-dimensional) integral curves of this topology are used as starting points for a number of three-dimensional integral curves. By tessellating the space between two adjacent integral curves, integral surfaces can be constructed. In this way surfaces of separation and reattachment can be visualized.

Streamlines and stream surfaces

The standard technique for the calculation of a streamline $\mathbf{r}(t)$ is based on the use of standard integration techniques, such as first order Euler, or second or fourth order Runge-Kutta. However, the use of low order integration techniques, large step sizes, and simple trilinear interpolation lead to unacceptable results, while higher order techniques, small or variable step sizes and cubic spline interpolation are computationally expensive. Kenwright and Mallinson [26] have developed an ingeneous technique for stream line calculation that is both exact and efficient. They consider a streamline as the line of intersection between two stream surfaces. A stream surface is a surface that is everywhere tangent to the flow. Here these surfaces are represented as implicit surfaces $f(\mathbf{x}) = f_1$ and $g(\mathbf{x}) = g_1$. For each cell that the streamline crosses, the stream functions f and g are determined. They are represented in a so-called f-g diagram. Each grid point has an associated value of f and g, so the cell can be mapped to this 2D diagram. The faces of the cell are quadrilaterals. Next the image of the entry point of the streamline into the cell is located in the f-g diagram, and the two faces of the cell which enclose it (inlet and exit face) are determined. Finally the exit coordinates are determined via an inverse transformation.

In contrast to conventional integration techniques, this technique is mass conservative and does not use time stepping. The authors claim that this offers a considerable speed advantage.

The display of stream surfaces themselves is a useful visualization technique. Images of stream surfaces are much easier to understand than images of a large number of streamlines, because shading can be used to convey more information on the three-dimensional structure of the flow.

The most obvious way to construct a stream surface is to view it as a parametric surface $\mathbf{r}(s, t)$, where s is a parameter along the originating curve $\mathbf{r}(s, 0)$, and t is a parameter along the streamlines that make up the surface. Curves $\mathbf{r}(C, t)$ are streamlines, curves $\mathbf{r}(s, C)$ are time lines. To construct the surface, first the originating curve is discretized into points $\mathbf{r}_{i0}$, and next discrete approximations of the streamlines for each originating point are calculated. This gives a set of points $\mathbf{r}_{ij}$, from which a polygonal mesh can be derived that approximates the stream surface $\mathbf{r}(s, t)$.

Unfortunately, this simple approach often gives a poor result. Convergent flow, divergent flow, shear flow, and especially the flow around objects all lead to badly shaped meshes. The density of the mesh varies from too sparse to too dense, and the polygons that make up the mesh are elongated and skinny. Increasing the sampling density in s and t direction is not efficient, and only partially solves the problems.

Various solutions have been proposed to construct better behaved stream surfaces. One solution is to insert new streamlines over the full length of the stream surface if two adjacent streamlines diverge [20, 21]. Another solution [27] is to compute streamlines, resample them, and to remove extraneous points with a filtering step.

The advancing front method of Hultquist [13] is a more efficient method for the construction of stream surfaces than those based on streamlines. The originating curve is discretized, and advanced downstream. The spacing and the number of particles at the front is adjusted to maintain the same distance between adjacent particles. The advancement step of the particles is chosen such that the advancing front is locally orthogonal to the streamlines. If two adjacent particles are headed in opposite directions, a critical point is assumed to be in between and the surface is split.

Another approach is to approximate stream surfaces with surface particles, which we discuss in the section on particles.

Yet another approach is described by Van Wijk [28]. Here the stream surface is modeled as an implicit surface $f(\mathbf{x}) = C$. In other words, the stream surface is considered as an iso-surface of the stream function f. The values of f at inflow boundaries are specified by the user, next the values of f are calculated simultaneously for all remaining grid points. This in contrast to Kenwright and Mallinson [26], who calculate the values of f per cell. Finally, a family of stream surfaces can be generated efficiently by varying C. For the calculation of f two techniques are proposed: based on the solution of the convection equation and based on backward tracing. The flow around objects poses special problems for which some solutions are discussed.

Compared with Hultquist's method, the method of implicit stream surfaces is not efficient, and sensitive to sampling artefacts, because it is based on a grid. However, irregular topologies, both of the originating curve as well as of the resulting surfaces, do not pose special problems. As an example, figure 1 shows stream surfaces that visualize the heat-induced convective air

Figure 1: Airflow visualized with implicit stream surfaces. (Data courtesy C.J.M. Lasance, Philips CFT.)

flow in a tv-set.

