# Design of Tangent Vector Fields

Matthew Fisher
Caltech

Peter Schröder
Caltech

Mathieu Desbrun
Caltech

Hugues Hoppe
Microsoft Research

## Abstract

Tangent vector fields are an essential ingredient in controlling surface appearance for applications ranging from anisotropic shading to texture synthesis and non-photorealistic rendering. To achieve a desired effect one is typically interested in smoothly varying fields that satisfy a sparse set of user-provided constraints. Using tools from Discrete Exterior Calculus, we present a simple and efficient algorithm for designing such fields over arbitrary triangle meshes. By representing the field as scalars over mesh edges (*i.e.*, discrete 1-forms), we obtain an intrinsic, coordinate-free formulation in which field smoothness is enforced through discrete Laplace operators. Unlike previous methods, such a formulation leads to a linear system whose sparsity permits efficient pre-factorization. Constraints are incorporated through weighted least squares and can be updated rapidly enough to enable interactive design, as we demonstrate in the context of anisotropic texture synthesis.

**Keywords:** Discrete exterior calculus, discrete differential 1-forms, constrained Laplace and Poisson problems for 1-forms, texture synthesis

## 1 Introduction

Smoothly varying tangent vector fields appear in many applications that control the appearance of surfaces. Examples include anisotropic shading [Schlick 1994], texture synthesis [Turk 2001; Wei and Levoy 2001], non-photorealistic rendering [Hertzmann and Zorin 2000], line integral convolution [Cabral and Leedom 1993], and spot noise [van Wijk 1991] among many others. In some settings one starts with a given vector field, while in others the user must first *design* a vector field. We present a method for designing smooth vector fields over triangle meshes from interactively specified user constraints (see Figures 1 and 9 for some examples from texture synthesis).

**Vector field interpolation** A common approach in user-driven vector field design calls for specifying a sparse set of vector constraints at selected mesh vertices, followed by interpolation of these vectors over all vertices of the mesh. Praun *et al.* [2000] use radial basis functions to form this interpolation, while Turk [2001] uses hierarchical low pass filtering. One may also perform such interpolation or alignment optimizations directly in the (2D) tangent spaces [Pedersen 1995; Hertzmann and Zorin 2000; Wei and Levoy 2001], though coordinate transformations between different tangent spaces must be accounted for carefully. All these approaches rely on explicit coordinate frames and vectors represented through coefficients in these frames, be they 2D or 3D. The parallel transport of tangent vectors between these coordinate frames generally makes vector field optimization a nonlinear problem, and hence prior techniques have not supported interactive global design.

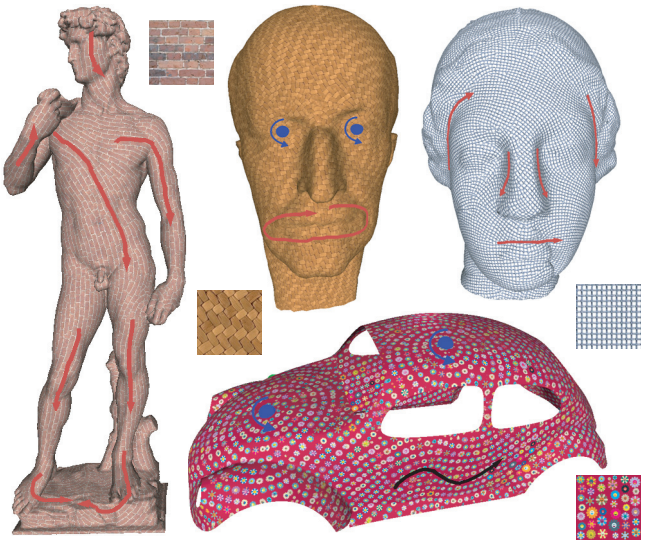


Figure 1: *Examples from texture synthesis. Curves are drawn on the surface to control the flow lines of the underlying vector fields. Features, such as vortices (blue), can be added by marking them on the surface. (Insets show the exemplars.)*

In contrast, we are able to formulate vector field design as a linear problem by using an intrinsic, *coordinate-free* approach based on discrete differential forms and the associated Discrete Exterior Calculus (DEC) [Desbrun et al. 2006]. The central concept in this approach is the representation of fields through measurements on cells: a 0-form represents a scalar function through its values at vertices (0-dim cells), while a 2-form represents area density through its area integral over triangles (2-dim cells). More relevant to our goals, a 1-form represents a *tangent vector field* through its line integral along edges (1-dim cells). This implies that tangent vector fields are specified as a *single scalar per edge* on the mesh. Since measurements are by their very nature independent of the coordinate frame in which they were taken, so are these scalar coefficients. All relevant computations are then performed on these coefficients and the results reconstructed with piecewise linear (PL) interpolation. For relatively fine triangle meshes such PL interpolation is generally sufficient and certainly the easiest. For coarser meshes, smoother interpolants are more appropriate, and suitable constructions were introduced by [Wang et al. 2006]. (If desired, these smoother bases can be used with the same mathematical framework that we describe in this paper.)

**Vector field topology** Vector fields can also be designed using vector field topology, *i.e.*, by considering the relative arrangement of singularities, their indices and how singularities can be merged, split, and moved [Theisel 2002; Zhang et al. 2006]. In particular, Zhang *et al.* [2006] design smooth tangent vector fields on non-flat triangle meshes. Their principal design tool is the placement of singular basis vector fields, which are mapped to the surface from their planar domain via the use of polar geodesic maps and parallel transport. This allows precise placement of sink,

source, vortex, and saddle singularities, resulting in a superposed field that generally has additional singularities. Emphasis is then placed on reducing these singularities through careful vector field topology analysis and harmonic smoothing over appropriately defined regions.

While our approach also supports direct placement of sinks, sources, vortices, and saddles, we focus less on the relationships between singularities as a way to drive the design, and instead place emphasis on shaping the field through direct interactive control. Specifically, we let the user draw sparse flow curves on the surface to intuitively guide the vector field creation. We then find the global field as the minimizer of the Dirichlet vector field energy subject to the provided constraints. While Zhang and co-workers use *a posteriori* harmonic smoothing to remove singularities, we directly seek harmonic vector fields, thus preventing the appearance of superfluous singularities.

**Foundations of our approach** The desired smoothly varying fields are constructed as minimizers of a quadratic energy with user supplied constraints incorporated through weighted least squares (wLSQ). When applied to 0-forms, *i.e.*, functions given as values at vertices, such approaches are well known in geometric modeling and editing [Botsch and Kobbelt 2004; Sorkine et al. 2005], where they are based on the Laplace and higher-order Laplace operators. In essence the same ideas work for 1-form data using the corresponding Laplace operators, as we demonstrate here for the first time.

We chose a wLSQ approach for robustness reasons: it is easy for a user to specify constraints that "fight" one another or which are outright incompatible. For example, on a genus 0 surface one cannot have a harmonic field with a single source and no matching sink. In such situations the wLSQ approach still produces the "best possible" solution that can be further manipulated through adjustment of the relative weights of the constraints. This is much preferable over a more stringent setup where incompatible constraints would lead to no solution at all. We will show in Section 2.2 that a wLSQ approach to constraints is naturally compatible with the construction of harmonic vector fields since the latter can be understood as LSQ solutions to a global requirement of zero curl and divergence.

**Contributions** We give simple and efficient algorithms for the design of smoothly varying discrete differential 1-forms on unstructured triangle meshes (2-manifold with or without boundary, compact, bounded, and of arbitrary genus). Smoothness is achieved through the use of the 1-form Laplace operator, requiring only the solution of a sparse linear system. We introduce appropriate boundary conditions and in particular a novel free-boundary condition. In a PL reconstruction step the discrete 1-forms are turned into tangent vector fields. User supplied constraints are enforced through wLSQ. Solution of the linear systems relies on pre-factored Cholesky decompositions, with incremental factor updates, to accommodate rapidly changing constraint sets during interactive design sessions. For ease of implementation we use a black-box solver. For design we provide a small but rich set of intuitive "shape" constraints. The technical details (system setup, details of constraint enforcement, boundary conditions, solver, *etc.*) as well as performance metrics are discussed in the context of interactive, anisotropic texture synthesis on arbitrary-topology surface meshes.

