Poisson-based Tools for Flow Visualization

Janick Martinez Esturo* Maik Schulze Christian Rössl Holger Theisel

University of Magdeburg

Abstract

This paper applies Poisson-based methods to assist in interactive exploration of steady flow fields. Using data-driven deformations we obtain flow-orthogonal and flow-tangential surfaces by a flux-based optimization. Surfaces are positioned interactively and deformed in real-time according to local flow. The deformed surfaces are particularly useful for defining seed structures. We show how the same gradient-based computational framework can be applied to obtain parametrizations of flow-aligned surfaces. This way it is easy to define nontrivial seed structures for integration-based flow visualization methods. Additionally, the flow-aligned parametrizations are employed for view-independent surface-based LIC visualizations. We apply our method to a number of data sets to show the effectiveness of our deformations and parametrization-based seed extraction methods for interactive flow exploration.

Index Terms: Computer Graphics [I.3.5]: Computational Geometry and Object Modeling—Geometric algorithms

1 Introduction

A variety of scientific, engineering, and medical application areas study problems in which 3D flow phenomena play a key role for their comprehension and solution. As problems of relevant size often do not permit an automatic analysis, the flow phenomena need to be investigated by domain experts. An effective analysis requires flow visualization techniques to process the oftentimes huge amounts of data. In recent years, visualization experts have proposed a variety of advanced and powerful flow visualization techniques. For example, characteristic surfaces, such as stream surfaces, are an established visualization approach that is well-studied in the research community. However, many important professional engineering tools used in practice do rarely provide domain experts with such advanced visualization techniques. In fact, a lot of tools (like, e. g., the ANSYS CFD-Post© package [2]) provide users with only basic visualization techniques, e. g., iconic or glyph-based, slice-based, or stream line or path line visualizations. Visualizations by characteristic surfaces are infrequently used in practice despite their advantages over basic visualizations.

We believe that one reason for the less prominent representation in professional tools is the more complex interaction that is required for surface-based visualizations: a lot of methods exist for the automatic seeding of stream lines and only a few limited approaches exist to automatically seed, e. g., stream surfaces. Stream surfaces are usually defined by seed curves from which the surfaces are integrated. The exact placement of these curves is based on assumptions and experience of domain experts and is usually performed manually. Most often the choice of seed curve geometry is very limited, e. g., to simple straight line segments or circles. Moreover, not every seed curve is suitable to define stream surfaces: in fact, seed curves that the user erroneously positions tangentially to the flow result in degenerate stream surfaces. These factors make direct seed curve specification challenging in practice. Therefore, it is necessary to provide users with additional interaction tools that overcome limitations of seed curve manipulation in order to make surface-based visualizations simpler to use in practice.

In contrast to existing methods that require seed curve manipulation, our approach is based on direct interaction with entire surfaces that are near flow-aligned. We do not perform surface integration but instead propose an interactive surface deformation-based method allowing free positioning of surfaces. This way we achieve a more direct interaction with the resulting flow-aligned surfaces. The set of flow-aligned surfaces under consideration does not only contain flow-tangential surfaces, which correspond to classical stream surfaces, but also flow-orthogonal surfaces that can in general not be obtained by a simple surface integration. Our approach is generalized to computing both types of surfaces by deformations that are steered by flux optimization criteria. Our flow-tangential surfaces are particularly useful for interactive flow exploration. Flow-orthogonal surfaces are well-suited to provide seed structures for integration-based methods, because every embedded curve is flow-orthogonal by construction. These curves can be obtained as iso-contours of special flow-aligned parametrizations. Moreover, the same global parametrizations can further be used for, e. g., texture-based visualization, and we show how they define animated and view-independent LIC-like and illustrative renderings.

In this work we show that *both*, flow-aligning deformations and parametrizations, can be computed efficiently and in a unified way by a Poisson-based optimization framework. Poisson-based methods are well-known with many applications in image and geometry processing. This is the first approach that applies this technique in the context of flow visualization.

2 Related Work

Flow-aligned orthogonal structures. Most related to the orthogonal surfaces that we compute by flux maximization are the recently proposed as-perpendicular-as-possible surfaces (APAPs) by Schulze et al. [32], who integrate along a scaled vector field to obtain orthogonally aligned surfaces. In contrast to our method, APAP surfaces cannot be positioned freely due to the integration-based approach. We will further compare both approaches for orthogonal surfaces in this work. A simpler method that does not consider energy minimization has been proposed before by Palmerius et al. [26]. Additionally, orthogonal structures were used to improve animation [3], to seed stream lines on two-manifolds [29], and they are well-known in the computer vision community [10] in the context of shape-from-shading. This paper introduces a flux-optimizing and deformation-based method to extract both, flow-orthogonal and flow-tangential surfaces. The classical approach to computing the later is surface integration.

Flow-tangential surface integration. There is a large body of research on stream surface integration as well as stream surface rendering [28, 22]. In the field of flow visualization, the seminal work of Hultquist [16] marks the foundation for the development of stream surface integrators [34, 31]. We refer to the survey by McLoughlin et al. [25] for an overview on stream surface integration methods. More elaborate approaches for path surface integration [30, 14], and streak and time surface integration [36, 8, 19, 38, 37] have been proposed recently.

*e-mail: martinez@isg.cs.ovgu.de

Flow exploration. Various methods have been proposed for automatic stream line seeding, which is in contrast to stream surface seeding that is generally performed by a manual search [25]. A dense set of stream surfaces can be selected automatically by clustering local flow features [13]. Seeds for simpler stream ribbons and particles can be interactively moved around for real-time exploration [20]. The method we propose also enables interactive flow exploration with the difference that the user can directly manipulate entire surfaces.

Texture-based visualization. A classical technique to visualize the flow on integral surfaces are texture-based methods, such as line integral convolution (LIC) approaches. Recent methods range from generation of texture atlases [23] to image-space techniques [21] that can be frame-coherent [15]. The survey by Laramee et al. [22] discusses the various techniques. We introduce LIC-like flow structure visualizations and illustrations that are defined by globally flow-aligned parametrizations. The method provides frame-coherence and supports stream and path surfaces as well as animation.

Poisson-based modeling. Poisson-based surface deformations were first proposed by Yu et al. [40]. They apply the idea of Poisson-based image editing by Perez et al. [27] in the context of geometric modeling. Using their basic technique a number of geometry processing methods were proposed, e. g., user-defined deformations [41], pose interpolation [39], and deformation transfer [35, 41]. All these methods are based on similar Poisson-based computations. They differ in the way local gradients are transformed prior to reconstruction. We refer to the survey of Botsch and Sorkine [7] for related linear deformation approaches. This work shows how gradient transformations are determined in a data-driven way to align surfaces to a flow field using deformations.

Before we present details of our approach we review the Poisson optimization method in general, which is required by both our deformation and parametrization approach.

3 Poisson-based Optimization

Given is a guidance vector field $\mathbf{h}$ in a two-manifold $\mathcal{M}$ with associated gradient operator ∇. Then Poisson-based methods search for the scalar field u on $\mathcal{M}$ whose gradient ∇u best fits $\mathbf{h}$ in least-squares sense. Formally, the energy

$$e(u) = \int_{\mathcal{M}} ||\nabla u - \mathbf{h}||^2 \, d\mathbf{x} \tag{1}$$

is minimized. A computationally attractive and sufficient condition for a minimum is given by the solution of Poisson's elliptic linear partial differential equation

$$\Delta u = \nabla^{\mathrm{T}} \mathbf{h} \,, \tag{2}$$

which has to be solved subject to suitable boundary constraints (see, e. g., [1]). In (2) Δ denotes the Laplace-Beltrami operator on $\mathcal{M}$, which can be represented by $\Delta = \nabla^{\mathrm{T}} \nabla$, i. e., as the divergence of the gradient.

