Generation of Implicit Flow Representations for Interactive Visual Exploration of Flow Fields

Molchanov Vladimir
The University of Münster
Schlossplatz 2
48149 Münster, Germany
molchano@uni-muenster.de

Lars Linsen
The University of Münster
Schlossplatz 2
48149 Münster, Germany
linsen@uni-muenster.de

ABSTRACT

A stream function is an implicit flow representation in form of a function, whose values are constant along streamlines of the underlying velocity field. To generate a stream function, a common approach is to use a streamline tracking technique after assigning scalar function values on the inflow/outflow domain boundary (pre-processing step). However, non-trivial flows generally have streamlines that do not start or end at the domain boundary. We propose an automatic approach that defines a stream function along such streamlines. To do so, we construct optimal termination surfaces inside the domain and assign scalar values to all streamlines crossing these surfaces. Furthermore, we propose a proper functional to characterize the quality of the approximated stream function. Using a variational approach, we derive a partial differential equation for the minimization of the derived functional. This minimization procedure is an effective tool to improve the stream function. It can also be used to significantly improve the pre-computation times by creating a high-quality high-resolution stream function from a low-resolution estimate. Once the implicit flow representation is established and improved, we can efficiently extract flow geometry such as stream ribbons, stream tubes, stream surfaces, etc. by applying fast marching algorithms. Tracking time recorded during the pre-processing step can be coupled with the stream function or used directly to extract time surfaces. Thus, the entire flow field can be explored interactively. There is no need for time-consuming particle tracking and mesh refinement during the visual exploration process.

Keywords
Flow visualization, streamlines and -surfaces, implicit representation, stream function.

1 INTRODUCTION

Modern flow visualization systems are required to handle large volumetric datasets of high complexity, to extract and transform requested information fast and accurately, and to meet users' intuition and expectation when rendering. The enormous demand on such systems caused an intensive research on this topic over the last decades. As a result there have appeared various visualization algorithms combining ideas from numerical methods, fluid dynamics, geometry, and other fields.

Most of the existing approaches can be classified into four large groups: direct, geometric, texture-based, and feature-based methods [LHD$^+$03]. All these approaches have their own application areas and differ in efficiency, generality, and expressiveness. Direct methods are intuitive but only allow for local comprehension of the flow and are of limited use when considering volume data. Texture-based techniques produce dense flow representations by applying filters to three-dimensional textures. Occlusion becomes an issue. Scalar characteristics are in the focus of feature-based methods, which often require more experience from the user. Geometric approaches are considered to be quite intuitive and expressive.

Our paper is devoted to three-dimensional geometric flow visualization using an implicit flow representation. The core of most geometric approaches is an integration of the flow field, which can be extremely time consuming when postulating high accuracy. To allow for an interactive visualization that involves many geometric objects, the integration needs to be executed in a pre-processing phase. Our algorithm takes advantage of an implicit representation of flow, thus, effectively converting the problem to a scalar field visualization task. Given the implicit flow representation in the form of a collection of stream functions, an extraction and rendering of geometric stream elements is performed efficiently using the available pre-integrated information.

Implicit flow representation is a collection of stream functions together with advection times and lengths recorded for each node. A (generalized) stream function is a non-trivial function, whose values are constant along streamlines of the underlying velocity field. To generate a stream function for a given velocity field, a common approach is to use a streamline tracking technique after assigning scalar function values (parametrization) on the inflow/outflow domain boundary, see Section 3. However, non-trivial flows generally have streamlines that do not start or end at the domain boundary. We propose an automatic approach that defines a stream function along such streamlines. To do so, we construct optimal termination surfaces inside the domain and assign scalar values to all streamlines crossing these surfaces, see Section 4. We also support the interactive modification of position and parametrization of the termination surfaces by the user based on the information obtained by the automatic procedure. After having computed one or several distinct stream functions for gridded data, marching algorithms can be applied to the grid to visualize implicit stream elements, such as streamlines, stream tubes, stream ribbons, stream surfaces, etc, see Section 6. Moreover, tracking time can be recorded during the pre-processing step, which allows for the extraction of time surfaces or for enhancing other stream elements with time information.

Another aspect of our work is concerned with the quality of the stream functions. To our knowledge, there exists no tool to measure and improve the quality of the pre-integrated data. Our efforts were concentrated on developing such an approach that improves a stream function with respect to the underlying velocity field. Using a variational approach, we derive a partial differential equation to optimize the derived quality measure, see Section 5. The procedure can be useful in many regards, including the following:

- *Improvement*: A stream function constructed by tracking of samples may contain noise, exhibit sampling artifacts, or have high local errors due to a non-uniform behavior of the velocity field. Our minimization procedure improves the quality of the stream function and can eliminate these artifacts.
- *Refinement*: Computing a stream function over a large domain can be rather expensive when tracking all nodes. Using our approach, we can downsample the data, compute a coarse approximation of the stream function, use interpolation for upsampling to the original resolution, and correct the interpolated stream function values via the proposed minimization procedure.

The main contributions of the paper can be summarized as follows: (1) Automatic generation of implicit flow representation for the entire flow domain; (2) Termination surfaces to generate a parametrization for streamlines not crossing the domain's boundary; (3) Proper functional to control the stream function quality; (4) Variationally derived procedure for stream functions improvement; (5) Effective algorithms for extraction of various stream elements with the possibility to represent advection-time information in form of color (transparency) encoding or extraction of time surfaces.

2 RELATED WORK

Nontrivial real-world and modeled flows have variations in velocity and curl magnitude, an inhomogeneous distribution of helicity and divergence, and a non-degenerated determinant of the gradient tensor. All these scalar fields associated with a flow are features that play an important role in flow analysis. An approach to highlight regions of a non-uniform flow behavior is to use a multi-dimensional transfer functions [PBL$^+$04, PBL$^+$05], or glyphs [GRT17].

Flow direction – one of the simplest flow characteristics – is hardly described by a scalar quantity. To depict this information the Line Integral Convolution method was proposed by Cabral and Leedom [CL93]. The idea is to blur textures along a given vector field over the domain producing intuitive patterns, especially in two spatial dimensions. In the case of a volumetric flow, the method can be combined with other approaches. For instance, Schafhitzel et al. [STWE07] computed and rendered stream surfaces and path surfaces of a three-dimensional flow with a texture-based surface flow structure.

