Over Two Decades of Integration-Based, Geometric Flow Visualization

Tony McLoughlin[1], Robert S. Laramee[1], Ronald Peikert[2], Frits H. Post[3], and Min Chen[1]

[1]Visual and Interactive Computing Group
Department of Computer Science, Swansea University, United Kingdom
{*cstony, r.s.laramee, m.chen*}@swansea.ac.uk

[2]Institute of Visual Computing
ETH Zurich, Switzerland
peikert@inf.ethz.ch

[3]Computer Graphics and CAD/CAM Group
Faculty of Electrical Engineering, Mathematics and Computer Science
TU Delft, The Netherlands
F.H.Post@tudelft.nl

Abstract
With ever increasing computing power, it is possible to process ever more complex fluid simulations. However, a gap between data set sizes and our ability to visualize them remains. This is especially true for the field of flow visualization which deals with large, time-dependent, multivariate simulation datasets. In this paper, geometry based flow visualization techniques form the focus of discussion. Geometric flow visualization methods place discrete objects in the velocity field whose characteristics reflect the underlying properties of the flow. A great amount of progress has been made in this field over the last two decades. However, a number of challenges remain, including placement, speed of computation, and perception. In this survey, we review and classify geometric flow visualization literature according to the most important challenges when considering such a visualization, a central theme being the seeding algorithm upon which they are based. This paper details our investigation into these techniques with discussions on their applicability and their relative merits and drawbacks. The result is an up-to-date overview of the current state-of-the-art that highlights both solved and unsolved problems in this rapidly evolving branch of research. It also serves as a concise introduction to the field of flow visualization research.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling

1. Introduction

Flow visualization is a classic branch of scientific visualization. Its applications cover a broad spectrum ranging from turbomachinery design to the modeling and simulation of weather systems. The goal of flow visualization is to present the behavior of simulation data in a meaningful manner from which important flow features and characteristics can be easily identified and analyzed.

Given the large variety of techniques currently utilized in visualization applications, selecting the most appropriate visualization technique for a given data set is a non-trivial task. Considerations have to be made taking into account the type of information the user wishes to extract from the visualization along with the spatial and temporal characteristics of the data set being analyzed. Different approaches have to be designed for different types of data. For example, visualization of 2D data is much different from visualizing 3D data. On top of this is the further complication of temporal dimensionality, with varying techniques more suited to steady flow compared to time-dependent flow fields and vice versa. To this end different tools have been developed according to the

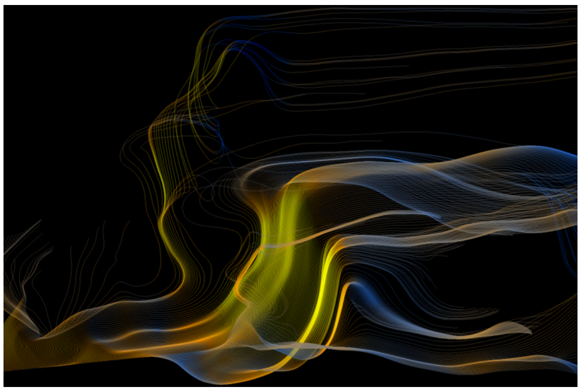

Figure 1: *A set of streamlines exploiting the power of modern programmable GPUs for faster computation [DGKP09].*

different needs of the users and the differing dimensionality of velocity field data.

1.1. Challenges in Flow Visualization

Flow simulations, or CFD (computational fluid dynamics) simulations, are computed using methods such as the Navier-Stokes equations [CSvS86] and are used to simulate experiments such as wind-tunnel tests on cars and airplanes. The visualization of these simulations poses many challenges, the most important of which we outline here.

Large Data Sets. A major technical issue arises from the sheer volume of data that may be generated from complex simulations. Velocity data comprises scalar values for each x, y, z velocity component at each sample point within the data domain. When coupled with several scalar data attributes and consisting of many time-steps, a large amount of data results. Advances in hardware are leading to more computational power and the ability to process larger, more complex simulations with faster computation times. Therefore, flow visualization algorithms must be able to handle this large amount of data and present the results (ideally) at interactive frame rates in order to be most useful in the investigation and analysis of simulation data.

Interaction, Seeding and Placement. One of the main challenges specific to geometric flow visualization is the seeding strategy used to place the objects within the data domain. Geometric flow visualization techniques produce discrete objects whose shape, size, orientation, and position reflect the characteristics of the underlying velocity field. The position of the objects greatly affects the final visualization. Different features of the velocity field may be depicted depending on the final position and the spatial frequency of the objects in the data domain. It is critical that the resulting visualization captures the features of the velocity field, e.g., vortices, turbulence, sources, sinks and laminar flow, which the user is interested in. This aspect becomes an even greater challenge in the case of 3D where a balance of field coverage, occlusion, and visual complexity must be maintained. Time-dependent data also raises a challenge because the visualization then depends on when objects are seeded.

Computation Time and Irregular Grids. Another challenge stems from the computation time. Most of the visualizations compute a geometry that is tangential to the velocity field, e.g., the path a massless particle would take when placed into the flow. Computing such curves through 3D, unstructured grids is non-trivial. Thus much research has been devoted to this and similar topics, see Section 2.4. A recent trend to increase performance has been to move computations from the CPU and perform them on the graphics processing unit (GPU) [BSK*07]. Although the resultant rendering looks the same as a CPU version (see Figure 1), this may offer a significant improvement in performance as the vector calculations are suited to the multiple execution units on the GPU – resulting in high performance parallel processing.

Perception. A central challenge in flow visualization (and visualization in general) relates to perceptual challenges in visualizing 3D and 4D velocity fields as well as multi-variate data sets. If streamlines are used to visualize flow in 3D, too many lines causes clutter, visual complexity, and occlusion. If too few are rendered, important characteristics may not be visualized. Thus an optimal balance between coverage and perception must be achieved. Animated flow presents its own unique challenges to the perception of the user. For instance, it may not be intuitive what a user sees from a cloud of moving particles or a surface deforming to the local flow characteristics. It may also be difficult to discern the movement direction is 3D space.

1.2. Contributions

In light of these challenges and the more than two decades of research the main benefits and contributions of this paper are:

- We review the latest developments in geometry-based flow visualization research.
- We introduce a novel classification scheme based on challenges including seeding. This scheme lends itself to an intuitive grouping of papers that are naturally related to each other. This allows the reader to easily extract the relevant literature without having to read the entire survey.
- Our classification highlights both unsolved problems in the area of geometric flow visualization and mature areas where many solutions have been provided.
- This survey is the most up-to-date presentation on this popular topic. The last time this topic was addressed in the literature was over six years ago [PVH*03].
- We provide a very concise introduction and overview in the area of vector field visualization for those who are new to the topic and wishing to carry out research in this area.

We have made a great effort not to provide simply an enumeration of related papers in integration-based, geometric flow visualization, but to compare different methods, related to one another and weigh their relative merits and weaknesses.

1.3. Classification

One of the main challenges of a survey is classifying these approaches and presenting them in a meaningful order. There are four general categories into which vector-field visualization approaches can be divided: *direct, dense texture-based, geometric, and feature-based*. This paper focuses on the geometric approaches to flow visualization, which received little coverage in previous surveys [LHD*04, PVH*03, LHZP07]. A large volume of research work has been undertaken in geometry-based vector field visualization. We use four tiers of categorization. Our top level of classification groups the literature by the dimensionality of the object used in the resulting visualization, i.e. curves, surfaces and volumes. We then subdivide the literature further according to the spatial dimensionality of the data domain, i.e, 2D velocity fields, velocity fields on surfaces and in 3D volumes. Temporal dimensionality is also used to group papers together, i.e., steady vs. unsteady flow. And lastly, those papers belonging to the same sub-class appear chronologically (See Table 1). Classifying the literature in this way facilitates comparison of similar papers with one another. It also highlights unaddressed challenges and problems for which a range of solutions exist. We give a brief overview and comparison of the four main categories before analyzing the geometric approaches in more detail.

1.4. Direct, Texture-based, and Feature-based Flow Visualization

Direct techniques are the most primitive methods of flow visualization. Typical examples involve placing an arrow glyph at each sample point in the domain to represent the vector data or mapping color according to local velocity magnitude. Direct techniques are simple to implement and computationally inexpensive. They allow for immediate investigation of the flow field. However, direct techniques may suffer from visual complexity and imagery that lacks in visual coherency. They also suffer from serious occlusion problems when applied to 3D data sets.

Dense, texture-based techniques, as the name implies, exploit textures to form a representation of the flow. The general approach uses texture (generally a filtered noise pattern) which is smeared and stretched according to the local properties of the velocity field. Texture-based approaches provide a dense visualization result, provide lots of detail, and capture many flow characteristics even in areas of intricate flow such as vortices, sources, and sinks. Texture-based methods generally cover the entire domain. They also share some of the same weakness of 3D domain representation as direct methods and are generally more suited to 2D or surfaces. A thorough investigation of texture-based flow visualization is presented by Laramee et al. [LHD*04].

Feature-based algorithms focus the visualization on selected features of the data such as vortices or topological information rather than the entire data set. This may result in a large reduction of the required data and thus these techniques are suited to large data sets that may consist of many time-steps. Since they generally perform a search of the domain, these techniques require considerably more processing before visualization. A survey of feature-based approaches is presented by Post et al. [PVH*03].

1.5. Integration-based, Geometric Flow Visualization

Geometric methods define sets of seeding points from which trajectories (streamlines or pathlines) are computed. Trajectories are then used for building geometric objects, in contrast to other methods where they are used for filtering or advecting textures or for topological analysis.

Geometric approaches compute discrete objects within the data domain. Velocity, $\mathbf{v} = \frac{d\mathbf{x}}{dt}$, is a derivative quantity. If we imagine tracking a massless particle through a velocity field, the displacement of such a point can be described by:

$$d\mathbf{x} = \mathbf{v} \cdot dt \tag{1}$$

where $\mathbf{x}$ is the position of the point, t is the time and $\mathbf{v}$ is the velocity field. The analytical solution is approximated using a numerical integration method. Thus geometry-based techniques are also known as integration-based and characterize the flow field with their geometry. It is a non-trivial task to automatically distribute the objects such that all of the important features of the velocity field are captured in the resulting visualization.

Two main aspects of geometric flow visualization dominated research for a decade. In the first decade, the focus was on particle tracing, i.e., the numerical computation of trajectories for various types of data discretization. In the second decade, interest shifted to particle seeding strategies. Geometric visualization techniques are suited to all spatial and temporal dimensions. However, without careful use they are susceptible to visual clutter and occlusion problems. These problems mainly arise from poor seeding strategies and thus considerable effort has been put into researching seeding strategies that provide clear, detailed visualizations. We start out our survey of the literature with the point-based seeding algorithms in 2D velocity field domains.

Table 1 provides a concise overview of the literature grouped according to our classification scheme. Literature is divided up based upon both the dimensionality of the geometry-based objects in the domain and the dimensionality of the data domain itself. Organizing the literature in

Integration-based Geometric Object	Curves				Surfaces	Volumes
	2D	On surfaces	3D Particle Tracing	3D Rendering and Placement		
Steady Data Field	🟥 [TB96]	🟩 [vW92]$_p$	🟨 [BS87]	🟩 [HP93]	🟩 [Hul92]	🟩 [SVL91]
	🟥 [JL97a]	🟩 [vW93a]$_p$	🟨 [RBM87]	🟩 [ZSH96]$_p$	🟩 [vW93b]	🟩 [MBC93]
	🟥 [JL97b]	🟥 [MHHI98]	🟨 [Bun89]	🟩 [FG98]	🟩 [BHR*94]	🟩 [XZC04]
	🟥 [JL01]	🟥 [SLCZ09]	🟨 [BMP*90]	🟥 [MT*03]	🟩 [LMG97]	
	🟥 [VKP00]	🟥 [RPP*09]	🟨 [KM92]	🟩 [LWSH04]	🟨 [WJE00]	
	🟥 [LJL04]		🟨 [USM96]	🟩 [MPSS05]	🟨 [SBH*01]	
	🟥 [MAD05]		🟨 [LPSW96]	🟩 [LGD*05]	🟩 [GTS*04]	
	🟥 [LM06]		🟨 [SvWHP97]	🟩 [LH05]	🟩 [LGSH06]	
	🟥 [LHS08]		🟨 [SdBPM98]	🟥 [YKP05]	🟩 [PS09]	
			🟨 [SRBE99]	🟥 [CCK07]		
			🟨 [NJ99]	🟥 [LS07]		
			🟨 [VP04]			
Unsteady Data Field	🟥 [JL00]		🟨 [Lan93]	🟩 [BL92]	🟩 [STWE07]	🟩 [BLM95]
			🟨 [Lan94]	🟩 [WS05]	🟩 [GKT*08]	
			🟨 [KL95]	🟥 [HE06]	🟩 [vFWS*08]	
			🟨 [KL96]		🟩 [MLZ09b]	
			🟨 [TGE97]		🟩 [KGJ09]	
			🟨 [TGE98]		🟩 [BFTW09]	
			🟨 [TE99]			
			🟨 [SGvR*03]			
			🟨 [KKKW05]			
			🟨 [BSK*07]			

Table 1: *An overview and classification of integration-based geometric methods in flow visualization along the x-axis. Research is grouped based on the temporal dimensionality along the y-axis. Each group is then split into techniques that are applicable to steady or unsteady flow. Finally the entries are grouped into chronological order. Each entry is also colored according to the main challenge, as outlined in Section 1.1, that they address. The color coding scheme used is* 🟥 *red for seeding strategies,* 🟩 *green for techniques addressing perceptual challenges and* 🟨 *yellow for methods aimed at improving application performance. The subscript "p" indicates visualization using particles. This table provides an overview of research and highlights unsolved problems as well as challenges for which a range of solutions have been provided.*

this way points out the mature areas where many solutions are offered and those areas still rich with unsolved problems.