In this work we discretize the surface $\mathcal{M}$ by a triangular mesh $(\mathcal{V}, \mathcal{T})$ defined by a set $\mathcal{V}$ of m vertices v_i with coordinates $\mathbf{x}_i \in \mathbb{R}^3$ and a set $\mathcal{T}$ of n triangles t_j. Scalar fields u are piecewise linear functions on triangulated surfaces, i. e., on a triangle $t_j = (a, b, c)$ with coefficients u_i at the vertices v_i we have $u(\mathbf{x}) = \sum_{i \in t_j} \phi_i{}^j(\mathbf{x})\, u_i$ with barycentric coordinates $\phi_i{}^j(\mathbf{x})$ as linear basis functions. The piecewise constant gradient field on t_j is given by $\nabla u = \sum_{i \in t_j} \nabla \phi_i{}^j\, u_i$. One way to compute the constant gradients $\nabla \phi_i{}^j$ of the basis functions is to solve the linear system

$$\begin{pmatrix} (\mathbf{x}_a - \mathbf{x}_c)^{\mathrm{T}} \\ (\mathbf{x}_b - \mathbf{x}_c)^{\mathrm{T}} \\ \mathbf{n}_j{}^{\mathrm{T}} \end{pmatrix} \left(\nabla \phi_a{}^j, \nabla \phi_b{}^j, \nabla \phi_c{}^j \right) = \begin{pmatrix} 1 & 0 & -1 \\ 0 & 1 & -1 \\ 0 & 0 & 0 \end{pmatrix}$$

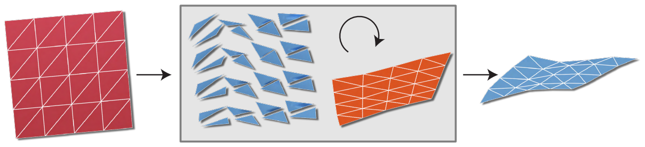


Figure 1: Deformation principle. An initial surface mesh (●) is iteratively deformed by *conceptually* aligning each triangle individually to the flow and reconstructing the mesh (●) from these transformed gradients until the iteration converges to a flow aligned mesh (●).

(see, e. g., [7] for a derivation). Here $\mathbf{n}_j$ is the unit normal of the triangle. If all scalar field coefficients u_i are stacked in a vector $\mathbf{u}$ one can assemble a $3n \times m$ gradient operator matrix $\mathbf{G}$ from the basis function gradients such that $\mathbf{G}\,\mathbf{u}$ is the vector of stacked scalar field gradients on each triangle.

Since the scalar field gradients on triangular meshes are constant per triangle the integral of (1) simplifies to an area-weighted sum of quadratic differences, and we can rewrite the integrated energy to

$$e(\mathbf{u}) = ||\mathbf{G}\,\mathbf{u} - \mathbf{h}||^2_{\mathbf{A}} = (\mathbf{G}\,\mathbf{u} - \mathbf{h})^{\mathrm{T}} \mathbf{A} \,(\mathbf{G}\,\mathbf{u} - \mathbf{h}) \,, \tag{3}$$

where $\mathbf{h}$ is the vector of stacked guidance gradient field vectors per triangle and $\mathbf{A}$ is a $3n \times 3n$ diagonal inner-product matrix of triangle areas that performs the integration. Then the optimal scalar field minimizing $e(\mathbf{u})$ is obtained by solving the linear system $\nabla_{\mathbf{u}}\, e(\mathbf{u}) = \mathbf{0}$, which results in the system

$$\mathbf{G}^{\mathrm{T}} \mathbf{A}\, \mathbf{G}\, \mathbf{u} = \mathbf{G}^{\mathrm{T}} \mathbf{A}\, \mathbf{h} \,. \tag{4}$$

This linear system is a discretization of the Poisson equation (2) on triangular meshes with the discrete Laplace-Beltrami operator matrix $\mathbf{L} = \mathbf{G}^{\mathrm{T}} \mathbf{A}\, \mathbf{G}$ and the discrete divergence operator matrix $\mathbf{D} = \mathbf{G}^{\mathrm{T}} \mathbf{A}$. The matrix $\mathbf{L}$ is sparse and positive semi-definite.

For Poisson-based surface deformations the gradients of all three scalar coordinate functions are modified, and the coordinates of the deformed mesh are reconstructed by solving (4). More precisely, let $\mathbf{Y}_j$ be the 3×3 matrix of the mesh coordinate gradients of triangle t_j, $\mathbf{Y}^{\mathrm{T}} = \left(\mathbf{Y}_1^{\mathrm{T}}, \cdots, \mathbf{Y}_n^{\mathrm{T}} \right)$ the $3n \times 3$ matrix of all stacked gradients, and $\mathbf{X}^{\mathrm{T}} = (\mathbf{x}_1, \cdots, \mathbf{x}_m)$ the $n \times 3$ matrix of stacked mesh coordinates such that $\mathbf{Y} = \mathbf{G}\mathbf{X}$. Then the per-triangle gradients are modified using *local gradient transformations* $\mathbf{T}_j$ to give $\mathbf{Y}'_j = \mathbf{Y}_j\, \mathbf{T}_j^{\mathrm{T}}$. Finally, the deformed mesh with coordinates $\mathbf{X}'$, whose gradients best conform to the transformed gradients in least-squares sense, is reconstructed by solving

$$\mathbf{L}\,\mathbf{X}' = \mathbf{D}\,\mathbf{Y}' \,. \tag{5}$$

As the gradients are translation-invariant the coordinates of at least one vertex need to be prescribed when solving this system.

Different applications rely on this general Poisson-based surface reconstruction technique and differ only in the way the local transformations $\mathbf{T}_j$ are specified. Usually, the $\mathbf{T}_j$ are combined rotation and scaling operations [40, 35, 41, 39]. In this work we use a data-driven specification of the local transformations $\mathbf{T}_j$ to obtain either flow-tangential or flow-orthogonal surfaces.

4 Interactive Deformation-based Flow Alignment

In this work we provide tools for the explorative analysis of 3D steady vector fields $\mathbf{v}(\mathbf{x})$ over a spatial domain $\mathcal{D}$. A popular and well-studied family of methods for the visualization of vector fields are integral surfaces. For steady vector fields these are usually stream surfaces [25]. Stream surfaces are surfaces $\mathcal{S} \subset \mathcal{D}$ that are tangential to the flow, i. e., given the normal $\mathbf{n}(\mathbf{x})$ of the stream surface the local flux condition $\mathbf{n}(\mathbf{x})^{\mathrm{T}} \mathbf{v}(\mathbf{x}) = 0$ holds for all points $\mathbf{x}$ on $\mathcal{S}$. Therefore, the total flux through a stream surface

$$f = \int_{\mathcal{S}} \mathbf{n}(\mathbf{x})^{\mathrm{T}} \mathbf{v}(\mathbf{x}) \, d\mathbf{x}$$

vanishes.

Stream surfaces are usually constructed by advancing front algorithms that start at seed curves, such as straight line segments, which are defined by domain experts [25]. However, even when users are restricted to simple seed geometries it is still possible to specify flow-tangential curves that lead to degenerate stream surfaces.

We propose an interaction metaphor that is different to this classical approach: we give the user direct, interactive control over complete flow-aligned surfaces. In this work the term *flow-aligned* surfaces denotes both, flow-tangential and flow-orthogonal surfaces. Flow-orthogonal surfaces will be used to specify seed curves that are near flow-orthogonal to define non-degenerate stream surfaces. Our approach is deformation-based. The basic idea is to start from an initial surface $\mathcal{M}_0$, which is a procedurally generated planar triangle mesh. The user positions this surface within the area of interest of the flow domain. Then this surface is iteratively deformed into intermediate surfaces $\mathcal{M}_k$ in such a way that the result converges to a surface that is aligned with the flow. Figure 1 illustrates the basic principle. At any time the user can interactively reposition the surface to more interesting locations, and simultaneously the surface shape adapts according to the altered local flow. We apply Poisson-based deformations and achieve interactivity by prefactorization of the involved linear operators. In contrast to other Poisson-based deformations our approach is data-driven in that it is steered by the local flow.