Rendering of flow-related geometrical objects is an extremely helpful visualization method. Colored points, curves, and surfaces may be used to define the topological skeleton of a vector field, i.e., critical points, periodic orbits, separatrices, etc. Existing approaches focus on topological segmentation of two-dimensional [SHJK00] and three-dimensional steady vector fields [MBS$^+$04], an analysis of time-dependent vector field topology [SRP09], and extraction of two-dimensional separatrices of three-dimensional saddles and saddle type periodic orbits [PS09].

The basic underlying principle of topology-based and geometric methods is the tracking of imaginary particles introduced into the flow. The idea was adopted from real-world experiments on injection of an extraneous, clearly visible fluid material into a stream. Propagation of the material displays the stream- or pathline structure of the flow. A proper optical model for smoke advection in an unsteady flow was proposed [vFWTS08]. Li et al. [LTH08] developed a dye propagation scheme overcoming non-physical artifacts of integration. Cuntz et al. [CKSW08] advected a dye in an unsteady three-dimensional flow using a hybrid particle-mesh formalization. A dye released into the flow at fixed positions at different times results in *streak lines* [WT10].

An integration along particle paths is commonly done by a fourth-order Runge-Kutta method [PYH+06]. The paths describe streamlines or pathlines and can be extended to stream ribbons or stream tubes [RLN+17]. An improvement of stream ribbon triangulation in divergent or shearing flows was studied in the seminal paper by Hultquist [Hul92].

The construction of a flow topology skeleton is mainly done without any user interaction. Although it also requires significant computational efforts, extraction of stream surfaces and lines is more user-oriented, since the seeding points can be defined arbitrarily. Typically, the number of simultaneously extracted stream elements needs to be limited to allow interactive frame rates. One step towards an interactive visualization application that allows the simultaneous extraction of many stream elements can be taken by moving all time-consuming integration to a pre-processing phase and encoding the flow implicitly in a scalar stream function.

An implicit surface representation is the key idea of a wide class of level-set methods, e.g., [CKSW08, WJE00, WJE01]. Early attempts in implicit representations of stream surfaces go back to van Wijk [vW93]. All grid nodes were tracked in the direction opposite to the flow until they reach the domain boundary. The velocity field was evaluated via a trilinear interpolation from the grid. Values of a smooth scalar function defined at the boundary are then assigned to the nodes based on the assumption that they remain constant along each streamline. Alternatively, a convection equation is solved on a regular grid. Isosurfaces of the resulting gridded volumetric function are then proven to be stream surfaces of the underlying flow. Xue et al. [XZC04] adapted the approach by van Wijk to render implicit volumes. Instead of assigning scalar values on the inflow region, the user is asked to paint a two-dimensional texture on the boundary (termination surface). Properly constructed boolean fields which remain unchanged along streamlines allow for effective flow topology exploration as shown in [SS07]. In this paper, we present an approach that computes stream functions fully automatically. Moreover, we define a quality measure and present an approach for improving stream functions.

A streamline can be found as an intersection of two stream surfaces called dual. A cell-wise trilinear approximation of dual stream functions (f and g) was used by Kenwright et al. to render streamlines [KM92]. A concept of an fg-diagram was then generalized to an irregular tetrahedral mesh [KM96].

3 STREAM FUNCTIONS

A stream function $\psi(\mathbf{x})$ of a two-dimensional potential flow $\mathbf{w}(\mathbf{x}) = (w_1(\mathbf{x}), w_2(\mathbf{x}))$, $\mathbf{x} = (x_1, x_2)$, is known to satisfy the Poisson equation $\triangle\psi(\mathbf{x}) = \dfrac{\partial w_2(\mathbf{x})}{\partial x_1} - \dfrac{\partial w_1(\mathbf{x})}{\partial x_2}$, where $\triangle = \frac{\partial}{\partial x_1} + \frac{\partial}{\partial x_2}$ stands for the Laplace operator. The right-hand side of the equation has the meaning of vorticity with a negative sign. The stream function $\psi(\mathbf{x})$ remains constant along streamlines and the magnitude of its gradient is proportional to the flux. This property holds exceptionally for potential flow, i.e., for velocity field $\mathbf{w}(\mathbf{x})$ with $\nabla \times \mathbf{w}(\mathbf{x}) = 0$. However, a generalized notion of a stream function is still applicable for non-potential flows in spatial dimensions higher than two.

A (nontrivial) scalar function $f(\mathbf{x})$ is said to be a (generalized) *stream function* of a given vector field $\mathbf{u}(\mathbf{x})$ (interpreted as velocity), if $\nabla f(\mathbf{x}) \perp \mathbf{u}(\mathbf{x})$ everywhere in a domain $D \in \mathbb{R}^d$, $d \geq 2$. It implies that $f(\mathbf{x})$ is constant along any streamline of the flow $\mathbf{u}(\mathbf{x})$, i.e., an implicit relation $f(\mathbf{x}) = f_{\text{iso}}$ with some constant f_{iso} defines a streamline or a stream surface for $d = 2$ or $d = 3$, correspondingly.

We assume that the underlying vector field is sufficiently smooth, i.e., its components have continuous first derivatives. Since the stream function definition above is invariant under arbitrary scaling of the velocity $\mathbf{u}(\mathbf{x})$, it is convenient to normalize the flow introducing a new field $\mathbf{v}(\mathbf{x}) = \mathbf{u}(\mathbf{x})/\|\mathbf{u}(\mathbf{x})\|$. The boundary ∂D of the flow domain D can be split into two parts, the *inflow boundary region* ∂D_{in} and the *outflow boundary region* ∂D_{out}, i.e., $\partial D = \partial D_{\text{in}} \bigcup \partial D_{\text{out}}$. By definition, $\mathbf{y} \in \partial D_{\text{in}}$ iff $\mathbf{y} \in \partial D$ and $\mathbf{v}(\mathbf{y}) \cdot \mathbf{n}(\mathbf{y}) \leq 0$, where $\mathbf{n}$ is a normal to the boundary ∂D pointing outwards and "$\cdot$" denotes the inner product of vectors in $\mathbb{R}^d$.