Tensor fields

The visualization of tensor fields is a difficult problem. A three-dimensional tensor field consists of nine scalar functions of position. These functions can be shown independently, but this is meaningless. The values of a real tensor can be interpreted in terms of the distortion of an infinitesimal element. Decomposition is an important technique to achieve more insight in this distortion. An often used decomposition is to split tensors in a symmetrical and an anti-symmetrical part. The symmetrical part is called the stress-strain tensor. The eigenvectors are called the principal directions of the tensor, and along these vectors the deformation is pure stretching. The axis with the largest stretching is called the major principal direction. The anti-symmetrical part of the tensor depicts the rotation of the element.

Methods for the visualization of tensor fields can be roughly divided into two classes. The first class of methods [29, 30, 8] visualize a tensor at a single point as an icon or glyph. The aim is to show as much information of the tensor as possible. Multiple instances of the icon are used to visualize the field, and the user can interactively change the selected position. Haber [29] represents a symmetric stress tensor by an object consisting of a shaft and a disc. The direction of the shaft is the direction of the major principal direction, the axes of the elliptical disk are aligned with the other principal directions. Geiben and Rumpf [30] visualize tensor data via the indicatrix surface.

De Leeuw and Van Wijk [8] developed a probe for flow fields. The tensor to be visualized is the Jacobian of the velocity field. The components of the Jacobian are the spatial derivatives of the velocity components. This tensor is first decomposed into components parallel and perpendicular to the flow, and next into five physically meaningful components. These

Figure 2: A probe for local flow field visualization.

components are visualized as geometric objects with an intuitively meaningful interpretation. A curved arrow with candy-stripes depicts velocity, curvature, and rotation, a half ellipsoid depicts acceleration, a ring depicts shear, and a paraboloid depicts convergence and/or divergence (see figure 2).

The second class of methods visualize tensor data along characteristic lines. The aim here is to show the structure of the tensor field. Dickinson [31, 32] introduced the concept of tensor field lines, i.e. lines that are everywhere tangent to one of the eigenvectors. Delmarcelle and Hesselink [33] generalize this idea in order to represent more of the tensor data along these trajectories. For the visualization of real, symmetric tensor fields a geometric primitive is swept along the tensor field line. Two types are used: an ellipse and a cross. In both cases the axes align with the principal directions and their length is proportional to the magnitude of the corresponding eigenvalues. The ellipse sweeps out a tube, the cross sweeps out a helix. The color of the surface is a function of the three eigenvalues, generally the amplitude of the longitudinal eigenvalue. Further, some suggestions are given for the visualization of collective behavior of hyperstreamlines. Topology extraction yields an objective and unique representation of vector fields. Much more research remains to be done before similar representations can be obtained of tensor fields.

The *stream polygon* of Schroeder et al. [34] is a special case, in the sense that it fits into both classes of methods. It is used to visualize local deformation, such as occurs for instance in velocity fields. The stream polygon is a regular, n-sided polygon, oriented normal to the local velocity vector. This geometric primitive can be used in several ways. The polygon can be deformed according to local strain and rotation. Also, the local strain can be projected onto the plane perpendicular to the local vector. The polygon can be rotated according to the projection of the vorticity onto the local vector. The radius of the stream polygon can vary as a function of some property of the field. Further, if the stream polygon is swept along a

Figure 3: A combination of different particle sources.

streamline a streamtube is produced. This streamtube has n sides. The color of each side can be set according to a specified scalar function. If used in combination with the techniques noted previously, a large amount of information can be represented.

Particles

The use of particles is well-known in computer graphics. Recently several variations of the concept were developed for visualization purposes.

Moving particles give a good indication of the direction and magnitude of the velocity. Streamlines and stream surfaces visualize the structure of the flow field. Stolk and Van Wijk [35, 36] have introduced a technique that shares those advantages. *Surface particles* integrate particle and surface based approaches: a particle is a small part of a flat surface. Due to its small size, its shape is irrelevant, and hence the surface particle can be described as a point with a normal. The path of the particle and the normal of surface particles are calculated by integration. Surface particles can be used to visualize flow in several ways by variation of the properties of the particle sources. For instance, a line source gives a stream surface, a circle source gives a stream tube, provided that these sources continuously emit particles (figure 3). The surfaces that result are not explicitly calculated, but emerge from the collective behavior of many separate particles. This technique is less efficient for the generation of stream surfaces than e.g. Hultquist's advancing front technique. On the other hand, the moving texture that results gives a strong additional cue for the direction and magnitude of the velocity.