1.6. Terminology

Here we introduce some common, important flow visualization terminology. The most common geometric technique is the streamline. A ***streamline*** is a curve that is everywhere tangent to the steady-state flow field. A ***critical point*** is a location in the velocity field where the velocity magnitude is zero. The behavior of the flow in the region around the critical point is used to classify its type. Some examples of critical points are sources, sinks and saddle points.

Velocity fields are either steady (static) or unsteady (changing over time). There are a variety of techniques that are usually more suited to one temporal dimensionality over the another. Unsteady flow is generally more challenging than steady flow. A natural way of representing time-dependent flow is through animation, which explicitly shows the changes over time. Animation can also be used to visualize steady flows where the animation is used to indicate the downstream motion of the flow or to depict local velocity magnitude.

Unsteady flow is not restricted to being visualized by animation. Streaklines and pathlines, for instance, are computed from successive time steps together so that multiple time-

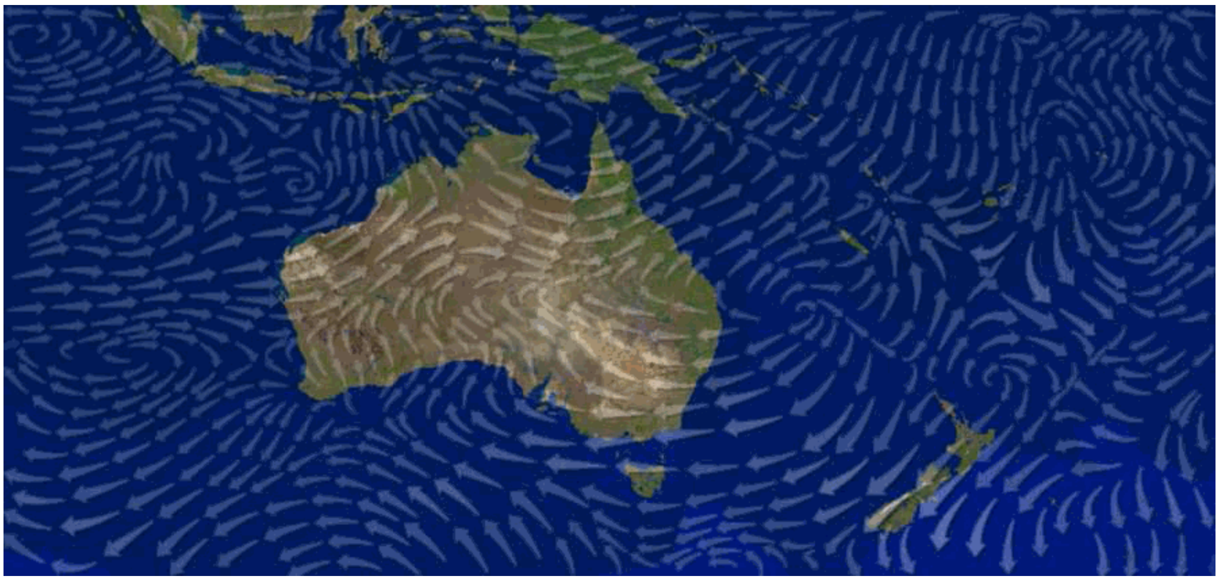

Figure 2: *Arrows showing the wind direction and magnitude over Australia. The arrows are placed along streamlines generated using the image-guided placement technique of Turk and Banks [TB96]. Image courtesy of Greg Turk.*

steps may be displayed in a single static image. A ***pathline*** or particle trace is the trajectory that a massless particle takes in an unsteady fluid flow. A ***streakline*** is the line joining a set of particles that have all been seeded at the same spatial location (but at successive times). If seeded at the same location in a steady flow field streamlines, pathlines and streaklines are identical. A ***timeline*** is a line connecting all particles that have been simultaneously released in a velocity field. The timeline is advected through the flow field and is deformed according to the local velocity variations. ***Streamlets*** are short streamline segments. A ***stream surface*** is a surface that is everywhere tangent to the vector field, it is the locus of a set of streamlines from a shared seeding curve (see Figure 10). ***Path surfaces*** and ***streak surfaces*** are extensions of pathlines and streaklines that are obtained by seeding from a curve instead of a point. For a path surface, seeding is done at a fixed time and particle positions are traced over time, while for a streak surface seeding is done continually and particle positions correspond to a fixed time. A ***time surface*** is the generalization of timelines, connecting particles that have been released from positions on a surface.

2. Integral Curve Objects in 2D

All of the algorithms in this section use points to place streamlines in the vector field domain. Data is usually obtained from flow simulations. Most of the research here focuses on automatic seeding. However, interactive seeding strategies are possible, as demonstrated in [BL92] [SGvR*03] [BSK*07].

2.1. Streamlines in 2D Steady-State Flow Fields

Here we group those methods restricted to two-dimensional, steady state domains.

Streamline placement for 2D flow fields greatly affects the final image(s) produced by visualization applications. Streamlines that are seeded in arbitrary locations may provide an unsatisfactory result. Critical features in the flow field may be missed if there are regions containing only a sparse amount of streamlines. Conversely, where there is a large number of streamlines in a localized region, a cluttered image may result making it difficult to distinguish flow behavior.

An image-guided streamline placement algorithm was introduced by Turk and Banks in 1996 [TB96]. One of the goals of this algorithm is to produce visualizations similar to hand-drawn illustrations found in textbooks. Prior seeding algorithms were simply based on regular grids, random sampling or interactive seeding [BL92]. The seeding of the streamlines is influenced by the resultant image. The goal is to obtain a uniformly dense streamline coverage. It is formulated as an optimization problem where the objective is to minimize the variation of a low-pass filtered (blurred) image. Starting from a random initial streamline seed, the problem is solved by iteratively performing one of the operations *move* (displace a seed), *insert, lengthen, shorten* and *combine* (connect two streamlines with sufficiently close end points) on the set of streamlines. Figure 2 shows one result.

A follow-up technique is presented by Jobard and Lefer [JL97a]. The motivation is to introduce a new streamline seeding strategy that was computationally efficient and less costly than the previous streamline seeding strategy [TB96] and allows the user to control the density of the displayed streamlines. The authors introduce two user-controlled parameters d_{sep} and d_{test}. These parameters are used to control the distance between adjacent streamlines. Existing streamlines are used to seed new streamlines and candidate seed points are chosen that are at a distance $d = d_{sep}$, from a streamline. All candidate points of one streamline are used before moving on to the next streamline. This process stops when there are no candidate points generated. The d_{test} parameter is a proportion of d_{sep}. d_{test} is used to control the closest distance that streamlines are allowed to one another. Sufficient coverage (i.e., a minimum density) is ensured by seeding streamlines at a distance d_{sep} from one another. The method was also combined with texture advection techniques [JL97b] for animating steady flow fields.

Jobard and Lefer [JL01] introduce a novel algorithm that produces images of a vector field with multiple simultaneous densities of streamlines. The paper builds upon the previous technique [JL97a]. The multi-resolution property is ideal for vector field exploration as it allows for sparse streamline placement for a quick overview of the vector field, the streamline density can then be increased to allow for a more detailed investigation into areas of interest. They also demonstrate the use of the technique for zooming and enrichment.

Lefer et al. provide a novel technique for producing variable-speed animations [LJL04]. This method encodes the motion information in a so-called motion map and a color table is utilized to animate the streamlines. Once streamlines have been computed they remain valid for all frames due to the steady vector field, so the challenge of animating the streamlines is simplified to a coloring the streamlines appropriately. The algorithm begins by creating a dense set of streamlines that cover every single pixel. Animating the streamlines is achieved by shifting the color table entries so that the color pattern appears to travel down the streamlines. The local velocity magnitude is taken into account when computing how fast to move the color pattern.

Verma et al. present a novel method of streamline placement that focuses on capturing flow patterns in the vicinity of critical points [VKP00]. Templates are defined for types of critical points that may be present in 2D flow fields. The algorithm begins by determining the location and type of critical points in the field. Verma et al. use Voronoi partitioning around the critical points that contain regions that exhibit similar flow behavior. A random Poisson disk seeding strategy is finally used to populate streamlines in sparse regions. Seeding in regions of critical points first ensures that they are covered by a sufficient amount of streamlines and that these streamlines have a longer length. In this implementation FAST [BMP*90] is used to compute critical point locations and determine their nature, Voronoi diagrams are computed using *triangle* [She96].

The work of Mebarki et al. [MAD05] builds upon previous research by Turk and Banks [TB96] and Jobard and Lefer [JL97a]. The results produced are of comparable quality to the work of Turk and Banks [TB96] while being produced faster [MAD05]. The algorithm uses a farthest seeding point strategy. Roughly speaking, when a new streamline is created, the point furthest away from all current streamlines is used as the seed point for the subsequent streamline. Using a farthest point seeding strategy ensures that long streamlines are produced. To determine the farthest point, the points of the streamlines are inserted in a 2D Delauney triangulation. The incident triangles of a newly integrated point are used to generate a minimal circumdiameter. Any diameter that is above the desired spacing distance and below a saturation level is pushed onto a priority queue that is sorted by length. The top circle is then popped out of the queue, the center may then be used as the seeding point for the next streamline.

The work of Liu et al. [LM06] introduces another evenly-spaced streamline algorithm. It builds upon the work of Jobard and Lefer [JL97a] and Mebarki et al. [MAD05]. Cubic Hermite polynomial interpolation is used to create fewer evenly-spaced streamline samples in the neighborhood of each previous streamline in order to reduce the amount of distance checking. Placement quality is enhanced by double queues to favor long streamlines (this minimizes discontinuities). The presented method is faster than that of Jobard and Lefer [JL97a]. In addition, it incorporates the detection of streamline loops.

Li et al. [LHS08] introduced a novel method for streamline placement, which is different from its predecessors in that its goal is to generate the fewest number of streamlines possible while still capturing the most important flow features of the vector fields (see Figure 3). The images produced use a small amount of streamlines and are intended

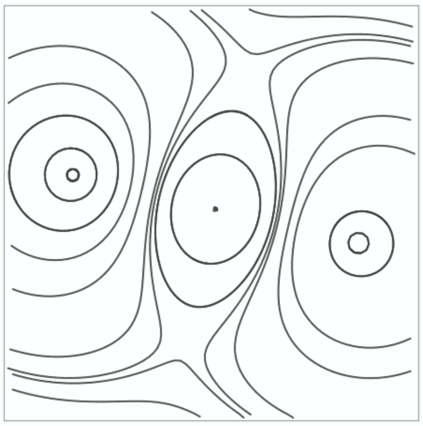

Figure 3: *A representative set of streamlines generated by the Illustrative Streamline Placement algorithm [LHS08]. Image courtesy of Han-Wei Shen.*

to be similar to hand-drawn diagrams. This is achieved by taking advantage of spatial coherence and by using distance fields to determine the similarity between streamlines. New streamlines are only created when they represent flow characteristics that are not already shown by neighboring streamlines. This way repetitive flow patterns are omitted. Similarity between streamlines is measured locally by the directional difference between the original vector at each grid point and an approximate vector derived from nearby streamlines, and globally by the accumulation of local dissimilarity at every integrated point along the streamline path.

Reflection: The two most notable contributions to streamline seeding in planar flows are by Turk and Banks [TB96] and Jobard and Lefer [JL97a]. Turk and Banks are the first to introduce the notion of a user-controlled spatial frequency for streamlines, while Jobard and Lefer accelerate the idea to fast rendering times. The contributions that follow are all variations on these two themes. Overall, we believe the challenge of streamline seeding in 2D steady flow to be a solved problem.

2.2. Integral curves in a 2D, Time-Varying Domain

Jobard and Lefer extend their evenly-spaced streamline technique [JL97a] to unsteady flow [JL00]. Streamlines across several time-steps represent the global nature of the flow at each step and give insight into the evolution of the vector field over time. However, simply generating a set of streamlines at each step and cycling between them leads to an incoherent animation. The authors present a set of parameters that are used to choose a suitable set of streamlines for the next time-step using the current set of streamlines as a basis. A so-called feed forward method is used which selects an appropriate subset of streamlines from the subsequent time-step that correlate with the current set. A technique is employed that quantitatively evaluates the corresponding criterion between streamlines. The best candidates are then used for the next time-step. Several methods are used to improve the animation quality, such as giving priority to circular streamlines and adding tapering effects to the streamlines. A cyclical texture is also applied and this is animated to indicate the downstream direction of the flow on the streamlines.

We have not discovered any literature describing the solution to explicit pathline or streakline seeding algorithms for 2D, unsteady flow. This is still an open challenge.

