Tangent Vector Fields on Triangulated Surfaces - An Edge-Based Approach

Alexandre Djerbetian

Tangent Vector Fields on Triangulated Surfaces - An Edge-Based Approach

Research Thesis

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Computer Science

Alexandre Djerbetian

Submitted to the Senate
of the Technion — Israel Institute of Technology
Shebat 5776 Haifa January 2016

This research was carried out under the supervision of Prof. Mirela Ben-Chen in the Faculty of Computer Science.

The generous financial help of the Technion is gratefully acknowledged.

Contents

List of Figures

Abstract

Since Tron in 1982, Computer-Generated Imagery (CGI) gradually infiltrated most of today's productions. From purely animated movies like Ratatouille to integrated special effects in Star Wars, CGI is present in almost every movie. Partly, the immense visual progress from Tron to Ratatouille can be attributed to advances made in Geometry Processing.

Geometry Processing covers several subfields, such as texture handling, mesh animation or remeshing. In many of those applications, the algorithms rely on the ability to design and handle vector fields, and more precisely, tangent vector fields to surfaces. Even though tangent vector fields are intuitive to visualize, their handling has been, and still is, extensively studied. Indeed, the discretization of curved surfaces into triangulations leads for instance to ill-defined tangent planes on edges and at vertices. The main question at the origin of this work is therefore the question of representation: *how do we discretize / sample a vector field on a discrete surface?*

The first and most popular choice of representation is **face-based sampling**. The representation requires one vector per face, and considers the vector field constant on each face. Since the tangent plane of a face is the face itself, this representation alleviates the problem of tangency and therefore has simple formulations of most of the common operators. However, it also leads to lower accuracy and to difficulties in defining derivatives and smoothness energies.

Another choice is to sample the vector field on **vertices**, and then "linearly" interpolate the values inside faces. This representation yields better results in terms of accuracy and allows the direct computation of derivatives and smoothness energies. However, one needs to redefine the tangent plane at the vertex, which is a nonlinear operation and therefore leads to a complicated formulation.

Finally, another representation used in Geometry Processing comes from **Discrete Exterior Calculus** (DEC), and stores on every edge the integrated projection of the vector field along the edge. This representation is linear, and therefore simple and accurate but has one major drawback: contrary the previous representations, it is not clear how DEC can represent N-RoSy vector fields. N-RoSy fields are N Rotationally Symmetric vectors associated to one point. For instance, a 4-RoSy vector field assigns a "cross" to every point of the mesh. N-RoSy vector fields are very useful in remeshing and non-photorealistic rendering, for instance.

In our work, we propose a simple yet powerful **new representation** which stores vectors on **edge mid-points** and linearly interpolates inside faces. Interestingly, the tangency problem can be trivially resolved at the edges, and the resulting vector fields are linear per face. Our representation is therefore simple, accurate and can handle N-RoSy vector fields and therefore provides a simple unified framework for tangent vector field processing.

Chapter 1

Introduction

Tangent vector fields are used ubiquitously in geometry processing, with applications ranging from texture synthesis [WLKT09, KCPS15] through non-photorealistic rendering [HZ00] and quadrangular remeshing [BLP+12] to physically based simulation [AWO+14, AVW+15]. It is therefore of interest to design a discrete representation of tangent vector fields which is on the one hand simple and thus widely applicable, and on the other hand accurate and robust.

One of the main challenges in tangent vector field representation is to reconcile the need for *smoothness* of the vector field, with the nature of the tangent space on triangle meshes, which is not well-defined at vertices and edges. Smoothness is important for *vector field design*, where it serves as a regularizer for finding tangent vector fields given some user constraints. Another challenge is to define a space of tangent vector fields which is compatible with the space of *scalar functions*, such that, for example, it is possible to find a function whose gradient is most similar to a given vector field. This is important, for example, for *quadrangular remeshing*, where two functions are sought after which are aligned with a given input. Finally, also for remeshing, it is of interest to be able to handle *rotationally symmetric* vector fields, also known as *N-RoSy fields*, which can represent vector fields that are known only up to some rotational ambiguity.

Existing representations of tangent vector fields excel at one or more of the previously mentioned challenges, however, to the best of our knowledge, there is no single representation which can handle all three. Specifically, vertex-based representations [ZMT06, KCPS13, dGDT15] yield smooth vector fields with well defined derivatives, yet are not directly applicable to texture mapping and quadrangular remeshing, due to the lack of relationship to a scalar function space. Furthermore, they are somewhat complicated to analyze due to the intricate constructions involved in the representation. On the other hand, face-based representations [PP03, War06] are piecewise-constant, and thus are compatible to gradients of piecewise-linear functions, and are additionally simple and easy to manipulate. However defining their derivatives is not straight-forward and leads to various difficulties.