There exist two main approaches to construct a stream function $f(\mathbf{x})$. A first approach solves the partial differential transport equation with boundary condition

$$\frac{\partial f(\mathbf{x},t)}{\partial t} + \mathbf{u} \cdot \nabla f(\mathbf{x},t) = 0; \quad f(\mathbf{y},t) = f_0(\mathbf{y}), \quad \mathbf{y} \in \partial D_{\text{in}},$$

to track boundary values throughout the domain along streamlines. Alternatively, all grid nodes $\mathbf{g}_i$ can be tracked backwards in the flow (so called, *anti-particles*). Here, the ordinary differential equation

$$\frac{\mathrm{d}\mathbf{g}_i(t)}{\mathrm{d}t} = -\mathbf{u}, \qquad \mathbf{g}_i(0) = \mathbf{g}_i, \tag{1}$$

is to be solved until each tracked particle reaches ∂D_{in} at some time t_i. After that, the inflow boundary region is parametrized, i.e., some scalar values are prescribed to all inflow boundary points. Then, all grid nodes $\mathbf{g}_i$ are assigned with the same scalar values as their footprints $\mathbf{g}_i(t_i)$. The established volumetric scalar field is

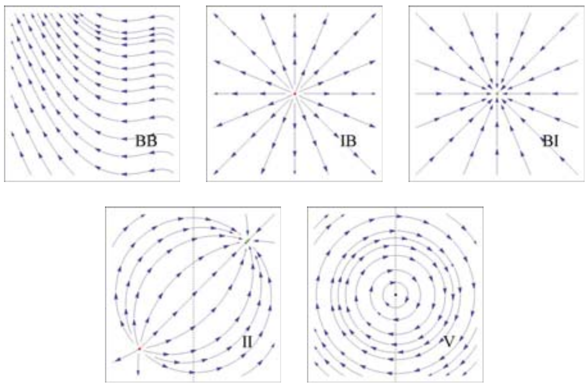

Figure 1: Classification of streamlines with respect to their start and end points lying on the boundary (B) or in the interior (I) of the domain. Case V describes closed streamlines around vortex. By proper splitting (dashed lines), II and V regions can be reduced to two subregions of BB, BI, or IB.

the stream function $f(\mathbf{x})$. The collection of the gridded values t_i determines another scalar field $t(\mathbf{x})$ called the *advection-time function*. In both approaches t stands for an artificial time.

Usually, both the coordinates of the footprints $\mathbf{g}_i(t_i)$ and the advection times t_i are recorded after the backwards tracking step. The user chooses a proper parametrization of ∂D_{in} and specifies an iso-value f_{iso} to extract the implicit stream surface $f(\mathbf{x}) = f_{\text{iso}}$. The advection times t_i can be used either to extract time surfaces $t(\mathbf{x}) = t_{\text{iso}}$ or to color extracted stream surfaces.

A dual technique is to track particles forward in the flow until they reach ∂D_{out} and to record their tracking time. Since we use both of the methods simultaneously in our approach, we denote by t_{in} and t_{out} the advection times by inverse and original flow, respectively.

In most cases, finding a parametrization that results in a globally smooth stream function is not easy for two reasons: First, the domain D is usually chosen to be a rectangular box, which, obviously, has a non-smooth boundary. Second, many flows have streamlines, which do not start on the boundary. We reproduce the flow diagrams from [vW93] in Figure 1. Based on whether a streamline starts/ends on the boundary (B) or in the interior (I), or it forms a loop around a vortex (V), one can classify them in five types: BB, BI, IB, II, and V.

The methods described above require that all streamlines of the flow $\mathbf{u}(\mathbf{x})$ start and/or end at the domain boundary ∂D. However, the presence of sources, sinks, or vortices may lead to stream curves belonging to the domain interior (cases II and V) which remain non-parametrized. These cases can be solved by a proper splitting of domain D into subdomains, see Figure 1, and/or by surrounding singularities with *termination surfaces*.

4 TERMINATION SURFACES

Parametrization of streamlines of type II and V was stated as an open problem by van Wijk [vW93]. Splitting of the flow domain as in Figure 1 (lower row) becomes impractical for three-dimensional fields, since critical points (sinks, sources, vortex cores) can build complicated geometry, e.g. vortex filaments. To handle the II-case with isolated sinks/sources, Xue et al. [XZC04] constructed termination surfaces surrounding the critical points. The streamlines approaching one of these points intersect the corresponding termination surface and pick up a value from its parametrization. However, Xue et al. did not present a methodology on how the radius of the spherical surface should be chosen and left the placement of termination surfaces to the user. Moreover, the streamline density on small spheres is extremely high, which makes the parametrization process unstable with respect to unavoidable tracking errors. Our approach automatically creates termination surfaces inside II or V regions optimally placed with respect to the locations of critical points, which is based on pre-processed information.

In the pre-processing step, we track each grid node forward and backward in the flow to define its type: The type of a node is the type of the streamline the node belongs to. For the nodes of types BB, BI, and IB we record the footprint point(s) and the two advection times. For the nodes of type II we record the grid voxels being visited, the tracking times t_{in} and t_{out} and the advection lengths l_{in} and l_{out}. For the nodes of type V we just record the voxels being visited. As such, we classify all grid nodes. Setting value 1 to all nodes of one class and value 0 to nodes of the other classes, we can extract *separating surfaces* as isosurfaces with respect to the isovalue 0.5. These are stream surfaces that provide important information about the flow structure. However, their quality is low, since they are extracted from a boolean field.

Flow regions that have been categorized as being connected to the domain boundary (types BB, BI, and IB) are then parametrized according to scalar field(s) that are assigned to the domain boundary ∂D. The next step is to create a smooth scalar field for regions of type II and V by constructing proper termination surface(s). For that purpose we first look for a *seeding voxel* S (discussed below). Let $\mathbf{c}$ be the center of S and $\mathbf{m} = \mathbf{v}(\mathbf{c})$ the velocity at $\mathbf{c}$. Starting from the seeding voxel, we grow the termination surface by marking neighboring voxels if they (a) have a non-empty intersection with the plane $\beta : \mathbf{x} \cdot \mathbf{m} = \mathbf{c} \cdot \mathbf{m}$, (b) have unparametrized streamlines crossing them, and (c) their velocity $\mathbf{v}$ has the same orientation as the velocity at $\mathbf{c}$, i.e., $\mathbf{v} \cdot \mathbf{m} > 0$. The procedure results in that part of plane β that is connected with voxel S and has unparametrized streamlines crossing it. Let $\{\mathbf{k}, \mathbf{l}\}$ be an orthonormal basis in plane β.