## 2 Mathematical Tools

Our approach has its mathematical foundation in Discrete Exterior Calculus [Mercat 2001; Hirani 2003; Desbrun et al. 2006], which defines discrete differential $k$-forms (here $k = 0, 1, 2$) on triangle meshes and expresses relevant operators such as divergence, curl, gradient, and Laplacian, as simple sparse matrices acting on intrinsic (coordinate-free) coefficients "living" on vertices, edges, and triangles. It also gives us discrete versions of such important tools as the Hodge decomposition. Previously this machinery has been used successfully in settings such as surface parameterization (*e.g.*, [Gu and Yau 2003; Gortler et al. 2006; Tong et al. 2006]), physical simulation [Elcott et al. 2007], and vector field decomposition and smoothing [Tong et al. 2003; Polthier and Preuß 2003]. A rigorous treatment of the connection between discrete and continuous settings can be found in the survey of Arnold *et al.* [2006], where finite element techniques are used to establish such essential properties as consistency, stability, and convergence of DEC based approaches.

### 2.1 Setup

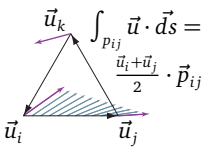
The main ingredients we need for our treatment here are the discrete $k$-forms, the discrete differential $d$ and its $L_2$-adjoint, the co-differential $\delta$, as well as PL interpolators to perform reconstruction on the discrete data.

**The triangle mesh** on which we work is assumed to be of arbitrary topology, orientable, bounded, compact, and 2-manifold (possibly with boundary). We denote its vertex set $V = \{v_i\}$, its edge set $E = \{e_{ij}\}$ and its triangle set $T = \{t_{ijk}\}$ ($1 \le i, j, k \le n = |V|$). Each triangle and edge carry an arbitrary but fixed intrinsic orientation, while vertices always have positive intrinsic orientation. Note that index order matters since $e_{ij}$ has opposite orientation from $e_{ji}$ (and similarly for triangles). Vertices are given positions $P = \{p_v \in \mathbb{R}^3 | v \in V\}$, which define the surface through PL interpolation over each triangle. The intrinsic volumes of edges and triangles will be denoted $|e|$ (length) and $|t|$ (area), and we assume these are all nonzero, *i.e.*, there are no degenerate edges. For a vertex, $|v| = 1$ by definition.

**Discrete $k$-forms** are given as scalars on $k$-cells, which represent measurements

$$c_i = \boldsymbol{\omega}^0(p_i), \quad c_{ij} = \int_{p_{ij}} \boldsymbol{\omega}^1, \quad c_{ijk} = \int_{p_{ijk}} \boldsymbol{\omega}^2.$$

Here $\boldsymbol{\omega}^k$ denotes a $k$-form ($k = 0, 1, 2$) and the scalars the corresponding measurements: $c_i$ is the value of a scalar function ($k = 0$) at position $p_i$ belonging to vertex $v_i$; $c_{ij}$ the line integral of a vector field ($k = 1$) along the segment $p_{ij}$ belonging to $e_{ij}$ (the inset Figure at right illustrates the computation of the line integral of a PL vector field specified with 2-vectors at vertices); and $c_{ijk}$ an area integral of a density ($k = 2$) over $p_{ijk}$ belonging to $t_{ijk}$. (We reserve bold symbols for continuously defined objects while discrete objects will be typeset non-bold.) Note that coefficients change sign when the orientation of their respective integration cells changes ($k = 1, 2$). We treat these coefficients as arrays $c_v$, $c_e$, and $c_t$ using an arbitrary but fixed indexing for the vertices ($v = 1, \ldots, |V|$), edges ($e = 1, \ldots, |E|$) and triangles ($t = 1, \ldots, |T|$).




**The discrete differential** $d$, which maps $k$-forms to $(k+1)$-forms, is given by the transpose of the signed incidence matrices of the triangle mesh: $d_0 = (\partial^1)^T$ maps 0-forms (coefficients at vertices) to 1-forms (coefficients at edges), while $d_1 = (\partial^2)^T$ maps 1-forms (coefficients at edges) to 2-forms (coefficients at triangles). Here $(\partial^2)_{et} = \pm 1$ if edge $e$ is incident on triangle $t$ and their intrinsic orientations agree/disagree and zero otherwise (and correspondingly for $\partial^1$). In standard vector calculus $d_0 \equiv \nabla$ and $d_1 \equiv \nabla\times$ and the fact that the boundary of a boundary is

empty results in $d_1 d_0 = 0$, which in turn corresponds to the vector calculus fact that $\nabla \times \nabla = 0$. (To simplify notation we will generally drop the subscript on $d$ since the type of $d$, *i.e.*, $d_0$ or $d_1$, follows from its argument.)

We also need inner products for discrete forms $c_v$, $c_e$, resp. $c_t$, which correspond to the $L_2$ inner products of continuous forms $\boldsymbol{\omega}^0$, $\boldsymbol{\omega}^1$, resp. $\boldsymbol{\omega}^2$. For our purposes the diagonal matrices

$$(\star_0)_{vv} = |v^*|/|v|, \quad (\star_1)_{ee} = |e^*|/|e|, \quad (\star_2)_{tt} = |t^*|/|t|,$$

are sufficient (the superscript $^*$ denotes the Voronoi dual of a given cell). These so called *diagonal Hodge-star* matrices may be regarded as lumped mass matrices [Bossavit and Kettunen 1999] and are nothing more than ratios of dual to primal intrinsic volumes. (We will use the symbol $\star$ to denote either $\star_0$, $\star_1$, or $\star_2$ depending on the type of its argument.)

We can now consider the $L_2$-adjoint of the exterior derivative $\boldsymbol{d}$. This *co-differential* $\boldsymbol{\delta}$, maps $(k+1)$-forms to $k$-forms and is defined through

$$\langle \boldsymbol{d}\boldsymbol{\omega}^k, \xi^{k+1} \rangle = \langle \boldsymbol{\omega}^k, \boldsymbol{\delta}\xi^{k+1} \rangle.$$

The corresponding discrete operators are $\delta_2 = \star_1^{-1} d_1^T \star_2$, the adjoint of $d_1$, and $\delta_1 = \star_0^{-1} d_0^T \star_1$, the adjoint of $d_0$ (and generically $\delta = \star^{-1} d^T \star$). In the language of vector calculus we have $\delta_1 \equiv \nabla\cdot$ and $\delta_2 \equiv \nabla$.

### 2.2 Harmonic Vector Fields

A vector field is harmonic iff it is both curl- and divergence-free. In the language of 1-forms this corresponds to requiring

$$\boldsymbol{d}\boldsymbol{\omega}^1 = 0 \quad \text{and} \quad \boldsymbol{\delta}\boldsymbol{\omega}^1 = 0.$$

(We ignore boundary issues for now and postpone their discussion to Section 4.1.) To incorporate this requirement into our wLSQ setup we consider the bilinear form

$$\boldsymbol{E}(\boldsymbol{\omega}^1, \xi^1) = \langle \boldsymbol{d}\boldsymbol{\omega}^1, \boldsymbol{d}\xi^1 \rangle + \langle \boldsymbol{\delta}\boldsymbol{\omega}^1, \boldsymbol{\delta}\xi^1 \rangle.$$

A harmonic 1-form $\boldsymbol{\omega}^1$ then satisfies $\boldsymbol{E}(\boldsymbol{\omega}^1, \boldsymbol{\omega}^1) = 0$. The connection with the Laplace operator, $\Delta = \boldsymbol{\delta}\boldsymbol{d} + \boldsymbol{d}\boldsymbol{\delta}$, is established through its weak formulation: a form $\boldsymbol{\omega}$ has vanishing Laplacian if its inner product with any of a set of test forms $\xi$ vanishes

$$0 = \langle \Delta\boldsymbol{\omega}, \xi \rangle = \langle \boldsymbol{\delta}\boldsymbol{d}\boldsymbol{\omega}, \xi \rangle + \langle \boldsymbol{d}\boldsymbol{\delta}\boldsymbol{\omega}, \xi \rangle = \langle \boldsymbol{d}\boldsymbol{\omega}, \boldsymbol{d}\xi \rangle + \langle \boldsymbol{\delta}\boldsymbol{\omega}, \boldsymbol{\delta}\xi \rangle.$$

In other words, $\boldsymbol{E}$ describes the bilinear form appearing in the weak formulation of the Laplacian: $\boldsymbol{E}(\boldsymbol{\omega}^1, \boldsymbol{\omega}^1)$ measures the Dirichlet energy of the underlying vector field.