For shading a correction is applied for the distance and the orientation of the particles [36]. The point-shaped particles are pre-filtered with Gaussian filters to prevent spatial and temporal artefacts. Various optimizations can be used to speed up the scan-conversion. Two versions of dealing with occlusion are discussed. First, one can assume that particles are so small that they do not overlap and mask each other, and that a dark background is used. Thus, all

Figure 4: Airflow visualized with surface particles. (Data courtesy C.J.M. Lasance, Philips CFT.)

intensities can just be added. However, this technique falls short if the particles must be used in combination with surfaces. Further, opaque or semi-transparent particles provide a strong, additional depth. Therefore, a second technique is described. First the surfaces are rendered and the results are stored in a Z-buffer. Next, the particles are sorted back to front, and finally they are scan-converted. During the scan-conversion the Z-buffer is used to test if the particles are hidden by surfaces or not. If not, the intensities of the particles are merged with the image. For the opacity a Gaussian mask is used. This technique gives satisfactory results, but it is not perfectly sound. Implicitly the particles are considered as Gaussian clouds instead of as moving points. Further, the use of depth of field as a tool for selective data-display is discussed. In figure 4 the same data set is visualized as in figure 1, but here surface particles are used.

Instead of cyclic display of a set of pre-calculated images, other techniques can be used to show animations on low-cost workstations. Van Gelder and Wilhelms [37] use hardware color table interpolation and color table animation. The paths of particles over a short time-interval are rendered using successive entries in the color look-up table. A narrow black border and Z-buffering are used to obtain a cue for depth.

Sims [38] treats the rendering of particles in a general and flexible way. Each particle has a head and a tail. Both have position, radius, color, and opacity as attributes. The shape of a moving particle consists of two circles, connected with tangent lines. All parameters are interpolated linearly from head to tail, and further, the opacity falls off from 1.0 at the center to 0.0 at the edges, according to a linear or Gaussian function. The implementation of particle animation and rendering on a massively parallel computer is discussed.

If many particles are used, the individual particles cannot be discerned any more, and texture is perceived instead. The concept of *spot noise* [39] provides more insight in this effect. The aim here was to use texture for data visualization. The requirements are, first, the

Figure 5: Turbulent channel flow visualized with particle path lines.

properties of the texture can be parameterized by the data, second, the scientist can specify this relation and other properties of the texture easily, third, the resulting texture clearly expresses the underlying data. The proposed solution is to generate texture by adding a large number of randomly positioned and randomly scaled spots. The properties of the spot directly control the properties of the texture: they have the same autocorrelation. Therefore, by modifying the shape of the spot as a function of the data, the data are visualized by texture. Several examples are given.

In the previous examples the particles were convected by the flow field. The work of Hin and Post [40] shows that variations on this strategy can provide more insight in the data. Here particles are used for visualization of three-dimensional turbulent flow fields. The method is based on Reynolds decomposition of a turbulent flow field into a convective and a turbulent motion. The convective motion is described by a mean velocity field, and the turbulent motion by an additional eddy-diffusivity field. This gives a measure for the amount of random motion to be superimposed on the convective motion. At each step of particle path generation a stochastic perturbation is added, dependent on local eddy-diffusivity. This results in random-walk motions of particles. The collective behavior of sets of particles provides a good visual cue for turbulence. Figure 5 shows an example of a turbulent channel flow. Path lines are shown of particles released from two sources at the inlet. It is clearly visible that perturbations are much smaller near the walls of the channel. The colors of the path lines indicate eddy-diffusivity.

Another example of non-standard use of particles is given by Szeliski and Tonnesen [41]. Here oriented particles (similar to the surface-particles described before) were used, not for flow visualization, but for surface modeling. The motion of the particles is governed by Newtonian dynamics. Particles attract each other at a long range, and repulse each other at a short range. By prescribing non-isotropic potentials the particles can be forced to align in smooth surfaces.

Feature visualization

One very promising line of research is what we propose to call *feature visualization*. Instead of direct visualization of raw data, algorithmic techniques are employed to extract meaningful patterns, structures, or objects. Interesting parts of the data can be selected for closer examination or more emphasis in visualization. The main features of a flow field can be schematically or symbolically visualized, while retaining quantitative accuracy. In this way an abstract visual representation can be achieved, which can replace the original data. Such a representation can have a much higher information content, enabling the user to ignore or discard most redundant or uninteresting data. This approach can reduce complexity, and also help to reduce the need to manipulate large data sets in interactive visualization.