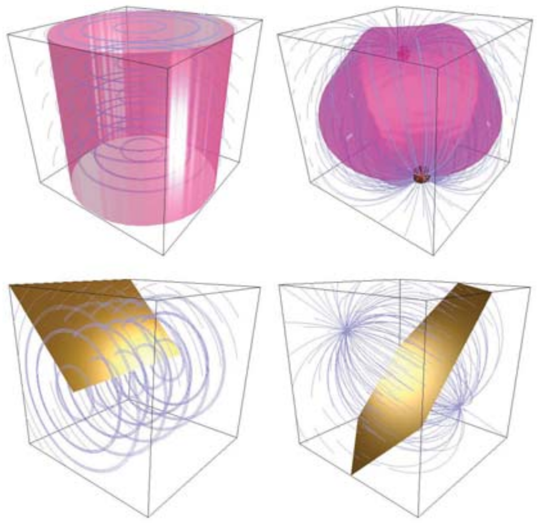
Figure 2: Automatic parametrization of V- and II-regions (left and right column, correspondingly). Upper row: Stream surfaces that separate the V- and II-region from the surrounding regions. Lower row: Termination surfaces provide scalar values for streamlines intersecting them, which allows for the extraction of stream elements.

A streamline crossing the termination surface at point **g** gets assigned a scalar according to $f_1 = (\mathbf{g}-\mathbf{c})\cdot\mathbf{k}$ or $f_2 = (\mathbf{g}-\mathbf{c})\cdot\mathbf{l}$. Both scalars are needed for extracting stream tubes and ribbons as discussed below.

For the selection of seeding voxel S, our aims are (1) to provide scalar values for a maximal number of streamlines at once, and (2) possibly avoid an overly dense local concentration of streamlines on the surface. In other words, we want to parametrize the largest part of the domain and make our parametrization less sensitive to computational errors. Several approaches to choose the seeding voxel were tested in our experiments. We came to the conclusion that for II-region with single source and sink the seeding voxel S should lie half way between the sink and the source on the shortest connecting streamline. Thus, S should contain the grid node with minimal total tracking length $(l_{\text{in}} + l_{\text{out}})$ and minimal tracking length difference $|l_{\text{in}} - l_{\text{out}}|$. In a V-region, on the other hand, tracking time for the streamlines has no meaning , since the streamlines are closed. Thus, the choice of S is arbitrary. Generally, one can find the largest termination surface with the maximal number of streamline crossing it by a brute force algorithm testing all possible seed points. Results are shown in Figure 2.

For each streamline we record a label of the termination surface from which it received the scalar values. If not all streamlines were parametrized, we iteratively build further termination surfaces until all streamlines are parametrized.

5 STREAM FUNCTION CONTROL AND IMPROVEMENT

Streamline tracking introduces numerical errors due to imprecise velocity interpolation and integration. The longer a streamline, the larger the error. To our knowledge, there exists no effective procedure to improve a constructed stream function $f(\mathbf{x})$ other than to reconstruct it again using a smaller integration step size, which is an extremely time-consuming process. Our goal is to develop a method to control and improve the quality of a stream function.

5.1 Functional for measuring stream function quality

We start with a construction of a functional measuring the quality of a stream function $f(\mathbf{x})$ with respect to the underlying normalized vector field $\mathbf{v}(\mathbf{x})$. The fundamental characteristic of a stream function is that its isolines (surfaces) are tangent to the flow direction. Therefore, we define

$$E_1(f) = \frac{1}{2}\int_{D'} \left| \frac{\nabla f(\mathbf{x})}{\|\nabla f(\mathbf{x})\|} \cdot \mathbf{v}(\mathbf{x}) \right|^2 \mathrm{d}\mathbf{x}. \tag{2}$$

Here and in the following D' denotes a subregion in D covered by streamlines of the same type. Streamlines within D' either have a common termination surface or start or end at the boundary $\partial D'$. Obviously, the functional takes values from interval $[0,\, 1]$ and vanishes for a perfect stream function. Our goal is to obtain a method, which allows us to minimize E_1 for a given approximation of $f(\mathbf{x})$.

5.2 Minimization algorithm

A standard technique to minimize a functional of the form $E(\phi) = \int L(\mathbf{x},\phi,\nabla\phi)\,\mathrm{d}\mathbf{x}$ is to construct its Euler-Lagrange equation

$$\frac{\partial L}{\partial \phi} - \mathrm{div}_{\mathbf{x}}\left[\frac{\partial L}{\partial \nabla\phi}\right] = 0. \tag{3}$$

Equation (3) expresses the necessary condition for a stationary point ϕ_0 of the functional and can be derived by usual differentiation of $E(\phi) = E(\phi_0 + \varepsilon\psi)$ with respect to ε.

To simplify the resulting equation, we omit the normalization of the gradient field in Equation (2). Our tests show that this modification reduces the computational costs and still serves the goal of minimization of $E_1(f)$. The simplified functional depends only on the gradient of the function $f(\mathbf{x})$, thus the associated Euler-Lagrange equation reduces to the form

$$-\mathrm{div}_{\mathbf{x}}\left[\mathbf{v}(\mathbf{x})\,(\nabla f(\mathbf{x})\cdot\mathbf{v}(\mathbf{x}))\right] = 0. \tag{4}$$

We introduce an artificial time τ and set the partial temporal derivative of $f(\mathbf{x},\tau)$ to the left-hand side of Equation (4) taken with a negative sign. The resulting evolutional equation

$$\frac{\partial f(\mathbf{x},\tau)}{\partial \tau} = \operatorname{div}_{\mathbf{x}} \left[\mathbf{v}(\mathbf{x}) \left(\nabla f(\mathbf{x},\tau) \cdot \mathbf{v}(\mathbf{x})\right)\right], \quad (5)$$

$$f(\mathbf{x},0) = f_0(\mathbf{x}), \quad (6)$$

describes a transformation of an initial approximated stream function $f_0(\mathbf{x})$ towards a local minimum of functional E_1. The algorithm is similar to the steepest descent method for root search, where the divergence term stands for the opposite gradient direction. The governing equation has the form of diffusion in the direction of $\mathbf{v}(\mathbf{x})$ with the diffusion rate $\nabla f(\mathbf{x},\tau)\cdot\mathbf{v}(\mathbf{x})$. Clearly, the diffusion rate vanishes for the perfect stream function. Thus, the perfect stream function is a stationary point of the evolution process.