2.3. Streamline Seeding on Surfaces

In practice, most vector field domains consist of either 2 or 3 spatial dimensions. Some approaches are more suitable for one spatial dimension over the others. Typically, as we move from 2D to 3D, the complexity of algorithms increases. This is due, in part, to the effort required to minimize visual clutter and occlusion and, to the extra complexity of another spatial dimension.

A novel visualization scheme based upon a particle system is introduced by Van Wijk [vW92]. The particles can be seeded from a variety of geometric objects. The most obvious objects are: points, lines, circles, rectangles and spheres. The sources also have a temporal attribute too, the particles can be injected at discrete time pulses or as a continuous stream. A continuous point source will result in streamlines being created by the particles and a stream surface will be created if a curve-based continuous source is used. The particles have a normal, which allows the lighting equations to be used in order to apply shading and provide greater depth cues. A Gaussian filter is used to smooth the visualization, softening aliasing and strobing artifacts.

Van Wijk [vW93a] builds upon his previous work in [vW92]. Here an improved shading model is used to reduce the aliasing and strobing artifacts that were found in [vW92]. A more detailed discussion of the seeding objects is also presented detailing the flexibility of using the surface particles to emulate an array of visualization techniques such as streamlines, stream surfaces and stream tubes. We classified the work of Van Wijk [vW92, vW93a] into point-based objects on a surface domain because the focus of the research is on how to effectively render particles on stream surfaces.

Mao et al. [MHHI98] present an evenly-spaced streamlines technique for curvilinear grids. They expand upon the work of Turk and Banks [TB96] by applying the seeding strategy to parameterized surfaces. This algorithm takes the vectors from the 3D surface and maps them to computational space. An extended 2D image-guided algorithm is then applied and streamlines of a desired density are generated. The streamlines are then mapped back onto the 3D curvilinear surface. However, curvilinear grids cells can vary significantly in size. This means that streamlines distributed evenly in computational space won't necessarily be evenly spaced when they are mapped back to physical space. This challenge is overcome by altering the computational-space streamline density. The streamline density is locally adapted to the inverse of the grid density in physical space [MHHI98]. This is achieved by using Poisson ellipse sampling which distributes a set of rectangular windows in computational space.

Spencer et al. [SLCZ09] extend the 2D evenly-spaced streamlines technique by Jobard and Lefer to surfaces, Figure 4 shows evenly-spaced streamlines on the boundary surface of a gas engine simulation. It starts by projecting the vector data onto the image plane, by rendering a so-called velocity image into the frame buffer. Performing the streamline computations in image-space effectively reduces the complexity of the 3D problem into a 2D one. This also has the advantage of a simpler implementation using the programmable portions of the graphics pipeline and inherently

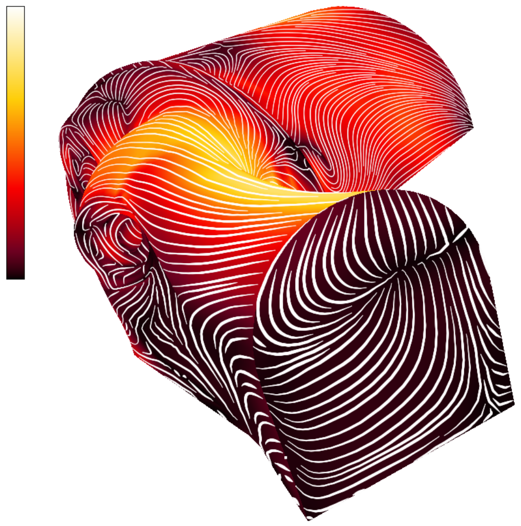

Figure 4: *Evenly-spaced streamlines on the boundary surface of a gas engine simulation. Perspective foreshortening is utilized and the density of streamlines further away from the viewpoint is increased [SLCZ09].*

takes advantage of the hardware interpolators. Another major performance benefit arises from the z-buffer and frustum culling. These discard any occluded fragments and clipped geometry, this prevents unnecessary streamline computations on areas that aren't visible to the user. A combination of seeding strategies is employed. A grid traversal strategy, checking each cell as a candidate seeding position is used in combination with seeding based on current streamlines, like Jobard and Lefer's algorithm [JL97a]. This ensures that all visible regions of the geometry are processed.

More recently Rosanwo et al. [RPP*09] provide a solution to streamline seeding based on dual streamlines. *Dual streamlines* run orthogonal to the vector field instead of tangential. Two sets of streamlines are maintained, a set of tangential streamlines, S, and a set of dual streamlines, D. This is similar to the technique used by Mebarki et al. [MAD05] where the largest voids are found in order to place a new seed. Streamlines are seeded in a similar way using the dual streamline segments. As this method only uses the arc-length of distance metric it may be efficiently applied to curved surfaces where other distance metrics such as geodesic distance are more computationally expensive and/or are hard to apply correctly. This algorithm requires that a suitable starting set of streamlines are used in order to be efficient and to ensure complete domain coverage and critical points. This is achieved by computing the topological skeleton of the dual field as the initial dual streamline set. In cases where the vector field contains no topology randomly seeded dual streamlines are seeded. This method shows a slower growth in computation time when increasing the streamline density compared to algorithms by Verma et al. [VKP00] and Mebarki et al. [MAD05].

Reflection: An important solution to the challenge of streamline seeding on surfaces comes from Spencer et al. [SLCZ09]. This is the only solution of its kind that handles general surfaces and unstructured, adaptive resolution meshes. It is also a fast algorithm that supports exploration through user-interaction. Pathline and streakline seeding on surfaces remains an open challenge however.

2.4. Efficient Particle Tracing in 3D

This subsection summarizes particle tracing strategies in 3D space. The volume cell types used in simulations vary. Depending upon the model used to generate them. The simplest grid type is a Cartesian grid. Curvilinear grids, commonly used in flow simulations, contain the same cell type but the grid is usually distorted (usually curving) so that is fits around a geometry. Unstructured data may contain several different cell types, tetrahedra and hexahedra are commonly used. Unstructured grids are generally more challenging than structured grids and some algorithms are specifically aimed at particle tracing solutions on them. Computations on unstructured grids may either be performed in physical-space or computational-space on a per-cell basis. Physical-space uses the velocity field and grid as output from the simulation. Computational-space transforms a grid cell to make it axis-aligned and unit length, and adjusts the velocities accordingly. Computational-space methods are used to simplify certain operations such as point-location.

This collection of papers focuses on computational methods, addressing the challenge of providing fast, accurate results that can be utilized by other visualization methods to improve their performance. This is in contrast to the other methods that directly provide novel visualizations. The forerunners to these techniques along with some of their applications can be found in [BS87, RBM87, Bun89].

2.4.1. 3D Particle Tracing in Steady Vector Fields.

PLOT3D [BS87, WBPE90] is a command line driven program for displaying results of CFD simulations on structured and unstructured grids. Besides a wide range of graphics functionality, e.g., hidden line and hidden surface techniques, PLOT3D offers 2D and 3D streamlines of the velocity field, the vorticity field (vortex lines), and the wall shear stress field (skin friction lines). The software was designed to run on supercomputers, e.g., for computing movies, but also on the first graphics terminals and workstations with hardware supported viewing transformations. PLOT3D was the precursor of FAST (Flow Analysis Software Toolkit) [BMP*90], a modular redesign which added a GUI and distributed processing. Visual2 [GH90] and Visual3 [HG91] were packages written by Haimes and Giles for the visualization of 2D and 3D flow fields. Linked to user-written main program, they provided interactive X-windows based visualization of steady or unsteady flow fields given on unstructured grids. Visual3 was later adapted to network computing and renamed to pV3 [Hai94]. The techniques for vector fields available in Visual3 include streamlines and variants such as ribbons and tufts (or streamlets).

Ueng et al. [USM96] present an efficient method of streamline construction in unstructured grids. This method uses calculations performed in computational-space to reduce computational cost of the streamline generation. To perform the calculation in computational-space the physical-space coordinates of a cell and its corresponding vector data must be transformed into canonical coordinates. A cell searching strategy similar to that used in [KL96] takes advantage of the canonical coordinates to simplify and speed up the operation. A specialized Runge-Kutta integrator is also presented for use in the canonical coordinate system which and offeres improved computation times compared to the second- and fourth-order Runge-Kutta integrators that perform in physical space.

The techniques used for **streamtube** and **streamribbon** construction are also described in [USM96]. Streamtubes are created by placing circular curves, oriented normal to the flow, at the streamline integration points. The circular cross sections are then connected to form an enclosed tube object. The radius of the streamtube illustrates the local cross flow divergence and is calculated at each streamline point, i.e., when the circular glyphs are created. Streamribbons are created using the streamline for one edge and then using a constant length normal vector generate the position of the opposite edge. The constant length normal rotates around the initial streamline in order to depict local flow vorticity.

UFLOW is a system introduced by Lodha et al. [LPSW96] to analyze the changes resulting from different integrators and step-sizes used for computing streamlines. A pair of streamlines are interactively seeded by the user and each streamline is generated using a different integrator or integration step-size. It is also possible to create a single streamline and then trace it backwards from its end point and compare it to the initial streamline (See Figure 5).

Sadarjoen et al. present a comparison of several algorithms used for particle tracing on 3D curvilinear grids [SvWHP97]. The particle tracing process is broken down, with a brief description, into basic components: *point-location*, locating which cell a point is in, *interpolation*, and *integration*. A more thorough discussion and comparison of *physical-space* and *computational-space* algorithms then ensues. Results for the implemented algorithms are also given showing that *physical-space* computation algorithms generally perform better than their *computational-space* counterpart.

Sadarjoen et al. [SdBPM98] present a 6-tetrahedra decomposition method for σ-transformed grids. σ-transformed grids are structured hexahedral curvilinear grids in which the x and y dimensions differ by 2-3 orders of magnitude from the z dimension, thus resulting in very thin cells. The method presented here is more accurate and allows for faster operations to be performed than the more common 5-tetrahedra decomposition that is usually employed [SvWHP97]. Decomposing the hexahedral cells into 6 tetrahedra prevents a center tetrahedron covering the center of the cell and thus makes point location much easier. This method also reduces the chances of infinite loops between two cells when using the 5-decomposition approach [SdBPM98].

Schulz et al. [SRBE99] present a set of flow visualization techniques that are tailored for PowerFLOW, a lattice-based CFD simulator. The grids in these simulations are multi-resolution Cartesian grids, where finer voxels are used in areas of interesting flow or boundary surface geometry. Particle tracing and collision detection are discussed, demonstrating the need for collision detection between the particle and the object surface. When a collision occurs the particle tracing for that line may either terminate or follow a path along the object boundary. The system has several seeding types, that can be interactively manipulated, ranging from rakes, planes and cubes.

Nielson et al. introduce efficient methods for computing tangent curves for three-dimensional flow fields [NJ99]. This technique is an extension to 3D of their previous research [NJS*97]. The techniques are designed to be used on tetrahedral grids. Incremental methods are used for stepping along the analytic solution of the streamline ODE and as a result produce exact results. Techniques for both Cartesian and barycentric coordinates are presented, allowing the user to use the tools for the coordinate system that is most suited to the current application. Several cases are defined based on the types of the eigenvalues found at a particular point, these in turn are used for the calculation of a tangent curve through a tetrahedron. Results are presented that compare the accuracy of the presented algorithms compared to Euler and fourth-order Runga-Kutta integrators.

Verma and Pang present methods for comparing streamlines and streamribbons [VP04] (Figure 5), and some of their methods are loosely based on those that appear in the UFLOW system [LPSW96]. Large CFD simulations are generally run on supercomputers, however the applications used to visualize these simulations are generally run on workstations. Some of these simulations have to be approximated with smaller data sets to make their use on workstations more feasible. Different datasets are compared simultaneously, with the second dataset being a subsampled version of the first dataset. A metric for measuring difference is needed and here the Euclidean distance between associated streamline points is used. Associated points are connected by lines, giving a ladder effect, which aids the visual representation of the differences between the streamlines. Strip envelopes, which fill in the ladder sections and spheres are also used to depict the difference when comparing a pair of streamlines.

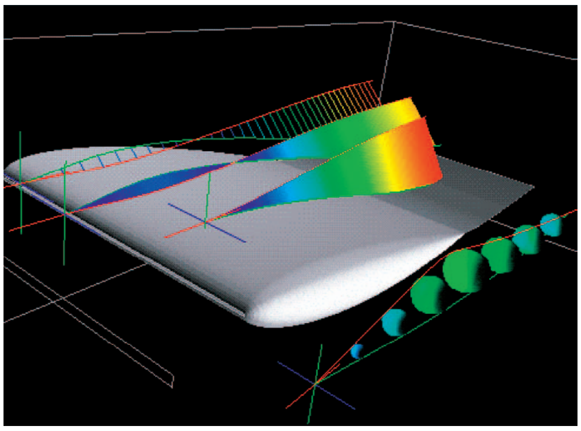

Figure 5: *Comparing streamlines of two datasets simulated using different turbulence models. The streamlines are compared using line glyphs, strip envelopes, and sphere glyphs to highlight the differences between them. Image Courtesy of Alex Pang [VP04].*

Reflection: We consider the challenge of particle tracing to be solved for the case of steady-state structured grids only. This is because many streamlines can be traced interactively for steady-state fields. The same cannot be said for large unstructured grids however. Tracing many streamlines (hundreds) tends to be non-interactive. This is still and unsolved problem.

