As-Perpendicular-as-Possible Surfaces for Flow Visualization

Maik Schulze[1*] Christian Rössl[1] Tobias Germer[1,2] Holger Theisel[1]

[1]University of Magdeburg [2]think-cell Software GmbH

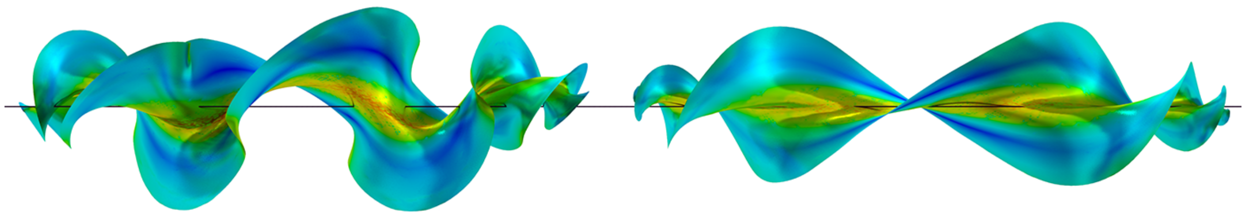


Figure 1: Flow swirling around a straight line. Two APAP surfaces are seeded near the core line at different distances.

Abstract

We define APAP surfaces, surfaces that are *as perpendicular as possible* to steady 3D vector fields, and present a method to construct discrete representations of them. Since, in general, a perfectly perpendicular surface to a vector field does not exist, we propose and minimize an error metric to enforce perpendicularity as much as possible. Our algorithm constructs an APAP surface by deforming a seed surface anchored in a domain point. In the discrete setting this minimization results in iteratively solving linear least-squares problems and integrating a locally scaled version of the vector field. The definition of the error metric and its numerical minimization guarantee that the minimum zero is attained for the perfectly perpendicular surface if it exists. Otherwise, the minimization converges to the same local minimum independent of the seed configuration, and the resulting surface is – in a least-squares sense – as perpendicular as possible to the flow. We apply these APAP surfaces as an interactive flow visualization tool which we demonstrate on a number of synthetic and real flow data sets.

Index Terms: I.3.5 [Computational Geometry and Object Modeling]: Geometric algorithms, languages, and systems—

1 Introduction

Vector fields play a vital role in many applications in Vision, Graphics and Visualization. Given a 3D vector field $\mathbf{v}$, there is a number of characteristic surfaces describing different aspects of the flow, such as stream, path, streak, and time surfaces. They are exhaustively studied in the literature. They all have in common that they evolve along the flow direction. In addition, surfaces that mainly evolve perpendicularly to the vector field have moved into the focus of consideration in different fields of application. Informally spoken, such a surface is characterized by the fact that at every surface point the direction of the surface normal (approximately) coincides with the direction of the vector field $\mathbf{v}$. Keeping in mind that this is not a formal and unique definition yet, we call such surfaces *as-perpendicular-as-possible* (APAP) surfaces.

APAP surfaces are considered in different areas of application: in Computer Vision they are used for surface reconstruction from vector fields in shape-from-shading approaches. In the visualization of diffusion tensors they are considered in regions where fibers cross as an alternative to fiber tracking. In flow visualization, they are considered to show local or global properties of the flow.

APAP surfaces are well-defined for conservative vector fields, i.e., for fields $\mathbf{v}$ which can be expressed as the gradient of a certain scalar field s. In this case, APAP surfaces are (parts of) the isosurfaces of s. However, if $\mathbf{v}$ is not conservative, a surface perfectly perpendicular to the flow does not exist. Existing approaches either modify $\mathbf{v}$ to be conservative, or they provide extraction algorithms whose results depend on arbitrarily chosen parameters and seed configurations. A review and discussion of existing methods and their limitations is given in section 2. We believe that such ambiguities in the extraction methods are due to the lack of a formal definition of the concept of APAP surfaces.

In this paper we give a mathematically rigorous definition of APAP surfaces as surfaces minimizing a certain error functional. The main idea is to start with a simple initial surface $\mathbf{s}_0$, on which a time-surface integration of a scaled version $\alpha\,\mathbf{v}$ of the vector field is applied. The scaling field α is chosen such that the time surfaces converge to a state of minimizing the error under certain boundary conditions – usually fixing a particular surface point. Based on this definition we present an approach to compute discrete APAP surfaces. The user provides a triangle mesh representing an arbitrary seed surface which is anchored at a specified domain point. Our algorithm integrates this surface along $\alpha\,\mathbf{v}$ until the error becomes minimal. For this, the field α has to be defined on the surface. In the discrete setting it is obtained from solving a sparse linear least-squares problem. We preserve accuracy of the APAP surface approximation by an adaptive refinement and relaxation of the triangulation during the integration. We also allow for growing the surface across its boundaries. Our implementation enables the use of APAP surfaces as an interactive flow exploration tool. We discuss its applicability on a number of synthetic and real 3D flow fields.

2 Related Work

APAP surfaces have been considered in Computer Vision, tensor visualization, and flow visualization. We review them here together with their problems and shortcomings. In addition, we give a short overview of flow visualization to classify APAP surfaces as an interactive flow exploration tool.

2.1 APAP Surfaces in Computer Vision

In Computer Vision, shape-from-shading approaches aim in reconstructing surfaces from measured illumination conditions of a single image or multiple images [5]. Deriving surface orientation is a difficult non-linear and ill-posed problem with no general solution. Most approaches use variational methods [12, 8], Fourier and wavelet basis functions [9, 15] or a direct solution to the Poisson equation [27]. [19] uses belief propagation to obtain an integrable gradient field.

*e-mail: maik@isg.cs.uni-magdeburg.de

For these approaches it can be assumed that the unknown surface satisfies the integrability condition. The non-integrability is treated as an error and removed from the field yielding a conservative normal field. This property does not hold for visualization of general vector fields that can be far from conservative and where a projection onto a conservative field is meaningless.

2.2 APAP Surfaces in DT-MRI Visualization

In DT-MRI visualization, fiber tracking is a standard technique. However, in regions where fibers cross, this method tends to fail because the diffusion tensor becomes planar, i.e., the larger two eigenvalues have approximately the same size. For these regions, so-called DT-MRI stream surfaces have been proposed [28, 33, 38] which are the surfaces perpendicular to the (in these regions well-defined) minimum eigenvector. Even though this has been widely referenced in DT-MRI visualization [22, 34, 39], [26] shows that these surfaces generally do not exist and that existing extraction methods depend on arbitrarily chosen parameters.

2.3 APAP Surfaces in Flow Visualization

In flow visualization, APAP surfaces of local first order approximations at particular points are considered for glyph techniques [7] or for computing local differential geometric properties [36]. [18] describes APAP surfaces as surfaces where the local angle between flow and surface normal is smaller than a certain threshold. Unfortunately, such a surface is not uniquely defined: even in a small neighborhood of a point there are infinitely many such surfaces which pass through it. This leads – similar to DT-MRI stream surfaces – to the fact that the proposed extraction methods depend on arbitrary parameters, namely the direction in which the proposed growing starts. In fact, the counterexample for DT-MRI stream surfaces in [26] applies to the proposed surfaces in [18] as well.