Translating the expression for $\boldsymbol{E}$ into the setting of discrete forms we arrive at

$$E(c_e, c_e) = c_e^T M c_e = c_e^T \left( d_1^T \star_2 d_1 + \star_1 d_0 \star_0^{-1} d_0^T \star_1 \right) c_e, \qquad (1)$$

and $M$ is the discrete version of the 1-form Laplace operator. For later use we note that $M = M^{\nabla\times} + M^{\nabla\cdot}$ consists of a first summand encoding the squared curl magnitude, while the second summand encodes the squared divergence magnitude.

**Assembly of $M$** Practically speaking $M$ is a straightforward assembly of diagonal matrices and signed incidence matrices. More concretely, for a given edge $e_{ij}$ with associated 1-form coefficient $c_{ij}$ a row of $M$ reads as (see also Figure 2)

$$M_{e_{ij}} = \frac{c_{ij} + c_{jk} + c_{ki}}{|t_{ijk}|} - \frac{c_{ij} + c_{jl} + c_{li}}{|t_{ijl}|} + \frac{|e_{ij}^*|}{|e_{ij}|} \left( \frac{|v_j|}{|v_j^*|} \sum_{e_{jl} \ni v_j} \frac{|e_{jl}^*|}{|e_{jl}|} c_{jl} - \frac{|v_i|}{|v_i^*|} \sum_{e_{ik} \ni v_i} \frac{|e_{ik}^*|}{|e_{ik}|} c_{ik} \right). \qquad (2)$$

$M$ has an average of $\approx 11$ nonzero entries per row.

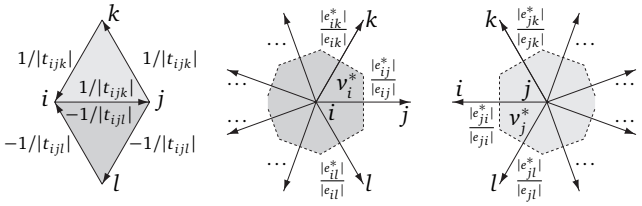



Figure 2: *The 1-form Laplace stencil, Eq.* (2)*, for a given $e_{ij}$ depends on data at the two incident triangles (left) as well as the edges incident on $v_i$ and $v_j$ (middle and right).*

### 2.3 Hodge Decomposition

Given proper boundary conditions, any 1-form can be written uniquely as an orthogonal sum

$$\boldsymbol{\omega}^1 = \boldsymbol{d}\boldsymbol{\omega}^0 + \boldsymbol{\delta}\boldsymbol{\omega}^2 + \boldsymbol{h}$$

where $\boldsymbol{\omega}^0$ and $\boldsymbol{\omega}^2$ are scalar potentials (0- and 2-forms respectively), and $\boldsymbol{h}$ a harmonic 1-form. The latter is nontrivial only in the case of surfaces with genus $g > 0$, for which the space of such harmonic 1-forms is $2g$ dimensional. The corresponding statement for *discrete* 1-forms holds as well

$$c_e = dc_v + \delta c_t + h. \qquad (3)$$

Such decompositions have been used to manipulate individual components of given vector fields [Polthier and Preuß 2003; Tong et al. 2003]. In the context of design we will use this decomposition to give us access to the curl-free ($dc_v$) and divergence-free ($\delta c_t$) parts of a vector field (see Section 3.1).

### 2.4 Reconstruction

Discrete $k$-forms can be PL interpolated with the aid of Whitney elements [Whitney 1957]. For 0-forms these are the standard PL hat functions $\phi^v = \{\phi_i | v_i \in V\}$. Appropriate 1-form interpolators $\phi^e = \{\phi_{ij} | e_{ij} \in E\}$ follow as

$$\phi_{ij} = \phi_i \boldsymbol{d}\phi_j - \phi_j \boldsymbol{d}\phi_i.$$

The $\phi_{ij}$ are supported on the triangle(s) incident to the given edge and vary linearly within them. Their projection along edges is continuous when crossing an edge while their line integral is 1 along $e_{ij}$ and 0 along all other edges, giving us linear interpolators for discrete 1-form data. For data at triangles (discrete 2-forms) the corresponding Whitney interpolators are the constant functions $\phi^t = \{\phi_{ijk} | t_{ijk} \in T\}$ supported on individual triangles (Figure 3).

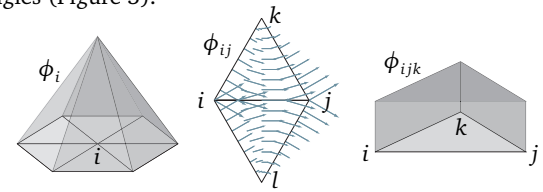



Figure 3: *The bases for 0- and 2-forms are standard PL hat functions resp. constant functions. 1-form bases may be visualized as vector fields once a metric is supplied (Euclidean in this case).*

Given the support of $\phi_{ij}$ the interpolated 1-form $\boldsymbol{\omega}^1$ within a single triangle is determined by the three incident edge coefficients

$$\boldsymbol{\omega}^1|_{t_{ijk}} = c_{ij}\phi_{ij} + c_{jk}\phi_{jk} + c_{ki}\phi_{ki}.$$

Using a metric (the standard Euclidean metric in all our examples) one can turn the 1-form $\boldsymbol{\omega}^1$ into a vector field $\vec{u} \in \mathbb{R}^2$. With barycentric coordinates $\alpha_i$, $\alpha_j$, $\alpha_k$ (associated to vertices $i$, $j$, and $k$ resp.) we get

$$2\,|t_{ijk}|\,\vec{u}(\alpha_i, \alpha_j, \alpha_k) = (c_{ki}\alpha_k - c_{ij}\alpha_j)\vec{p}_{jk}^{\perp} + (c_{ij}\alpha_i - c_{jk}\alpha_k)\vec{p}_{ki}^{\perp} + (c_{jk}\alpha_j - c_{ki}\alpha_i)\vec{p}_{ij}^{\perp} \qquad (4)$$

where $\vec{p}_{ij} = p_j - p_i$ represents the edge as a vector; $\perp$ indicates a $\pi/2$ rotation in the plane of $p_{ijk}$; and we used $\boldsymbol{d}\phi_i \equiv \nabla\phi_i = \vec{p}^{\perp}_{jk}/(2|t_{ijk}|)$. This can be understood as an equation giving a 3-vector. However, since the vector lives in the plane of $p_{ijk}$ it can equally well be treated as a 2-vector equation when expressing it in a tangent plane frame. From now on we will no longer distinguish between these two interpretations unless stated otherwise.

In some applications vector fields computed on the surface are used in the texture domain. In that case tangent vectors can be pushed forward through the Jacobian of the mapping from the surface to the parametric domain. However, there is an even simpler approach. It suffices to replace all spatial quantities in Eq. (4) with their corresponding texture domain counterparts

$$2\,|\tilde{t}_{ijk}|\,\tilde{\vec{u}}(\alpha_i,\alpha_j,\alpha_k) = (c_{ki}\alpha_k - c_{ij}\alpha_j)\tilde{\vec{p}}^{\perp}_{jk} + (c_{ij}\alpha_i - c_{jk}\alpha_k)\tilde{\vec{p}}^{\perp}_{ki} + (c_{jk}\alpha_j - c_{ki}\alpha_i)\tilde{\vec{p}}^{\perp}_{ij}$$

where we used $\tilde{}$ to indicate that these quantities are to be taken in the texture domain rather than on the mesh in 3-space. This automatically accounts for the effects of the Jacobian.