The flow topology analysis of Helman and Hesselink [20, 22] is an early recognition of the importance of this idea, applied to vector fields. Their technique is an excellent seminal example of the potential of feature visualization. Vector field topology and critical point theory provide a sound theoretical basis for their method. It is generic in the sense that it is not restricted to fluid dynamics, but can be used for different types of vector fields. A variety of physical flow phenomena can be studied with it, such as separation topology.

There are several other examples of feature visualization. Silver and Zabusky [42, 43, 44] have applied methods from computer vision, image processing, and mathematical morphology for extraction and time tracking of coherent structures from flow data sets. The goal of this approach is to develop methods for automatic searching, identification, and classification of interacting amorphous structures in vector fields. The methods have been applied to study causal effects in vector fields, and vortex dynamics.

To study causal effects [42], ellipsoids were fitted around regions of high velocity or vorticity magnitude, showing the relationship between different vector quantities evolving in time. For vortex dynamics [43], ellipsoid fitting and skeletal extraction of vorticity tubes was used. Although the primary area of application is fluid dynamics, it is claimed that the methods are generic for other disciplines with similar data sets.

Vortex dynamics also is the subject of a study by Villasenor and Vincent [45]. They used morphological techniques for extracting the skeleton of a vortex tube from 3D time dependent vector fields. The vortex tubes are tracked in time using a spiral search technique.

Pagendarm [46] has developed an algorithm for detection and localization of discontinuities, and applied it to shock waves. A simple integral expression is used to detect high gradients in scalar fields. Shock waves are determined by finding local gradient maxima in pressure fields, and these areas can be visualized as iso-surfaces.

A different strategy is employed by Van Walsum [7]. He did not extract topological or morphological structures, but developed a data selection technique to algorithmically identify ’interesting’ areas in a data set. Grid points are selected in a vector field that satisfy a user specified criterion of interest, formulated as a logical expression consisting of conditions on data values. These may be source data, but also derived data calculated from the source data, including gradient functions such as partial derivatives, divergence, or rotation. The expression

Figure 6: Data selection using a criterion of interest, and streamlines passing through the selected area. (Data courtesy A. Segal, Delft University, Applied Mathematics)

is evaluated for each grid point, yielding a selection of grid points. Subsets of the data obtained in this way can be used to focus the attention on important features of the data, and highlight these features in visualization. Figure 6 shows the use of this technique in a 'backward facing step' geometry. As a criterion if interest, a high value of (normalized) helicity density is specified, indicating a spiralling flow pattern. Selected grid nodes satisfying this criterion are shown in the recirculation zone behind the step. Streamlines are tracked in both directions through seed points in this area, clearly showing the characteristic flow pattern.

Although not strictly in this category, there are also examples of visualization where features are not extracted algorithmically in a pre-processing stage, but the visualization mapping is chosen such that the features emerge in the images. Comte [47] studied coherent structures in turbulent flows. Hairpin vortices were visualized as iso-surfaces in vorticity magnitude fields. In a similar way, Wilhelms et al. [48] have visualized hairpin vortex structures by applying gradient shading to a scalar field of a single velocity component.

Feature visualization is concerned with the content of the data, and thus the techniques are always liable to be application-specific. In visualization research, generic techniques must be developed, allowing the users to specify their features or selection criteria according to the specific application area and the purpose of analysis. It is too early to see to what extent this is possible, but a wider application of techniques from image processing and mathematical morphology, and from the mathematics of vector and tensor fields may be very fruitful.

Research Issues

The results described in the previous section show that flow visualization has gained much attention and that significant progress has been made. Nevertheless, it is our opinion that the subject has not been exhausted by far. This holds for nearly all topics discussed, but especially for feature visualization and for tensor visualization. Both are complex problems, for which a single solution probably does not exist.

In this section we present some other areas where new ideas, methods, and techniques would be welcomed.

Interaction

So far we focused on presentation issues. For the visual exploration of data however, the interaction of the user with the visualization system is almost equally important. Our opinion is that for an optimal result standard interaction methods (such as sliders, buttons, and other widgets) fall short, and that in addition interaction methods especially developed for flow visualization are required.

An excellent example in this respect is the Virtual Windtunnel, developed by Bryson [11]. Here virtual reality techniques are used and customized for flow visualization. The user can inspect the flow data around him by pointing and gesturing. In this way for instance streamlines can be generated.