Orthogonal alignment. Surfaces that we align orthogonally to the flow *maximize* total flux. Exactly flow-orthogonal surfaces exist in conservative but do not exist in general vector fields. Before considering the entire surface mesh we start with the analysis of aligning a single triangle orthogonally to the flow. The only admissible class of deformations are (rigid) rotations, since we strive to preserve the shape of the triangle and only align it locally to the flow. Let $\mathbf{R}(\mathbf{a},\gamma) \in \mathrm{SO}(3)$ be the transformation describing a rotation around the axis $\mathbf{a}$ with angle γ and let $\gamma(\mathbf{p},\mathbf{q})$ be the angle between the vectors $\mathbf{p}$ and $\mathbf{q}$. If we assume a linear vector field $\mathbf{v}(\mathbf{x})$ on a triangle $t_j = (a,b,c)$ given by $\mathbf{v}^j(\mathbf{x}) = \sum_{i\in t_j} \phi_i{}^j(\mathbf{x})\,\mathbf{v}(\mathbf{x}_i)$, then the flux f_j through t_j can be expressed as

$$f_j = \int_{t_j} \mathbf{n}_j{}^{\mathrm{T}} \mathbf{v}^j(\mathbf{x})\,\mathrm{d}\mathbf{x} = \frac{A_j}{3}\,\mathbf{n}_j{}^{\mathrm{T}}\left(\mathbf{v}(\mathbf{x}_a)+\mathbf{v}(\mathbf{x}_b)+\mathbf{v}(\mathbf{x}_c)\right)$$

with triangle area A_j and normal $\mathbf{n}_j$. Since $\frac{1}{3}(\mathbf{v}(\mathbf{x}_a)+\mathbf{v}(\mathbf{x}_b)+\mathbf{v}(\mathbf{x}_c))$ is the value of the linearized vector field at the center of the triangle the nonlinear flux through the triangle can be approximated by dropping the linearity property and evaluating the approximate flux by a single point quadrature that evaluates the vector field at the triangle center: $f_j \approx A_j\,\mathbf{n}_j{}^{\mathrm{T}}\,\mathbf{v}_j$ with $\mathbf{v}_j := \mathbf{v}(\frac{1}{3}(\mathbf{x}_a+\mathbf{x}_b+\mathbf{x}_c))$. To maximize the flux and to align the triangle orthogonally to the flow a rotation that minimizes $\gamma(\mathbf{n}_j,\mathbf{v}_j)$ has to be performed around the axis $\mathbf{a}_j = \mathbf{n}_j \times \mathbf{v}_j$. We can therefore rotate the triangle (around its center) by the transformation $\mathbf{R}(\mathbf{a}_j,\gamma(\mathbf{n}_j,\mathbf{v}_j))$ such that the approximated flux f_j is maximized.

Using Poisson-based deformations this consideration for a single triangle can directly be applied to align triangle meshes with the flow. Instead of transforming the vertex coordinates directly the basic idea is to transform the gradients of the coordinate function of each triangle. This means that the local gradient transformations of each triangle of the mesh are given by

$$\mathbf{T}_j = \mathbf{R}(\mathbf{a}_j,\gamma(\mathbf{n}_j,\mathbf{v}_j))$$

(see Section 3). The deformed mesh that optimally approximates these modified gradients (in least-squares sense) is then reconstructed by solving (5). In Section 8 we show that our method for computing orthogonal surfaces achieves higher flux rates compared to APAP [32] surfaces.

Tangential alignment. Only a slight modification of the previous argument is required to directly obtain flow-tangential surfaces in the same framework, as flow-tangential surfaces *minimize* the total flux. Therefore, using

$$\mathbf{T}_j = \mathbf{R}(\mathbf{a}_j,\gamma(\mathbf{n}_j,\mathbf{v}_j)-\pi/2)$$

to minimize the deviation of $\gamma(\mathbf{n}_j,\mathbf{v}_j)$ from $\pi/2$ as the local gradient transformation for each triangle minimizes the flux through each triangle, and approximately flow-tangential surfaces are obtained.

Interactive iterative deformation. Surfaces will in general not directly be aligned with the flow when the mesh is reconstructed using (5) together with the proposed local gradient transformations. This is because the gradients of the reconstruction only comply with the prescribed gradients in least-squares sense. Moreover, and more importantly, the rotations are only flux-optimizing if triangles undergo no translation (unless the vector field is constant). This is because otherwise the local flow after reconstruction differs from the one that was used to determine the rotations. In fact, the reconstruction inevitably has to introduce slight translations for the mesh to be continuous. Therefore, one single reconstruction is, in general, not sufficient to obtain a deformed and aligned mesh. Yet, if only *small* deformations are performed, then the flux is iteratively optimized until the mesh converges to an aligned configuration. This is justified by a mild continuity assumption on the vector field, i. e., a matrix norm of the Jacobian of the vector field is bounded. The convergence in interactive sessions is further quantified in Section 7. We perform small deformations by limiting the maximal absolute value of the rotation angle by a constant small value η. Hence, small deformations are performed iteratively and we compute new local gradient transformations for each intermediate deformed configuration $\mathcal{M}_k$. From the user's perspective these iterative deformations are continuous. Convergence is achieved if the maximal rotation angle is smaller than a constant value ε. This iterative deformation approach is, in essence, similar to other deformation [33] and parametrization methods [42, 24], which also minimize nonlinear measures.

There are two cases that still need to be handled by corrections of the local gradient transformations: First, triangle area may vary between iterations due to the least-squares reconstruction. We avoid this artifact by performing an additional damped rescaling transformation of the prescribed gradients by using $\mathbf{T}'_j = \left(A_j^0/A_j^k\right)^{1/2}\mathbf{T}_j$ as local gradient transformations, where A_j^k is the area of triangle t_j in the k-iteration. Second, for tangentially aligned surfaces it is possible that two neighboring triangles converge to a flow-aligned, but oppositely directed configuration, because both configurations minimize the local flux. This is due to the fact that local gradient transformations are computed independently of each other. To correct this artifact we prescribe a maximal dihedral angle $\theta_{\max}$. If for two neighboring triangles t_p and t_q with dihedral angle θ_{pq} we detect that the angle defect $\delta = \theta_{pq} - \theta_{\max} > 0$, i. e., the dihedral angle is greater than the prescribed maximum angle, we modify the local gradient transformations as $\mathbf{T}'_p = \mathbf{R}(\mathbf{n}_p\times\mathbf{n}_q,\delta/2)\,\mathbf{T}_p$, and $\mathbf{T}'_q = \mathbf{R}(\mathbf{n}_q\times\mathbf{n}_p,\delta/2)\,\mathbf{T}_q$. This way the dihedral angle is minimized in consecutive iterations until the fold is resolved. The video demonstrates the stability of this unfolding technique. In the unlikely event of sampling a critical point, i. e., $\mathbf{v}_j = \mathbf{0}$, or if the triangle is located outside of the flow domain we simply set $\mathbf{T}_j = \mathbf{I}$. In all experiments we use $\eta = 0.1$, $\varepsilon = 10^{-6}$, and $\theta_{\max} = \pi/2$. These are all parameters of our approach.

Numerical solution. For the reconstruction of the deformed mesh using (5) the coordinates of at least one vertex need to be prescribed to account for the translation invariance of the Poisson-based method. We found that fixing vertices of the triangle that is closest to the barycenter of the mesh works well in practice. We use "soft" constraints on these vertices such that the system in (5) becomes positive-definite.