**Continuity** The vector fields reconstructed with Whitney 1-forms are piecewise linear, but in general not continuous along edges or at vertices. Traditional approaches to vector fields on meshes often use 2 coefficients $(u^1_i, u^2_i)$ per vertex and define the vector field coordinate-wise by interpolating these linearly over each triangle. As pointed out by Zhang *et al.* [2006], this does not result in continuous tangent vector fields either when a vertex has nonzero Gaussian curvature. Zhang and co-workers addressed this issue by using geodesic polar maps and nonlinear interpolants on triangle interiors to maintain continuity. Instead we opt to stay in a linear framework, taking advantage of the fact that the edge coefficients arise from the solution of a discrete Poisson problem for which we, in effect, use a discontinuous Galerkin method [Arnold et al. 2000]. This connection with finite element theory ensures that the Whitney-interpolated fields approximate the underlying smooth solution arbitrarily well so long as the mesh is suitably fine. More generally, Dodziuk *et al.* [1976] show that Whitney 1-forms can approximate any smooth vector field arbitrarily well in the $L_\infty$ norm (see also [Hildebrandt et al. 2006; Arnold et al. 2006]). For very coarse triangulations, however, it is likely better to use either the nonlinear interpolants of Zhang *et al.* or smoother DEC bases [Wang et al. 2006]. The latter are always continuous and have the advantage that their discrete differential is *still* realized with the adjoint of the incidence matrices, though a discrete Hodge-star with higher-order accuracy (and necessarily larger support) is required to avoid a drop in approximation order.

For display purposes it may be desirable to post-process the computed solution the same way one computes vertex normals from (discontinuous) incident triangle normals: using Eq. (4) we can evaluate the vector field at $v_i$ in each triangle incident to $v_i$ and average based on areas

$$\vec{u}_i = |v^*_i|^{-1} \sum_{t_{ijk}\ni i} \frac{|v^*_i \cap t_{ijk}|}{2|t_{ijk}|}(c_{ij}\vec{p}^{\perp}_{ki} - c_{ki}\vec{p}^{\perp}_{ij}). \tag{5}$$

This equation is well defined only if the tangent spaces of the triangles incident to $v_i$ can be identified, as is the case when $v_i$ has zero Gaussian curvature. For a non-flat surface the necessary identification of tangent spaces can be performed with the help of a parameterization. If an application calls for tangent vectors at vertices expressed in $\mathbb{R}^3$ one may still use Eq. (5) by treating the $\vec{p}_{ij}$ as 3-vectors. The unavoidable errors due to curvature may remain acceptable if the neighborhood of $v_i$ is sufficiently flat.

## 3 Constrained Design of Vector Fields

In this section we discuss the details of the different constraints we have found useful during design, and the overall setup of the equations we need to solve. In all cases we seek a field that minimizes a quadratic energy while meeting the given constraints in a wLSQ sense.

Consider first an arbitrary discrete 1-form $c_e = dc_v + \delta c_t + h$ (Eq. (3)). Its curl is $dc_e = d\delta c_t =: r_t$ while its divergence is $\delta c_e = \delta dc_v =: s_v$ since $h$ is both curl- and divergence-free. Allowing arbitrary linear functionals $Zc_e$—which will account for constraints as discussed below—we arrive at a "stack" of matrices acting on the unknown vector of edge coefficients $c_e$

$$\begin{pmatrix} d \\ \delta \\ Z \end{pmatrix} c_e = \begin{pmatrix} r_t \\ s_v \\ c_z \end{pmatrix}$$

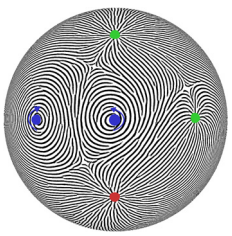
where the right hand side encodes the desired curl (at triangles), divergence (at vertices) and the inhomogeneous part of the linear functionals ($c_z$). The solution with minimum residual norm satisfies the associated normal equations

$$(\star\delta\ \ \star d\ \ Z^T W)\begin{pmatrix} d \\ \delta \\ Z \end{pmatrix} c_e = (\star\delta\ \ \star d\ \ Z^T W)\begin{pmatrix} r_t \\ s_v \\ c_z \end{pmatrix} \iff$$
$$(M + Z^T W Z)c_e = \star\delta r_t + \star d s_v + Z^T W c_z. \tag{6}$$

where $W$ is a diagonal weight matrix containing non-negative weights for the constraints, *i.e.*, how much each constraint should be enforced relatively to others.

### 3.1 Control of Sources, Sinks and Vortices

We first examine the setting of no constraints (absence of any $Z$). In that case the right hand side can encode desired locations for sources and sinks through $s_v$ and similarly for vortices through $r_t$ as follows. Let $s_v$ be a vector of vertex coefficients with positive entries for selected sink vertices and negative entries for selected source vertices with their magnitude representing strengths and satisfying a weighted (by Voronoi area) zero mean condition, *i.e.*, $\bar{s}_v = A^{-1}\sum_i s_i|v^*_i| = 0$, where $A$ is the total area of the mesh. ("Anything that is produced must be consumed.") In practice, if the user specifies a vector $s_v$ with $\bar{s}_v \neq 0$ we perturb it as $\hat{s}_i = s_i - \bar{s}_v$. This deals with "impossible" right hand sides and ensures that the computed solution $c_e$ is LSQ optimal: the perturbation from $s_v$ to $\hat{s}_v$ minimizes $\langle \delta c_e - s_v, \delta c_e - s_v\rangle$. For vortices we proceed similarly. Let $r_t$ be a vector of (selected) triangle coefficients with positive entries for vortices whose orientation coincides with the triangle orientation and negative entries for opposing orientation (with $r_{ijk}/|t_{ijk}|$ indicating the strength). This time we enforce an unweighted zero mean for $r_t$, if not already satisfied, through a perturbation $\hat{r}_{ijk} = r_{ijk} - \bar{r}_t|t_{ijk}|$ for $\bar{r}_t = A^{-1}\sum_{ijk} r_{ijk}$, minimizing $\langle dc_e - r_t, dc_e - r_t\rangle$. If the user supplied $s_v$ and $r_t$ satisfy the zero mean condition up front then the resulting vector field has *only* those sources, sinks and vortices. (This follows from the uniqueness of the underlying 0-form and 2-form Poisson problem.) An example demonstrates this for the sphere where two vortices (blue marks), one source (red mark) and two sinks (green marks) were set. For the horse (genus 0), with only a single source placed by the user, adjustment of $s_v$ leads to a solution with additional singularities at the hoofs (Figure 4 shows a comparison of single source and matching source/sink behavior).



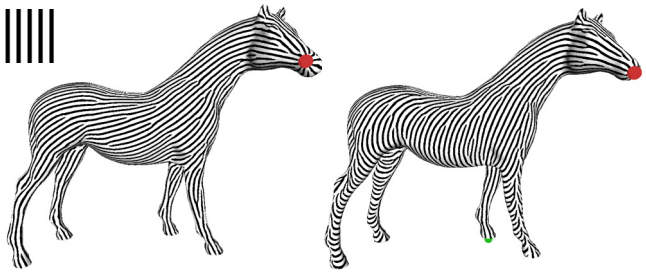


Figure 4: *Example of user specified single source (left) and matching source/sink (right) on the horse. The LSQ solution in the case of only a single source places sinks at the hoofs. The inset shows the texture synthesis exemplar used for all visualizations in this Section.*

Vortices, sources and sinks are important features in a vector field and can produce an overall field with little user supplied data, but they do not provide enough detail control. For this we need direct constraints on the edges.

## 4 Edge Constraints

Consider now the case of using constraints $Z$. If both $s_v$ and $r_t$ are zero then we are seeking a field which is harmonic, *i.e.*, free of curl and divergence, while approximating the given $Z$ constraints. What constraints are useful?

In the least squares setup constraining an individual edge value amounts to the trivial row

$$Z c_e := c_{ij} = \text{chosen value.} \tag{7}$$

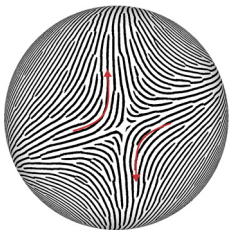
Fixing an edge coefficient in this manner leads to a vector field whose line integral along the edge matches the given value, but does not imply that the vector field is parallel to the edge. Stronger control though can be imposed by using several edge constraints in concert (Figure 5).