The extraction of interesting features and the selection of interesting areas in the data is an important aspect of visualization, and often a very time-consuming process. In the ideal case, a system should be able to detect interesting features automatically, and present them clearly to the user. However, this requires a highly intelligent system that understands some high-level specification of the user ("is this simulation stable?") and is able to transform this into simpler rules. The custom approach now is that the user explicitly specifies which data must be visualized and also in what way ("show an iso-surface $f = 5.0$"). We think that tomorrow's visualization system should have more knowledge of the problem domain, and should not only be a slave that produces pictures, but also a sensible assistant.

CFD simulations are typical examples of compute-intensive jobs, even for today's most powerful supercomputers. A run often takes several hours to several days. Computational steering, in the sense of interactive change of input parameters and immediate visualization of the result, is therefore only feasible for small simulations at low resolution. In the future, however, we can expect further progress in the performance of the computing hardware, and then we expect that computational steering will provide new perspectives. We can imagine for instance that a designer modifies the geometry of a car, and continuously sees the aerodynamical effects.

A more down-to-earth issue, which is closely related to interaction, is efficiency. For smooth interaction a high throughput is essential. Thus, more efficient algorithms for, for instance, particle tracing but also volume rendering, are important. Other solutions are the use of parallel computing, and the use of previews, either of simplified data or using simplified visualizations.

Validation

Another important research issue concerns the comparison between experimental and numerical data. Visualization can play a role in the experimental validation of numerical techniques by presenting results of numerical simulation and experiments in the same format, thus allowing a side-by-side or superimposed comparison [49]. For a more than very global and qualitative indication of similarity, this would require a close compatibility between the two visualizations, that can be achieved in two ways.

First, one can attempt to simulate experimental visualization techniques as realistically as possible. The experimental results can thus be presented by digitized photograph or video, and the comparison is made on the level of images. This form imposes severe restrictions on the visualization techniques, both experimental and computer graphical.

Second, one can attempt to represent the data produced by experiment and simulation in a compatible format. Visualization is done by the same computer-based technique, and comparison is possible on many levels, qualitative as well as quantitative. However, the problems to achieve a high compatibility are severe. The use of advanced techniques in velocimetry, and digital image processing to extract quantitative data from experimentally obtained imagery are primary conditions. From the numerical simulation, data generation facilities are quite flexible. But even simple adaptations such as grid resampling may introduce new errors.

We think that comparisons only on the image level are not satisfactory, because all semantics must be supplied by the user, and important similarities or differences are not always tied to pixels. A better approach would be to use feature-based visualization, allowing direct comparison of essential features, which are less dependent on position and orientation, and which can contain the essential semantics of the flow. For example, the occurrence of vortices in a flow field, or a particular separation topology give much better criteria for similarity than techniques based on pattern extraction from images.

A related issue is the development of indicators of accuracy and reliability in visualization. In most experimental sciences, it is traditional practice to document this, but for computer generated visualizations this practice is not yet followed. Possibly, these indicators should be an integrated part of the visual presentation. The sources of error or unstability can be indicated through the whole visualization process, from measurement or numerical simulation to the visualization process itself. A paper by Buning [50] is a rare example of an error analysis in flow visualization.

Evaluation

In addition to validation of the numerical simulation, also the visualization techniques themselves must be validated. We think this issue has not yet received due attention from the visualization community, though the session on ’how to lie and confuse with visualization’ at the Visualization ’92 conference was a first step in the right direction.

Computer visualization has a high pretence of insight and communication. Though the results of visualization are often impressive, the main effect of the images is credibility and persuasiveness, rather than quantitative fidelity. From a quantitatively correct visualization a user should be able to read about ordering and structure, but also about trend (increase and decrease), and about rate of change (is it linear? is it more or less than linear?). In case of 2D visualization, this goal can be achieved by relatively simple means, though there are notorious pitfalls, such as the order and non-linearity of the spectral color scale.

In case of 3D or even higher dimensional visualization, this is much more difficult. For example, in case of a 3D scalar field, would a viewer be able to tell whether it varies linearly in a given area? Or, indeed, from a semi-transparent rendering of a constant field in a box-shaped area, could the viewer tell that the field is constant?