Therefore, common practice is (as suggested e.g. in the recent review [dGDT15]) to

first design a vector field using a vertex based representation, and then sample it to get a face based representation for further applications. Unfortunately, this approach both complicates the pipeline, as not all processing is done within the same framework, and additionally introduces error due to the conversions between representations.

The goal of this thesis is to bridge this gap, and introduce a representation of tangent vector fields which is applicable both for vector field design and for quadrangular remeshing, and can additionally be used for function and vector field transport, which are important for physically-based simulation.

We therefore propose (Section 2) a tangent vector field representation which is based on the *non-conforming piecewise-linear* basis functions for triangle meshes [War06]. Our representation shares various properties with existing approaches, e.g. it leverages a complex formulation and thus is easily generalizable to N-RoSy fields, it is based on a piecewise-smooth basis and can thus be derivated directly inside triangles, and it is additionally closely related to discrete edge-based one-forms. We can use it to define first order derivative operators such as divergence, curl (Section 3.3) and covariant derivative (Section 5.1) as well as to define second order derivative operators such as the connection Laplacian (Section 3.1), and finally to define operators which act on functions, such as gradient (Section 4.2) and the Laplace-Beltrami operator (Section 4.2).

As broad applicability is our main goal, we introduce the operators in the context of three applications: vector field design (Section 3), quadrangular remeshing (Section 4) and transport (Section 5), and show how to compute all the required quantities for implementing these applications within our framework. In addition, we provide comparisons of our approach with state-of-the-art methods, showing that our approach is simpler, more efficient, and yields comparable results, within a unified framework.

1.1 Other Vector Field Representations at a Glance

We provide here a brief classification of existing vector field representations in Computer Graphics, following the recent review [dGDT15]. We will provide a detailed analysis of the differences between these methods and ours in Section 6.

Face-based representation. This is perhaps the simplest representation of tangent vector fields, which assigns a single constant vector per *face* of the mesh. The advantages of this representation are its simplicity, and its relation to piecewise linear functions, which are given by values at the vertices or at the edges of the mesh. However, since the vector field is constant per face, directly computing derivatives on triangles is not meaningful, and it is therefore required to resort to representation in a smooth basis [AOCBC15] in order to have well defined derivatives. Some notion of smoothness has been proposed [RVLL08], but it relies on a non convex energy which is therefore difficult to minimize.

Vertex-Based representations. In these approaches, the vector field is defined at the *vertices* of the mesh, and interpolated to the neighboring faces. Since the tangent space at the vertex is not canonically defined, various methods have been proposed to tackle this difficulty. In [ZMT06], the authors use a *geodesic polar map* to create a local chart on the one ring of a vertex and map it to the plane. Unfortunately, this construction is not well defined in the triangle near the vertex, which led the authors to "cut out" a small piece of the triangle near the vertex for the computations. A different approach to address this issue was proposed in [KCPS13], where instead of performing integration on the flat triangle, the curvature of the vertex was "pushed" to the triangle, and the computations were done on these curved triangles (see section 6.2). Both formulations lead to somewhat complicated expressions for evaluating the vector field inside the triangle, and in general to difficulties in tailoring specific operators to specific applications.

Finally, most recently, an extrinsic approach was discussed in [dGDT15], where given additional tangent planes and reference frames at the vertices and edges, it is possible to define a smoothly varying vector field which is defined everywhere. Unfortunately, this requires additional information beyond the input mesh, which has to be provided together with the vector field representation for the vector field description to be usable. Furthermore, the resulting expressions are somewhat complex as well.

Our approach shares the same design guidelines as [ZMT06, KCPS13], namely: we define an intrinsic tangent space with a corresponding local frame at chosen locations, and then represent the vector field as a linear combination of the frame-based representations. However, by choosing the local frames to be at the *edges* where the mesh is locally flat, instead of the at vertices where the angle deficit complicates matters, the flattening procedure is considerably simpler. Specifically, curved triangles are not required, thus considerably simplifying the resulting formulation.

Discrete 1-forms. Discrete Exterior Calculus [Hir03] provides a *coordinate-free* approach to tangent vector field representation, by encoding the line integral of the projection of the vector field on the oriented mesh edges. Thus, this approach is, like ours, *edge-based*. However, using only discrete one-forms, representing symmetric tensors and first order derivatives requires additional, somewhat intricate, machinery [dGLB$^+$14], whereas in our setup the formulation of all the operators is quite straightforward, using standard variational formulations and piecewise linear basis functions. Interestingly, we show in Section 6.3, that our representation can be understood as a generalization of DEC, which spans a larger space of vector field and allows to easily represent N-RoSy fields.