**Vector constraints** At times it is desirable to specify a particular vector at a particular location as a constraint. Suppose we want to specify a vector $\vec{u}$ on some triangle $t_{ijk}$. Taking advantage of $dc_e = c_{ij} + c_{jk} + c_{ki} = 0$ on $t_{ijk}$, *i.e.*, the zero curl condition, and using Eq. (4) we find the desired constraints as

$$Z_1 c_e := c_{ij} = \vec{u} \cdot \vec{p}_{ij}, \quad Z_2 c_e := c_{jk} = \vec{u} \cdot \vec{p}_{jk}, \quad Z_3 c_e := c_{ki} = \vec{u} \cdot \vec{p}_{ki},$$

where the inner products are taken in the plane of $p_{ijk}$.

**Strokes** The most important means to control the appearance of a vector field are sequences of edge constraints given through a curve drawn on the mesh, signifying the vector field should follow it. Let $\vec{t}$ be the velocity vector to the curve as it crosses edge $e_{ij}$, then



$$Z c_e := c_{ij} = \vec{t} \cdot \vec{p}_{ij}$$

gives the constraint for each edge crossed. In this manner, *e.g.*, it is easy to place a saddle (inset image) or perform editing operations such as on the bunny (Figure 6).

### 4.1 Boundaries

To properly deal with boundaries we must modify $M$ (Eq.(1)) according to the type of boundary condition we wish to enforce. We support two different types: *free* and *arbitrary (fixed) angle*. For free boundaries neither fluxes across boundary edges, nor line integrals along them are constrained. The angle boundary conditions enforce a zero line integral in a freely chosen direction relative to each boundary edge. Special cases of this setting include *tangential* boundary conditions, which force the vector field to have zero flux across the boundary edges, and *normal* boundary conditions, which force the field to meet the boundary edges orthogonally. Examining $M$ we see that $M^{\nabla\times}$ needs no modification since it is defined on a per triangle basis. On the other hand, $M^{\nabla\cdot}$, which encodes the square of the divergence at every vertex, must be changed for boundary vertices.

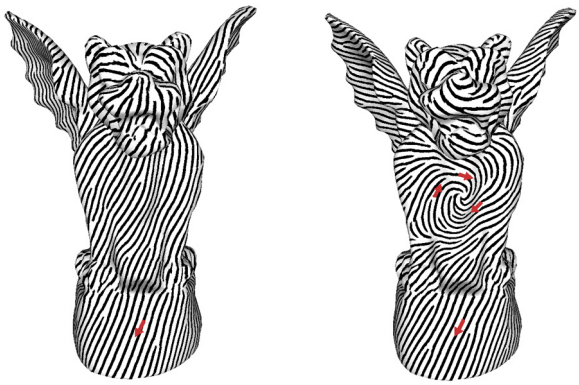


Figure 5: *A single edge constraint induces a global field (left). Adding 3 closely spaced edge constraints creates a spiral.*

To see what is needed for the different boundary conditions we first consider the general relationship between edge coefficients on boundary triangles and the resulting flux across the boundary edge. Given a boundary edge $e_{ij}$ with incident triangle $t_{ijk}$ the line integral along the boundary edge is simply $c_{ij}$. The flux across the boundary edge can be found by exploiting the curl-free condition ($c_{ij}+c_{jk}+c_{ki}=0$) on $t_{ijk}$ to eliminate $c_{ij}$ in Eq. (4). Integrating the inner product against $\nabla\phi_k/|\nabla\phi_k|$ along $e_{ij}$ yields the net flux

$$f_{ij} = c_{jk}\cot\theta^i_{jk} - c_{ki}\cot\theta^j_{ki},$$

where $\theta^i_{jk}$ indicates the angle at $v_i$ across from $e_{jk}$.

Given the line integral $c_{ij}$ along $e_{ij}$ and the flux $f_{ij}$ across $e_{ij}$ the line integral in *any* direction is completely determined

$$c_{\beta_{ij}} = c_{ij}\cos\beta_{ij} + f_{ij}\sin\beta_{ij}$$

where $\beta_{ij}$ is the angle through which $p_{ij}$ is rotated in the plane of $p_{ijk}$. Setting this line integral to zero for a given angle forces the vector field to be orthogonal to the direction of $p_{ij}$ rotated through $\beta_{ij}$ without constraining the magnitude in the orthogonal direction. Special cases of interest are normal boundary conditions ($\beta_{ij}=0$) and tangential boundary conditions ($\beta_{ij}=\pi/2$).

For free boundary conditions we do not want to constrain the field in any direction but rather have it "choose" the direction. To get the necessary linear equations at the boundary we consider the computation of divergence at boundary vertices directly. At

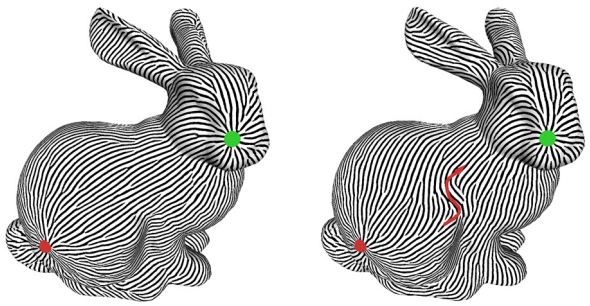
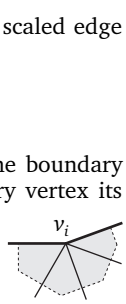


Figure 6: *A single source/sink pair produced the field on the left which is then reshaped with a curve constraint drawn on the body.*

an interior vertex $v_i$ the divergence is a sum over scaled edge coefficients incident to $v_i$

$$(d_0^T \star_1 c_e)_i = \sum_{e_{im}} \frac{|e_{im}^*|}{|e_{im}|} c_{im},$$

which is equivalent to the sum of fluxes across the boundary of the associated Voronoi cell. If $v_i$ is a boundary vertex its Voronoi cell is bounded by edges dual to the incident edges, just as in the sum above, *and* 1/2 of the two incident boundary edges (see inset Figure). The flux across these two incident boundary edges (say $e_{ij}$ and $e_{il}$) must be accounted for and makes an additional contribution of $1/2(f_{ij} + f_{il})$ to the above sum.



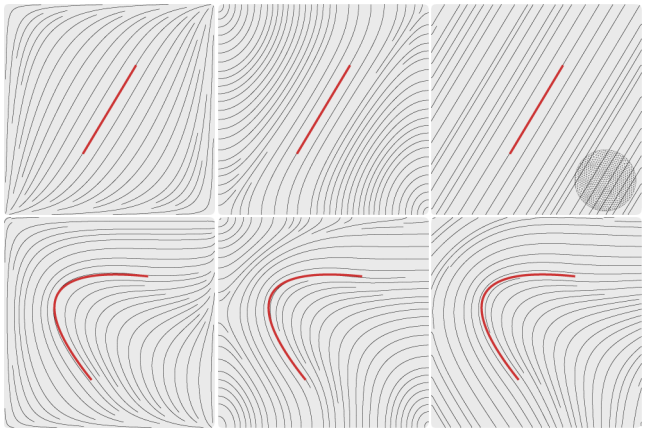



Figure 7: *Comparison of tangential, normal, and free boundary conditions with two different strokes as constraints, rendered with the method of [Mebarki et al. 2005] (inset shows mesh resolution).*

### 4.2 Assembly of Boundary Modified Laplacian

As indicated earlier, $M^{\nabla\times}$ is assembled without modification. $M^{\nabla\cdot}$ is assembled as the product of $d_0^T \star_1$ with its transpose using an intervening $\star_0^{-1}$ and must be modified.

For free boundaries all vertices participate in $d_0^T \star_1$, but each boundary vertex "picks up" the extra flux contributions from its incident boundary edges

$$(d_0^T \star_1)_{v_i e_{jk}} \mathrel{+}= \tfrac{1}{2}\cot\theta_{jk}^i \quad (d_0^T \star_1)_{v_i e_{ki}} \mathrel{-}= \tfrac{1}{2}\cot\theta_{ki}^j$$
$$(d_0^T \star_1)_{v_j e_{jk}} \mathrel{+}= \tfrac{1}{2}\cot\theta_{jk}^i \quad (d_0^T \star_1)_{v_j e_{ki}} \mathrel{-}= \tfrac{1}{2}\cot\theta_{ki}^j$$

*before* multiplying with $\star_0^{-1}$ and the transpose of the modified $d_0^T \star_1$ (and addition of $M^{\nabla\times}$) to yield $M_{\text{free}}^b$.