In the area of vector and tensor fields, our understanding of quantitatively correct visualization is even less. We know that a 3D direction vector cannot be unambiguously represented by an arrow in a 2D projection plane. But how can complex 3D directional structures be visualized correctly and effectively? And can the velocities of moving particles be visualized and judged correctly? Can we faithfully show the rates of change represented by the deformation tensor?

Currently we do not know the limits to our capacity to perceive spatial and temporal variance quantitatively, and it is therefore important to realize that visualization today is mainly qualitative. We can show objects and structures, shapes and relative positions and motions in space, and comparative attributes. Better quantitative visualizations would require more insight in the way visual imagery is processed. The problem is not that we deliberately deceive with visualization, but that we do not really know how to be faithful to the data.

It is highly desirable that metrics be devised of correctness, validity, and accuracy for visualizations, and also visual means to indicate error margins, uncertainties, and other limitations, to be communicated directly with a visual representation. The users then cannot escape the dilemmas caused by the imperfections inherent in modeling, measurement, and visualization. Visualization thus can become less liable to abuse in science and politics.

In conclusion ...

In the preceding sections we showed that much work has been done, but also that many problems still remain. How to continue?

The general answer is in the creative integration of knowledge from several scientific domains. Visualization is an interdisciplinary subject, and for further progress more knowledge from related domains should be brought in. A non-exhaustive overview of such domains follows.

Mathematics provides a wide range for inspiration. Partial differential equations, differential geometry, geometric design, and topology have been studied in great depth. The transposition of this knowledge is hard for the non-initiated, but can lead to really innovative visualization techniques, as Helman and Hesselink have shown.

Fluid Dynamics and CFD: the user of a flow visualization package will typically be a fluid dynamics expert. Trivial, but sometimes forgotten by over-enthusiastic graphics people: it is the user who determines the requirements. Also, we expect that using knowledge from fluid dynamics can lead to new visualization techniques. As an example: the implicit stream surfaces technique of Van Wijk [28] can be described as the use of CFD-simulations for visualization purposes. Generally the demarcation between fluid dynamics and visualization research tends to be blurred. CFD researchers are increasingly active in visualization, computing new objects and quantities for visualization; visualization experts use more CFD knowledge and are more concerned with CFD-like activities such as computing particle traces and stream surfaces.

Experimental flow visualization is important for two reasons. First, to allow for comparison of the results of experiments and simulations it is helpful if the computer visualization mimics the experimental visualization. Second, the results of experimental visualization are often impressive with respect to the clarity of the presentation.

All visualizations must pass through the human visual system. This is an incredibly powerful parallel image processing device, however, with its own limits and pecularities. The issue of perception is studied in *psychophysics*. Neglecting this knowledge can lead to poor results, and even severe errors. Topics such as the use of color have been studied in depth, and from this we can learn for instance that the rainbow color scale, which is very popular in visualization, is often not a good choice.

Graphics design has a rich tradition in the graphical presentation of numerical data [51]. The main focus here is on the presentation of tabular data, but the basic guidelines can possibly be transposed.

Also in *animation*, considerable experience exists in the expression of motion. Cartoonists have developed a visual language to show motions such as translations and rotation, which is easy to understand.

From *image synthesis* we can learn how to generate realistic images. The subtle effects of light in interaction with surfaces and fluids have been studied in depth. Realism as such does not guarantee a meaningful result. The aim of visualization is insight, not a pretty picture. On the other hand, if many data have to be shown simultaneously with different techniques, every depth cue is welcome. And our visual system is trained in the interpretation of realistic images.

We can consider the results of simulations as samples of a continuum, just like bitmaps are samples of a continuous image. Therefore, if it comes to the enhancement of data and the extraction of features, we probably can learn a lot from *digital image processing* and *mathematical morphology*. Rules for what to enhance and extract should come from other disciplines, but the basic techniques to do this are available.

With these hints for sources of inspiration, and the problem areas described before, we hope to have provided stimuli for future research.

References

[1] M. van Dyke. *An Album of Fluid Motion*. The Parabolic Press, 1982.

[2] W. Merzkirch. *Flow Visualisation, second edition*. Academic Press Inc., New York, 1987.

[3] W.J. Yang, editor. *Handbook of Flow Visualization*. Hemisphere Publishing Corporation, 1989.

[4] F.H. Post and T. van Walsum. Fluid flow visualization. In H. Hagen, H. Mueller, and G.M. Nielson, editors, *Focus on Scientific Visualization*. Springer Verlag, Berlin, 1993.