2.4.2. 3D Particle Tracing in Unsteady Vector Fields.

Lane introduces a system for using streaklines (refer to Section 2.2 for the visualization of unsteady flows) [Lan93]. Lane presents the numerical background for particle integration over many time-steps as well as integration over simulations that involve a moving grid, a feature demonstrated in very few systems. The two datasets visualized are in excess of 15GB and 64GB including both their grids and solutions. Seeding points are positioned manually. Lane shows that only two time-steps need to be loaded at once to perform integration for one step, thus enabling this technique to be applied to large datasets. The tools in this application build upon similar systems such as the Virtual Wind Tunnel [BL92] (see also Section 2.6).

UFAT [Lan94] is a system that is used to generate streaklines on datasets with a large number of time-steps. One of the major challenges of unsteady flow is the size of the datasets that may be produced. The size makes them difficult to store in memory. This is also true when a time-dependent grid is used. Examples of moving grids are shown, such as engine cylinder simulations with a moving piston, and turbine simulations with rotating blades. A second-order Runge-Kutta with adaptive step size is used to advect the particles through the flow field. The system stores the streaklines at each time on disk so that they can be recovered without re-computation and used to create an animation. This system builds upon work done on systems such as the Virtual Wind Tunnel [BL92], pV3 [Hai94] and FAST [BMP*90].

Kenwright and Lane [KL95, KL96] present methods that increase the efficiency of particle tracing for simulations on curvilinear grids. Many simulations output the vector data on curvilinear grids. In this case, particle tracing can be calculated in physical space, i.e., on the curvilinear grid in its original state, or in computational space, which transforms the curvilinear grids coordinates into Cartesian space. Calculations in computational space are easier to perform but tend to be less accurate due to the vector field transformation using approximated Jacobian matrices. Physical space computation is more accurate but point location, can be an expensive operation if done naively (e.g. a brute force linear search in every cell). The authors overcome this barrier by implementing a more efficient point-location strategy for tetrahedral grids.

Teitzel et al. [TGE97] describe an analysis of integration methods used in scientific visualization. The integration methods investigated are both adaptive and non-adaptive Runge-Kutta integrators of orders 2, 3 and 4. A robust integration scheme is found by establishing the link in numerical errors between the integration method and the linear interpolation of the vector field values between the discretely sampled grid points. Their approach is shown to be more efficient than that of [BLM95] and [KL95]. The authors also describe implicit integration methods for use in stiff problems (areas of strong shear or vorticity).

Teitzel et al. [TGE98] introduce a particle tracing method for sparse grids built upon their previous work [TGE97]. The main difficulty in this task is the interpolation operation to find the vector values along an integral path. On a full grid the tri-linear interpolation is done as a local operation. To help with the efficiency the authors have used an array to store the contributing coefficients of the sparse grid as values can be accessed directly. Functions are added to calculate the contributing samples and to accumulate them over the different levels of the sparse grid. The flow is visualized with color-coded streak balls, streak tubes and streak bands (or ribbons). **Streak balls** follow the same path as streaklines, however, the spacing between objects depicts acceleration and the size of the ball depicts local flow convergence and divergence. **Streak tubes** (a dericative of stream tubes) use a closed-curve seeding object resulting in a tube that follows a streakline path. The diameter of the tube depicts flow divergence and convergence. A **streak band** uses a short line segment as a seeding object. This results in a ribbon when traced through unsteady flow whose twisting depicts the vorticity (or swirling motion) of the flow.

Teitzel et al. [TE99] also introduce an improved method to accelerate particle tracing on sparse grids and introduce particle tracing on curvilinear sparse grids. An adaptive evaluation of the sparse grids is implemented. This is achieved by omitting contribution coefficients with a norm below a given error criterion during the interpolation process. The combination technique is also used to improve the efficiency. Streaklines, streak balls and streak tetrahedra are used to visualize flow on curvilinear sparse grids. **Streak tetrahedra** attempt to combine the advantages of streaklines, ribbons, tubes, and balls. The displacement of the tetrahedra along a streakline path depicts acceleration, rotation depicts vorticity, and volume reflects convergence and divergence.

Reflection: We consider the challenge of particle tracing in 3D, unsteady vector fields to still be a challenge for the case of unstructured grids. Tracing many integral curves is generally still not interactive from a performance point of view.

2.5. Streamline Rendering and Placement in a 3D Steady-State Domain

This section surveys streamlines used to visualize 3D vector fields. Here, the challenges are perceptual. Rendering too many field lines results in clutter, complexity, occlusion, and other perceptual problems. Rendering too few field lines may lead to missing important characteristics of the data. Conveyance of depth and spatial orientation are also challenges.

In 1993, Hin and Post introduce a method for depicting turbulent flow using a particle system [HP93]. Turbulence is a common feature of flow fields, however, there are relatively few techniques that are specifically focused on this flow feature. Turbulence is modeled on Reynolds' decomposition [Rey95], which expresses turbulence of flow into mean flow and fluctuation, where the fluctuation represents local turbulent motion. This was implemented using a stochastic process whereby a compound velocity was composed of the mean velocity and a random perturbation generated using random-walk models. Tracing the random-walk particles over many steps leads to an effect representing turbulent behavior. The seeding of particles is based on a uniform Cartesian grid aligned with the domain boundary.

Zöckler et al. introduce a method of illuminating streamlines [ZSH96]. Graphics APIs such as OpenGL support hardware acceleration for lighting when applied to surface primitives. OpenGL uses the Phong reflection model which typically uses the orientation of the surface (i.e., its normal) with respect to the light direction and the viewing angle. However, there is no native support for the lighting of line primitives in these libraries, due to the fact that line primitives have no unique normal vector.

From the set of possible normal vectors, the method chooses the ones that maximize diffuse and specular reflection, respectively. For this, two products $t_1 = \mathbf{L} \cdot \mathbf{T}$ and $t_2 = \mathbf{V} \cdot \mathbf{T}$ are computed from the light, tangent and view unit vectors $\mathbf{L}, \mathbf{T}$ and $\mathbf{V}$ on the vertices. By using specially constructed textures and t_1 and t_2 as texture coordinates, diffuse and specular terms are obtained per pixel.

A streamline placement algorithm is also introduced. For the placement technique a stochastic seeding algorithm is applied. The degree of interest in each cell is defined on some scalar value (i.e., velocity magnitude). An equalization strategy is then employed to distribute the seed points more homogeneously. See Weinkauf et al [WT02, WHN*03] for applications of this seeding strategy.

Mattausch et al. [MT*03] combine the illuminated streamlines technique of [ZSH96] with an extension of the evenly-spaced streamlines seeding strategy of Jobard and Lefer [JL97a] to 3D. With the 2D version of evenly-spaced streamlines presented by Jobard and Lefer [JL97a] the d_{sep} parameter is used in connection to the current streamline point for the candidate streamline seed points. In 2D there are only two possible positions for this new candidate seed position (one on either side of the streamline). When this is extended to 3D there are an infinite number of positions around a line at an orthogonal distance of d_{sep}. The authors simplify the extension to 3D by defining six points around a streamline that may be used for the candidate seed point generation.

Mallo et al. present an improved illuminated lines technique [MPSS05]. This method builds upon the previous illuminated lines by Zöckler et al. [ZSH96] and the cylinder averaging technique presented by Schussman and Ma [SM04]. This method calculates the diffuse and specular components

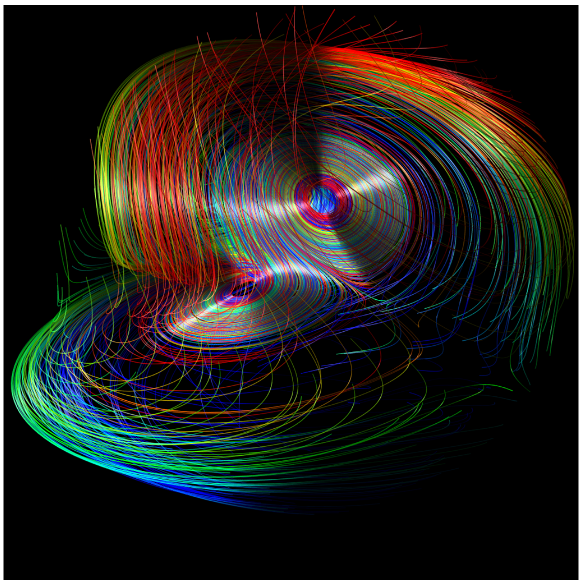

Figure 6: *A Lorenz attractor visualized using streamlines. The streamlines are illuminated using a cylinder averaging presented by Mallo et al. [MPSS05].*

of lighting from the infinitesimal facets of a cylinder. The authors take advantage of programmable GPUs and implement shader programs. This technique improves upon Zöckler et al's technique which used maximal reflection due to the fact that the maximal reflection technique produces bi-directional lighting. The cylinder averaging technique does not produce bi-directional lighting and thus provides clearer orientation and depth information without having to use a strong specular component. Figure 6 shows an example of illuminated streamlines.

Fuhrmann and Gröller [FG98] present a technique for virtual environments that aims to reduce perceptual problems in visualizing 3D data such as occlusion and visual clutter. The concept of a **dashtube** is introduced. A dashtube is an animated, opacity mapped streamline. The dashtubes are seeded using a straightforward extension of the evenly-spaced streamlines algorithm [JL97a] to 3D. For simplicity the tube portions are set to either being fully opaque or fully transparent. The opacity mapping is achieved using textures with animation taking place in texture space to improve efficiency and ease of implementation. This method, like most texture based algorithms, can suffer from aliasing problems. The authors present two methods for resolving this. The first method is a variation of well known mip-mapping, which instead of filtering the mip-maps, produces sub-maps. The second method uses a texture with bands of varying sizes for different sized regions on the streamline (regions further away from the user appear smaller). The authors also present focus and context techniques: magic lenses and magic boxes. The region within the lens contains a higher density of dashtubes and allows the user to investigate selected areas in more detail. The magic box shows a discrete volume which forms the focus and works on the same principle as the lens while allowing the user to change viewing position and orientation.

Laramee and Hauser present a set of geometric visualization techniques including the introduction of two novel approaches: the streamcomet and a fast animating technique [LH05]. These techniques are demonstrated in the context of CFD simulation data. Oriented streamlines improve upon standard streamlines by depicting the downstream direction of the flow in a static image. Animation of streamlines is achieved by a stipple pattern. The streamcomet is a metaphor that offers a large amount of flexibility and interaction from the user. A streamcomet is comprised of a head section and a tail section.

Ye et al. [YKP05] present a method for streamline placement in 3D flow domains. This paper addresses the common goals of streamline placement, namely, the generation of uncluttered visualizations, and sufficient coverage of the domain to ensure that all important features are captured by the visualization. Conceptually, this algorithm can be viewed as an extension of Verma et al's method [VKP00] to 3D.

This approach scans the vector field for critical points and extracts them, identifying important areas of interest. Different seeding templates are defined a priori and positioned around the vicinity of critical points. This approach also contains an operation which detects the proximity of one critical point to another. A proximity map is then used to merge the two most appropriate templates. Poisson sphere seeding is used to add streamlines to regions of low streamline density. Filtering of the streamlines is then used to remove redundant streamlines and to avoid visual clutter. The filtering process is multi-staged and considers both geometrical and spatial properties. First the streamlines with short lengths and small winding angles are removed. The next step considers the similarity of the remaining streamlines. The streamlines are ranked in order of winding angle. The distance between endpoints and centroids of streamlines with similar winding angles are then considered. If the distance is below a predefined threshold then one of them is filtered out.

Chen et al. [CCK07] present a novel method for the placement of streamlines. Unlike many other streamlines placement methods this technique does not rely solely on density placement or feature extraction. Streamline generation methods relying on a density measure may contain redundant streamlines. Strategies based on the extraction of critical points in the field require binary filtering of data based on whether or not they describe a feature. This approach is based on a similarity method which compares candidate streamlines based on their shape and direction as well as their Euclidean distance from one another.