The global iterative deformation can be performed in real-time as each single deformation is very cheap. This is because for each iteration only the right-hand side of (5) varies. In particular the matrices $\mathbf{L}$ and $\mathbf{D}$ are constant as we discretize these operators on the

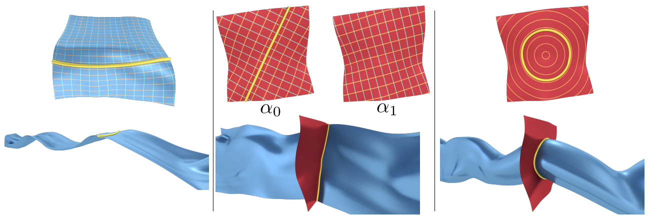


Figure 2: Parametrization types. Iso-contours (●) of flow tangential surfaces are used to integrate larger exact stream surfaces (●) (left). Orthogonally aligned surfaces (●) can be parametrized using different angular rotations (middle) or using circular geodesics-based distance fields (right).

initial mesh $\mathcal{M}_0$. This approach has two benefits: First, since the system matrix $\mathbf{L}$ (augmented with a soft constraints diagonal term $\mathbf{W}$) is symmetric positive definite, we are able to perform one single sparse Cholesky factorization only *once* in a preprocessing step of the interaction. This factorization $\mathbf{R}^{\mathrm{T}}\mathbf{R} = \mathbf{L} + \mathbf{W}$ yields the sparse triangular Cholesky factor $\mathbf{R}$. Then only back substitutions have to be performed in each iteration for updated right-hand sides to update the mesh coordinates and guarantee interactive deformations: $\mathbf{X}'_{k+1} = \mathbf{R}^{-1}\mathbf{R}^{-T}\left(\mathbf{D}\mathbf{Y}'_k + \mathbf{W}\mathbf{X}_c\right)$ with the matrix $\mathbf{X}_c$ of constrained mesh coordinates. Second, in each iteration $\mathcal{M}_0$ is deformed according to the updated gradients, and we conceptually do not deform intermediate meshes. This way possible errors introduced in a single deformation step cannot accumulate in the iteration and the mesh discretization of the deformed surface is based on $\mathcal{M}_0$.

Interaction. During the iterative deformation the user can interactively position and orient the current surface $\mathcal{M}_k$ inside the domain. In the next iteration the surface is deformed according to the changed local flow. We provide rigid translation and rotation operations and interleave user operations and deformation iteration. Usually deformations converge after a few iterations (quantified experimentally in Section 7). The surface can also be grown in a user-defined direction. We use a growing strategy that is similar to the one by Schulze et al. [32] by computing an offset curve at the boundary that is tangential to the surface. The offset curve is then tessellated to obtain the grown surface. Note that we perform this update operation of mesh coordinates on both the current mesh $\mathcal{M}_k$ and the base mesh $\mathcal{M}_0$ to be able to update the differential operators, which are discretized on $\mathcal{M}_0$. However, since growing changes the connectivity of the mesh, the discretized Laplace-Beltrami operator has to be refactored in each growing step. This is a more expensive operation, especially for large meshes. It turns out that growing a tangentially aligned surface in a direction orthogonal to the flow is advantageous for the seed curve extraction discussed in the next section (see Figure 4). The accompanying video shows examples of interactive sessions.

5 Surface Parametrization for Seed Extraction

The flow-aligned deformations presented in the previous section are well-suited for interactive exploration of flow data sets. However, as we do not perform mesh adaption during deformation, we cannot formally guarantee exact flow alignment. Nevertheless, the converged surfaces are well-suited to provide *orthogonal* seed curves for an additional front line-based stream surface integration using either classical [16] or more advanced [31] surface integrators. Stream surfaces seeded from curves in approximately flow-tangential surfaces coincide locally, but they span a larger part of the domain due to integration. Flow-orthogonal surfaces are even more suited for stream surface seeding as any curve in such a surface is also flow-orthogonal by construction. Seed curves on flow-orthogonal surfaces can either be manually "drawn" by the user (see Figure 4) or computed automatically. To automatically calculate seed curves we first perform different kinds of parametrizations

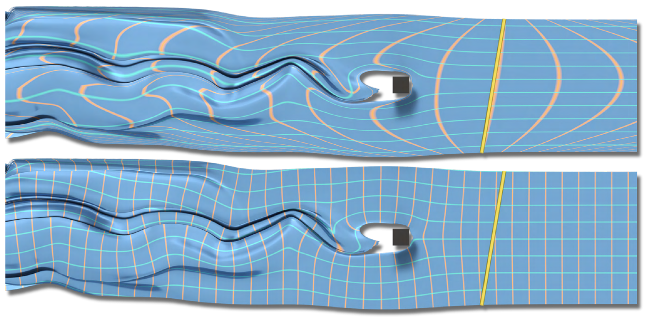
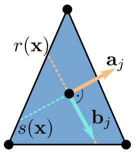


Figure 3: Stream surface parametrizations. The exact unnormalized time line/stream line parametrization obtained by front line-based stream surface integrators (top) exhibits more distortion than our least-squares tangential parametrization (bottom).

of flow-aligned surfaces. Flow-orthogonal seed curves are then extracted as iso-contours of these parametrizations.

Note that the following computations are based on the same computational framework of Poisson-based scalar field optimization. In fact, our approach to parametrization of flow-aligned surfaces is similar to the surface quadrangulation approach by Bommes et al. [4]. This approach first computes a normalized orthogonal cross field $\mathbf{C}_j = \left(\mathbf{a}_j, \mathbf{b}_j\right)$ on each triangle t_j. If these are interpreted as the parametrization gradient and cogradient, we can directly compute the corresponding parametrization scalar fields. To do so a globally integrated parametrization energy



$$e_p(\mathbf{r},\mathbf{s}) = ||\mathbf{G}\,\mathbf{r} - \mathbf{a}||^2_{\mathbf{A}} + ||\mathbf{G}\,\mathbf{s} - \mathbf{b}||^2_{\mathbf{A}} \quad (6)$$

similar to (3) is minimized for the parametrization scalar fields $r(\mathbf{x}), s(\mathbf{x})$ with stacked coefficients vectors $(\mathbf{r},\mathbf{s})$. This is equivalent to solving

$$\mathbf{G}^{\mathrm{T}}\mathbf{A}\,\mathbf{G}\,(\mathbf{r},\mathbf{s}) = \mathbf{G}^{\mathrm{T}}\mathbf{A}\,\mathbf{C} \quad (7)$$

with the $3n \times 2$ matrix $\mathbf{C}^{\mathrm{T}} = \left(\mathbf{C}_1^{\mathrm{T}}, \cdots, \mathbf{C}_n^{\mathrm{T}}\right)$. Once again scalar fields need to be constrained at a single vertex v_c. Unless otherwise specified by the user we constrain the vertex nearest to the barycenter onto the parameter origin. In general only the gradients $\mathbf{a}_j$ need to be determined, and the second orthogonal cross direction follows from $\mathbf{b}_j = \mathbf{a}_j \times \mathbf{n}_j$. For quadrangulations the computation of the guiding cross fields turns out to be the most complex part since they need to be determined from the shape geometry only. In this work, however, we make use of the flow-alignment property of the considered surfaces from which guidance fields can directly be determined. Figure 2 illustrates the different parametrization types we propose: tangential, orthogonal, and circular parametrizations.

Tangential parametrization. For tangentially flow-aligned surfaces a natural choice for the parametrization guidance field $\mathbf{a}_j$ is simply the vector field itself: $\mathbf{a}_j = \mathbf{P}_j\mathbf{v}_j / ||\mathbf{P}_j\mathbf{v}_j||$ with $\mathbf{P}_j = \mathbf{I} - \mathbf{n}_j\mathbf{n}_j^{\mathrm{T}}$. The projection $\mathbf{P}_j$ into the triangle plane is only required if the surface is not yet aligned exactly with the flow. Normalization of the gradient field guarantees that iso-contour lines are near-equidistant on the surface. Iso-contours of $r(\mathbf{x})$ are near-perpendicular to the flow and can therefore be used as seed curves of stream surfaces, whereas iso-contours of $s(\mathbf{x})$ should not be used as seed curves as they are near-tangential to the flow. Note that the extracted iso-contours are general curves, which is a much greater set of possible seed curves compared to the typically used straight line segments. The number of possible selectable stream surfaces is therefore also much higher. This natural Poisson-based parametrization technique for flow-aligned surfaces is applicable to all other integral surfaces as well and is useful in its own right: Figure 3 demonstrates that our normalized parametrizations are less distorted compared to the unnormalized time line/stream line parametrizations, which are generated by common stream surface integrators [16]. We use this type of parametrization to compute LIC-like visualizations in Section 6.