[5] D. Speray and S. Kennon. Volume probes: Interactive data exploration on arbitrary grids. *Computer Graphics*, 24(5):5–12, 1990. Proceedings San Diego Workshop on Volume Visualization.

[6] R.B. Haber. Visualization techniques for engineering mechanics. *Computing Systems in Engineering*, 1(1):37–50, 1990.

[7] T. van Walsum. Content-based grid node selection for vector field visualization. In *Proceedings of the Fourth Eurographics Workshop on Visualization in Scientific Computing*, Abingdon, UK, April 1993.

[8] W.C. de Leeuw and J.J. van Wijk. A probe for local flow field viusalization. In *Proceedings Visualization '93*. IEEE Computer Society Press, Los Alamitos, CA, 1993.

[9] A. Kaufman, editor. *Volume Visualization*. IEEE Computer Society Press, Los Alamitos, CA, 1990.

[10] L. Westover. Footprint evaluation for volume rendering. *Computer Graphics*, 24(4):367–376, Aug 1990.

[11] S. Bryson and C. Levit. The virtual windtunnel: An environment for the exploration of three-dimensional unsteady flows. In *Proceedings Visualization '91*, pages 17–24. IEEE Computer Society Press, Los Alamitos, CA, 1991.

[12] G. Volpe. Streamlines and streamribbons in aerodynamics. In *27th Aerospace Sciences Meeting*, Reno, NV, January 1989. AIAA Paper 89-0140.

[13] J.P.M. Hultquist. Constructing stream surfaces in steady 3-d vector fields. In *Proceedings Visualization '92*, pages 171–178. IEEE Computer Society Press, Los Alamitos, CA, 1992.

[14] J.J. van Wijk. A raster graphics approach to flow visualization. In C.E. Vandoni and D.A. Duce, editors, *Proceedings Eurographics '90*, pages 251–259, 1990.

[15] G. Sakas. Modeling and animating 3-d turbulence using spectral synthesis. *The Visual Computer*, 9(4):200–212, January 1993.

[16] G. Sakas and M. Gerth. Sampling and anti-aliasing discrete 3-d volume density textures. In F.H. Post and W. Barth, editors, *Proceedings Eurographics '91*, pages 87–102, 1991.

[17] K.-L. Ma and P.J. Smith. Virtual smoke: An interactive 3d flow visualization technique. In *Proceedings Visualization '92*, pages 46–53. IEEE Computer Society Press, Los Alamitos, CA, 1992.

[18] N. Max, R. Crawfis, and D. Williams. Visualizing wind velocities by advecting cloud textures. In *Proceedings Visualization '92*, pages 171–178. IEEE Computer Society Press, Los Alamitos, CA, 1992.

[19] R. Crawfis and N. Max. Direct volume visualization of three-dimensional vector fields. In A. Kaufman and W.E. Lorenson, editors, *1992 Workshop on Volume Visualization*, pages 55–60. ACM, 1992.

[20] 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.

[21] J.L. Helman and L. Hesselink. Surface representations of two- and three-dimensional fluid flow topology. In *Proceedings of Visualization '90*, pages 6–13, San Diego, CA, October 1990.

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

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

[24] R.R. Dickinson. Interactive analysis of the topology of 4d vector fields. *IBM Journal of Research and Development*, 35(1/2):59–66, 1991.

[25] A. Globus, C. Levit, and T. Lasinski. A tool for visualizing the topology of three-dimensional vector fields. In *Proceedings of Viualization '91*. IEEE Computer Society Press, Los Alamitos, CA, 1991.

[26] D.N. Kenwright and G.D. Mallinson. A 3-d streamline tracking algorithm using dual stream functions. In *Proceedings Visualization '92*, pages 62–69. IEEE Computer Society Press, Los Alamitos, CA, 1992.

[27] E. Eder. Visualisierung von teilchenstroemungen mit hilfe eines vektorrechners. Master's thesis, Fachhochschule Muenchen, Fachbereich Informatik, Munich, February 1991. (Visualization of Flow Fields Using Vector Computers).

[28] J.J. van Wijk. Implicit stream surfaces. In *Proceedings Visualization '93*. IEEE Computer Society Press, Los Alamitos, CA, 1993.

[29] R.B. Haber and D.A. McNabb. Visualization idioms: A conceptual model for scientific visualization systems. In G.M. Nielson, B. Shriver, and L.J. Rosenblum, editors, *Visualization in Scientific Computing*, pages 74–92. IEEE Computer Society Press, Los Alamitos, CA, 1990.