In the case of fixed angle conditions only interior vertices participate in $M^{\nabla\cdot}$ before addition to $M^{\nabla\times}$. Only afterward do we add the boundary edge conditions

$$0 = c_{ij}\cos\beta_{ij} + (c_{jk}\cot\theta_{jk}^i - c_{ki}\cot\theta_{ki}^j)\sin\beta_{ij}$$

as $Z_{\beta_e} c_e = 0$ type constraints, one for each boundary edge, to yield $M_\beta^b = M + Z_{\beta_e}^T W Z_{\beta_e}$. (For the special case of $\beta = 0$, *i.e.*, $c_{ij} = 0$ for boundary edges, one may instead simply drop all boundary edge rows and columns.)

Figure 7 demonstrates the effect of $\beta = \pi/2$ (tangential), $\beta = 0$ (normal), and free boundary conditions on two prototypical fields specified with a curve constraint. Shown here are the integral curves of the underlying vector fields. Figure 8 shows the effect of different boundary conditions on the (damaged) neck of the Planck dataset. In the rightmost image we exploited the ability to set $\beta_e$ on a per edge basis, choosing the desired angle for each boundary edge to match a global down direction.

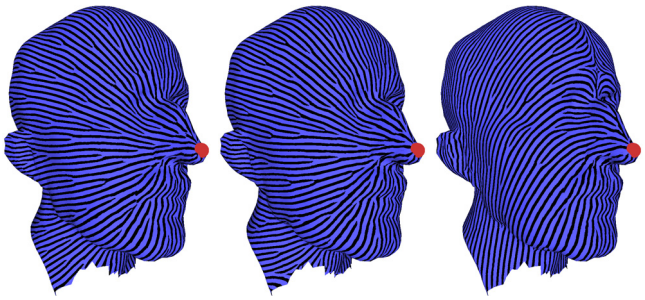


Figure 8: *The Planck dataset with a single source and varying boundary conditions: free (left), normal (middle), and fixed angle (right). In the latter case the desired angle was set on a per edge basis ensuring that the field flows off the boundary in the global down direction.*

## 5 Numerical Implementation

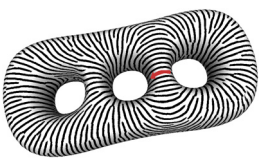
In this section we document some of our numerical, algorithmic, and implementation choices, and include details on the "translation" from mathematical statement to concrete computation.

### 5.1 Numerical Linear Algebra

The linear system of Eq. (6) is symmetric positive (semi-)definite and sparse. In the case of 0-form Laplace (as well as bi-Laplace) systems, approaches based on Cholesky factorization have been shown to exhibit excellent performance [Botsch et al. 2005], and we chose such a (black-box) solver as well (TAUCS [Toledo 2003] with a modification supplied by Olga Sorkine).

One could factor the entire left hand side of Eq. (6) as one monolithic matrix. Changing right hand sides then requires only cheap backward/forward substitution. Unfortunately only the $M$ matrix is constant for a given mesh while the constraint part $(Z^T W Z)$ changes frequently due to user interaction. In general, the number of constraints $|Z| =: k$ will be far smaller than $|E|$, the dimension of $M$. This favors a Cholesky pre-factorization strategy for $M = CC^T$ coupled with incremental modification of the Cholesky factor $C$ [Davis and Hager 1999; Davis and Hager 2001]. For general $Z_i$ this can be quite involved and may even change the sparsity structure of the Cholesky factor. We are in the fortunate setting that all our constraints are very simple: each $Z_i$ is a vector (of length $|E|$) that is zero in every slot but one. Updating a Cholesky factorization when only a diagonal entry changes through addition of a (positive) weight $w_i$ is particularly efficient and numerically stable [Sorkine et al. 2005, Section III].

To summarize, we compute the Cholesky factorization $CC^T = M$ (where $M$ may be boundary modified) upfront. At runtime changing right hand sides require only backward/forward substitution while changing $Z$ (add/remove/change weight) constraints require an incremental update of $C$.



**Rank deficiency of $M$** There is one final issue to deal with: $M$ is only positive *(semi-)*definite, as it has a $2g$ dimensional kernel for a genus $g$ surface. We address this issue by fixing very small values ($\approx 10^{-9}$ relative magnitude) for $2g$ random edges. This "ties down" the kernel with negligible impact on the computed solutions for edges in general position. (The probability of failure vanishes in the limit of increasing mesh size and we have not observed any failures.) No less, fields in the kernel of $M$ can still be recovered by specifying an appropriate curve constraint, even if short, as illustrated with the 3-holed torus, showing a harmonic vector field induced by the red curve constraint.

### 5.2 Operator Assembly

While it is possible to assemble the $M$ matrix directly (Eq. (2)) it is needlessly complex (and prone to implementation errors). Instead we assemble $M$ as a sum of products of the much simpler constituent matrices (Eq. (1)) along the lines of the approach described in [Elcott and Schröder 2006] and only then "compile" them into compressed column storage format for use by TAUCS. This only requires knowledge of the incidence matrices as well as the diagonal Hodge-star matrices.

**Hodge-star** matrices, since they are diagonal, are represented as vectors. For $\star_2$ the corresponding vector of length $|T|$ simply contains the inverses of the triangle areas, while $\star_0$ (length $|V|$) contains the areas of the Voronoi cells of the vertices. These cells are defined by sequences of dual vertices at the triangle circumcenters. Note that these are not guaranteed to be interior to their triangles (even in an intrinsically Delaunay [Bobenko and Springborn n. d.] mesh) and may even "fold back" on themselves. Nonetheless, there is a simple approach to compute these areas correctly by iterating over all triangles and accumulating signed elementary areas into each vertex [Meyer et al. 2002, Section 3.3]. Consider $p_{ijk}$ together with its circumcentric dual vertex $p^*_{ijk}$. Decompose $p_{ijk}$ into 6 smaller triangles each formed by a corner (*e.g.*, $p_i$), the midpoint of an adjacent edge (*i.e.*, $(p_i + p_j)/2$), and the circumcenter $p^*_{ijk}$. The signed height of $p^*_{ijk}$ above $p_{ij}$ is $h^k_{ij} = |e_{ij}|/2 \cot \theta^k_{ij}$, where the sign depends on whether $p^*_{ijk}$ is to the triangle interior side of $p_{ij}$ (+) or not (−). The signed area of such a sub-triangle is $|e_{ij}|^2/4 \cot \theta^k_{ij}$ (and correspondingly for the other 5 sub-triangles induced by $t_{ijk}$). A simple iteration over all triangles then computes the correct Voronoi areas. To summarize, knowledge of the cotangents of each triangle corner angle together with the lengths of all edges is sufficient to compute the Voronoi areas.

Figure 9: *Gallery of surface texture synthesis results based on vector fields specified with a variety of constraints, demonstrating that even just a few constraints can quickly build overall fields with pleasing flows.*

Similarly, we can reduce the computation of $\star_1$ to knowing these cotangents since $|e^*_{ij}|/|e_{ij}| = 1/2(\cot \theta^k_{ij} + \cot \theta^l_{ij})$ for incident triangles $t_{ijk}$ (and $t_{ijl}$). Note that this expression represents the correct lumped mass matrix and preserves the positive (semi-)definiteness of $M$, even if there are negative coefficients in $\star_1$. (The only drawback of negative cotan weights is the increased condition number of the resulting matrices [Fisher et al. 2006].)

## 6 Results

We have implemented all the algorithms described above and used the resulting fields to drive interactive texture synthesis, employing the method of Lefebvre and Hoppe [2006]. In this application only a direction field is needed so all vectors were normalized before being passed on to the texture synthesis code. (See Section 6.3 for a scenario where magnitudes are taken into account.)