Contrary to the surfaces in the 3D case, curves perpendicular to the flow in the plane or on a surface are well-defined and used, e.g., in [2]. [23] use perpendicular curves for streamline seeding in 2D vector fields.

2.4 Flow Visualization

Flow visualization is a well-researched field of scientific visualization, see, e.g., the surveys [17, 21]. Among the existing techniques, an interactive exploration of the flow is enabled by stream surfaces [13, 25, 29], path surfaces [10, 24], and streak surfaces [4, 16, 35, 37]. However, contrary to these surfaces our approach explores the flow perpendicular to the flow direction.

One straightforward solution of the problem of non-integrable vector fields may come into one's mind: given a vector field, decompose it into its conservative and divergence-free part by a Helmholtz-Hodge decomposition [20, 32] and compute the perfectly perpendicular surface from the conservative flow part. Note that this approach does not give solutions if $\mathbf{v}$ is divergence-free, since in this case the conservative part vanishes. (We consider a divergence-free flow in the example shown in Figure 7.)

3 As-perpendicular-as-possible Surfaces

In this section we provide the definition of as-perpendicular-as-possible surfaces, first in the continuous case and then in the discrete case.

3.1 Continuous Formulation

Given is a 3D vector field $\mathbf{v}(\mathbf{x})$ over a domain D and a regularly parametrized surface $\mathbf{s}(u,v)$ in D with parameters $(u,v) \in [u_0,u_1] \times [v_0,v_1]$. In addition, we require that in no surface point of $\mathbf{s}$ the vector of $\mathbf{v}$ lies in the tangential plane of $\mathbf{s}$ and that $\mathbf{v}$ does not have critical points on $\mathbf{s}$, i.e., $\det(\mathbf{s}_u, \mathbf{s}_v, \mathbf{v}(\mathbf{s})) \neq 0$ for any $(u,v) \in [u_0,u_1] \times [v_0,v_1]$. We consider a vector $\mathbf{r}$ in the tangent plane of $\mathbf{s}$,

$$\mathbf{r}(u,v,\gamma) = \frac{\cos\gamma}{\sqrt{E}}\,\mathbf{s}_u + \frac{\sin\gamma}{\sqrt{EG-F^2}\sqrt{E}}(F\,\mathbf{s}_u - E\,\mathbf{s}_v) \qquad (1)$$

where $\mathbf{s}_u, \mathbf{s}_v$ denote the partial derivatives of $\mathbf{s}$ and $E = \mathbf{s}_u^T\mathbf{s}_u$, $F = \mathbf{s}_u^T\mathbf{s}_v$, $G = \mathbf{s}_v^T\mathbf{s}_v$. (1) is constructed such that $\|\mathbf{r}\| = 1$ and $\mathbf{r}$ is moving with constant angular velocity while linearly increasing γ. If $\mathbf{s}$ was a surface perfectly perpendicular to $\mathbf{v}$ then the relation $\mathbf{r}^T\mathbf{v} = 0$ holds for any (u,v,γ). For the general case we define the local error function

$$e(u,v,\gamma) = \mathbf{r}(u,v,\gamma)^T\,\tilde{\mathbf{v}}(\mathbf{s}(u,v)) \qquad (2)$$

with $\tilde{\mathbf{v}} = \frac{\mathbf{v}}{\|\mathbf{v}\|}$. It is a measure of how perpendicular $\mathbf{v}$ and $\mathbf{r}$ are locally. Note that the error term in (2) is equipped with a sign. From (2) we define the global error

$$E = \int_0^{2\pi}\int_{v_0}^{v_1}\int_{u_0}^{u_1} \|\mathbf{s}_u \times \mathbf{s}_v\| \cdot (e(u,v,\gamma))^2\; du\,dv\,d\gamma\,, \qquad (3)$$

which describes how perpendicular $\mathbf{v}$ and $\mathbf{s}$ are globally. (Note that E does not depend on the parameterization of $\mathbf{s}$.) The main idea to define APAP surfaces is to integrate $\mathbf{s}$ along a vector field $\alpha\,\tilde{\mathbf{v}}$ until E is minimized. For this, α is a certain scalar field which has to be constructed to minimize E but preserves certain properties and boundary constraints inherent to $\mathbf{s}$. In order to define the scalar field α, we realize that it is evaluated only on the surface $\mathbf{s}$, i.e., it is sufficient to consider $\alpha(u,v,t)$ where t is the integration time. This gives the time-dependent surface $\mathbf{s}(u,v,t)$ satisfying

$$\frac{\partial \mathbf{s}(u,v,t)}{\partial t} \;=\; \alpha(u,v,t)\cdot\tilde{\mathbf{v}}(\mathbf{s}(u,v,t))\,,$$

with the initial condition $\mathbf{s}(u,v,0) = \mathbf{s}_0(u,v)$. We want to choose α such that (the signed quantity) e is corrected towards 0 during the integration. We express this by demanding

$$\frac{\partial e(u,v,t)}{\partial t} + e(u,v,t) \;=\; 0\,, \qquad (4)$$

which can be interpreted as follows: the larger the error e is, the more it decreases towards 0 during the integration of $\alpha\,\tilde{\mathbf{v}}$. What remains to be done is to find conditions for α to satisfy (4). Using the fact that

$$\frac{\partial e}{\partial t} \;=\; \left(\frac{\partial \mathbf{r}}{\partial t}\right)^T \tilde{\mathbf{v}} \;+\; \mathbf{r}^T\frac{\partial \tilde{\mathbf{v}}}{\partial t} \qquad (5)$$

and keeping in mind that the directional derivative of $\tilde{\mathbf{v}}$ in a certain direction $\mathbf{d}$ can be written as $\tilde{\mathbf{J}}\,\mathbf{d}$ where $\tilde{\mathbf{J}}$ is the Jacobian matrix of $\tilde{\mathbf{v}}$, we get

$$\begin{aligned}\frac{\partial \tilde{\mathbf{v}}}{\partial t} &= \tilde{\mathbf{J}}\,(\alpha\,\tilde{\mathbf{v}})\\ \frac{\partial \mathbf{r}}{\partial t} &= \alpha\cdot\tilde{\mathbf{J}}\,\mathbf{r} + \frac{\partial\alpha}{\partial \mathbf{r}}\,\tilde{\mathbf{v}}\,.\end{aligned}$$

Inserting the above into (5) and this into (4), we get

$$\begin{aligned}f(\alpha) \;&:=\; \alpha\cdot\mathbf{r}^T(\tilde{\mathbf{J}}^T + \tilde{\mathbf{J}})\tilde{\mathbf{v}} + \frac{\partial\alpha}{\partial\mathbf{r}}\cdot\tilde{\mathbf{v}}^T\tilde{\mathbf{v}} + \mathbf{r}^T\tilde{\mathbf{v}} \qquad (6)\\ &=\; \alpha\cdot\mathbf{r}^T(\tilde{\mathbf{J}}^T + \tilde{\mathbf{J}})\tilde{\mathbf{v}} + \frac{\partial\alpha}{\partial\mathbf{r}} + \mathbf{r}^T\tilde{\mathbf{v}} \qquad = 0\,,\end{aligned}$$

where $\frac{\partial\alpha}{\partial\mathbf{r}}$ is the directional derivative of α in direction $\mathbf{r}$. The fact that the only derivatives of α in (6) are the directional derivatives in $\mathbf{r}$-direction gives that the computation of α can perfectly be combined with the integration of $\mathbf{s}$: given $\mathbf{s}$ at a certain time $\hat{t}$, we compute $\alpha(u,v,\hat{t})$ by solving (6) in a least-squares sense at $\hat{t}$. From this we can integrate $\alpha\,\tilde{\mathbf{v}}$ to get $\mathbf{s}$ at the next time step.