Orthogonal parametrization. One application of orthogonally aligned surfaces for flow analysis is their applicability as general seed structures. Therefore, points on the interactively specified surfaces can also directly be used for seeding, e. g., *illuminated stream lines* [25] (see Figure 5). To extract seed curves for surface integration we again rely on a parametrization. By construction the vector field does not directly provide directions that are tangential to the surface and usable for parametrization. To obtain these directions we perform a global surface-dependent rotation in the following way: for the normal $\mathbf{n}_c$ at the constrained vertex v_c as the reference, we compute a global rotation axis $\mathbf{r}_c$ by using one basis vector of the null space of $\mathbf{n}_c$. The basis of the null space can, e. g., be obtained by using a QR-factorization $\mathbf{n}_c = \mathbf{Q}_c\,\mathbf{R}_c$ and taking the second and third column of $\mathbf{Q}_c$. Then the unnormalized guidance field is given by $\mathbf{a}_j = \mathbf{P}_j\,(\mathbf{R}(\mathbf{n}_c, \alpha)\,\mathbf{r}_c) \times \mathbf{v}_j$. Here we have introduced one additional degree of freedom for the user in that $\mathbf{r}_c$ can additionally be rotated in the null space by an angle α to align the resulting linear iso-contours differently in the orthogonal surface. The resulting iso-contours are general curves as they are embedded in orthogonally aligned surfaces. However, they have the tendency to span the surface in a straight way (see Figure 2).

Circular seeds. For certain flow phenomena straight seed curves may not be the desired type of seed structures. For example, recently circular seed curves were successfully used for illustrative flow visualization by Hummel et al. [17]. This type of curves can also be extracted in our computational framework. Note that circular curves can be regarded as iso-contours of a geodesic distance field centered at v_c. We use the recent method by Crane et al. [12], who compute geodesics using a heat flow method, because it uses the same gradient-based operators we use in our work. We only sketch an outline of this method here. In essence, their method first performs a heat flow integration from a point source to obtain a heat distribution scalar field. Then a guidance vector field is obtained by normalizing the negative gradient of the heat distribution. The heat integration is performed by a single implicit backward Euler step by solving $(\mathbf{I} - t\mathbf{L})\,\mathbf{h} = \mathbf{h}_0$ for the heat scalar field $\mathbf{h}$. Here $\mathbf{h}_0$ is the initial heat distribution that is one at v_c and vanishes on all other vertices. As proposed by the authors we use a scale-invariant time step of $t = {}^{5A_{\mathcal{M}}}/_{|\mathcal{T}|}$, where $A_{\mathcal{M}}$ is the total surface area. Then the guidance vector field is obtained by normalizing the negative gradients $\mathbf{a} = -\mathbf{G}\,\mathbf{h}$ on each triangle and a final Poisson system is solved with these gradients for a distance field $d(\mathbf{x})$. We use iso-contours of $d(\mathbf{x})$ to extract circular seeds on orthogonally aligned surfaces.

6 Parametrization-based LIC-like Visualization

The Poisson-based parametrization presented in the previous section can directly be used to compute LIC-like texture-based flow structure visualizations for stream and path surfaces [22]. For the unnormalized tangential case ($\mathbf{a}_j = \mathbf{v}_j$) the key observation is that the energy (6) approximates flow directions with gradients of $r(\mathbf{x})$ in least-squares sense. The scalar field $r(\mathbf{x})$ can therefore be interpreted as an approximation for the time coordinate of a *global* space-time parametrization of stream as well as path surfaces. Such space-time parametrizations are either hard to compute or are of poor quality (see Figure 3). For this reason surface-based LIC techniques rely on chart-packing or screen-space techniques to overcome the absence of a high quality parametrization [21, 23, 15].

We use global tangential parametrizations to efficiently compute LIC-like visualizations. These special parametrizations allow to simply map precomputed seamless structured textures onto the integral surface to visualize the flow structure. One choice is a high contrast texture of colored noise that is anisotropically low-pass filtered in time direction and generates LIC-like flow structure patterns. Alternatively, an illustrative visualization of flow directions is obtained by using a texture of flow-aligned arrows. The textures are

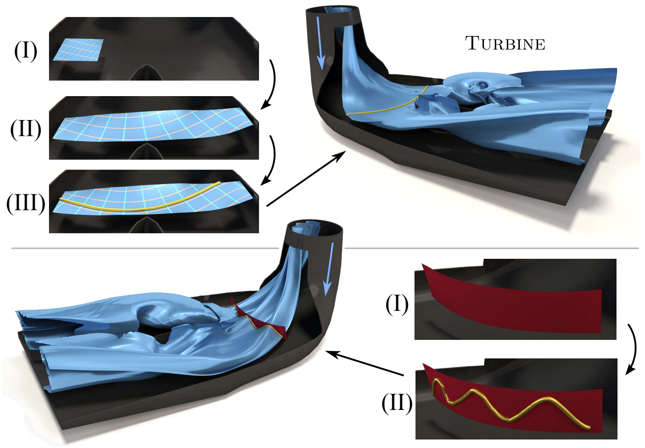


Figure 4: Seeding at the TURBINE. Top: a tangential surface is interactively placed into the inflow area of the flow (I). It is then grown *orthogonally* to the flow (II) and an iso-contour of a tangential parametrization yields a non-straight seed curve (III) for the integration of the final stream surface (top right). Bottom: onto an interactively placed orthogonal surface (I) a orthogonal seed curve is manually “drawn” (II) and used for stream surface integration.

shown in Figure 8 (bottom right). As texture structures are approximately aligned to flow directions due to the energy-minimizing parametrizations the visualization have a LIC-like character.

Flow directions are only approximated in least-squares sense and are not necessarily exactly interpolated everywhere by the minimizers of (6). However, in all our experiments we found that the gradients of $r(\mathbf{x})$ are generally very well fitted to the flow and will only diverge in small regions of the surface. The time component of the energy e_p therefore vanishes almost everywhere on the surface and the LIC-like visualization is only incorrect in small localized regions. To remedy this limitation we mask these regions by blending the texture with the average texture color at each pixel. The blending function $b(\gamma)$ depends on the local angle $\gamma(\mathbf{v}(\mathbf{x}), \nabla r(\mathbf{x}))$ between the flow and the gradient of the time component of the parametrization. A smooth cubic sigmoid-shaped blending polynomial that interpolates $b(0) = 0$ and $b(\pi/2) = 1$ turns out to be sufficient to hide the parts of the surface where $\nabla r(\mathbf{x})$ is not exactly aligned with $\mathbf{v}(\mathbf{x})$. The remaining structures of the visualization are aligned with the flow and are also distributed proportional to $||\mathbf{v}||$ since we use $\mathbf{a}_j = \mathbf{v}_j$ to fit the parametrization.

This technique does not require any integration nor costly texture advection and only amounts to optimizing one global parametrization energy. As we only have to sample the vector field once to setup the resulting linear system (7) the visualization is very efficient to compute. Additionally, the visualization is frame-coherent and the texture map can also be used to animate the flow structures in steady stream surfaces: an animation of flow structures is obtained by simply offsetting the parametrization time-coordinate for each frame, resulting in a movement of the flow structure along approximated stream lines. Note that animation is only meaningful for stream and not for path surfaces.