Li et al. [LS07] present a streamline placement strategy for 3D vector fields. The motivation is drawn from the fact that streamlines that are generally well-organized in 3D space may still produce a cluttered visualization when projected to the screen. This is the only approach of its kind – where an image-based seeding strategy is used for 3D flow visualization. The approach presented here places the

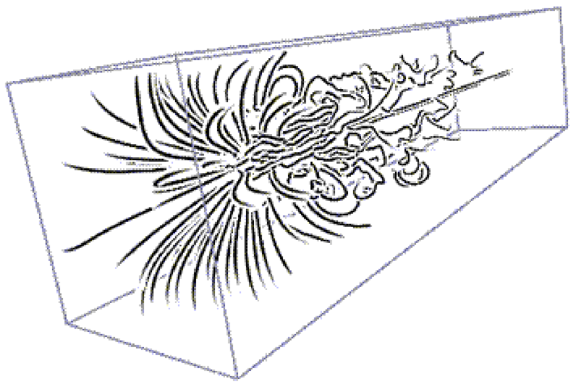

Figure 7: *Streamlines seeded in a 3D domain using an image-based technique [LS07]. Image courtesy of Han-Wei Shen.*

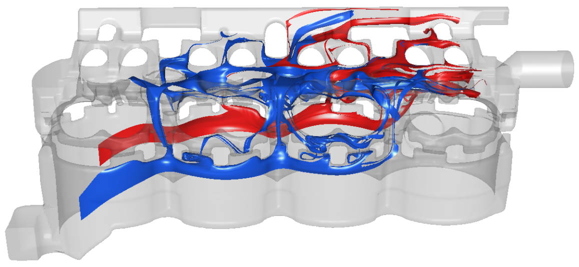

Figure 8: *Stream surfaces showing the flow through an engine cooling jacket. Either side of the surface is colored differently to easily identify the orientation of the surface [LGD*05].*

streamline seeds in image-space and then unprojects them back onto object-space. The first stage of this algorithm is to randomly select a seed point on the image plane for the initial streamline. This position is then unprojected back into object space. Switching between image-space and object-space is made possible by exploiting a depth map. Once the initial seed has been placed, the streamline is integrated and placed into a queue. The oldest streamline is then removed from the queue and used to generate a new seed position for another streamline. There are two candidate positions for the seeds, one on either side of the streamline. The new streamline is integrated until it is within a threshold, d_{sep}, from other streamlines, it is then placed in the queue. This process is then repeated. Complications arise when 3D streamlines in object-space then overlap in image-space. Halos are one of the tools used to address this problem.

Interactive, seeding strategies have been used in various modern, real-world applications including the investigation and visualization of engine simulation data [Lar02, LWSH04, LGD*05] (See Figure 8).

Reflection: One of the major milestones to perception in 3D vector fields came from Zöckler et al [ZSH96]. They were the first to introduce an enhanced lighting and illumination model for 3D streamlines. However, we still consider perception of 3D vector fields to be an open problem with various unsolved challenges. For example no user-study evaluation of illuminated streamlines exists.

2.6. Integral Curve Placement in a 3D, Time-varying Domain

The following research focuses on point-based seeding in unsteady, 3D vector fields. Bryson and Levit [BL92] introduce the Virtual Wind Tunnel. The tunnel is a virtual environment for the exploration of vector fields. It utilizes a mounted head-tracked stereoscopic display. This serves two main purposes: The stereoscopic display provides depth information to the user and the head-tracking allows the user to change their view point within the application by physically changing the position and orientation of their head. This system also allows the user to interact and manipulate objects (such as seed positions) in the system through the use of a glove with input based on gestures from the user. Visualization techniques that are used include tufts, streaklines, particle paths and streamlines. Performance issues arise due to the nature of time-dependent visualization, large data sets and result in high bandwidth and memory requirements when using techniques that simultaneously depict many time-steps. This problem is exaggerated more by the head-tracking feature, the application needs to maintain a minimum execution performance rate (A minimum of 10fps is recommended) to prevent the user from losing coordination within the virtual environment.

Wiebel and Scheuermann present two methods for static visualization of unsteady flow [WS05]. This is opposed to using animation which is much more commonly used to visualize unsteady flow. The first method involves bundles of streaklines and pathlines that pass through one point in space (the eyelet) at different times. A group of pathlines or streaklines passing through the eyelet point (at different times) form the basis of a tangent surface. This method is similar to the technique proposed by Hultquist [Hul92], however in the cases of convergence, a line trace is not terminated, it is just simply ignored for the purpose of surface construction. In the case of divergence a test is made to see if there are any pathlines that are currently being ignored and if so and they are in the correct place they are then used again. If no appropriate pathline exists then a new line must be traced from the eyelet. It is not adequate to simply interpolate a new position between two pathline for a seed point, this is because this new seed point won't necessarily pass through the eyelet. Regions of high activity are of more interest for investigation in general and iso-surfaces are used to separate regions of high activity from regions of (nearly) steady flow. This effectiveness of this visualization technique depends greatly upon the placement of the eyelets within the flow field. Positioning the eyelet is based on sharp edges or corners of objects in the simulation, vortices, critical points, and regions of high activity i.e., rapidly fluctuating flow direction.

Helgeland and Elbroth present a hybrid geometric and texture based method for visualizing unsteady vector fields [HE06]. The seed positions for the field lines are computed as a pre-processing step. A random initial seed position is used to prevent visual artifacts that may arise when using a uniform distribution of seed points. The seeding algorithm is based upon the evenly-spaces streamline strategy introduced by Jobard and Lefer [JL97a]. As a seed point is placed it is advected both upstream and downstream a certain distance. If the field lines don't maintain a minimum distance, d_i, from all other field lines, the seed point is removed. The final set of seed points are stored in a 3D texture. Particle advection is implemented using a fourth-order Runge-Kutta integrator. Particles are added at inflow boundaries using the same scheme as the initial seeding strategy. During the course of the visualization, particles may cluster together producing both regions of high particle density and regions of low particle density. To prevent this particles are removed in regions of high density and new particles are injected into sparse regions of particles. A texture-based approach is then used to generate the field lines.

Reflection: Overall, there has been very little work in seeding of integral curves in 3D, unsteady flow fields. We consider this an open problem. Challenges related to both interactive computation time and perception remain. Also, there is no general consensus on an optimal seeding strategy in 3D.

3. Surface-based Integral Objects

This section describes geometric methods involving surface-based integral objects. Increasing the seeding object dimensionality increases the dimensionality of the resulting integral object. Surfaces have the added benefit of providing greater perceptual information over line primitives, as shading provides better depth cues. Surfaces also suffer to a lesser extent from visual complexity when compared to line primitives as many lines can be replaced by a single surface, providing more spatial coherency. We note that a significant amount of related work has also been carried out in the applied mathematics community. See Krauskopf et al. [KOD*05] for an overview.

3.1. Surface-based Objects in 3D Steady-State Domain

In 1992, Hultquist introduced a novel stream surface construction algorithm [Hul92]. Streamlines are seeded from a curve and are advanced through the vector field. The sampling frequency is updated at the integration step if necessary. This is achieved using distance tests for neighboring streamline front points. For convergent flow the distance between neighboring points reduces and conversely for divergent flow. In the case of divergent flow a new streamline is seeded when the points exceed a pre-determined distance. In the case of convergent flow, the advancement of a streamline may be terminated if neighboring streamlines come too close. These operations help control the density of the points of the advancing front and maintain a sampling frequency that accurately reconstructs the vector field. The streamline points are used for the stream surface mesh. A locally-greedy tiling strategy is used to tile the mesh with triangles to construct the surface. The stream surfaces may also split apart in order to visualize flow around highly divergent areas such as the flow around an object boundary. The stream surfaces are seeded using an interactive seeding rake.

In contrast to the local method of stream surface presented by Hultquist [Hul92], Van Wijk presents a global approach for stream surface generation [vW93b]. A continuous function $f(x,y,z)$ is placed on the boundaries of the data set. A scalar field is then computed throughout the domain by streamlines placed at all grid boundary points and propagating the value of f along the streamline. An iso-surface of this so-called stream function can then be extracted to construct the stream surface. One drawback of this approach is that it only generates stream surfaces that intersect the domain boundary.

Scheuermann et al. present a method of stream surface construction on tetrahedral grids [SBH*01] that builds upon previous work introduced by Hultquist [Hul92]. This method advances the surface through the grid one tetrahedron at a time and calculates where the surface intersects with the tetrahedron. When the surface passes through the tetrahedron the end points are traced as streamlines. For each point on a streamline, a line is added connecting it to its counterpoint. These are then clipped against the faces of the tetrahedron cell and the result is the surface within the cell. Due to the nature of this method, i.e., using the underlying grid in the surface construction process, this method is inherently compatible with multi-resolution grids and thus benefits from the increased grid resolution in interesting flow regions, such as near object boundaries within the flow field.

Brill et al. [BHR*94] introduce the concept of a streamball and apply them to the visualization of steady and unsteady flow fields. Streamballs are defined by a set of discrete points in the vector field based on the metaballs of Wyvill et al. [WMW86]. A ***streamball*** follows the same path of a streamline, however acceleration and deceleration are depicted by the amount of displacement between neighboring spheres. Other local properties of the flow can be mapped to the radius of the sphere.

Using discrete streamball placement it is possible to construct streamlines and pathlines by ensuring that the center points for each streamball are close enough so that they blend together and form an implicit surface [BHR*94]. This technique can also be applied to stream surface construction by advancing a set of streamballs that are seeded along a curve. This produces a smooth surface where the streamballs merge automatically in areas of convergence and split in areas of flow divergence.

Westermann et al. [WJE00] present a level-set method for the visualization of flow fields. Vector-field data is converted into a scalar level-set representation. These level-sets are used to create implicit time surfaces. This approach is similar to the implicit stream surfaces method of van Wijk [vW93b]. Once we have this scalar representation the surfaces can then be computed using an iso-surface extraction technique.

Garth et al. [GTS*04] introduce a stream surface technique that handles areas of intricate flow more accurately than previous techniques such as the method presented by

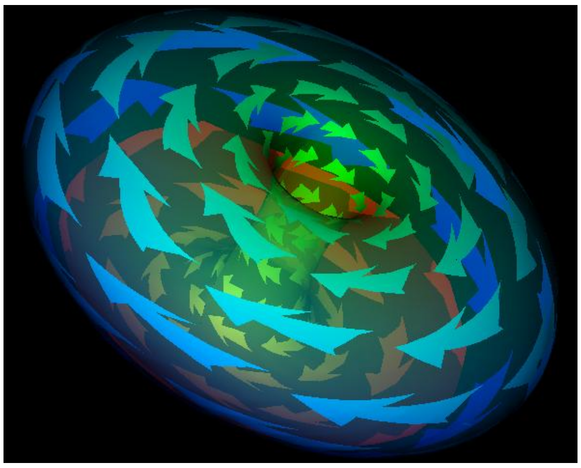

Figure 9: *Stream-arrows textured to a streamsurface. The portions outside the arrows are semi-transparent reducing the occlusion by the surface [Löf98]. Image courtesy of Helwig Hauser.*

Hultquist [Hul92]. The algorithm is demonstrated in the context of vortex structures and is based upon an advancing front with the insertion and deletion of points at each integration step in order to maintain sufficient resolution for the construction of an accurate stream surface. In order to handle more complex flow regions, a high-order integrator is used as well as streamline integration being based upon arc length as opposed to parameter length, as used in Hultquist's method [Hul92]. This results in an improved triangulation for the stream surface mesh, i.e. triangles are more regular throughout the mesh. For the insertion of a new streamline the authors also introduce a new rule based upon the angle between triples of neighboring points on the stream surface front.

Löffelmann et al. [LMG97] introduce an extension called stream-arrows for the enhancement of stream surfaces. ***Stream-arrows*** are partitions that are removed from the surface in order to convey inner flow structure within the surface. The removal of the partition also help to reduce occlusion as the area behind the removed portion is visible. In the original stream-arrows algorithm the arrows were placed on the stream surface using regular tiling. However, this technique may provide unsatisfactory results in regions of convergence and divergence, as the arrows may become too small or too large. The hierarchical extension overcome this shortfall by generating a stack of stream-arrow textures, where each texture contains a unique resolution of arrows. The most appropriate texture is then chosen according to the size of the stream surface, this ensures that the stream-arrows all keep a similar size.

Laramee et al. [LGSH06] present a hybrid method by applying texture advection to stream surfaces. This hybrid technique enables the visualization to convey more information about its inner flow structure. This technique has been used on iso-surfaces [LSH04] but the textures can mislead the user. If an iso-surface is generated using velocity magnitude, there is no guarantee that the surface will be everywhere tangent to the flow. This means there may be a component of the flow vector that is, at least in part, orthogonal to the iso-surface and thus the advection may be misleading. Stream surfaces don't suffer from this weakness as they are, by definition, always aligned with the flow field.

Peikert and Sadlo [PS09] present a hybrid seeding and construction method for topologically relevant stream surfaces. The topology of the velocity field is used to compute the seeding curve of the most expressive stream surfaces. Seeding for cases such as periodic orbits and critical points is described to ensure that the stream surface is not multiply covered. Cases of open and closed seeding curves are described. A quad-based construction method is used to compute the stream surfaces. Quadrilateral cells are added at the front of the surface enclosed by streamlines and orthogonal curves. Sinks within the velocity field are detected when the orthogonal edge segments become too small. These edges are flagged as inactive. Integration terminates when there are no more active edges. Saddle points are handled by tearing the surface resulting in a closed surface front becoming an open one and an already open front splitting into separate portions.

Reflection: A milestone in this category was that of Hultquist who introduced the first stream surfaces for 3D-steady flow [Hul92]. The work that follows improves that work in terms of performance and perception. A few of the open challenges here relate to performance and perception. How to minimize occlusion while maximizing coverage is a big challenge. Interactive computation of stream surfaces for unstructured grids still remains to be seen.

3.2. Surface-based Objects in 3D Time-Dependent Domain

Schafhitzel et al. [STWE07] introduce a point-based stream surface construction and rendering method. This method is also applicable to unsteady flow fields, for which it generates path surfaces. This method was implemented to exploit graphics hardware acceleration and therefore all data structures used lend themselves to being stored in textures. Like the method presented by Hultquist [Hul92] this algorithm includes operations to adjust the density of the surface front. By adding or removing points this method updates the sampling frequency and allows for accurate surface construction in flow exhibiting local convergent or divergent behavior. A method and conditions for splitting the surface are also implemented which are similar to the technique Hultquist [Hul92] used in his algorithm.