Now we can give a formal definition of APAP surfaces.

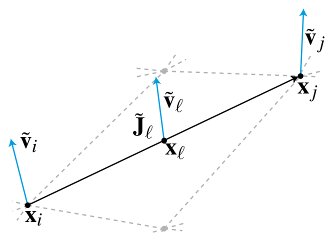


Figure 2: Quantities required for the local discrete error at an edge $\ell = \{i, j\}$: vertex positions $\mathbf{x}_i$, $\mathbf{x}_j$, normalized flow vectors $\tilde{\mathbf{v}}_i, \tilde{\mathbf{v}}_j$ evaluated at these locations as well as in the midpoint ($\tilde{\mathbf{v}}_\ell$), and the Jacobian $\tilde{\mathbf{J}}_\ell$ of $\tilde{\mathbf{v}}$ at the midpoint of the edge.

Definition 1 *A surface* $\mathbf{s}$ *is an* as-perpendicular-as-possible surface (APAP surface) *iff the scalar field* $\alpha(u,v) = 0$ *minimizes the integral*

$$\bar{E}(\alpha) = \int_0^{2\pi} \int_{v_0}^{v_1} \int_{u_0}^{u_1} f(\alpha)^2 \, du \, dv \, d\gamma \,. \tag{7}$$

Note that this definition conforms to the special case, i.e., if a surface is perfectly perpendicular then it is an APAP surface because it minimizes (7) to $\bar{E}(\alpha) = 0$. Informally spoken, this definition states that any integration of $\mathbf{s}$ in $\alpha \tilde{\mathbf{v}}$ with $\alpha \neq 0$ must increase the error $\bar{E}$.

Furthermore, the equation (7) provides a way to construct a unique APAP surface for a given domain point $\hat{\mathbf{x}}$. Consider an initial seed surface $\mathbf{s}_0$ through $\hat{\mathbf{x}}$ and set $\alpha(\hat{\mathbf{x}}) \equiv 0$ as boundary condition, making sure that the integrated surface still passes through $\hat{\mathbf{x}}$, then the construction of α and the integration of $\alpha \tilde{\mathbf{v}}$ converges to an APAP surface through $\hat{\mathbf{x}}$.

3.2 Discretization

In the discrete setting the surface is approximated by a triangulation of vertices $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}, \mathbf{x}_i \in \mathbb{R}^3$. In the following, we require only the notion of edges $\{i, j\}$ which connect two adjacent vertices $\mathbf{x}_i$ and $\mathbf{x}_j$ in the triangulation. Note that this definition does explicitly not provide an orientation of edges, $\{i, j\}$ and $\{j, i\}$ are equivalent. We assume a valid triangulation without degenerated triangles or fold-overs. Furthermore, we assume that initially the vertices are distributed equally on the surface such that edges are of almost equal length. In the remainder of this section, we consider one particular time step, and we leave out parameters in the notation if they are clear from the context.

We search for a piecewise linear scalar field $\alpha(u, v, \hat{t})$ which allows us to make the mesh as perpendicular as possible to the flow field $\tilde{\mathbf{v}}$ in the current time step $t = \hat{t}$. The scalar field is represented by values $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n)$ at the vertices. We obtain the APAP surface iteratively by integrating the vertex positions along the scaled field $\alpha_i \tilde{\mathbf{v}}(\mathbf{x}_i)$.

We first derive the error term for each edge. For this, we consider the error function e in the midpoint of each edge $\ell = \{i, j\}$ between the vertices $(\mathbf{x}_i, \mathbf{x}_j)$. Setting

$$\tilde{\mathbf{v}}_i = \tilde{\mathbf{v}}(\mathbf{x}_i) \,,\ \tilde{\mathbf{v}}_j = \tilde{\mathbf{v}}(\mathbf{x}_j) \,,\ \mathbf{x}_\ell = \frac{\mathbf{x}_j + \mathbf{x}_i}{2} \,,\ \tilde{\mathbf{v}}_\ell = \tilde{\mathbf{v}}(\mathbf{x}_\ell) \,,\ \tilde{\mathbf{J}}_\ell = \tilde{\mathbf{J}}(\mathbf{x}_\ell) \,,$$

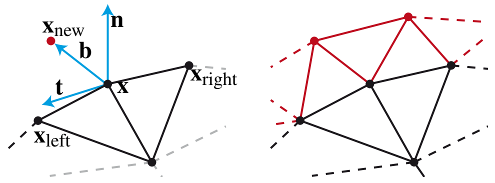
we obtain the discrete error value at $\mathbf{x}_\ell$ as

$$e_\ell = (\mathbf{x}_j - \mathbf{x}_i)^T \tilde{\mathbf{v}}_\ell \tag{8}$$

and demand

$$\frac{\partial e_\ell}{\partial t} + e_\ell = 0 \,. \tag{9}$$

Figure 2 illustrates the setup.



$\mathbf{x}_{\mathrm{new}}$
$\mathbf{n}$
$\mathbf{b}$
$\mathbf{t}$
$\mathbf{x}$
$\mathbf{x}_{\mathrm{right}}$
$\mathbf{x}_{\mathrm{left}}$

Figure 3: Surface growing. The position of the offset vertex $\mathbf{x}_{\mathrm{new}}$ is computed from the discrete boundary curve and its unit tangent $\mathbf{n}$ and normal $\mathbf{n}$ vectors (left). The boundary polygon and the offset points form a triangle strip (red) which is included in the triangulation.

Using

$$\begin{aligned}
\frac{\partial e_\ell}{\partial t} &= \frac{\partial (\mathbf{x}_j - \mathbf{x}_i)}{\partial t}^T \tilde{\mathbf{v}}_\ell + (\mathbf{x}_j - \mathbf{x}_i)^T \frac{\partial \tilde{\mathbf{v}}_\ell}{\partial t} \\
\frac{\partial \tilde{\mathbf{v}}_\ell}{\partial t} &= \tilde{\mathbf{J}}_\ell \frac{\alpha_i \tilde{\mathbf{v}}_i + \alpha_j \tilde{\mathbf{v}}_j}{2} \\
\frac{\partial (\mathbf{x}_j - \mathbf{x}_i)}{\partial t} &= \alpha_j \tilde{\mathbf{v}}_j - \alpha_i \tilde{\mathbf{v}}_i \,,
\end{aligned}$$

equation (9) can be written as

$$\alpha_i \, p_\ell + \alpha_j \, q_\ell + r_\ell = 0 \tag{10}$$

with

$$\begin{aligned}
p_\ell &= \frac{1}{2} \left(\mathbf{x}_j - \mathbf{x}_i\right)^T \tilde{\mathbf{J}}_\ell \, \tilde{\mathbf{v}}_i - \tilde{\mathbf{v}}_i^T \tilde{\mathbf{v}}_\ell \\
q_\ell &= \frac{1}{2} \left(\mathbf{x}_j - \mathbf{x}_i\right)^T \tilde{\mathbf{J}}_\ell \, \tilde{\mathbf{v}}_j + \tilde{\mathbf{v}}_j^T \tilde{\mathbf{v}}_\ell \\
r_\ell &= \left(\mathbf{x}_j - \mathbf{x}_i\right)^T \tilde{\mathbf{v}}_\ell \,.
\end{aligned}$$

Equation (10) is the discrete version of equation (6), and we can express the continuous surface integral $\bar{E}(\alpha)$ in (7) as a discrete sum over all edges. (In the following we do this in matrix notation.)