[30] M. Geiben and M. Rumpf. Visualization of finite elements and tools for numerical analysis. In F.H. Post and A.J.S. Hin, editors, *Advances in Scientific Visualization*, pages 1–21. Springer, 1992.

[31] R.R. Dickinson. A unified approach to the design of visualization software for the analysis of field problems. In *SPIE Proceedings*, volume 1083, pages 173–180, January 1989.

[32] R.R. Dickinson. Interactive analysis of transient field data. In *SPIE Proceedings*, volume 1459, pages 166–176, February 1991.

[33] T. Delmarcelle and L. Hesselink. Visualization of second-order tensor fields and matrix data. In *Proceedings Visualization '92*, pages 316–323. IEEE Computer Society Press, Los Alamitos, CA, 1992.

[34] W.J. Schroeder, C.R. Volpe, and W.E. Lorenson. The stream polygon: A technique for 3d vector field visualization. In *Proceedings Visualization '91*, pages 126–132. IEEE Computer Society Press, Los Alamitos, CA, 1991.

[35] J. Stolk and J.J. van Wijk. Surface-particles for 3d flow visualization. In F.H. Post and A.J.S. Hin, editors, *Advances in Scientific Visualization*, pages 119–130. Springer, 1992.

[36] J.J. van Wijk. Flow visualization with surface particles. *IEEE Computer Graphics and Applications*, 13:18–24, July 1993.

[37] A. van Gelder and J. Wilhelms. Interactive animated visualization of flow fields. In A. Kaufman and W.E. Lorenson, editors, *1992 Workshop on Volume Visualization*, pages 47–54. ACM, 1992.

[38] K. Sims. Particle animation and rendering using data parallel computation. *Computer Graphics*, 24(4):405–413, July 1990.

[39] J.J. van Wijk. Spot noise – texture synthesis for data visualization. *Computer Graphics*, 25(4):309–318, July 1991.

[40] A.J.S. Hin and F.H. Post. Visualization of turbulent flow with particles. In *Proceedings Visualization '93*. IEEE Computer Society Press, Los Alamitos, CA, 1993.

[41] R. Szeliski and D. Tonnesen. Surface modeling with oriented particle systems. *Computer Graphics*, 26(2):185–194, July 1992.

[42] D. Silver, M. Gao, and N. Zabusky. Visualizing causal effects in 4d space-time vector fields. In *Proceedings Visualization '91*, pages 12–16. IEEE Computer Society Press, Los Alamitos, CA, 1991.

[43] D. Silver and N.J. Zabusky. Quantifying visualizations for reduced modeling in nonlinear science: Extracting structures from data sets. *Journal of Visual Communication and Image Representation*, 4(1):46–61, March 1993.

[44] N.J. Zabusky and D. Silver. Case study: Visualizing classical problems in cfd. In *Proceedings Visualization '92*, pages 436–440. IEEE Computer Society Press, Los Alamitos, CA, 1992.

[45] J. Villasenor and A. Vincent. An algorithm for space recognition and time tracking of vorticity tubes in turbulence. *CVGIP: Image Understanding*, 55(1):27–35, January 1992.

[46] H.G. Pagendarm and B. Seitz. An algorithm for detection and visualization of discontinuities in scientific data fields applied to flow data with shock waves. In P. Palamidese, editor, *Scientific Visualization - Advanced Software Techniques*, pages 161–177. Ellis Horwood Ltd., 1993.

[47] P. Comte. Visualization of turbulent coherent structures in numerical wind tunnels. In *Summer Course on Principles of Visualization in Fluid Mechanics*. J.M. Burgers Centre for Fluid Mechanics, Delft, Netherlands, Aug. 1992.

[48] J. Wilhelms, J. Challinger, N. Alper, and S. Ramamoorthy. Direct volume rendering of curvilinear volumes. *Computer Graphics*, 24(5):41–47, Dec. 1990. Proceedings San Diego Workshop on Volume Visualization.

[49] T. Strid, A. Rizzi, and J. Oppelstrup. Development and use of some flow visualization algorithms. In *Computer Graphics and Flow Visualization in Computational Fluid Dynamics*. Von Karman Institute Lecture Series 1989-07, 1989.

[50] P.G. Buning. Sources of error in the graphical analysis of cfd results. *Journal of Scientific Computing*, 3(2):149–162, 1988.

[51] E.R. Tufte. *The Visual Display of Quantitative Information*. The Graphics Press, Cheshire, Connecticut, 1983.