We discretize Equation (5) in space and time to derive a numerical scheme. In our tests we use central differencing for spatial and forward differencing for temporal discretization resulting in an explicit scheme with second-order accuracy in space. The discretized partial differential equation has the form

$$\frac{f^{n+1}_{i,j,k} - f^{n}_{i,j,k}}{\delta\tau} = \operatorname{div}_{i,j,k} \left[\mathbf{v}^n_{i,j,k} \left(\nabla_{i,j,k} f^n \cdot \mathbf{v}^n_{i,j,k}\right)\right], \quad (7)$$

where gradient $\nabla_{i,j,k}$ and divergence operator $\operatorname{div}_{i,j,k}$ are discretized using central differences, $\delta\tau$ is the time step, the upper indices denote the time, and the lower indices indicate the position in space.

Equation (5) is a parabolic partial differential equation. Thus, both initial and boundary conditions are required for the well-posedness of the problem. The initial condition is given by Equation (6). Imposing a proper boundary condition is not a trivial task, since numerical instabilities can develop close to the boundary $\partial D'$ of the considered region.

The simplest and the safest approach is to fix the values of the stream function at $\partial D'$ for all τ by imposing the Dirichlet boundary condition: $f(\mathbf{y},\tau) = f_0(\mathbf{y})$ for all $\mathbf{y} \in \partial D'$ and all $\tau \geq 0$. The numerical scheme becomes simple and the functional decreases over the first iterations. Moreover, the initial parametrization of the boundary is not affected.

5.3 Application

The minimization procedure described above can be applied to an already generated stream function to make its level sets be better aligned to the given vector field. Since the governing Equation (5) models a diffusion process, the procedure also has a smoothing effect. Van Wijk [vW93] applied an isotropic smoothing filter to the generated stream function to enhance its rendering quality at the cost of losing detailed information. In the proposed method, the smoothing is performed in accordance with the underlying flow field decreasing the error defined in Equation (2).

Another main application of the minimization algorithm can be the reduction of computation time in the pre-processing stage. Accurate advection of all grid nodes can take hours for large data sets. Even if the pre-processing has to be performed only once, the computational efforts are an issue. We propose to construct a rough approximation to the stream function which subsequently can be improved by applying our minimization procedure. The steps of the algorithm are the following:

1 We perform an advection of three subsets of nodes: (a) the boundary nodes $\mathbf{g}_i \in \partial D'$, (b) nodes having vorticity or absolute divergence values larger than specified thresholds, and (c) an evenly distributed sparse subset of nodes in D'.

2 The advected nodes are parametrized according to their footprints at the boundary.

3 The scalar field sampled at the parametrized nodes is interpolated linearly to the nodes which were not tracked producing a rough approximation to a globally defined stream function.

4 The approximate stream function is improved according to Equation (7). The values at the advected nodes (from Step 1) remain unchanged during this optimization.

5 The minimization process is stopped as soon as the error (2) reaches its minimum.

The vorticity used in Step 1 are given by norm of $\nabla \times \mathbf{v}(x,y,z) = \left(\frac{\partial \mathbf{v}_y}{\partial z} - \frac{\partial \mathbf{v}_z}{\partial y}, \frac{\partial \mathbf{v}_x}{\partial z} - \frac{\partial \mathbf{v}_z}{\partial x}, \frac{\partial \mathbf{v}_y}{\partial x} - \frac{\partial \mathbf{v}_x}{\partial y}\right)$, where derivatives are computed by central differencing. High vorticity values indicate that locally the stream function is highly curved. In the neighborhood of large absolute values of divergence, the norm of the gradient of the stream function can grow quickly. To avoid possible instabilities when evolving $f(\mathbf{x})$ according to Equation (7), we explicitly advect and parametrize grid nodes in regions of high vorticity and high absolute divergence values. Analogously, we can parametrize the streamlines crossing a termination surface.

6 EXTRACTION OF STREAM ELEMENTS

Stream elements are the instruments of geometric flow visualization methods. The most commonly used elements are stream surfaces, streamlines, stream tubes, and stream ribbons. Different stream elements serve for an adequate exploration of different flow characteristics and structures. Since the flow through any stream surface vanishes, i.e., $\mathbf{v} \cdot \mathbf{m} = 0$ with surface normal $\mathbf{m}$, these surfaces can be widely used to identify and separate different regions of flow. However, displaying several stream surfaces usually leads to occlusion. One can easily show a direction and magnitude of the local flow using colored stream elements. Stream tubes and ribbons are the proper tools to reflect divergence and torsion of the field, correspondingly. Combination of these basic elements can be applied to provide a versatile picture of the flow. Given the derived implicit flow representation, stream elements can be directly extracted from these scalar fields.

All points satisfying the relation $f(\mathbf{x}) = f_{\text{iso}}$ for arbitrary $f_{\text{iso}} \in \mathbb{R}$ define a stream surface of the flow $\mathbf{v}(\mathbf{x})$. We use standard marching technique to derive a triangulated representation of stream surfaces.

It is well-known that an intersection line of two non-parallel stream surfaces is a streamline [KM92]. Different parametrizations of the boundary (or termination surface) lead to different stream functions for the same flow. Given two stream functions $f_1(\mathbf{x})$ and $f_2(\mathbf{x})$ with the property $\nabla f_1 \cdot \nabla f_2 \neq 0$ in D, a set of streamlines can be obtained by intersection of isosurfaces $f_1(\mathbf{x}) = c_1$ and $f_2(\mathbf{x}) = c_2$ for various constants c_1 and c_2. Therefore, each streamline is uniquely defined by two stream coordinates $\mathbf{c} = (c_1, c_2)$. However, an explicit integration of a single streamline is much easier. This observation changes as soon as one is interested in extracting certain sets of streamlines.

Usually, a stream tube is generated as a collection of streamlines with seeding points lying on an ellipse. An alternative construction of a stream tube can be obtained by generating a proper stream function. Let $f_1(\mathbf{x})$ and $f_2(\mathbf{x})$ be two stream functions. It is easy to show that any smooth function $h(f_1(\mathbf{x}), f_2(\mathbf{x}))$ is also a stream function: $\nabla h(\mathbf{x}) \cdot \mathbf{v}(\mathbf{x}) = 0$ [vW93]. Let us assume that isosurfaces $f_1(\mathbf{x}) = c_1$ and $f_2(\mathbf{x}) = c_2$ are orthogonal in a neighborhood of the termination surface: $\nabla f_1(\mathbf{x}) \cdot \nabla f_2(\mathbf{x}) = 0$. Then, the stream surface $h(\mathbf{x}) = 1$ for the function

$$h(\mathbf{x}) = \frac{(f_1(\mathbf{x}) - c_1)^2}{a^2} + \frac{(f_2(\mathbf{x}) - c_2)^2}{b^2}$$

is the desired stream tube with radii a and b.