In order to render the surface, particles are distributed

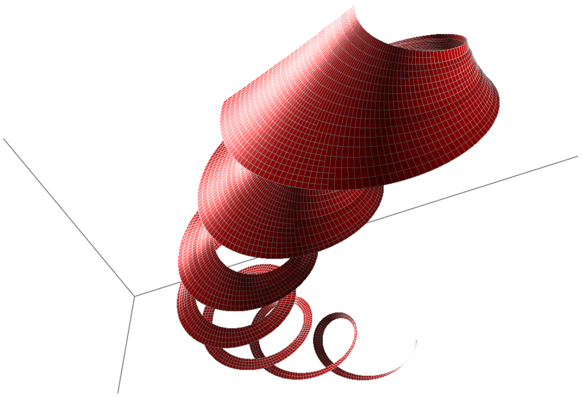

Figure 10: *A stream surface visualizing a tornado simulation [MLZ09b]. Surfaces reduce visual complexity compared to line primitives.*

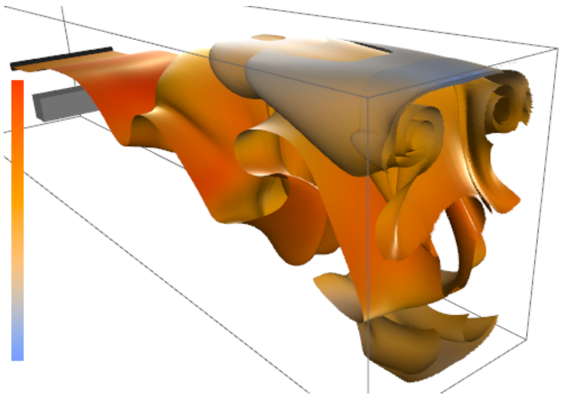

Figure 11: *A streaksurface depicting a flow behind a square cylinder simulation. Streaksurfaces are well suited for visualizing the complex structures occurring in unsteady flow [MLZ09a].*

with a sufficient density (to cover the image space represented by the surface) so that point sprites can be used, this results in a closed surface. Surface normals can be estimated for each particle and this allows the surface to benefit from shading and its associated advantages, i.e. greater depth cues etc. A surface-based LIC algorithm was also applied in order to provide internal visual structure for the surfaces.

Von Funck et al. present a novel technique for smoke surface construction that provides nearly interactive framerates by avoiding the expensive mesh re-triangulation at every time step [vFWS*08]. No re-triangulation of the mesh leads to irregular triangles due to flow characteristics such as divergence. This method exploits these irregular triangles and maps the opacity of the triangle to its size and shape. This provides a fair approximation of the optical model for smoke resulting in surfaces that give a smoke-like effect. A number of enhancements are also given, showing how simple modifications to the core algorithm can be used to simulate smoke injection from nozzles and wool tufts attached to the boundary surface of geometries within the flow domain.

Garth et al. present a novel stream and path surface technique focusing on accuracy [GKT*08]. This method defines refinement criteria that are based upon the order of continuity of the surface lines. New points are added when the curves need to be refined. When discontinuities of first and second order are encountered, the portions of the surface either side of the discontinuity are treated independently of each other, like the surface tearing as handled by Hultquist [Hul92]. This method has no mechanism for removing points from the surface, where sampling is more than sufficient. They trade the cost of the extra integrations for the cost of performing the tests for redundant points and their removal. The meshlines are stored as polynomials and the mesh is discretized after the entire advection stage.

McLoughlin et al. [MLZ09b] present a simplified stream and path surface construction technique (Figure 10). This method is closely related to Hultquist's technique [Hul92]. This technique makes use of simpler data structures – a 2D array, compared to the tracer and ribbon structures used by Hultquist. This simpler approach is made possible by the use of quad primitives for the surface construction. Quads lend themselves naturally to a 2D array, which forms an implicit parameterization of the surface. However, memory may be wasted as elements of the 2D array may be empty, containing no geometry information, but maintain the parameterization property of the surface. Insertion and deletion of vertices is performed to maintain a sufficient sampling of the vector field. This is achieved by processing the mesh quad by quad, dividing and merging quads when a change in resolution is required. Shear flow is also handled by an adaptive step-size integration technique along the quad edge to ensure the quads are more regular.

Krishnan et al. [KGJ09] present a novel streak and time surface algorithm (See Figure 11). This technique guarantees a C^1 continuous curve for the integral curves due to the use of an adaptive step-size 5th-order Runge-Kutta outputting a sequence of fourth-order polynomials. This allows for interim points to be computed easily and provides more accurate results than gained from a simple piecewise linear interpolation between sample points. Three basic operations are defined for the surface adaptation process, these are *edge split*, *edge flip* and *edge collapse*. An edge split ensures that no edge on a triangle is longer than a prescribed threshold. A new vertex is inserted and this is used to create a new integral curve. Egde flipping locally refines an area to maximize the minimum angles within the triangles such that triangles are more regular. Edge collapse removes edges from the mesh in regions where the density of triangles is too high. This prevents the unnecessary propagation of curves in future computations. The algorithm is demonstrated on large unsteady unstructured grid simulations, which inherently consume much processing effort. It is shown that this technique lends itself to parallelism.

Bürger et al. [BFTW09] present two streak surface techniques implemented on the GPU. These techniques provides interactive rates throughout the surface construction. The first technique is based on quads. Each quadrilateral patch contains four vertices. The same vertex is stored (and propagated) multiple times. Refinement of patches is achieved by splitting the longest edge of the quadrilateral and the edge opposite it. This may result in discontinuities within the mesh. A two-pass rendering operation ensures that the quads form a smooth surface during the rendering phase. The first pass creates a *depth imprint* of the enlarged quadrilateral patches in respect to the position of the viewer. On the second pass a biased depth test is used on the depth imprint to ensure that only patch samples close to the surface are used. A smooth transition for shading, coloring etc. is done by using a Gaussian kernel at each path centroid to weight the attributes which are finally accumulated using additive blending and normalization.

The second technique more closely follows the more common mesh approaches. Duplicate vertices are not stored explicitly. This is handled by memory layout in the vertex buffer. Surface refinement is performed in three stages: timeline refinement, connectivity update, streakline refinement. During timeline refinement particles may be inserted, spawning new streaklines, or particles may be removed. Connectivity is updated by each particle on a timeline searching along neighboring timelines for the closest match based on a uniqueness criterion. In the streakline refinement phase a test for the maximum Euclidean distance is performed for neighboring timelines. If the distance is above or below given thresholds, an *entire* streakline is added or removed respectively.

Reflection: Constructing and rendering of integral surfaces in 3D unsteady flow is clearly an unsolved problem with various challenges remaining including performance and perception. Current solutions do not handle shear flow very well. Also, the combination of large datasets and interaction still poses challenges. This category of techniques is currently and active area of research.

4. Volume Integral Objects

This section describes geometric methods involving 2D surface or planar-based seeding objects. Once more, increasing the dimensionality of the seeding object increases the dimensionality of the integral object. The result is a geometric object that sweeps a volume. The volume of these objects can be used to depict flow characteristics such flow convergence and divergence.

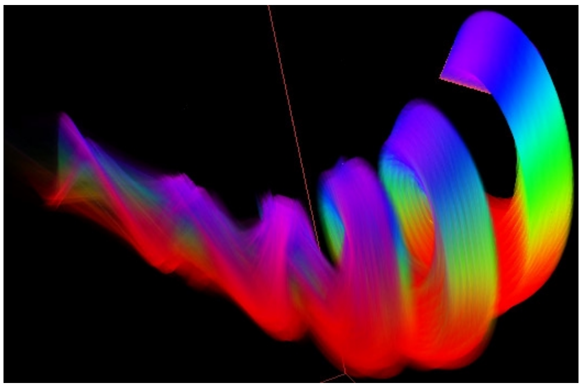
Figure 12: *A flow volume created using the tornado dataset. Image courtesy of Roger Crawfis [MBC93].*

4.1. Volumetric Integral in a 3D Steady-State Domain

Schroeder et al. introduce the *Stream Polygon* [SVL91]. A ***stream polygon*** is a regular n-sided polygon that is oriented normal to the vector field. The stream polygon can be used by placing a new polygon for each point of a streamline or it may be swept along the streamline to form a tube. The polygon is deformed according to the local flow properties. Rotation of the polygon reflects the local vorticity of the flow field. There are no constraints on the polygon maintaining a rigid body structure, therefore deformation of the polygon is used to illustrate the local strain of the flow field.

Max et al. introduce flow volumes [MBC93]. A ***flow volume*** (Figure12) is the volumetric equivalent of a streamline. This method draws inspiration from experimental flow visualization, using a tracer material released into a fluid flow. As the trace propagates through the flow it forms into a flow volume. The flow volume is divided into a set of tetrahedra, the projection of which are divided into triangles. Color and opacity are computed for each tetrahedra using the density emitter model of Sabella [Sab88]. The contributing pixel values can be composited in an arbitrary manner, thus negating the need for a complex sorting algorithm for the volumetric cells. This is suitable as it produces a reasonable approximation of the tracer material effect that the authors are aiming for. An interactive seeding object is used which is always oriented normal to the local flow field and allows the user to change attributes of the seed object such as: position, color and opacity of the smoke and the number of sides for the seeding polygon.

Xue et al. introduce implicit flow volumes [XZC04]. This idea builds upon flow volumes introduced by Max et al. [MBC93] and the implicit stream surface technique presented by Van Wijk [vW93b]. Two techniques are presented for the rendering of the implicit flow volume, a slice-based 3D texture mapping and interval volume rendering. The first approach renders the flow field directly without the inflow

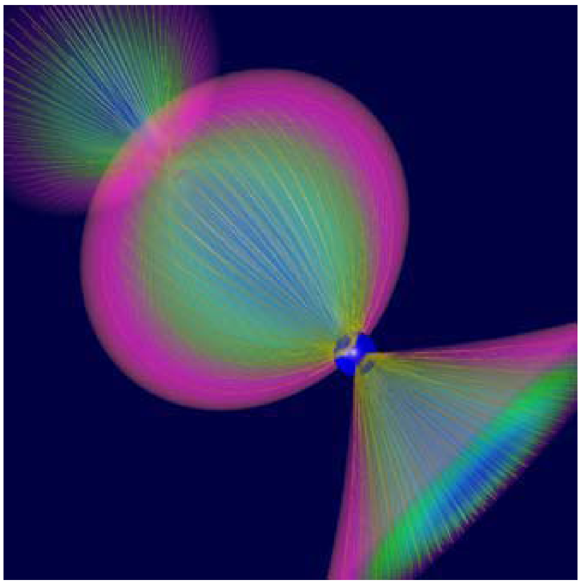

Figure 13: *An implicit flow volume based on the technique of Xue et al. [XZC04]. Image courtesy of Roger Crawfis*

mapping to a scalar field, as used by Van Wijk [vW93b]. Volume shaders can be used to change the appearance and representation of the flow volume. This method allows for high levels of interactivity and fine texture detail in all regions of the flow volume (See Figure 13). The second approach utilizes a flow mapping that produces a scalar field, the flow volume created from the interval volume enclosed between two iso-surfaces. The rendering of the volume is then achieved by using a tetrahedron-based technique.

4.2. Volumetric Integral in a 3D Time-Dependent Domain

Becker et al. extend the flow volume technique [MBC93] for the use with unsteady vector fields [BLM95]. Flow volumes in steady flow fields are created using a set of streamlines seeded from a polygon oriented normal to the local flow. To extend this to unsteady flow Becker et al. construct the flow volumes using streaklines. As in the steady case, the volume is divided up into tetrahedra and volume rendered using hardware. Using streaklines instead of streamlines introduces several complications to the initial flow volume strategy. In the steady case, only the end points of each streamline are advected and a new layer is added to the end of the flow volume in the downstream direction during each integration step. However, when streaklines are used, every point on the streakline must be advected (not just the end points). This may result in the flow volume geometry changing over time. The subdivision strategy used in [MBC93] was performed only at the end of the flow volume, but the changing geometry here requires a subdivision strategy that operates anywhere in the volume. A subdivision created in a previous time step may be unnecessary in future time steps. Subdivision in time is also required, if all particles within a given layer exceed a given distance threshold from the previous layer, a new layer is inserted between them. The reverse is also applicable with the possibility that a layer may be removed.

Reflection: Work on volume integral objects is clearly less mature in comparison to research using curve and surface-based solutions. Also, no individual contribution has triggered a chain of follow-up approaches offering enhancements to the original. Both computational and perceptual challenges remain in this area. Seeding is also a challenge that has not been addressed.

5. Discussion and Conclusions

A variety of techniques have been discussed, each with their own relative merits and shortcomings. As clearly illustrated in this literature, there is no single visualization tool which provides optimal results for all given phenomena. The most appropriate method is dependent upon several factors such as the data dimensionality (both spatial and temporal). The size of the simulation output and the goal of the user are also factors, i.e., is the visualization to be used for detailed investigation in specific regions, is the visualization intended for fast exploration of the vector field or for high quality presentation purposes.