7 Results

Seed selection. Figure 5 shows integration results starting from seed curves on orthogonal surfaces. The orthogonal surfaces were all placed interactively with number of triangles ($|\mathcal{T}|$) ranging from 1.7×10^3 (STALLING2D) to 3.5×10^3 (CYLINDER). Note that shapes of interactively deformed surfaces are rather insensitive to tessellation quality as the energy (3) is an integrated quantity and a discretization of a continuous energy. The STALLING2D flow is a conservative two-dimensional benchmark data set proposed by Stalling [34]. We show multiple uniformly distributed

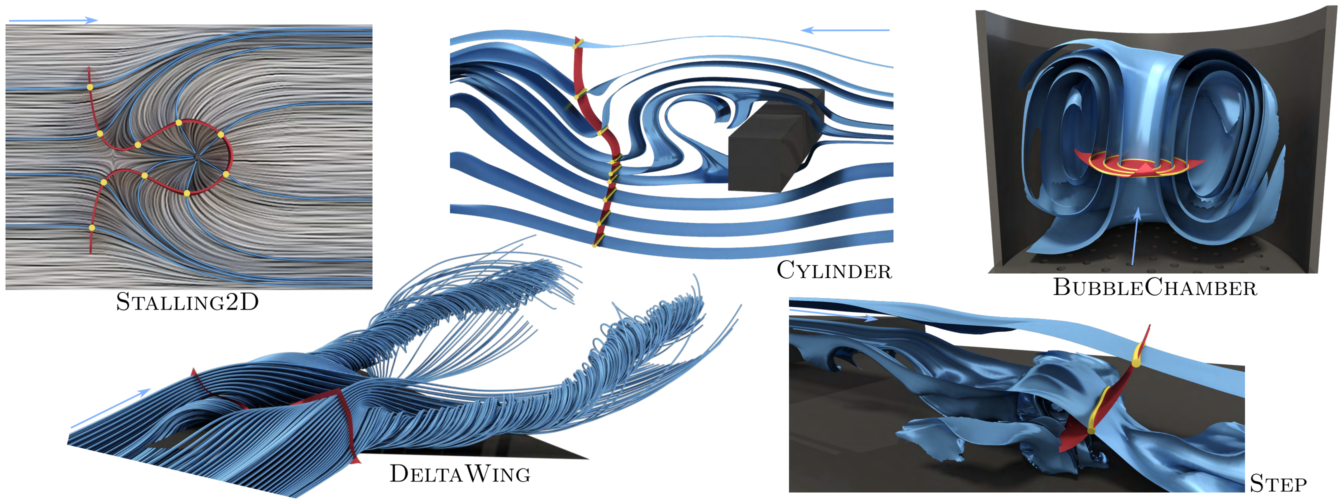


Figure 5: Interactive seeding results. The surfaces orthogonal surfaces (●) are positioned interactively by the user. Then stream surfaces and stream lines (●) are seeded from these orthogonal surfaces and integrated tangentially to the flow. Both orthogonal (STALLING2D, CYLINDER, and STEP) as well as circular parametrizations (BUBBLECHAMBER, cutaway view) were used to extract flow orthogonal seed curves.

stream surfaces on a highly curved orthogonal surface. Our approach for orthogonal surfaces does not suffer from "spiky" artifacts at fixed vertices that may appear using the APAP approach (see Figure 5 in [32]). A similar example is shown for the more complex CYLINDER flow [9] (see, e. g., [8, 13, 31]) where an orthogonal surface was placed near the flow vortex. Iso-contour variation reveals the narrow outflow region of the vortex through the orthogonal surface. Different levels of the tumbling flow in the measured BUBBLECHAMBER data set of a bioreactor can be visualized using circular seeding curves on an orthogonal surface located at the turnover point of the flow. The simulated DELTAWING data set of the flow at a triangularly-shaped airplane is known to contain two dominant vortical structures [14, 5, 17, 31]. Flow-orthogonal surfaces can be used to seed integrated quantities like stream lines near the assumed locations of the vortices for their concrete visualization. Stream surfaces of different complexity can be selected as different iso-contours in an orthogonal surface in the STEP data set of a flow behind a backward-facing step [18].

Figure 4 (top) illustrates an effective work flow to compute seed curves in prescribed flow tangential surface patches at the example of a hydroelectric TURBINE [31]: a tangentially aligned surface is placed in the inflow region and subsequently grown *orthogonally* to the flow. Then a seed curve can be extracted from the tangential parametrization defining a stream surface that covers a large part of the domain and interpolates the initial patch. Note that extending a stream surface orthogonally to the flow is not possible with classical stream surface integrators, but poses no problem for our deformation-based approach. Figure 4 (bottom) shows another stream surface integrated from an orthogonal seed curve "drawn" by the user on an orthogonal surface, which was positioned interactively. The video shows examples of interactive seeding.

LIC-like visualization. Figure 8 shows LIC-like and illustrative visualizations based on tangential parametrizations for both stream and path surfaces. The visualized structures capture flow directions in all non-masked regions and bifurcations are correctly represented by the parametrizations. We evaluate average angles (in degree) and their standard deviation $(\bar{\gamma}, \sigma_\gamma)$ between $\mathbf{v}$ and ∇r for the DELTAWING $(1.0, 0.9)$, STEP $(2.2, 2.4)$, and TURBINE $(2.7, 5.7)$ data sets, which are close to the optimal solution. The larger standard deviation is caused by outlier regions, which are masked by blending. We note that for a few examples the parametrization may not be onto, i. e., it may contain self intersections in the parameter domain. However, self intersections in the parametrization pose no problem in this application as they only

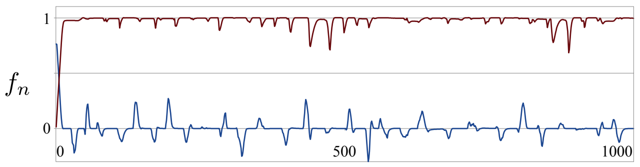


Figure 6: Flux convergence in interactive session. The graph shows the normalized flux f_n of each deformation iteration in an interactive session for tangentially (●) and orthogonally (●) aligned surfaces. Translations, rotations as well as growing operations were performed by the user, which result in quickly optimized "flux spikes".

result in texels mapping to multiple parts of the surface. As the textures have no structure except flow alignment, which is maintained even at self intersections, no visual artifacts or incorrect visualizations arise. See the accompanying video for examples of frame-coherence / viewport-independence and steady flow animation.

Convergence. The interactive deformations presented in Section 4 are designed to either maximize or minimize total flux iteratively. We quantify the convergence of the deformation iteration within interactive user sessions, the results are visualized in Figure 6. As flux depends on the vector field norm (which can significantly vary in the flow domain), we measure the normalized flux $f_n = A_{\mathcal{M}_k}^{-1} \sum_{t_j \in \mathcal{T}} A_j \, \mathbf{n}_j^{\mathrm{T}} \mathbf{v}_j / \|\mathbf{v}_j\|$ in each iteration k. Note that $f_n = 1$ for exactly orthogonal surfaces, and f_n vanishes for exactly tangential surfaces. We observe rapid decay for both surface types such that convergence is achieved after few iterations. Moreover, tangential surfaces always converge to exactly aligned surfaces, whereas orthogonal surfaces may not always reach an exact state. This is not surprising, since perfect orthogonality is generally not possible [26], except for, e. g., conservative flows.

Performance. We solve linear systems using the CHOLMOD [11] library to perform sparse Cholesky factorization with fill-in reducing reordering. Solving Poisson-type problems has, in theory, sub-quadratic complexity but scales even better in practice [6]. All our examples were computed on an Intel Core i7-2600 3.4GHz Linux PC. In Table 1 we list measured timings of our single-threaded deformation method. Using a mesh resolution of $|\mathcal{T}| \approx 5{,}000$ turns out to be sufficient for all tested data sets, but significantly larger meshes can still be handled without problems. The timings indicate that interactive results can be guaranteed. A linear system factorization (FACTOR) has to be computed only once and in each iteration

$\vert\mathcal{T}\vert$	Preprocess FACTOR	Iteration GTRANSF	SOLVE	IT/S	NCONV
8×10^2	0.5	0.26	0.16	2,300	5
5×10^3	2.3	1.5	0.25	570	9
2×10^4	13.2	6.4	0.83	130	16
8×10^4	932	24.2	4.5	35	19
3.2×10^5	8,000	91.1	23.0	9	22

Table 1: Deformation timings. For differently sized meshes ($\vert\mathcal{T}\vert$) we list the required times to perform system factorization (FACTOR), gradient transformation (GTRANSF), and system solving (SOLVE) in *ms*. IT/S indicates performed number of iterations per second, NCONV denotes the average number of iterations for convergence.