Similar to the stream tube construction, there are also two methods for extracting stream ribbons. One can seed a set of streamlines along a line segment of interest or one can extract a part of the stream surface $f_1(\mathbf{x}) = f_{\text{iso}}$ satisfying the condition $a \leq f_2(\mathbf{x}) \leq b$. In the latter case, we construct the stream surface with respect to the field $f_1(\mathbf{x})$ by means of a marching algorithm. For each triangle from the derived surface representation we compute values of $f_2(\mathbf{x})$ on its vertices. If all three values are in the range $[a, b]$, the triangle will be accepted; if none of the values belong to the interval,

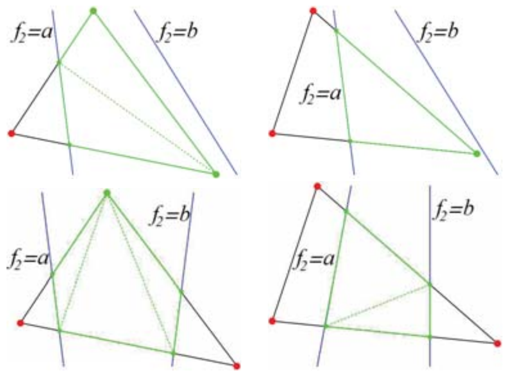

Figure 3: Extraction of stream ribbon $f_1(\mathbf{x}) = f_{\text{iso}}$, $a \leq f_2(\mathbf{x}) \leq b$. A marching algorithm produces a triangulation of the stream surface $f(\mathbf{x}) = f_{\text{iso}}$. These triangles are then rejected, accepted, or accepted with modification based on the values of function f_2 at their vertices. If a triangle intersects the ribbon boundary, it is trimmed producing up to 3 new triangles.

we reject the triangle; if some of the values are in the range, the triangle is trimmed producing up to 3 new triangles. All possible trimming scenarios are shown in Figure 3.

The advection-time field $t(\mathbf{x})$ is also available after the pre-processing step. Its isosurfaces — time surfaces — can be extracted in the same manner as stream surfaces. The advection time information can be encoded on the surface of stream element using color or transparency. Besides that, stream functions and advection-time field can be combined to extract flow volumes. A flow volume is a part of flow domain bounded by surface $S(f_1(\mathbf{x}), f_2(\mathbf{x}), t(\mathbf{x})) = 1$. In practice, we use flow tube and flow cube given by expressions

$$S_{\text{tube}} = \max\left\{ \frac{(f_1(\mathbf{x}) - c_1)^2}{a^2} + \frac{(f_2(\mathbf{x}) - c_2)^2}{b^2}, \frac{|t(\mathrm{x}) - t_0|}{r_t} \right\}$$
$$S_{\text{cube}} = \max\left\{ \frac{|f_1(\mathbf{x}) - c_1|}{r_1}, \frac{|f_2(\mathbf{x}) - c_2|}{r_2}, \frac{|t(\mathrm{x}) - t_0|}{r_t} \right\}.$$

Flow volumes have a meaningful interpretation for steady flows: They define the fluid portion that crosses the boundary at the specified location during the given time interval. For example, the fluid inside flow tube S_{tube} will flow through the elliptical part of the boundary withing time from $t_0 - r_t$ to $t_0 + r_t$, i.e., it will traverse the tube from one end to the other.

7 NUMERICAL EXPERIMENTS

All numerical tests presented in this and the following section were performed on a PC with an Intel Xeon 3.20GHz processor. For surface extraction, a marching cubes algorithm was used. Extraction of any stream element for any examples presented here took only a fraction of a second. Thus, the user experiences a highly interactive system for extracting stream elements. In all

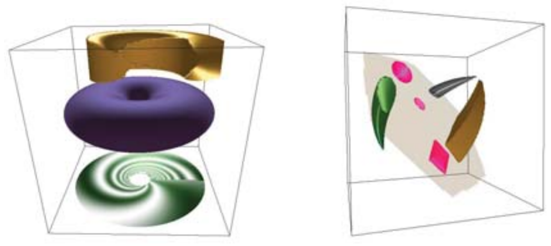

Figure 4: Extraction of flow volumes. *Left:* Type V region. Restricting a stream tube (purple) to a finite time-advection interval results in a flow volume (gold). To obtain an information about velocity magnitude, the sinus of the advection time is mapped to transparency of a stream surface (green). *Right:* Type II region. Three flow volumes together with their footprints (red) on the termination surface are shown.

our tests, we used trilinear interpolation of the velocity field and a fourth-order Runge-Kutta method for integration.

First, we looked into simple synthetic data sets. The first example is that of flow around a vertex line, which we sampled at 100^3 regularly distributed grid nodes. The flow is divided in two subdomains of type BB and V. To parametrize the latter, we construct a termination surface as shown in Figure 2 (left). Several extracted stream elements are shown in Figure 5 (left). Information about velocity magnitude can be obtained by analyzing the shape of flow volumes or by rendering of advection-time values on stream elements. In Figure 4 (left) transparency of the lower stream surface shows the sine of the advection time. Curved patterns show that the magnitude of velocity increases superlinearly with the distance to the vortex line. The same conclusion can be drawn when looking to the shape of the flow tube shown in gold.

A second example is that of flow from a single source to a simple sink. This flow field includes a subdomain of type II. It is parametrized as shown in Figure 2 (right). Extracted stream elements are shown in Figure 5 (center). Three flow volumes and their footprints on the termination surface are shown in Figure 4 (right).

Next, we demonstrate the speed-up of the preprocessing step when applying the minimization procedure presented in Section 5. The tornado dataset [CM93] was sampled on a uniform grid of resolution 50^3 and 128^3. After computing vorticity at all nodes, we set its threshold to 0.15. Then, we track those grid nodes, which belong to the domain boundary, have vorticity values larger than the threshold, or have an even grid index. The tracked nodes get scalar values equal to the z-coordinate of their footprint at the boundary. The resulting sparse scalar field is linearly interpolated to the rest of the nodes. Finally, we perform several iterations to minimize the stream function error.

The time spent at each step of the algorithm for both datasets is summarized in Table 1. When compared to tracking all nodes, we observe that our algorithm requires only 22% and 16% of the tracking time for the data sets with 50^3 and 128^3 nodes, correspondingly. The evolution of the average error during the minimization step is presented in Figure 7. Only few iterations with the artificial time step $\delta\tau = 2.0$ were enough to reduce the error to values that are even below the error one obtains when tracking all nodes. A result for extracted stream elements from this data set can be seen in Figure 5 (right). Areas of high vorticity are shown in Figure 6(left), while Figure 6(right) shows the error on a stream surface.