In the context of geometric approaches, a large volume of effort has been placed on streamlines. Streamlines are an effective tool and coupled with an effective seeding strategy may produce some insightful visualizations. The success of streamlines comes partly from their ease of implementation and the quality of the results produced. Streamline enhancements tend to fall in one of two categories, particle tracing or seeding algorithms. The particle tracing algorithms also have their own subset of classifications, with their contributions differentiating them from other tracing algorithms. Most effort has been undertaken on providing ever faster tracing, while other methods have focused tracing on specific grid types, while others on accuracy, producing exact results rather than approximations when using a numerical integration method. Particle tracing methods were a popular area of research in the 1990's with comparatively very little work been undertaken recently. This may be due to the efficiency of current methods, making it more difficult to obtain further significant gains in performance.

Seeding strategies have been heavily researched and are still an area of active research. Many algorithms are based on providing aesthetic, insightful visualizations in image-space. The focus of these papers tends to be on producing uncluttered visualizations that avoid bombarding the user with visual overload. The majority of seeding algorithms have been targeted at 2D domains with only a handful being extended or specifically aimed at higher spatial dimensions. We believe that seeding in 3D domains is still a fruitful area of research and seeding strategies may be extended to providing efficient algorithms for unsteady flow fields.

During the composition of this survey we identified an interesting trend. As the dimensionality of the integral object increases the volume of research decreases, this is evident from Table 1. We believe that this is due to the added complexity of creating higher dimensional geometric objects and ensuring that they maintain an accurate representation of the underlying vector field. We have already stated the advantages that surfaces and volumes present over line primitives, but these advantages are countered by the difficulty of creating a suitable mesh for a surface or volume. Stream surfaces are a very useful tool for flow visualization, however, research and their implementation in industrial applications is limited. While several algorithms exist for stream surface construction, their complexity is often a barrier for a developer. We believe better construction methods that are both efficient and simple are possible. The extension to unsteady flow fields is also an attractive prospect with relatively few papers explicitly showing a time-dependent implementation using surfaces. We also note that there are very few strategies that focus on automatic placement of these structures in the flow field. We expect this to be a very active area of research in the coming years.

Many of the techniques here are used in commercial applications. The array of tools is too vast for all of them to be combined into a single package and so the most appropriate subset must be chosen. This will be more successful if the application designer works closely with a CFD engineer who has expert knowledge of the simulations they will be creating and the phenomena they wish to investigate.

Some of the key areas we identified needing additional work are:

- Higher dimensional (both spatial and temporal) data domain seeding strategies.
- Uncertainty visualization tools for geometric techniques.
- Comparative visualization tools for geometric techniques.
- Improved surface and volume construction methods.
- Surfaces for visualizing unsteady flow fields.
- Automatic seeding for surfaces and volumes.
- Interactive, real-time visualization of unsteady flow fields. Specialized vector field compression techniques must be used to reduce the bottleneck of loading new time steps into the GPU.
- GPU-based methods on unstructured grids.

A large amount of success has been gained for 2D vector fields and more recent techniques are offering a less significant improvement over current methods. Three-dimensional vector fields have also attained a high level of progress. However due to the added challenges, there is still room for significant improvement in many areas. Similarly, many problems remain unsolved when visualizing unsteady flow and the barriers are constantly pushed as simulations are increasing the size of their output and ever more efficient methods are required to address this expansion.

6. Acknowledgements

This work was funded in part by EPSRC Research Grant EP/F002335/1 on the project Interactive, High-Dimensional Flow Visualization Using Image Space-Based Approaches.

References

[BFTW09] Bürger K., Ferstl F., Theisel H., Westermann R.: Interactive Streak Surface Visualization on the GPU. *IEEE Transactions on Visualization and Computer Graphics 15*, 6 (November-December 2009), 1259–1266. 4, 17

[BHR*94] Brill M., Hagen H., Rodrian H.-C., Djatschin W., Klimenko S. V.: Streamball Techniques for Flow Visualization. In *Proceedings IEEE Visualization '94* (Oct. 1994), pp. 225–231. 4, 14

[BL92] Bryson S., Levit C.: The Virtual Wind Tunnel. *IEEE Computer Graphics and Applications 12*, 4 (July 1992), 25–34. 4, 5, 10, 13

[BLM95] Becker B. G., Lane D. A., Max N. L.: Unsteady Flow Volumes. In *Proceedings IEEE Visualization '95* (1995), pp. 329–335. 4, 10, 18

[BMP*90] Bancroft G. V., Merritt F. J., Plessel T. C., Kelaita P. G., McCabe R. K., Globus A.: FAST: A Multiprocessed Environment for Visualization of Computational Fluid Dynamics. In *Proceedings of IEEE Visualization* (1990), pp. 14–27. 4, 6, 8, 10

[BS87] Buning P. G., Steger J. L.: Graphics and Flow Visualization in Computational Fluid Dynamics. In *Proc. American Institute of Aeronautics and Astronautics 8th Computational Fluid Dynamics Conf* (1987), pp. 814–820. 4, 8

[BSK*07] Bürger K., Schneider J., Kondratieva P., Krüger J., Westermann R.: Interactive Visual Exploration of Unsteady 3D Flows. In *Proc. EuroVis* (2007), pp. 251–258. 2, 4, 5

[Bun89] Buning P. G.: Numerical Algorithms for CFD Post-Processing. *van Karman Inst. for Fluid Dynamics* (1989), 1–20. 4, 8

[CCK07] Chen Y., Cohen J. D., Krolik J.: Similarity-Guided Streamline Placement with Error Evaluation. *IEEE Trans. Vis. Comput. Graph. 13*, 6 (2007), 1448–1455. 4, 12

[CSvS86] Cuvelier C., Segal A., van Steenhoven A.: *Finite Element Methods and Navier-Stokes Equations*. Springer, 1986. 2

[DGKP09] DUSSEL D., GRIFFITH E. J., KOUTEK M., POST F. H.: *Interactive Particle Tracing for Visualizing Large, Time-Varying Flow Fields*. Tech. rep., TU Delft Data Visualization Group, 2009. 2

[FG98] FUHRMANN A. L., GRÖLLER M. E.: Real-Time Techniques for 3D Flow Visualization. In *Proceedings IEEE Visualization '98* (1998), IEEE, pp. 305–312. 4, 12

[GH90] GILES M., HAIMES R.: Advanced Interactive Visualization for CFD. *Computing Systems in Education 1*, 1 (1990), 51–62. 8

[GKT*08] GARTH C., KRISHNAN H., TRICOCHE X., BOBACH T., JOY K. I.: Generation of Accurate Integral Surfaces in Time-Dependent Vector Fields. *Proceedings of IEEE Visualization 2008* (Oct. 2008). 4, 16

[GTS*04] GARTH C., TRICOCHE X., SALZBRUNN T., BOBACH T., SCHEUERMANN G.: Surface Techniques for Vortex Visualization. In *Data Visualization, Proceedings of the 6th Joint IEEE TCVG–EUROGRAPHICS Symposium on Visualization (VisSym 2004)* (May 2004), pp. 155–164. 4, 14

[Hai94] HAIMES R.: *pV3: A Distributed System for Large-Scale Unsteady CFD Visualization*. Paper 94-0321, AIAA, 1994. 9, 10

[HE06] HELGELAND A., ELBOTH T.: High-Quality and Interactive Animations of 3d Time-Varying Vector Fields. *IEEE Transactions on Visualization and Computer Graphics 12*, 6 (2006), 1535–1546. 4, 13

[HG91] HAIMES R., GILES M.: *VISUAL3 Interactive Unsteady Unstructured 3D Visualization*. Tech. Rep. 91-0794, AIAA Paper, 1991. 8

[HP93] HIN A. J. S., POST F. H.: Visualization of turbulent flow with particles. In *Proceedings of IEEE Visualization '93* (1993), pp. 46–53. 4, 11

[Hul92] HULTQUIST J. P. M.: Constructing Stream Surfaces in Steady 3D Vector Fields. In *Proceedings IEEE Visualization '92* (1992), pp. 171–178. 4, 13, 14, 15, 16

[JL97a] JOBARD B., LEFER W.: Creating Evenly–Spaced Streamlines of Arbitrary Density. In *Proceedings of the Eurographics Workshop on Visualization in Scientific Computing '97* (1997), vol. 7, pp. 45–55. 4, 5, 6, 7, 8, 11, 12, 13

[JL97b] JOBARD B., LEFER W.: The Motion Map: Efficient Computation of Steady Flow Animations. In *Proceedings IEEE Visualization '97* (Oct. 19–24 1997), IEEE Computer Society, pp. 323–328. 4, 6

[JL00] JOBARD B., LEFER W.: Unsteady Flow Visualization by Animating Evenly-Spaced Streamlines. In *Computer Graphics Forum (Eurographics 2000)* (2000), vol. 19(3), pp. 21–31. 4, 7

[JL01] JOBARD B., LEFER W.: Multiresolution Flow Visualization. In *WSCG 2001 Conference Proceedings* (Plzen, Czech Republic, February 2001), pp. 33–37. 4, 6

[KGJ09] KRISHNAN H., GARTH C., JOY K. I.: Time and streak surfaces for flow visualization in large time-varying data sets. *IEEE Transactions on Visualization and Computer Graphics 15*, 6 (Oct. 2009), 1267–1274. 4, 16

[KKKW05] KRÜGER J., KIPFER P., KONDRATIEVA P., WESTERMANN R.: A Particle System for Interactive Visualization of 3D Flows. *IEEE Transactions on Visualization and Computer Graphics 11*, 6 (2005), 744 – 756. 4

[KL95] KENWRIGHT D. N., LANE D. A.: Optimization of Time-Dependent Particle Tracing Using Tetrahedral Decomposition. In *Proceedings of IEEE Visualization 1995* (1995), pp. 321–328. 4, 10

[KL96] KENWRIGHT D. N., LANE D. A.: Interactive Time-Dependent Particle Tracing Using Tetrahedral Decomposition. *IEEE Transactions on Visualization and Computer Graphics 2*, 2 (June 1996), 120–129. 4, 9, 10

[KM92] KENWRIGHT D. N., MALLINSON G. D.: A 3-D Streamline Tracking Algorithm Using Dual Stream Functions. In *VIS '92: Proceedings of the 3rd conference on Visualization '92* (Los Alamitos, CA, USA, 1992), IEEE Computer Society Press, pp. 62–68. 4

[KOD*05] KRAUSKOPF B., OSINGA H. M., DOEDEL E. J., HENDERSON M. E., GUCKENHEIMER J., VLADIMIRSKY A., DELLNITZ M., JUNGE O.: A survey of methods for computing (un)stable manifolds of vector fields. *I. J. Bifurcation and Chaos 15*, 3 (2005), 763–791. 14

[Lan93] LANE D.: Visualization of Time-dependent Flow Fields. In *Proceedings of Visualization '93* (1993), pp. 32–38. 4, 10

[Lan94] LANE D. A.: UFAT: A Particle Tracer for Time-dependent Flow Fields. In *Proceedings of Visualization '94* (1994), pp. 257–264. 4, 10

[Lar02] LARAMEE R. S.: Interactive 3D Flow Visualization Using a Streamrunner. In *CHI 2002, Conference on Human Factors in Computing Systems, Extended Abstracts* (April 20–25 2002), ACM SIGCHI, ACM Press, pp. 804–805. 13

[LGD*05] LARAMEE R. S., GARTH C., DOLEISCH H., SCHNEIDER J., HAUSER H., HAGEN H.: Visual Analysis and Exploration of Fluid Flow in a Cooling Jacket. In *Proceedings IEEE Visualization 2005* (2005), pp. 623–630. 4, 13

[LGSH06] LARAMEE R. S., GARTH C., SCHNEIDER J., HAUSER H.: Texture-Advection on Stream Surfaces: A Novel Hybrid Visualization Applied to CFD Results. In *Data Visualization, The Joint Eurographics-IEEE VGTC Symposium on Visualization (EuroVis 2006)* (2006), Eurographics Association, pp. 155–162,368. 4, 15

[LH05] LARAMEE R. S., HAUSER H.: Geometric Flow Visualization Techniques for CFD Simulation Data. In

Proceedings of the 21st Spring Conference on Computer Graphics (May 2005), pp. 213–216. 4, 12

[LHD*04] LARAMEE R. S., HAUSER H., DOLEISCH H., POST F. H., VROLIJK B., WEISKOPF D.: The State of the Art in Flow Visualization: Dense and Texture-Based Techniques. *Computer Graphics Forum 23*, 2 (June 2004), 203–221. 3

[LHS08] LI L., HSIEN H. H., SHEN H. W.: Illustrative streamline placement and visualization. In *IEEE Pacific Visualization Symposium 2008* (2008), pp. 79–86. 4, 6

[LHZP07] LARAMEE R., HAUSER H., ZHAO L., POST F. H.: Topology Based Flow Visualization: The State of the Art. In *Topology-Based Methods in Visualization (Proceedings of Topo-in-Vis 2005)* (2007), Mathematics and Visualization, Springer, pp. 1–19. 3

[LJL04] LEFER W., JOBARD B., LEDUC C.: High-quality animation of 2d steady vector fields. *IEEE Transactions on Visualization and Computer Graphics 10*, 1 (2004), 2–14. 4, 6