Considering equation (10) for all edges simultaneously leads to a linear system in the unknown variables α_i:

$$\mathbf{A} \boldsymbol{\alpha} = -\mathbf{r} \tag{11}$$

with $\mathbf{A} \in \mathbb{R}^{m \times n}, \boldsymbol{\alpha} \in \mathbb{R}^n$, and $\mathbf{r} \in \mathbb{R}^m$, where m is the number of edges in the surface triangulation. The system matrix $\mathbf{A}$ is sparse, each row represents an edge, the columns correspond to vertices. With $1 \leq l \leq m$ denoting the index of a particular edge $\ell = \{i, j\}$ its structure is

$$\mathbf{A}_{l,v} = \begin{cases} p_\ell & \text{if } v = i \\ q_\ell & \text{if } v = j \\ 0 & \text{otherwise} \end{cases} \tag{12}$$

for vertices $1 \leq v \leq n$.

4 Construction of discrete APAP surfaces

In the previous section we gave the definition for APAP surfaces and derived a discrete version. In this section we show how to find the required discrete scalar field α in practice. Using this field we construct discrete APAP surfaces from an initial seed surface by an integration along the flow field $\alpha \tilde{\mathbf{v}}$.

We want to solve the linear system (11) in order to obtain α for a particular time step. This system is generally overdetermined, as the number of edges is generally $m \approx 3n$ for sufficiently large meshes with a single boundary. Counting the degrees of freedom reveals that also for the discrete setting we cannot expect a solution leading to a surface which is perfectly perpendicular to the flow. Instead of solving $\mathbf{A}\boldsymbol{\alpha} + \mathbf{r} = 0$ we determine the optimal solution $\boldsymbol{\alpha}$ in least-squares sense, i.e., we minimize

$$||\mathbf{A} \boldsymbol{\alpha} + \mathbf{r}||^2 \,, \tag{13}$$

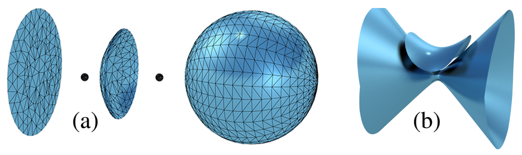


Figure 4: (a) Surface growing. We consider the conservative vector field $\mathbf{v}(\mathbf{x}) = \nabla \mathbf{x}$, the critical point $\mathbf{0}$ is marked. In this case the APAP surface is a spherical isosurface. We start with disk as seed surface (left) which evolves to part of a sphere (center) as APAP surface. After surface growing the mesh covers the whole sphere. (b) Two APAP surfaces in a conservative linear field with a saddle.

refering to the continuous $\bar{E}(\alpha)$ in (7). We do this by solving the normal equations

$$\mathbf{A}^T \mathbf{A} \boldsymbol{\alpha} = -\mathbf{A}^T \mathbf{r} \, .$$

The system matrix $\mathbf{A}^T\mathbf{A}$ is sparse, symmetric and positive-definite if (1.) the triangulation is not degenerate in the sense that for every edge $\{i, j\}$ the vector $\mathbf{x}_j - \mathbf{x}_i$ does not vanish, and (2.) the vector field $\mathbf{v}$ does not have critical points at $\mathbf{x}_i$ and $\mathbf{x}_j$. We assume that the first condition is satisfied due to our requirements on the seed (see section 3.2) and relaxation during the minimization (see section 4.2). We address the second condition by adding a regularization term $\mu_f ||\boldsymbol{\alpha}||^2$ to (13), i.e., we solve $\left(\mathbf{A}^T\mathbf{A} + \mu_f \mathbf{I}\right) \boldsymbol{\alpha} = -\mathbf{A}^T \mathbf{r}$ for a small scalar μ_f. This improves numerical robustness, and this way $\boldsymbol{\alpha}$ is well-defined also in the vicinity of critical points. In practice, if we hit a critical point exactly, the movement of the surface stops locally, vertices close to a critical point keep moving.

For vector field exploration we require an additional constraint: we want the APAP surface to interpolate a given point of interest. The easiest way to achieve this is to fix a certain vertex $\hat{\mathbf{x}} = \mathbf{x}_c \in \mathbf{X}$ by adding the constraint $\alpha_c = 0$, i.e., the vertex does not move during integration along the vector field $\alpha \tilde{\mathbf{v}}$. It is well-known that constraining the normal equations directly sacrifices smoothness of the resulting surface at the constraint, i.e., fixing a single vertex $\mathbf{x}_c$ would manifest in a small and thin "spike". As this behavior is not desirable, we relax the interpolation constraint and replace it by an approximation constraint: we would like to remain as close as possible near $\mathbf{x}_c$ or equivalently keep $||\alpha_c||^2$ as small as possible. Technically this results in adding another regularization term, or more precisely penalty term, this time weighted with a relatively high value $\mu_c \gg \mu_f$. The higher the value of μ_c, the less is the drift of a constrained vertex $\mathbf{x}_c$ with interpolation of the constraints in the limit case $\mu_c = \infty$. Note that it is sufficient for our purpose to constrain one single vertex.

As both regularization terms are of the same type, we can write the resulting linear system as

$$\left(\mathbf{A}^T \mathbf{A} + \mathbf{D}\right) \boldsymbol{\alpha} = -\mathbf{A}^T \mathbf{r} \, , \tag{14}$$

where $\mathbf{D} \in \mathbb{R}^{n \times n}$ is a diagonal matrix with

$$\mathbf{D}_{ii} = \begin{cases} \mu_c & \text{if vertex } i \text{ is constrained} \\ \mu_f & \text{if vertex } i \text{ is free .} \end{cases}$$

In our implementation we used values $\mu_f = 10^{-5}$ and $\mu_c = 10^4$.

The system matrix is symmetric and positive definite, and we solve the linear system by a sparse Cholesky factorization. There are established and highly efficient numerical algorithms and software libraries, which enable the processing of large meshes with tens or even hundreds of thousands of vertices. We use the CHOLMOD [6] library in our implementation.

4.1 Error Minimization by Integration

Given is a seed surface placed in domain D such that one vertex $\mathbf{x}_c$ is anchored to a point of interest $\hat{\mathbf{x}}$. We translate and rotate the initial surface such that $\mathbf{x}_c = \hat{\mathbf{x}}$ and the directions of the surface normal $\mathbf{n}(\mathbf{x}_c)$ and the flow field $\mathbf{v}(\mathbf{x}_c)$ coincide at the anchor point.