### 6.1 Performance

The following timings were taken on an AMD Athlon XP at 2.08GHz:

| Model | $\|E\|$ | $\|\neq 0\|$/ row $M$ | $\|\neq 0\|$/ row $C$ | Factor time (s) | Solve time (s) |
|---|---|---|---|---|---|
| Garg. | 15000 | 11.2 | 56.3 | 0.13 | 0.02 |
| Hygea | 24798 | 11.8 | 59.3 | 0.25 | 0.02 |
| David | 74985 | 11.2 | 64.6 | 0.89 | 0.06 |
| Planck | 76245 | 11.4 | 68.2 | 1.11 | 0.06 |
| Beetle | 88771 | 11.0 | 60.8 | 1.02 | 0.06 |
| Bunny | 104288 | 11.1 | 71.3 | 1.69 | 0.09 |
| Feline | 149598 | 11.5 | 64.0 | 1.89 | 0.13 |

(Here $C$ denotes the Cholesky factor of $M$.) Cholesky updates with $Z_i$ constraints with a single nonzero entry take $< 3$ ms.

From these timings we can see that the solve time is a linear function of the original model complexity with a constant depending on the sparsity of the Cholesky factors. The number of nonzero entries in $C$ is approximately 5 to 6 times larger than the number of nonzero entries in $M$. The pre-processing cost (factorization), while significant with respect to the other parts of the algorithm, is quite small even for meshes as large as Feline. Of the online parts of the algorithm, individual Cholesky updates are essentially negligible while solve times range from tens of ms to over 100 ms, and these times dominate the texture synthesis times ($20 - 40$ ms each). Given that texture synthesis is performed on the GPU, while our solver runs on the CPU, the user perceived time is dominated by the back substitution timings.

### 6.2 Discussion of Examples

Figures 1 and 9 show a gallery of textured models created in all cases by using only a few constraints. The VW model texture was controlled through placement of two vortices (top of hood and cabin) and a curve on the side. The boundary conditions were of the free type, allowing the vector field to approach the boundary, consisting of multiple connected components, in a natural way. In the case of the horse a source was placed at the nose (compare with the figure in Section 3.1) and more precise control over the flow on the legs exerted through two additional curve constraints. For the Planck head two vortices (on the eyes) and a curve constraint were placed while the boundaries used the

free boundary conditions. The initial field for the Gargoyle was created through a sink on each of the wing tips and a source on the nose. The upward flowing field on the torso was then turned by approximately $\pi/2$ through a additional curve constraint. In the Bunny example the vector field was constructed using just three curves placed on the side of the torso. Similarly the fields on the David and Hygea dataset used only curve constraints.

The quality of solutions depends little on the constraint weights $w_i$ (diagonal elements of the matrix $W$). However, finer design control can be obtained by changing the weights of our least-squares solution as they directly correspond to how well a given constraint is enforced. This behavior is indeed verified numerically; for the Bunny example we computed the maximum deviation along the curves when using given weights: $w_i = 1600$ leads to 9.63 % maximum deviation, while at $w_i = 10,000$ this reduces to 0.37 % and for $w_i = 1,000,000$ to 0.0018 %. The results for $w_i > 10,000$ are visually indistinguishable. (We note however that the fields are harmonic, *i.e.*, curl- and divergence-free *only* in the limit of increasing weights on the constraints.)

In these examples no attention was paid by the designer to questions of global topology or index theory and consequently the wLSQ framework was essential in producing any solutions at all (see the discussion in Section 3.1). Of course in the end the mathematical constraints imposed on smooth vector fields cannot be cheated ("you can't comb a hairy ball") and consequently singularities (vortices, sinks, sources, saddles) do appear at potentially undesirable locations, but it is then an easy matter to manipulate the field further with additional constraints to reshape the flow or place appropriate singularities in regions that are less objectionable. We have found that a fluid flow analogy—sources eject a certain amount of mass per unit time, *etc.*—is often helpful in understanding why a field forms as it does.

### 6.3 Extensions

For most texture synthesis scenarios, only the local direction of the field matters. We normalized all computed vectors before passing them on to the texture synthesis code, thus ignoring the computed magnitudes. Instead one can use the magnitudes to control the local scale of the texture. When using only the Laplacian the vector magnitudes tend to decay rapidly away from constraints. The corresponding observation for functions has led researchers in geometric modeling, for example, to consider solutions of the bi-Laplacian to provide more control over the change in magnitudes. In the case of vector fields that same philosophy can be applied. The corresponding bi-Laplacian, $\Delta^2 = (\boldsymbol{\delta d})^2 + (\boldsymbol{d\delta})^2$, then corresponds to

$$B = M \star^{-1} M.$$

This matrix has $\approx 44$ nonzero entries per row on average and, in the spirit of splines under tension, one may add a small multiple ($\gamma > 0$) of $B$ to $M$, yielding a modified system

$$(M + \gamma B + Z^T W Z)c_e = \star\delta r_t + \star d s_v + Z^T W c_z$$

Figure 10 demonstrates the effect of $B$ on the computed field magnitudes, while Figure 11 shows the resulting effects during texture synthesis. Manipulating the relative scale of the texture in this manner is a yet largely unexplored territory, but may prove crucial in applications based on flows such as texture advection and crowd control in animation. (Local scale factors for magnitude control could of course also be based on separately computed 0- or 2-form fields.)

The guarantees of topological simplicity coming with the use of the Laplacian alone are not valid anymore once a small multiple of the bi-Laplacian is added, so the value of setting $\gamma > 0$ is likely very application specific. It should also be noted that the sparsity of the overall linear system decreases to that of $B$ and the timings increase correspondingly.

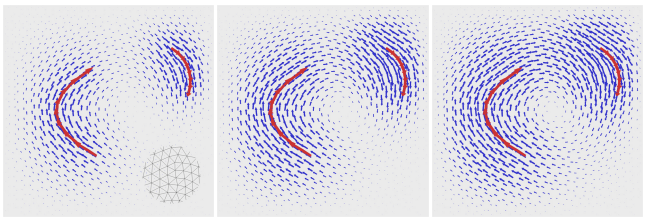


Figure 10: *Hedgehog rendering of a constrained vector field using the Laplacian (left). Adding a small multiple of the bi-Laplacian ($\gamma = 0.01$, middle; $\gamma = 0.05$, right), vector magnitudes decay more gently. (Here free boundaries were used and the inset on the left shows the mesh resolution.)*

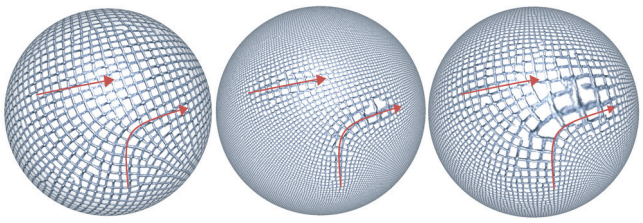


Figure 11: *The local scale of texture synthesis can be controlled through vector magnitudes. On the left the result of texture synthesis using no magnitude information. In the middle the magnitude of the Laplace solution is used for local scale control, while on the right the magnitude resulting from a $\gamma = 0.1$ solution is used.*

| Model | $\lvert \neq 0 \rvert$/ row $B$ | $\lvert \neq 0 \rvert$/ row $C$ | Factor time (s) | Solve time (s) |
|---|---|---|---|---|
| Garg. | 42.4 | 204.0 | 0.80 | 0.03 |
| Hygea | 47.1 | 227.0 | 1.78 | 0.05 |
| David | 42.5 | 241.4 | 6.13 | 0.19 |
| Planck | 44.1 | 268.9 | 8.67 | 0.20 |
| Beetle | 40.6 | 212.9 | 5.48 | 0.19 |
| Bunny | 41.8 | 271.4 | 11.92 | 0.30 |
| Feline | 44.3 | 246.7 | 13.88 | 0.41 |

(Here $C$ denotes the Cholesky factor of $M + \gamma B$ for $\gamma > 0$.) Now an individual Cholesky update takes $5 - 18$ ms, still sufficient for real time update. A subsequent solve though does take a good fraction of a second.

## 7 Conclusion

Methods of Discrete Exterior Calculus have become quite successful in geometry processing because of their algorithmic simplicity and their solid mathematical foundations. Here we put them to use for the design of tangent vector fields by solving for harmonic vector fields subject to user constraints in a wLSQ framework. Our method is straightforward to implement, as it involves only a single degree of freedom per edge, easy-to-assemble sparse matrices, and black-box sparse linear solvers.