[LM06] LIU Z. P., MOORHEAD, II R. J.: An Advanced Evenly-Spaced Streamline Placement Algorithm. *IEEE Transactions on Visualization and Computer Graphics 12*, 5 (Sep/Oct 2006), 965–972. 4, 6

[LMG97] LÖFFELMANN H., MROZ L., GRÖLLER E.: Hierarchical Streamarrows for the Visualization of Dynamical Systems. In *Visualization in Scientific Computing '97* (1997), Eurographics, Springer-Verlag, pp. 155–164. 4, 15

[Löf98] LÖFFELMANN H.: *Visualizing Local Properties and Characteristic Structures of Dynamical Systems*. PhD thesis, Technical University of Vienna, 1998. 15

[LPSW96] LODHA S. K., PANG A., SHEEHAN R. E., WITTENBRINK C. M.: UFLOW: Visualizing Uncertainty in Fluid Flow. In *Proceedings IEEE Visualization '96* (Oct. 27–Nov. 1 1996), pp. 249–254. 4, 9

[LS07] LI L., SHEN H.-W.: Image-Based Streamline Generation and Rendering. *IEEE Transactions on Visualization and Computer Graphics 13*, 3 (2007), 630–640. 4, 12

[LSH04] LARAMEE R. S., SCHNEIDER J., HAUSER H.: Texture-Based Flow Visualization on Isosurfaces from Computational Fluid Dynamics. In *Data Visualization, The Joint Eurographics-IEEE TVCG Symposium on Visualization (VisSym '04)* (2004), Eurographics Association, pp. 85–90,342. 15

[LWSH04] LARAMEE R. S., WEISKOPF D., SCHNEIDER J., HAUSER H.: Investigating Swirl and Tumble Flow with a Comparison of Visualization Techniques. In *Proceedings IEEE Visualization 2004* (2004), pp. 51–58. 4, 13

[MAD05] MEBARKI A., ALLIEZ P., DEVILLERS O.: Farthest Point Seeding for Efficient Placement of Streamlines. In *Proceedings IEEE Visualization 2005* (2005), pp. 479–486. 4, 6, 8

[MBC93] MAX N., BECKER B., CRAWFIS R.: Flow Volumes for Interactive Vector Field Visualization. In *Proceedings IEEE Visualization '93* (Oct. 1993), IEEE Computer Society, pp. 19–24. 4, 17, 18

[MHHI98] MAO X., HATANAKA Y., HIGASHIDA H., IMAMIYA A.: Image-Guided Streamline Placement on Curvilinear Grid Surfaces. In *Proceedings IEEE Visualization '98* (1998), pp. 135–142. 4, 7

[MLZ09a] MCLOUGHLIN T., LARAMEE R. S., ZHANG E.: *Constructing Streak Surfaces for 3D Unsteady Vector Fields*. Tech. rep., Swansea Univeristy Visual and Interactive Computing Group, 2009. 16

[MLZ09b] MCLOUGHLIN T., LARAMEE R. S., ZHANG E.: Easy Integral Surfaces: A Fast, Quad-based Stream and Path Surface Algorithm. In *Proceedings Computer Graphics International 2009* (2009), pp. 67–76. 4, 16

[MPSS05] MALLO O., PEIKERT R., SIGG C., SADLO F.: Illuminated Lines Revisited. In *Proceedings IEEE Visualization 2005* (2005), pp. 19–26. 4, 11

[MT*03] MATTAUSCH O., THEUSSL T., , HAUSER H., GROLLER E.: Strategies for Interactive Exploration of 3D Flow Using Evenly-Spaced Illuminated Streamlines. In *Proceedings of the 19th Spring Conference on Computer Graphics* (2003), pp. 213–222. 4, 11

[NJ99] NIELSON G. M., JUNG I.-H.: Tools for Computing Tangent Curves for Linearly Varying Vector Fields over Tetrahedral Domains. *IEEE Transactions on Visualization and Computer Graphics 5*, 4 (Oct. – Dec. 1999), 360–372. ISSN 1077-2626. 4, 9

[NJS*97] NIELSON G. M., JUNG I. H., SRINIVASAN N., SUNG J., YOON J. B.: *Scientific Visualization, Overviews, Methodologies, and Techniques*. IEEE Computer Society, 1997, ch. Tools for Computing Tangent Curves and Topological Graphs for Visualizing Piecewise Linearly Varying Vector Fields over Triangulated Domains, pp. 527–562. 9

[PS09] PEIKERT R., SADLO F.: Topologically Relevant Stream Surfaces for Flow Visualization. In *Proc. Spring Conference on Computer Graphics* (April 2009), Hauser H., (Ed.), pp. 43–50. 4, 15

[PVH*03] POST F. H., VROLIJK B., HAUSER H., LARAMEE R. S., DOLEISCH H.: The State of the Art in Flow Visualization: Feature Extraction and Tracking. *Computer Graphics Forum 22*, 4 (Dec. 2003), 775–792. 2, 3

[RBM87] ROGERS S. E., BUNING P. G., MERRIT F. J.: Distributed Interactive Graphics Applications in Computational Fluid Dynamics. *International Journal of Supercomputer Applications 1*, 4 (1987), 96–105. 4, 8

[Rey95] REYNOLDS O.: On the Dynamical Theory of Incompressible Viscous Fluids and the Determination of the Criterion. *Royal Society of London Philosophical Transactions Series A 186* (1895), 123–164. 11

[RPP*09] ROSANWO O., PETZ C., PROHASKA S., HOTZ I., HEGE H.-C.: Dual streamline seeding. In *Proceedings of the IEEE Pacific Visualization Symposium* (2009), Eades P., Ertl T., Shen H.-W., (Eds.), pp. 9 – 16. 4, 8

[Sab88] SABELLA P.: A Rendering Algorithm for Visualizing 3D Scalar Fields. In *SIGGRAPH '88: Proceedings of the 15th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1988), ACM, pp. 51–58. 17

[SBH*01] SCHEUERMANN G., BOBACH T., HAGEN H., MAHROUS K., HAMANN B., JOY K. I., KOLLMANN W.: A Tetrahedral-based Stream Surface Algorithm. In *Proceedings IEEE Visualization 2001* (Oct. 2001), pp. 151–157. 4, 14

[SdBPM98] SADARJOEN I. A., DE BOER A. J., POST F. H., MYNETT A. E.: Particle Tracing in σ-Transformed Grids Using Tetrahedral 6-Decomposition. In *Visualization in Scientific Computing '98* (1998), Eurographics, Springer, pp. 71–80. 4, 9

[SGvR*03] SCHIRSKI M., GERNDT A., VAN REIMERSDAHL T., KUHLEN T., ADOMEIT P., LANG O., PISCHINGER S., BISCHOF C.: ViSTA FlowLib - Framework for Interactive Visualization and Exploration of Unsteady Flows in virtual environments. In *In Proc. of the Eurographics Workshop on Virtual Environments* (2003), pp. 77–85. 4, 5

[She96] SHEWCHUK: Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator. In *WACG: 1st Workshop on Applied Computational Geometry: Towards Geometric Engineering, WACG* (1996), LNCS, pp. 203–222. 6

[SLCZ09] SPENCER B., LARAMEE R. S., CHEN G., ZHANG E.: Evenly-spaced Streamlines for Surfaces. *Computer Graphics Forum 28*, 6 (2009), 1618–1631. 4, 7, 8

[SM04] SCHUSSMAN G. L., MA K. L.: Anisotropic Volume Rendering for Extremely Dense, Thin Line Data. In *Proceedings IEEE Visualization '04* (2004), pp. 107–114. 11

[SRBE99] SCHULZ M., RECK F., BARTELHEIMER W., ERTL T.: Interactive Visualization of Fluid Dynamics Simulations in Locally Refined Cartesian Grids. In *Proceedings IEEE Visualization '99* (1999), pp. 413–416. 4, 9

[STWE07] SCHAFHITZEL T., TEJADA E., WEISKOPF D., ERTL T.: Point-Based Stream Surfaces and Path Surfaces. In *GI '07: Proceedings of Graphics Interface 2007* (New York, NY, USA, 2007), ACM, pp. 289–296. 4, 15

[SVL91] SCHROEDER W., VOLPE C. R., LORENSEN W. E.: The Stream Polygon: A Technique for 3D Vector Field Visualization. In *Proceedings IEEE Visualization '91* (1991), pp. 126–132. 4, 17

[SvWHP97] SADARJOEN I. A., VAN WALSUM T., HIM A. J. S., POST F. H.: *Scientific Visualization, Overviews, Methodologies, and Techniques*. IEEE Computer Society, 1997, ch. Practical Tracing Algorithms for 3D Curvilinear Grids, pp. 299–323. 4, 9

[TB96] TURK G., BANKS D.: Image-Guided Streamline Placement. In *ACM SIGGRAPH 96 Conference Proceedings* (Aug. 1996), pp. 453–460. 4, 5, 6, 7

[TE99] TEITZEL C., ERTL T.: New Approaches for Particle Tracing on Sparse Grids. In *Data Visualization '99*, Eurographics. Springer-Verlag, May 1999, pp. 73–84. 4, 10

[TGE97] TEITZEL C., GROSSO R., ERTL T.: Efficient and Reliable Integration Methods for Particle Tracing in Unsteady Flows on Discrete Meshes. In *Visualization in Scientific Computing '97* (1997), Eurographics, Springer-Verlag Wien New York, pp. 31–42. 4, 10

[TGE98] TEITZEL C., GROSSO R., ERTL T.: Particle Tracing on Sparse Grids. In *Visualization in Scientific Computing '98* (1998), Eurographics, Springer-Verlag Wien New York, pp. 81–90. 4, 10

[USM96] UENG S. K., SIKORSKI C., MA K. L.: Efficient Streamline, Streamribbon, and Streamtube Constructions on Unstructured Grids. *IEEE Transactions on Visualization and Computer Graphics 2*, 2 (June 1996), 100–110. 4, 9

[vFWS*08] VON FUNCK W., WEINKAUF T., SAHNER J., THEISEL H., HEGE H.-C.: Smoke Surfaces: An Interactive Flow Visualization Technique Inspired by Real-World Flow Experiments. *IEEE Transactions on Visualization and Computer Graphics (Proceedings Visualization 2008) 14*, 6 (November - December 2008), 1396–1403. 4, 16

[VKP00] VERMA V., KAO D., PANG A.: A Flow-guided Streamline Seeding Strategy. In *Proceedings IEEE Visualization 2000* (2000), pp. 163–170. 4, 6, 8, 12

[VP04] VERMA V., PANG A.: Comparative flow visualization. *IEEE Trans. Vis. Comput. Graph. 10*, 6 (2004), 609–624. 4, 9, 10

[vW92] VAN WIJK J. J.: Rendering Surface-particles. In *Proceedings of IEEE Visualization '92* (1992), pp. 54–61. 4, 7

[vW93a] VAN WIJK J. J.: Flow Visualization with Surface Particles. *IEEE Computer Graphics and Applications 13*, 4 (July 1993), 18–24. 4, 7

[vW93b] VAN WIJK J. J.: Implicit Stream Surfaces. In *Proceedings of Visualization '93* (1993), pp. 245–252. 4, 14, 17, 18

[WBPE90] WALATKA P., BUNING P., PIERCE L., ELSON P.: *PLOT3D User's Manual*. NASA, mar 1990. http://www.openchannelfoundation.org/ [September 2007]. 8

[WHN*03] WEINKAUF T., HEGE H.-C., NOACK B., SCHLEGEL M., DILLMANN A.: Coherent structures in a transitional flow around a backward-facing step. *Physics of Fluids 15*, 9 (September 2003), S3. 11

[WJE00] WESTERMANN R., JOHNSON C., ERTL T.: A Level-set Method for Flow Visualization. In *VIS '00: Proceedings of the conference on Visualization '00* (Los Alamitos, CA, USA, 2000), IEEE Computer Society Press, pp. 147–154. 4, 14

[WMW86] WYVILL G., MCPHEETERS C., WYVILL B.: Data Structure for *soft* Objects. *The Visual Computer 2*, 4 (1986), 227–234. 14

[WS05] WIEBEL A., SCHEUERMANN G.: Eyelet Particle Tracing - Steady Visualization of Unsteady Flow. In *Proceedings of IEEE Visualization 2005* (2005), pp. 77–84. 4, 13

[WT02] WEINKAUF T., THEISEL H.: Curvature measures of 3d vector fields and their applications. *Journal of WSCG 10*, 2 (2002), 507–514. WSCG 2002, Plzen, Czech Republic, February 4 - 8. 11

[XZC04] XUE D., ZHANG C., CRAWFIS R.: Rendering Implicit Flow Volumes. In *Proceedings IEEE Visualization 2004* (2004), pp. 99–106. 4, 17, 18

[YKP05] YE X., KAO D., PANG A.: Strategy for Seeding 3D Streamlines. In *Proceedings of IEEE Visualization 2005* (2005), pp. 471–476. 4, 12

[ZSH96] ZÖCKLER M., STALLING D., HEGE H.-C.: Interactive Visualization of 3d-Vector Fields using Illuminated Stream Lines. In *Proceedings of IEEE Visualization '96* (1996), pp. 107–ff. 4, 11, 13