We minimize the error – the deviation from a perfectly perpendicular surface – by integrating the surface in the field $\alpha \tilde{\mathbf{v}}$. In the discrete setting we do this in discrete time steps: in each time step we solve (14) for the current scalar field α. Then we apply a fourth-order Runge-Kutta integration step with fixed step size h and scale the result by α. Note that α may be negative, i.e., the surface may (locally) move backwards. We repeat this for a sequence of time steps until the change in the residual $||\mathbf{A}\boldsymbol{\alpha} + \mathbf{r}||^2$, which approximates the continuous error $\bar{E}(\alpha)$, is sufficiently small. (Figure 11 shows error plots over time; the error decreases rapidly, and generally only few time steps are required.) The additional least-squares constraint $\mu_c ||\alpha_c||^2$ penalizes movement of the constrained vertex $\mathbf{x}_c$, see (14). Effectively, this anchor vertex remains close to its initial position $\hat{\mathbf{x}}$, and the integrated surface stays globally smooth. In summary, we use the following algorithm to minimize $\bar{E}(\alpha)$.

- Translate/rotate seed s.t. for anchor $\hat{\mathbf{x}} = \mathbf{x}_c$: $\mathbf{n}(\mathbf{x}_c) \, || \, \mathbf{v}(\mathbf{x}_c)$
- Repeat over time steps ...
  - setup and solve the linear system (14) to get the field α
  - compute a step-size $\Delta^h(\mathbf{x}_i, t_i)$ for every vertex $\mathbf{x}_i$
  - integrate every vertex: $\mathbf{x}_i \leftarrow \mathbf{x}_i + \alpha_i \Delta^h(\mathbf{x}_i, t_i)$
  - *optionally* adapt the surface mesh (see below)
- ... until change of global error $||\mathbf{A}\boldsymbol{\alpha} + \mathbf{r}||^2$ is below a threshold

The optional adaptation step ensures the quality of the surface sampling and triangulation. In addition, it enables dilating or growing the surface. We describe surface mesh adaptation in the next section.

4.2 Surface Mesh Adaptation

The approximation of APAP surfaces depends on the discrete sampling: the number of vertices and their distribution over the surface. We start with an equal vertex distribution as seed. During integration, the vertices may move not only in surface normal direction but also tangentially. This tangential drift results in a change of the sampling rate: edge lengths and triangle areas increase or decrease locally, i.e., there are fewer or more vertex samples per local area. The effect depends on the flow field $\mathbf{v}$, and in fact, the tangential

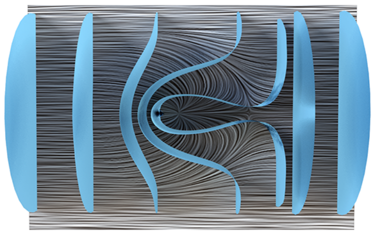


Figure 5: APAP surfaces in a conservative flow [29]. The LIC image of the underlying 2D vector field reveals the sink and saddle.

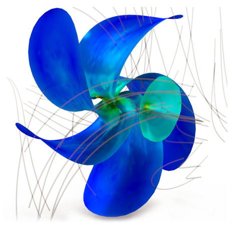


Figure 6: Five APAP surfaces combined with illuminated streamlines visualize the focus saddle flow.

drift can become significant over time. This can lead to undersampling and cannot be tolerated for a high quality approximation.

A standard approach to remedy this effect is remeshing, well-known in geometry processing. There are various approaches with different goals, and we refer to [1] for a review. A simple and efficient method [3] iterates edge collapses and edge splits for short and long edges, edge flips to balance vertex degrees near the optimal number 6, and a tangential relaxation of vertex positions to equalize edge lengths. The goal is a quasi-regular triangulation with all triangles being close to equilateral for a given target edge length.

We observed that undersampling due to large triangles is especially critical in our setting. Therefore, we apply a local refinement for all triangles (and boundary edges) whose area (length) exceeds a threshold. We use the $\sqrt{3}$-refinement [14] – a 1-3-split of triangles followed by edge flips – which provides a truly local refinement at low computational cost. The refinement can be combined with a tangential relaxation based on a uniform Laplacian smoothing. This results in a more equal vertex distribution.

4.3 Surface Growing

We implement another feature related to surface adaptation: surface growing – the user may interactively expand the APAP surface over its current boundaries. Starting with a relatively large seed surface captures the flow more globally and hence evolves to a different error function. This can result in a globally different surface. In order to overcome this restriction, we allow for a dynamic surface expansion in any time step of the integration of the APAP surface. The expansion is done progressively, layer by layer, with an interleaved error minimization based on the growing surface as new seed.

In order to expand the surface by one layer of triangles, we iterate over the boundary vertices. For each boundary vertex we estimate its normal vector and its tangent vector by taking central differences on the boundary edge loop. With the unit normal vector $\mathbf{n}$ and the unit tangent vector $\mathbf{t}$ we obtain a new offset point at position $(\mathbf{n} \times \mathbf{t}) \cdot \varepsilon$ within an approximate ε-distance based on the desired edge length. The value of ε is proportional to the average edge length. The new vertices and the boundary define a triangle strip which is added to the triangulation yielding a new offset boundary curve. Figure 3 illustrates this process.

This heuristic expansion works well in regions of small vector field curvature. In this case, the new vertices will move only slightly during the subsequent error minimization. However, in regions with high vector field curvature the heuristic may fail in a sense that the new points are far from their optimal position. In addition, the optimization process may suffer from triangle fold-overs which generally cannot be avoided for the offset. We remedy this by a global uniform Laplacian smoothing step on the current surface: every inner vertex is moved towards the barycenter of it neighbors, and every boundary vertex is moved towards the barycenter of its boundary neighbors. We damp the Laplacian displacements by a factor of $\frac{2}{5}$; the anchor vertex remains fixed. The rationale is that the tangential part of the displacements equilibrates the vertex distribution, while the normal part provides a geometric smoothing. Assuming that the optimal APAP surface is smooth, the latter improves the position of the heuristically inserted vertices in a sense that they get closer to their optimal position. At the same time, the subsequent error minimization corrects the normal displacement of the inner vertex. The global smoothing step provides numerical stability to the heuristic expansion of the surface to a new seed. This step is essentially the same step as the relaxation for mesh adaptation, the only difference is whether the Laplacian displacements are projected into the tangent plane or not.

In our experiments we observe that growing the surface by iterating boundary expansion, smoothing, and APAP error minimization is robust and stable, independent of the starting configuration. Figure 4a shows a simple example where APAP surfaces are spherical isosurfaces. Surface growing enables starting with a moderately sized disk-shaped seed which evolves to geometric sphere. (Of course, topologically, the surface remains a disk.) Our examples include conservative fields, see Figures 4b and 8a, for which we apply surface growing, and for which the discrete APAP surfaces converge to isosurfaces.

5 User Interaction

This section describes how a user can interact with APAP surfaces in our implementation, and how these surfaces are used for interactive flow visualization. Please see also the accompanying video which briefly demonstrates the visualization and interaction. The video shows parts of the interactive sessions played back at 1.5× real time (3×real time for cylinder example).