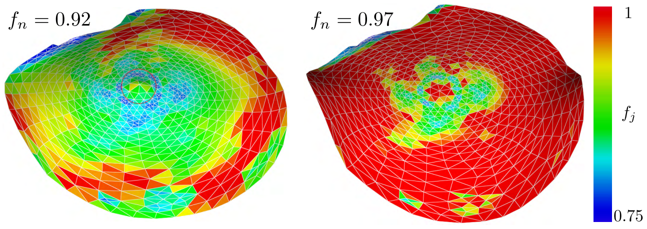


Figure 7: APAP comparison. Starting from a converged APAP [32, Figure 10] surface (left) our deformation converges to an even better aligned orthogonal surface (right). The color scale captures normalized $f_j \in [0.75, 1]$. The highlighted fixation artifact at the center vertex is also removed by our deformation.

only efficient gradient transformations (GTRANSF) and system solving (SOLVE) by back-substitution need to be performed. Note that local gradient transformation computations require vector field sampling (see Section 4), which turns out to be costly for high mesh resolutions. However, since each transformation can be computed independently, this operation is a natural candidate for parallelization, e. g., by parallel vector field sampling on the GPU. As almost identical systems are solved for Poisson-based parametrization timings for tangential/orthogonal parametrization as well as for the LIC-like visualizations are very similar to the presented deformation timings.

8 Discussion

Comparison to APAP. For flux optimization we search for the same class of surfaces as the APAP approach: orthogonally aligned surfaces. A key difference is that the vertices of our surfaces are not constrained to only move along (scaled) flow directions, but they move along general paths. This way we achieve higher flux rates compared to the APAP approach. Moreover, our method does not generate deformation artifacts at constrained vertices, whereas even with careful regularization these artifacts cannot be completely suppressed by the APAP approach. Figure 7 exemplifies both properties where we initialize our deformation method with a converged APAP surface. Additionally, we only need to perform the expensive linear system factorization *once*, whereas APAP requires to solve a new linear system in *every* integration step.

Our method does, however, not strive to replace classical front line-based stream surface integrators for tangential surfaces in terms of approximation quality. Rather, our approach applies Poisson-based methods to enhance the usability and effectiveness of the concept of surface integrators, e. g., by approximate interactive deformation-based selection of interesting flow regions as a "preview" and by parametrization-based nontrivial seed curve extraction. Growing tangential surfaces orthogonally to the flow is another technique that is not possible with classical methods.

Limitations. Our deformation method is based on local alignment of surface orientations to the flow. High curvature regions of the flow require tangential surfaces to change their normal rapidly. This is only possible for surfaces that are locally tessellated sufficiently high enough. However, since we perform no adaptive mesh refinement yet, the deformation iteration is unlikely to converge in these areas. An adaptive subdivision scheme (with costly refactorization) and criteria for its applications would therefore be required to also handle these regions. Note that this lack of refinement is not a severe problem for the current approach as meshes do not degenerate in these areas; the mesh is rather "pushed out" of high curvature areas due to folding penalization (see video).

The quality of LIC-like visualizations depends to a great extent on the quality of the underlying parametrization. Although exact flow alignment cannot be guaranteed by the least-squares parametrization energy, the presented results indicate that the parametrizations are flow-aligned almost everywhere in data sets of practical relevance. Still, there exist regions where flow alignment may not exactly be met and which we mask in the visualization. Our parametrization trades exact flow alignment, which is hard to compute, for more efficient computation, which may not be exact everywhere. In contrast, screen-space methods become unstable at silhouettes, and chart-packing approaches suffer from texture resolution and discontinuity artifacts. The explicit hierarchical handling of critical or hard-to-parameterize regions [23] and the analysis of better suited parametrization energies is, therefore, an interesting direction for further research.

9 Conclusions

In this work we showed that Poisson-based interactive deformations and parametrizations are well-suited to support explorative analysis and seed specification to study flow phenomena. Additionally, we introduce a LIC-like rendering technique for surface-based flow visualization. Since these methods are all based on a small set of well-established differential operators, it is easy to integrate our approach into existent visualization packages.

A possible direction for further research is the application of the deformation method to time-dependent vector fields. Another promising direction is the incorporation of additional application-dependent surface quality measures into the optimization. We believe that the proposed parametrizations are also beneficial for, e. g., applications like remeshing characteristic surfaces.

Acknowledgements

We thank Tino Weinkauf for resampling the CYLINDER data set. The DELTAWING data set is courtesy of Markus Rütten, DLR. We thank Axel Seeger and Klaus Affold for providing the BUBBLECHAMBER data set. The TURBINE flow is part of the visualization system AMIRA. The primary author has been partially funded by the Studienstiftung des deutschen Volkes.

References

[1] R. Abraham, J. Marsden, and T. Ratiu. *Manifolds, Tensor Analysis, and Applications*. Applied Mathematical Sciences. Springer, 1988.
[2] ANSYS Inc. CFD-Post, Sep. 2012. URL: `www.ansys.com`.
[3] S. Bachthaler and D. Weiskopf. Animation of orthogonal texture patterns for vector field visualization. *TVCG*, 14(4):741–755, 2008.
[4] D. Bommes, H. Zimmer, and L. Kobbelt. Mixed-integer quadrangulation. *TOG (Proc. SIGGRAPH)*, 28(3):78–87, 2009.
[5] S. Born, A. Wiebel, J. Friedrich, G. Scheuermann, and D. Bartz. Illustrative stream surfaces. *TVCG (Proc. Vis)*, 16(6):1329–1338, 2010.
[6] M. Botsch, D. Bommes, and L. Kobbelt. Efficient linear system solvers for mesh processing. In *Proc. MS*, pages 62–83, 2005.
[7] M. Botsch and O. Sorkine. On linear variational surface deformation methods. *TVCG*, 14(1):213–230, 2008.
[8] K. Bürger, F. Ferstl, H. Theisel, and R. Westermann. Interactive streak surface visualization on the gpu. *TVCG (Proc. Vis)*, 15(6):1259–1266, 2009.