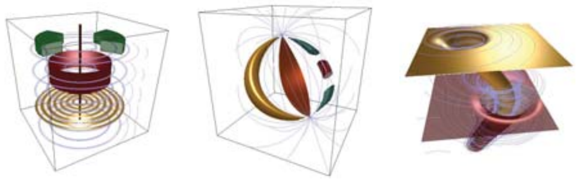

Figure 5: Flow representation of several data sets: flow around a vortex line (*left*), flow between a sink and a source (*center*), and tornado data set (*right*). A set of streamlines together with various stream elements are shown for each data set. The geometric features are extracted interactively from an implicit flow representation.

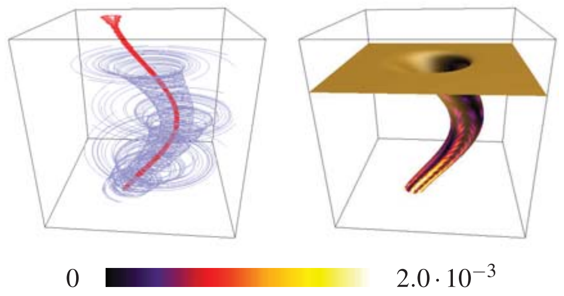

Figure 6: Left: Streamlines computed for tornado dataset. Red are the grid nodes which have vorticity values larger than threshold. A stream surface close to these nodes has high curvature that makes the error minimization procedure unstable in this region. Right: Error visualization on a stream surface. The error increases when the surface approaches the tornado center (red-yellow-white spots) and vanishes at larger distances (black-blue spots). Areas with negligible error remain gold.

Finally, we construct an implicit representation for a simulation of the flow of five jets (dataset courtesy of Kwan-Liu Ma, University of California, Davis). Figure 8 shows a set of streamlines, a constructed termination surface and some extracted stream elements.

	50^3 nodes	128^3 nodes
tracking time	177.53 s	4631 s
interpolation time	0.15 s	2.07 s
error minimization time	9.19 s	114.62 s
number of iterations	70	50
total time	186.87 s	4797.69 s
final error	$6.278 \cdot 10^{-4}$	$3.343 \cdot 10^{-4}$
time (tracking all nodes)	861 s	30625 s
error (tracking all nodes)	$7.475 \cdot 10^{-4}$	$6.456 \cdot 10^{-4}$

Table 1: Time consumption for different stages of our algorithm in Section 5 for the construction of an implicit flow representation when applied to the tornado dataset sampled at 50^3 and 128^3 nodes. The results show significant reduction of time when compared to the approach of tracking all nodes. Moreover, although we are tracking significantly less nodes, the average error decreases with our approach.

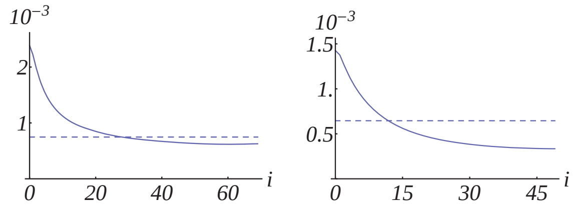

Figure 7: Evolution of average error during the minimization procedure applied to the interpolated data (solid lines). Dashed lines show the error values after tracking all grid nodes. Tornado dataset with 50^3 nodes (left) and 128^3 nodes (right) was tested.

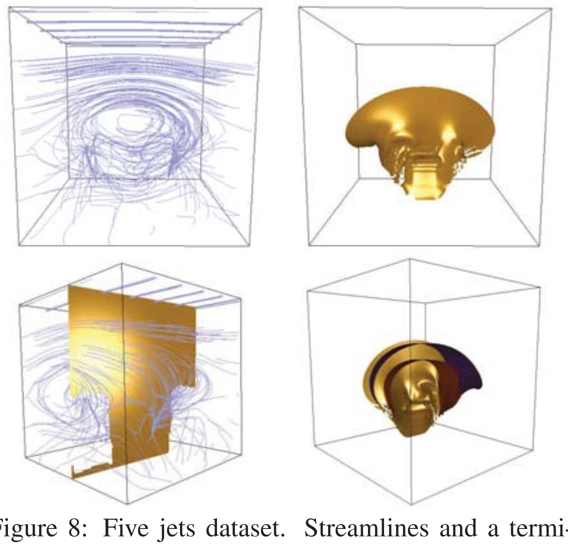

Figure 8: Five jets dataset. Streamlines and a termination surface are shown in the upper row. Extracted stream surface in combination with two stream ribbons is shown in the lower row.

8 CONCLUSION

We presented a method for automatic generation of implicit representation for volumetric flow. The method is based on the classification of streamlines in five types: BB, BI, IB, II, and V depending on whether they start/end on the boundary or in the domain interia, or form a closed trajectory. For the first three cases known techniques as in [vW93, XZC04] are applicable. We focused our efforts on effectively defining a stream function for regions of type II and V. To handle those, a termination surface is created starting with a proper seeding voxel. Two strategies for choosing seeding voxels are proposed: Maximization of number of unparametrized streamlines passing through the voxel (suitable for type V), and comparing advection-time values recorded in the pre-processing step (suitable for type II). Thus, some open problems concerning the construction of a stream function in the entire flow region have been solved. We have avoided any artificial splitting of the domain. Instead, the established subregions reflect the flow topology; they are separated from each other by special stream surfaces. We have also avoided termination surfaces isolating critical points, since it could lead to inaccurate parametrization due to the high density of the streamlines on such surfaces.

We also proposed a tool for improving of stream functions. It is based on variational minimization of a functional describing the function quality with respect to the underlying vector field. The governing diffusion equation is derived.

Various geometrical stream elements (e.g., stream surfaces) can be extracted and displayed interactively. In particular, we proposed novel algorithms for the extraction of stream tubes and ribbons. We also combined the stream function visualization with a visualization of the advection-time field. Both tracking the nodes in the pre-processing step and extraction of stream elements in run time allow for an efficient parallel computing.

Acknowledgments This work was supported in part by DFG grants LI 1530/6-2 and MO 3050/2-1.

9 REFERENCES

[CKSW08] Nicolas Cuntz, Andreas Kolb, Robert Strzodka, and Daniel Weiskopf. Particle level set advection for the interactive visualization of unsteady 3D flow. *Computer Graphics Forum*, 27(3):719–726, May 2008.