Control of APAP surfaces. Most useful are planar disk-shaped seed meshes where the user can interactively choose radius and sampling rate. The user can translate and rotate the seed mesh in the flow. Optionally, flow vectors are evaluated on the seed surface for immediate feedback. This is useful for orientation if no additional visualization such as vector or streamline samples is used. Generally, we propose to combine the visualization of APAP surfaces with other established techniques like illuminated streamlines [40].

The user can "anchor" the seed surface at a point of interest, usually the surface center, and compute an APAP surface passing through this point. Then, the surface normal at this point is aligned with the vector field by rotating the surface. Here, we also enable the user to choose a different orientation of the seed. This way, different parts of the vector field are sampled and therefore a different orientation may manifest in different APAP surfaces. A manual alignment was used in Figure 1 and near the vortex cores in Figure 9. (Note that the orientation has no influence for conservative flow fields.) After this, the user can start the error minimization process. After each iteration the surface is displayed. This results in an animation. Similarly, the user can interactively grow the surface according to his needs. In the video we grow several surfaces over *multiple* offset layers (see section 4.3) before we start the animated error minimization. Generally, we recommend alternate error minimization and growing to increase stability.

The user can place an arbitrary number of APAP surfaces. Each surface can be modified, e.g., by growing the surface or moving the anchor point and recomputing the surface, or by removing it from the visualization.

Rendering. For the evaluation of our algorithm we render APAP surfaces with the local error color-coded on the surface. For color coding we "normalize" the error, i.e., we compute the angular error $e_\ell / ||\mathbf{x}_j - \mathbf{x}_i||$ – cosine of angle between vector field and edge – and take averages of all incident edges for vertices. The color map in the pictures and in the video maps blue and red to low and high error, respectively. Note that for conservative flow fields a standard shading is preferable as the error is zero everywhere. Of course, any other scalar vector field quantity can be visualized on the APAP surfaces, e.g., curvature.

6 Results

In this section we show results from vector field visualization with APAP surfaces for synthetic and real world data sets. We provide timings to show that the approach is interactive, and we measure convergence of the global error. For some examples, we combine APAP surfaces with other flow visualization techniques like 2D LIC and illuminated streamlines. The color coding on surfaces – if applied – displays the local error.

We start with a couple of synthetic vector fields. The simplest one is the gradient field shown in Figure 4a to demonstrate surface growing.

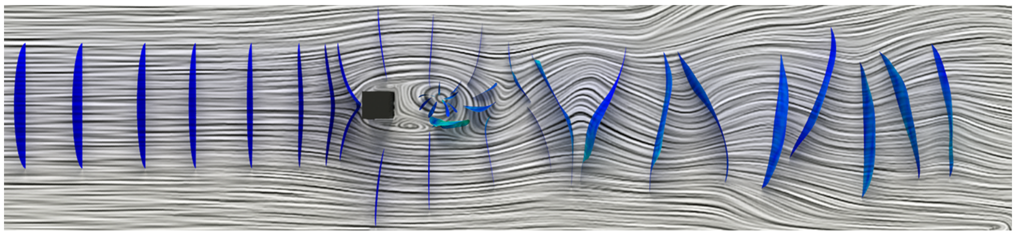


Figure 7: Simulated flow around a square cylinder. We show one time step of the simulation and combine APAP surfaces with a LIC image on a central plane.

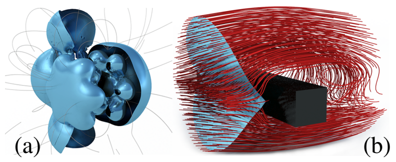


Figure 8: (a) APAP surface near critical point in front of square cylinder. 473 streamlines are seeded at the vertices resulting in an even distribution. (b) Four APAP surfaces on the benzene flow combined with illuminated streamlines.

Another conservative field is a linear vector field with a saddle (Figure 4b). We place two disk-shaped seed surfaces. Note that the double cone is formed by one surface with a single boundary: the difference to the smaller surface is the location of the anchor closer to the critical point and bigger amount of surface growing. This example is a test for regions of high curvature requiring local refinement (see section 4.2) during integration. This is handled well by our algorithm. The number of vertices ranges from $1,105$ to $27,082$ for the larger locally refined APAP surface.

The next example is modeled after a benchmark data set in [29]. It is essentially a 2D flow, constant in the 3rd dimension and requires significant deformation of the seed in order to converge to the APAP surface in the center domain region. Figure 5 shows the result combined with a LIC image of the underlying flow. The maximum number of vertices for the largest APAP surface is $38,874$.

For all conservative fields as well as for the inherently 2D fields (Figure 5) our algorithm converges to the “ground truth” solutions which are perfectly flow-perpendicular surfaces passing through the point of interest.

We show two more synthetic vector fields which are not conservative. Here, the global error converged to a non-zero minimum, the colors visualize the local error. Figure 6 shows five APAP surfaces seeded in the inflow plane of the focus saddle. In the visualization, we combine APAP surfaces with illuminated streamlines. The maximum number of vertices is $5,733$.

Figure 1 shows a flow swirling around a straight line. The APAP surfaces form helical structures which twist around the core line. The helix radius depends on the distance from the anchor point to the line. We show two APAP surfaces with seeds placed in different distance to the core line. We used a maximum of $37,990$ (left) and $25,787$ (right) vertices.

We also show a number of real data sets which stem from numerical simulations. Figure 8a shows the electrostatic field around a benzene molecule. This data set was using the fractional charges method described in [30]. This conservative vector field has 184 critical points. We show four APAP surfaces in combination with illuminated streamlines. We started with seeds of $1,901$ vertices and observe a maximum of $29,235$ vertices for the final APAP surfaces. Figure 7 shows a flow around a square cylinder. The time-dependent data set was obtained from a numerical simulation. The main flow direction is from left to right. We show one time step in the mid of the simulated time span. We combine a LIC image on a central intersection plane with multiple APAP surfaces which behave well in front of the obstacle and close to the critical point. There, we observe high curvature manifested in a sharp bend of the APAP surface. The resulting APAP surfaces in this region are aligned on ellipses which indicates a swirling behavior. The maximum number of vertices was $39,468$.

Figure 8b shows another APAP surface in the cylinder data set. Streamlines are seeded at the 473 surface vertices. We show that APAP surfaces are good seeding structures for streamlines because of the resulting equal density.

Figure 9 shows two different views of one time step of a simulated flow around an airplane wing. The main flow direction is from right to left (red arrow). Swirling flow is found above the wing. Five APAP surfaces are placed in combination with illuminated streamlines. The big, almost planar APAP surface closest to the inflow region shows the laminar flow above the front of the wing and contains 822 vertices. The swirling region is depicted by three surfaces that contain up to $11,551$ vertices. Near the vortex region the seed surfaces were oriented manually to capture the helical structure well. The high curvature in these regions causes significant refinement.

Finally, we show another simulated data set in Figure 10 at two different time steps. The data set consists of 100 time steps showing the relaxation and decay of magnetic Borromean rings. The four yellow APAP surfaces are anchored on the attracting plane of a focus saddle-like center. Two blue APAP surfaces are seeded along the repelling diagonal. The six APAP surfaces consist of up to $4,656$ vertices. The attracting region grows over time, closed structures appear along the repelling direction.