Anisotropic texture synthesis is an obvious application in which design of the detailed flow of the principal texture directions on the surface is highly desirable and we used this application to generate our examples using the GPU based method of Lefebvre and Hoppe [2006]. In future work we look forward to applying our method to other vector field applications. Examples include random vector noise generation (along the lines of [Cook and DeRose 2005]), flow based crowd control, and further texturing methods based on advection as well as reaction and diffusion. The latter integrate physical simulation ideas into the discrete

$k$-form setting which appears to be a promising avenue.

**Acknowledgment** This work was supported in part by NSF (CCF-0528101, CCR-0503786 and ITR DMS-0453145), DOE (W-7405-ENG-48/B341492 and DE-FG02-04ER25657), the Caltech Center for Mathematics of Information, the Alexander von Humboldt Stiftung, SUN Microsystems, Pixar, nVidia, and Autodesk. Some models courtesy Stanford University, Cyberware, and Leif Kobbelt. Special thanks to Ke Wang, Cici Koenig, Max Wardetzky, Yiying Tong, and the anonymous reviewers.

## References

Arnold, D. N., Brezzi, F., Cockburn, B., and Marini, D. 2000. Discontinuous Galerkin Methods for Elliptic Problems. In *Discontinuous Galerkin Methods: Theory, Comp. and Appl.*, vol. 11 of *LNCSE*. Springer Verlag, 101–89.

Arnold, D. N., Falk, R. S., and Winther, R. 2006. Finite Element Exterior Calculus, Homological Techniques, and Applications. *Acta Numerica 15*, 1–155.

Bobenko, A. I., and Springborn, B. A. A Discrete Laplace-Beltrami Operator for Simplicial Surfaces. Preprint (2005) http://www.arxiv.org/math/0503219; to appear in *Discr. & Comp. Geom.*

Bossavit, A., and Kettunen, L. 1999. Yee-Schemes on a Tetrahedral Mesh, with Diagonal Lumping. *Int. J. Num. Model. 12*, 129–142.

Botsch, M., and Kobbelt, L. 2004. An Intuitive Framework for Real-Time Freeform Modeling. *ACM Trans. Graph. 23*, 3, 630–634.

Botsch, M., Bommes, D., and Kobbelt, L. 2005. Efficient Linear System Solvers for Mesh Processing. In *Mathematics of Surfaces XI*, vol. 3604 of *LNCS*. Springer Verlag, 62–83.

Cabral, B., and Leedom, L. C. 1993. Imaging Vector Fields Using Line Integral Convolution. In *Proc. ACM/SIGGRAPH Conf.*, 263–270.

Cook, R. L., and DeRose, T. 2005. Wavelet Noise. *ACM Trans. Graph. 24*, 3, 803–811.

Davis, T. A., and Hager, W. M. 1999. Modifying a Sparse Cholesky Factorization. *SIAM J. Matr. Anal. Appl. 20*, 3, 606–627.

Davis, T. A., and Hager, W. M. 2001. Multiple-Rank Modifications of a Sparse Cholesky Factorization. *SIAM J. Matr. Anal. Appl. 22*, 4, 997–1013.

Desbrun, M., Kanso, E., and Tong, Y. 2006. Discrete Differential Forms for Computational Modeling. In Grinspun et al. [2006].

Dodziuk, J., and Patodi, V. K. 1976. Riemannian Structures and Triangulations of Manifolds. *J. Ind. Math. Soc. 40*, 1–4, 1–52.

Elcott, S., and Schröder, P. 2006. Building Your Own DEC at Home. In Grinspun et al. [2006].

Elcott, S., Tong, Y., Kanso, E., Schröder, P., and Desbrun, M. 2007. Stable, Circulation-Preserving, Simplicial Fluids. *ACM Trans. Graph. 26*, 1.

Fisher, M., Springborn, B., Bobenko, A. I., and Schröder, P. 2006. An Algorithm for the Construction of Intrinsic Delaunay Triangulations with Applications to Digital Geometry Processing. In Grinspun et al. [2006].

Gortler, S. J., Gotsman, C., and Thurston, D. 2006. Discrete One-Forms on Meshes and Applications to 3D Mesh Parameterization. *Comput. Aided Geom. Des. 33*, 2, 83–112.

Grinspun, E., Schröder, P., and Desbrun, M., Eds. 2006. *Discrete Differential Geometry*. Course Notes. ACM SIGGRAPH.

Gu, X., and Yau, S.-T. 2003. Global Conformal Surface Parameterization. In *Proc. Symp. Geom. Proc.*, 127–137.

Hertzmann, A., and Zorin, D. 2000. Illustrating Smooth Surfaces. In *Proc. ACM/SIGGRAPH Conf.*, 517–526.

Hildebrandt, K., Polthier, K., and Wardetzky, M. 2006. On the Convergence of Metric and Geometric Properties of Polyhedral Surfaces. Tech. Rep. ZR-05-24, Konrad-Zuse-Zentrum für Informationstechnik, Berlin.

Hirani, A. N. 2003. *Discrete Exterior Calculus*. PhD thesis, Caltech.

Lefebvre, S., and Hoppe, H. 2006. Appearance-Space Texture Synthesis. *ACM Trans. Graph. 25*, 3, 541–548.

Mebarki, A., Alliez, P., and Devillers, O. 2005. Farthest Point Seeding for Efficient Placement of Streamlines. In *IEEE Visualization*, 479–486.

Mercat, C. 2001. Discrete Riemann Surfaces and the Ising Model. *Comm. in Math. Physics 218*, 1, 177–216.

Meyer, M., Desbrun, M., Schröder, P., and Barr, A. 2002. Discrete Differential-Geometry Operators for Triangulated 2-Manifolds. In *Vis. and Math. III*, H.-C. Hege and K. Polthier, Eds. Springer Verlag, 35–57.

Pedersen, H. K. 1995. Decorating Implicit Surfaces. In *Proc. ACM/SIGGRAPH Conf.*, 291–300.

Polthier, K., and Preuß, E. 2003. Identifying Vector Field Singularities using a Discrete Hodge Decomposition. In *Vis. and Math. III*, H. C. Hege and K. Polthier, Eds. Springer Verlag, 113–134.

Praun, E., Finkelstein, A., and Hoppe, H. 2000. Lapped Textures. In *Proc. ACM/SIGGRAPH Conf.*, 465–470.

Schlick, C. 1994. An Inexpensive BRDF Model for Physically-Based Rendering. *Comp. Graph. Forum 13*, 3, 233–246.

Sorkine, O., Cohen-Or, D., Irony, D., and Toledo, S. 2005. Geometry-Aware Bases for Shape Approximation. *IEEE Trans. Vis. Comp. Graph. 11*, 2, 171–180.

Theisel, H. 2002. Designing 2D Vector Fields of Arbitrary Topology. *Comp. Graph. Forum 21*, 3, 595–604.

Toledo, S., 2003. TAUCS. Software at http://www.tau.ac.il/~stoledo/taucs/.

Tong, Y., Lombeyda, S., Hirani, A. N., and Desbrun, M. 2003. Discrete Multiscale Vector Field Decomposition. *ACM Trans. Graph. 22*, 3, 445–452.

Tong, Y., Alliez, P., Cohen-Steiner, D., and Desbrun, M. 2006. Designing Quadrangulations with Discrete Harmonic Forms. In *Proc. Symp. Geom. Proc.*, 201–210.

Turk, G. 2001. Texture Synthesis on Surfaces. In *Proc. ACM/SIGGRAPH Conf.*, 347–354.

van Wijk, J. J. 1991. Spot Noise Texture Synthesis for Data Visualization. *Comp. Graph. (Proc. of ACM/SIGGRAPH Conf.) 25*, 4, 309–318.

Wang, K., Weiwei, Tong, Y., Desbrun, M., and Schröder, P. 2006. Edge Subdivision Schemes and the Construction of Smooth Vector Fields. *ACM Trans. Graph. 25*, 3, 1041–1048.

Wei, L.-Y., and Levoy, M. 2001. Texture Synthesis over Arbitrary Manifold Surfaces. In *Proc. ACM/SIGGRAPH Conf.*, 355–360.

Whitney, H. 1957. *Geometric Integration Theory*. Princeton University Press.

Zhang, E., Mischaikow, K., and Turk, G. 2006. Vector Field Design on Surfaces. *ACM Trans. Graph. 25*, 4, 1294–1326.