[CL93] Brian Cabral and Leith Casey Leedom. Imaging vector fields using line integral convolution. In *SIGGRAPH ’93: Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 263–270, New York, NY, USA, 1993. ACM.

[CM93] Roger Crawfis and Nelson Max. Texture splats for 3D vector and scalar field visualization. In *Proceedings Visualization ’93*, pages 261–266, Los Alamitos, Oct 1993. IEEE Computer Society.

[GRT17] T. Gerrits, C. Rössl, and H. Theisel. Glyphs for space-time jacobians of time-dependent vector fields. *Journal of WSCG*, 25(1):31–38, 2017.

[Hul92] J. P. M. Hultquist. Constructing stream surfaces in steady 3d vector fields. In *VIS '92: Proceedings of the 3rd conference on Visualization '92*, pages 171–178, Los Alamitos, CA, USA, 1992. IEEE Computer Society.

[KM92] David N. Kenwright and Gordon D. Mallinson. A 3-D streamline tracking algorithm using dual stream functions. In *VIS '92: Proceedings of the 3rd conference on Visualization '92*, pages 62–68, Los Alamitos, CA, USA, 1992. IEEE Computer Society.

[KM96] David Knight and Gordon Mallinson. Visualizing unstructured flow data using dual stream functions. *IEEE Transactions on Visualization and Computer Graphics*, 2(4):355–363, 1996.

[LHD$^+$03] Robert S. Laramee, Helwig Hauser, Helmut Doleisch, Benjamin Vrolijk, Frits H. Post, and Daniel Weiskopf. The state of the art in flow visualization: Dense and texture-based techniques. *Computer Graphics Forum*, 23(2):203–221, 2003.

[LTH08] Guo-Shi Li, Xavier Tricoche, and Charles D. Hansen. Physically-based dye advection for flow visualization. *Comput. Graph. Forum*, 27(3):727–734, 2008.

[MBS$^+$04] Karim Mahrous, Janine Bennett, Gerik Scheuermann, Bernd Hamann, and Kenneth I. Joy. Topological segmentation in three-dimensional vector fields. *IEEE Transactions on Visualization and Computer Graphics*, 10:198–205, 2004.

[PBL$^+$04] Sung W. Park, Brian Budge, Lars Linsen, Bernd Hamann, and Kenneth I. Joy. Multidimensional transfer functions for interactive 3d flow visualization. In *PG '04: Proceedings of the Computer Graphics and Applications, 12th Pacific Conference*, pages 177–185, Washington, DC, USA, 2004. IEEE Computer Society.

[PBL$^+$05] Sung W. Park, Brian Budge, Lars Linsen, Bernd Hamann, and Kenneth I. Joy. Dense geometric flow visualization. In *EUROGRAPHICS - IEEE VGTC Symposium on Visualization*, pages 21–28, 2005.

[PS09] Ronald Peikert and Filip Sadlo. Topologically Relevant Stream Surfaces for Flow Visualization. In H. Hauser, editor, *Proc. Spring Conference on Computer Graphics*, pages 43–50, April 2009.

[PYH$^+$06] Sung W. Park, Hongfeng Yu, Ingrid Hotz, Oliver Kreylos, Lars Linsen, and Bernd Hamann. Structure-accentuating dense flow visualization. In Beatriz Sousa Santos, Thomas Ertl, and Kenneth I. Joy, editors, *EuroVis06: Joint Eurographics - IEEE VGTC Symposium on Visualization, Lisbon, Portugal, 8-10 May 2006*, pages 163–170. Eurographics Association, 2006.

[RLN$^+$17] Dylan Rees, Robert S. Laramee, Duong Nguyen, Lei Zhang, Guoning Chen, Harry Yeh, and Eugene Zhang. A Stream Ribbon Seeding Strategy. In *EuroVis 2017 - Short Papers*. The Eurographics Association, 2017.

[SHJK00] Gerik Scheuermann, Bernd Hamann, Kenneth I. Joy, and Wolfgang Kollmann. Visualizing local vector field topology. *SPIE Journal of Electronic Imaging*, 9:367, 2000.

[SRP09] Filip Sadlo, Alessandro Rigazzi, and Ronald Peikert. Time-Dependent Visualization of Lagrangian Coherent Structures by Grid Advection. In *Proceedings of TopoInVis 2009*. Springer, 2009.

[SS07] Tobias Salzbrunn and Gerik Scheuermann. Streamline predicates as flow topology generalization. In Helwig Hauser, Hans Hagen, and Holger Theisel, editors, *Topology-Based Methods in Visualization (Mathematics and Visualization)*, pages 65–78. Springer, July 2007.

[STWE07] Tobias Schafhitzel, Eduardo Tejada, Daniel Weiskopf, and Thomas Ertl. Point-based stream surfaces and path surfaces. In *Graphics Interface*, pages 289–296, 2007.

[vFWTS08] Wolfram von Funck, Tino Weinkauf, Holger Theisel, and Hans-Peter Seidel. Smoke surfaces: An interactive flow visualization technique inspired by real-world flow experiments. *IEEE Transactions on Visualization and Computer Graphics (Proc. IEEE Visualization)*, 14(6):1396–1403, Nov 2008.

[vW93] Jarke J. van Wijk. Implicit stream surfaces. In *VIS '93: Proceedings of the 4th conference on Visualization '93*, pages 245–252, Washington, DC, USA, 1993. IEEE Computer Society.

[WJE00] Rüdiger Westermann, Christopher Johnson, and Thomas Ertl. A level-set method for flow visualization. In *VIS '00: Proceedings of the conference on Visualization '00*, pages 147–154, Los Alamitos, CA, USA, 2000. IEEE Computer Society.

[WJE01] Rüdiger Westermann, Christopher Johnson, and Thomas Ertl. Topology-preserving smoothing of vector fields. *IEEE Transactions on Visualization and Computer Graphics*, 7(3):222–229, 2001.

[WT10] Tino Weinkauf and Holger Theisel. Streak lines as tangent curves of a derived vector field. *IEEE Transactions on Visualization and Computer Graphics*, 16:1225–1234, 2010.

[XZC04] Daqing Xue, Caixia Zhang, and Roger Crawfis. Rendering implicit flow volumes. In *VIS '04: Proceedings of the conference on Visualization '04*, pages 99–106, Washington, DC, USA, 2004. IEEE Computer Society.