6.1 Performance

We provide timings in Table 1 to show that our algorithm is interactive. Performance was measured on an AMD Dual-Core Opteron 2218 processor at 2.6GHz using a single core. Here, we consider meshes of different size from $n \approx 2,000$ to $n \approx 200,000$ vertices. The table shows typical times for a single time step: we solve the least-squares problem – this involves setup of the linear system, Cholesky factorization and solving using the factor (col. 2–4). Furthermore, we apply relaxation by a Laplacian smoothing step (col. 5). We also measure the time for a global refinement by splitting *all* triangles – in practice we refine locally, then the number of triangles involved depends on properties of the vector field – (col. 6), and we measure the time for growing the surface by one layer of offset vertices (col. 7). The table shows that our implementation is clearly interactive for moderate mesh sizes. In our examples the number of vertices does not exceed $40,000$ for an APAP surface. Even for relatively large meshes ($> 100,000$ vertices), the performance is acceptable. The total computation time is dominated by the solution of the linear system, in particular by the sparse Cholesky factorization.

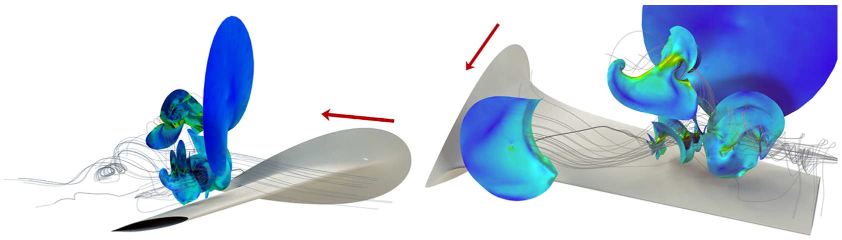


Figure 9: Two views of flow around a wing visualized by five APAP surfaces combined with streamlines. The red arrow shows the main direction.

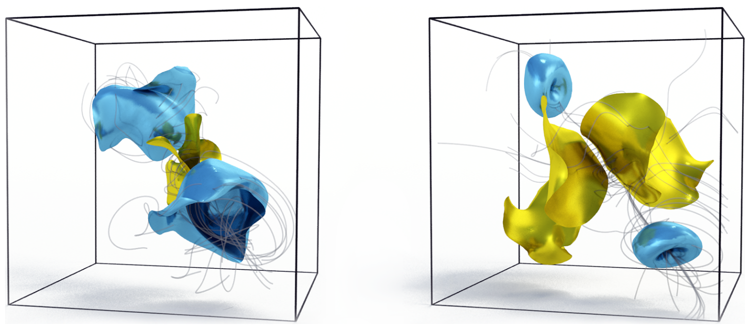


Figure 10: Six APAP surfaces at time-step 50 (left) and 78 (right) indicating the focus saddle-like structure of decaying magnetic Borromean rings.

6.2 Convergence and numerical stability

It remains to show that the minimization converges (sufficiently fast) and is numerically stable. We carried out the following experiment: we measure the residual $||\mathbf{A}\boldsymbol{\alpha}+\mathbf{r}||^2$ (see (13) in section 4) for a sequence of time steps. We expect this error to converge smoothly and rapidly to a minimum. Figure 11 shows plots of the residual for two different vector fields: the first field is conservative (blue), we chose the gradient field as for Figure 4a. We expect the error to drop to zero because every APAP surface is an isosurface perfectly perpendicular to the flow. This is confirmed by the plot. The second example (red) is the swirl around a straight line shown in Figure 1. Here, the global error converges to some minimal value. Note that the global scale of the residual (y-axis) depends on the triangulation and the vector field. Hence, only the relative change – not the absolute values – is meaningful and of interest. Our experiment confirms that our algorithm for computing APAP algorithm reduces the global error smoothly and finds a configuration with minimal error, i.e., the resulting surface is as perpendicular as possible to the flow. The experiment also shows that the rate of convergence is high, and that in practice only few $(5-20)$ integration steps are required.

7 Discussion and Conclusions

Although ad-hoc solutions for APAP surfaces have been applied in different fields of visualization, a rigorous definition of such surfaces together with a stably converging extraction method was not available yet in flow visualization. Our approach closes this gap.

From the standpoint of flow visualization, we see APAP surfaces as an interactive flow exploration tool which does not intend to replace existing curve and surface integration methods but acts as a complementary tool. As APAP surfaces are (nearly) orthogonal to the flow, there is relatively few interference with traditional techniques which evolve geometric objects along the flow. At the same time, sets of APAP surfaces visualize the vector field “implicitly” and provide an intuitive understanding of the flow. Therefore, we think that generally flow visualizations can be enhanced by combining established methods and APAP surfaces. We observe that APAP surfaces are ideal seeding structures for streamlines yielding even distributions.

Another argument in favor of APAP surfaces is the fact that for conservative flows the visualization of the isosurfaces of the underlying scalar field is a natural and well-established choice for visualization. APAP surfaces extend this to general vector fields. We remark that since our definition is based on the concept of a minimum of a function, different local minima for given configuration are theoretically possible. We observe that our approach is not sensitive to the input configuration in a sense that for small changes we find the same minimum and hence the same surface.

Our approach to the construction of APAP surfaces has a number of limitations. First, since the normal of the seeding surface must not be parallel to the flow, the size of the seeding surface is limited especially for complex flows. However, this limitation is attenuated by our surface growing approach (section 4.3).

Second, in areas of very high vector field curvature the discrete representation either suffers from undersampling or becomes too large to be practical: our triangular representation of APAP surfaces runs into excessive subdivision leading to extremely large meshes. In particular, APAP surfaces starting in a critical point of $\mathbf{v}$ are undefined. This, however is not a serious limitation because there are well-established topological methods to represent the flow in a neighborhood of 3D critical points [11, 31], and it only emphasizes that a combination of APAP surfaces and established techniques can lead to intuitive visualizations.

Third, our current implementation is limited to meshes with about $100,000$ vertices as an interactive tool. As shown in our examples, this is sufficient for practical data exploration. Size is less of an issue if the APAP surface is smooth enough such that the mesh can be coarsened adaptively. Challenging are the already mentioned regions of turbulent flow, where an appropriate sampling of APAP surfaces easily exceeds the practical limits. This is not unique to APAP surfaces but an issue for surface integration methods in general. Another challenge is the appropriate handling of “double covering” as, e.g., for the saddle (Figure 4b): this does not impair rendering but it “wastes” triangles increasing the size of the linear system. An automatic detection of such situations together with an adaptation of the APAP surface topology is an interesting problem for future research. Finally, the definition and extraction of

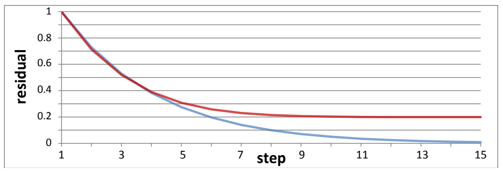


Figure 11: Plot of the residual over time steps to show convergence and numerical stability. Blue: for a conservative field (see Figure 4a), the global error becomes zero. The seed surface was a unit cube around the critical point. Red: The same experiment is carried out for the spiral flow (see Figure 1) and a disk-shaped seed surface. Here, the residual quickly converges to a minimum.

n	setup	factor	solve	smooth	refine	grow
2,513	6.2	6.0	1.1	1.9	24.2	3.4
5,041	12.2	15.5	3.3	2.1	42.9	6.2
15,121	36.6	70.3	12.8	7.5	115.4	11.5
35,640	116.6	267.9	28.0	35.7	286.4	34.2
70,200	200.3	663.7	73.5	56.2	445.7	33.1
136,080	385.0	2,052.0	133.8	82.3	899.4	52.6
210,600	690.5	8,172.3	238.7	147.4	1,579.6	92.8

Table 1: Timings for meshes with n vertices. We measured time to *setup* and *factor* $\mathbf{A}^T\mathbf{A}$, to *solve* the linear system, to *smooth*, *refine* (all triangles), and to *grow* one layer. Timings are given in milliseconds.