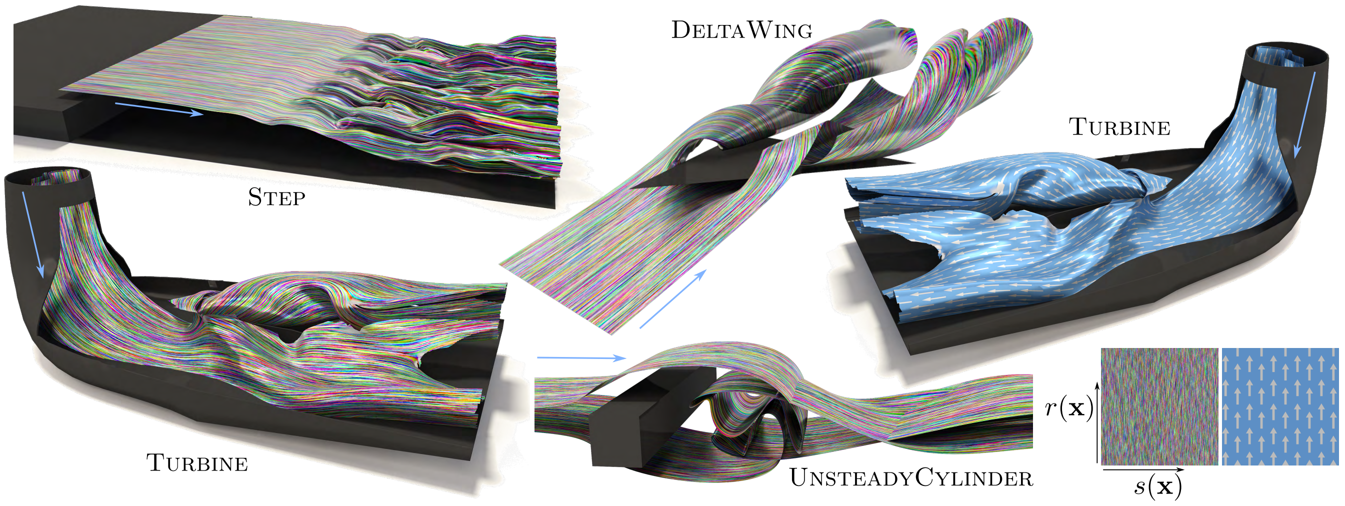


Figure 8: LIC-like visualization results. Precomputed textures of anisotropic noise and flow-aligned arrows (bottom right) are mapped to the tangentially parametrized stream surfaces (Step, Turbine, DeltaWing) and two path surfaces (UnsteadyCylinder). Resulting LIC-like and illustrative structures are flow-aligned everywhere except at locally masked regions (e. g., at the vortex of the DeltaWing).

[9] S. Camarri, M.-V. Salvetti, M. Buffoni, and A. Iollo. Simulation of the three-dimensional flow around a square cylinder between parallel walls at moderate Reynolds numbers. In *AIMETA XVII*, 2005.

[10] J. Y. Chang, K. M. Lee, and S. U. Lee. Multiview normal field integration using level set methods. In *Proc. CVPR*, pages 1–8, 2007.

[11] Y. Chen, T. A. Davis, W. W. Hager, and S. Rajamanickam. Algorithm 887: Cholmod, supernodal sparse cholesky factorization and update/downdate. *TOMS*, 35(3):1–14, 2008.

[12] K. Crane, C. Weischedel, and M. Wardetzky. Geodesics in heat. *CoRR*, 2012.

[13] M. Edmunds, R. S. Laramee, R. Malki, I. Masters, T. N. Croft, G. Chen, and E. Zhang. Automatic stream surface seeding: A feature-centered approach. *CGF (Proc. EuroVis)*, 31(3):1095–1104, 2012.

[14] C. Garth, H. Krishnan, X. Tricoche, T. Bobach, and K. Joy. Generation of accurate integral surfaces in time-dependent vector fields. *TVCG*, 14(6):1404–1411, 2008.

[15] J. Huang, W. Pei, C. Wen, G. Chen, W. Chen, and H. Bao. Output-coherent image-space lic for surface flow visualization. In *Proc. PacificVis*, pages 137–144, 2012.

[16] J. P. M. Hultquist. Constructing stream surfaces in steady 3d vector fields. In *Proc. VIS*, pages 171–178, 1992.

[17] M. Hummel, C. Garth, B. Hamann, H. Hagen, and K. Joy. Iris: Illustrative rendering for integral surfaces. *TVCG (Proc. Vis)*, 16(6):1319–1328, 2010.

[18] H.-J. Kaltenbach and G. Janke. Direct numerical simulation of flow separation behind a swept, rearward-facing step at $Re_H = 3000$. *POF*, 12(9):2320–2337, 2000.

[19] H. Krishnan, C. Garth, and K. Joy. Time and streak surfaces for flow visualization in large time-varying data sets. *TVCG (Proc. Vis)*, 15(6):1267–1274, 2009.

[20] J. Krüger, P. Kipfer, P. Kondratieva, and R. Westermann. A particle system for interactive visualization of 3D flows. *TVCG*, 11(6):744–756, 2005.

[21] R. S. Laramee, C. Garth, J. Schneider, and H. Hauser. Texture advection on stream surfaces: A novel hybrid visualization applied to cfd simulation results. In *Proc. EuroVis*, pages 155–162, 2006.

[22] R. S. Laramee, H. Hauser, H. Doleisch, B. Vrolijk, F. H. Post, and D. Weiskopf. The state of the art in flow visualization: Dense and texture-based techniques. *CGF*, 23(2):203–221, 2004.

[23] G.-S. Li, X. Tricoche, D. Weiskopf, and C. Hansen. Flow charts: Visualization of vector fields on arbitrary surfaces. *TVCG*, 14(5):1067–1080, 2008.

[24] L. Liu, L. Zhang, Y. Xu, C. Gotsman, and S. J. Gortler. A local/global approach to mesh parameterization. *CGF (Proc. SGP)*, 27(5):1495–1504, 2008.

[25] T. McLoughlin, R. S. Laramee, R. Peikert, F. H. Post, and M. Chen. Over two decades of integration-based, geometric flow visualization. *CGF*, 29(6):1807–1829, 2010.

[26] K. L. Palmerius, M. Cooper, and A. Ynnerman. Flow field visualization using vector field perpendicular surfaces. In *Proc. SCCG*, pages 27–34, 2009.

[27] P. Pérez, M. Gangnet, and A. Blake. Poisson image editing. *TOG (Proc. SIGGRAPH)*, 22(3):313–318, 2003.

[28] F. Post, B. Vrolijk, H. Hauser, R. Laramee, and H. Doleisch. The state of the art in flow visualization: Feature extraction and tracking. *CGF*, 22(4):775–792, 2003.

[29] O. Rosanwo, C. Petz, S. Prohaska, I. Hotz, and H.-C. Hege. Dual streamline seeding. In *Proc. PacificVis*, pages 9–16, 2009.

[30] T. Schafhitzel, E. Tejada, D. Weiskopf, and T. Ertl. Point-based stream surfaces and path surfaces. In *Proc. GI*, pages 289–296, 2007.

[31] M. Schulze, T. Germer, C. Rössl, and H. Theisel. Stream surface parametrization by flow-orthogonal front lines. *CGF (Proc. SGP)*, 31(5):1725–1734, 2012.

[32] M. Schulze, C. Rössl, T. Germer, and H. Theisel. As-perpendicular-as-possible surfaces for flow visualization. In *Proc. PacificVis*, pages 153–160, 2012.

[33] O. Sorkine and M. Alexa. As-rigid-as-possible surface modeling. In *Proc. SGP*, pages 109–116, 2007.

[34] D. Stalling. *Fast Texture-Based Algorithms for Vector Field Visualization*. PhD thesis, ZIB, 1998.

[35] R. W. Sumner and J. Popović. Deformation transfer for triangle meshes. *TOG (Proc. SIGGRAPH)*, 23(3):399–405, 2004.

[36] W. von Funck, T. Weinkauf, H. Theisel, and H.-P. Seidel. Smoke surfaces: An interactive flow visualization technique inspired by real-world flow experiments. *TVCG (Proc. Vis)*, 14(6):1396–1403, 2008.

[37] T. Weinkauf, H.-C. Hege, and H. Theisel. Advected tangent curves: A general scheme for characteristic curves of flow fields. *CGF (Proc. EG)*, 31(2):825–834, 2012.

[38] T. Weinkauf and H. Theisel. Streak lines as tangent curves of a derived vector field. *TVCG (Proc. Vis)*, 16(6):1225–1234, 2010.

[39] D. Xu, H. Zhang, Q. Wang, and H. Bao. Poisson shape interpolation. *GM*, 68(3):268–281, 2006.

[40] Y. Yu, K. Zhou, D. Xu, X. Shi, H. Bao, B. Guo, and H.-Y. Shum. Mesh editing with poisson-based gradient field manipulation. *TOG (Proc. SIGGRAPH)*, 23(3):644–651, 2004.

[41] R. Zayer, C. Rössl, Z. Karni, and H.-P. Seidel. Harmonic guidance for surface deformation. *CGF (Proc. EG)*, 24(3):601–609, 2005.

[42] R. Zayer, C. Rössl, and H.-P. Seidel. Discrete tensorial quasi-harmonic maps. In *Proc. SMA*, pages 278–287, 2005.