APAP surfaces in time-dependent fields remains an open question.

Acknowledgements

We kindly thank Fritz Kemmler for help with the video. The cylinder data is courtesy of Simone Camarri and co-workers, the borromean rings data is courtesy of Simon Candelaresi. The benzene and wing data are courtesy of Konrad-Zuse-Zentrum für Informationstechnik Berlin. We thank Tino Weinkauf for providing resampled versions of them.

References

[1] P. Alliez, G. Ucelli, C. Gotsman, and M. Attene. Recent advances in remeshing of surfaces. In *Shape Analysis and Structuring*. Springer, 2007.

[2] S. Bachthaler and D. Weiskopf. Animation of orthogonal texture patterns for vector field visualization. *IEEE TVCG*, 14:741–755, 2008.

[3] M. Botsch and L. Kobbelt. A remeshing approach to multiresolution modeling. In *Eurographics Symposium on Geometry Processing*, pages 189–196, 2004.

[4] K. Bürger, F. Ferstl, H. Theisel, and R. Westermann. Interactive Streak Surface Visualization on the GPU. *IEEE TVCG*, 15(6):1259–1266, 2009.

[5] J. Y. Chang, K. M. Lee, and S. U. Lee. Multiview normal field integration using level set methods. *CVPR*, 0:1–8, 2007.

[6] Y. Chen, T. A. Davis, W. W. Hager, and S. Rajamanickam. Algorithm 887: Cholmod, supernodal sparse cholesky factorization and update/downdate. *ACM Trans. Math. Softw.*, 35:22:1–22:14, 2008.

[7] W. C. de Leeuw and J. J. van Wijk. A probe for local flow field visualization. In *Proc. IEEE Visualization*, pages 39–45, 1993.

[8] D. A. Forsyth. Shape from texture and integrability. *Computer Vision (Proc. ICCV)*, 2:447–452, 2001.

[9] R. T. Frankot, R. Chellappa, and S. Member. A method for enforcing integrability in shape from shading algorithms. *PAMI'88*, 10:439–451.

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

[11] A. Globus, C. Levit, and T. Lasinski. A tool for visualizing the topology of three-dimensional vector fields. In *Proc. Visualization*, VIS '91, pages 33–40. IEEE Computer Society Press, 1991.

[12] B. K. P. Horn. Height and gradient from shading. *International Journal of Computer Vision*, 5(1):37–75, 1990.

[13] J. Hultquist. Constructing stream surfaces in steady 3D vector fields. In *Proc. IEEE Visualization*, pages 171–177, 1992.

[14] L. Kobbelt. $\sqrt{3}$ subdivision. In *SIGGRAPH*, pages 103–112, 2000.

[15] P. Kovesi. Shapelets correlated with surface normals produce surfaces. *Computer Vision (Proc. ICCV)*, 2:994–1001, 2005.

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

[17] 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, 2004.

[18] K. L. Palmerius, M. Cooper, and A. Ynnerman. Flow field visualization using vector field perpendicular surfaces. In *Proc. SSCG*, pages 35–42, April 2009.

[19] N. Petrovic, I. Cohen, B. J. Frey, R. Koetter, and T. S. Huang. Enforcing integrability for surface reconstruction algorithms using belief propagation in graphical models. *CVPR*, 1:743–748, 2001.

[20] K. Polthier and E. Preuß. Identifying vector field singularities using a discrete hodge decomposition. In H. Hege and K. Polthier, editors, *Visualization and Mathematics III*, pages 112–134. Springer, 2002.

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

[22] B. Preim and D. Bartz, editors. *Visualization in Medicine*. Morgan-Kaufmann-Verlag, 2007.

[23] O. Rosanwo, C. Petz, S. Prohaska, I. Hotz, and H.-C. Hege. Dual streamline seeding. In *Proc. Pacific Vis '09*, pages 9–16, 2009.

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

[25] D. Schneider, A. Wiebel, and G. Scheuermann. Smooth stream surfaces of fourth order precision. *CGF*, 28:871–878, 2009.

[26] T. Schultz, H. Theisel, and H.-P. Seidel. Crease surfaces: From theory to extraction and application to diffusion tensor MRI. *IEEE TVCG*, 16(1):109–119, 2010.

[27] T. Simchony, R. Chellappa, and M. Shao. Direct analytical methods for solving poisson equations in computer vision problems. *IEEE Trans. Pattern Anal. Mach. Intell.*, 12:435–446, May 1990.

[28] R. Sondershaus and S. Gumhold. Meshing of diffusion surfaces for point-based tensor field visualization. In *IMR*, 2003.

[29] D. Stalling. *Fast Texture-based Algorithms for Vector Field Visualization*. PhD thesis, 1998.

[30] D. Stalling and T. Steinke. Visualization of vector fields in quantum chemistry. Technical report, ZIB Preprint SC-96-01, 1996.

[31] H. Theisel, T. Weinkauf, H.-C. Hege, and H.-P. Seidel. Saddle connectors - an approach to visualizing the topological skeleton of compelx 3d vector fields. In *Proc. IEEE Visualization*, pages 225–232, 2003.

[32] Y. Tong, S. Lombeyda, A. N. Hirani, and M. Desbrun. Discrete multiscale vector field decomposition. *Proc. SIGGRAPH '03*, 22:445–452.

[33] A. Vilanova, G. Berenschot, and C. van de Pul. Dti visualization with streamsurfaces and evenly-spaced volume seeding. In *VisSym*, pages 173–182, 2004.

[34] A. Vilanova, S. Zhang, G. Kindlmann, and D. Laidlaw. An introduction to visualization of diffusion tensor imaging and its applications. In J. Weickert and H. Hagen, editors, *Visualization and Processing of Tensor Fields*, pages 121–153, 2006.

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

[36] T. Weinkauf and H. Theisel. Curvature measures of 3d vector fields and their applications. *Journal of WSCG*, 10:507–514, 2002.

[37] T. Weinkauf and H. Theisel. Streak lines as tangent curves of a derived vector field. *IEEE TVCG*, 16(6), November - December 2010.

[38] S. Zhang, C. Demiralp, and D. H. Laidlaw. Visualizing diffusion tensor mr images using streamtubes and streamsurfaces. *IEEE TVCG*, 9:454–462, 2003.

[39] S. Zhang, G. Kindlmann, and D. Laidlaw. Diffusion tensor MRI visualization. In *The Visualization Handbook*. Academic Press, 2004.

[40] M. Zöckler, D. Stalling, and H.-C. Hege. Interactive visualization of 3d-vector fields using illuminated stream lines. In *Proc. IEEE Visualization*, pages 107–113, 1996.