Operator-based representations. A more recent approach to tangent vector field representation [ABCCO13] has suggested to represent vector fields in a *coordinate-free* manner, as *linear derivation operators on scalar functions*. This point of view is

especially beneficial when the *transport* of quantities on the surface is required, e.g. for parallel transport, as has been shown in [AOCBC15]. Our representation is in fact *complementary* to the operator approach, as we can similarly construct the required global linear operators based on our representation, instead of using piecewise constant vector fields.

1.2 Contribution

A discrete 1-form is represented using one real number per edge and the other representations use one complex number per vertex or face. We propose to combine both approaches and store one complex number per edge. Our construction then linearly interpolates the complex values to faces therefore providing (1) a simple representation, thus widely applicable, (2) a piecewise-linear basis whose derivatives can be easily computed, and (3) a straightforward generalization to N-RoSy fields.

Chapter 2

Edge-Based Tangent Vector Fields

Given a triangulated surface $\mathcal{M}$, our goal is to define a discrete basis to represent tangent vector fields on $\mathcal{M}$, and then derive the corresponding operators. We first describe our representation, and then, in the context of three applications, we show how various standard operators can be computed in this representation.

2.1 Motivation

Two decisions are required for choosing a discrete representation: (1) where to sample the vector field, and (2) how to interpolate between the samples. Our goal is to enable the computation of first order derivatives, and thus we choose to interpolate the values using *piecewise linear functions*. On a triangle mesh, two options are available for piecewise linear functions [dGDT15], one which interpolates values at the *vertices* and is continuous everywhere on the mesh, and another which interpolates values at the *edges*, and is only continuous at edge-midpoints. In both cases, an interpolated vector field constructed with those functions must be continuous at the sample points by construction. Thus, for a curved surface, at the location of the sample points, the mesh should be mapped to a flat domain, as only there a tangent vector field can be continuous across triangles.

If one chooses to sample at the vertices (as has been done in [ZMT06, KCPS13]), this flattening procedure is non-trivial, since the discrete Gaussian curvature is *concentrated* at the vertices, and thus some distortion must be introduced when flattening, complicating the formulation. We on the other hand sample the vector field at the *edge-midpoints*, where the Gaussian curvature is zero, as two triangles can be easily flattened locally without introducing distortion. This leads to much simpler expressions for the vector field basis, as well as the resulting operators.

2.2 Formulation

Complex representation. On a planar domain, a vector field ψ is commonly represented as a function from $\mathbb{R}^2$ to $\mathbb{R}^2$. Following [KCPS13], we replace the tangent plane $\mathbb{R}^2$ with the complex line $\mathbb{C}$ and write $\psi(x, y) = z(x, y)\vec{X}$, with $z : \mathbb{R}^2 \to \mathbb{C}$.

Local frames. The tangent spaces of a surface do not have a canonical two dimensional coordinate system. Therefore, for every point $p \in \mathcal{M}$, we need to define an arbitrary unit norm *coordinate frame* $\vec{X}_p$. At an edge mid-point q between the vertices v_i and v_j, we choose $\vec{X}_q$ to be the normalized edge vector, with the orientation from the lower to higher vertex index (see figure 2.1). For any other point p inside a face of $\mathcal{M}$, we choose $\vec{X}_p$ to be the direction of the first edge of the face. Note, that we always work *per triangle*, and thus the local frame at the vertex has multiple values, one for every neighboring triangle.

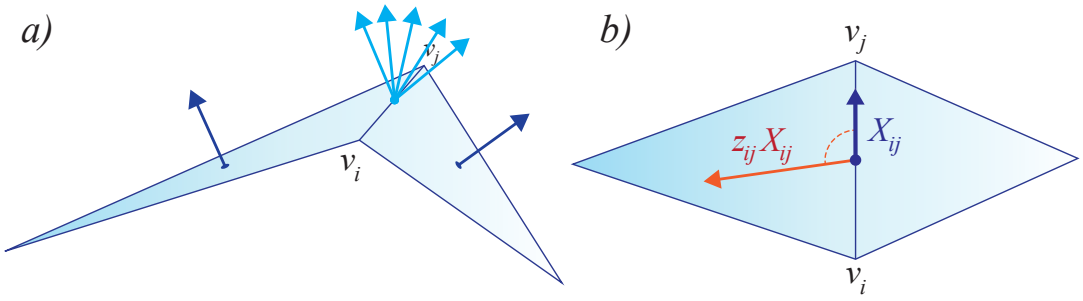

Figure 2.1: Although there is no well-defined tangent space on the edge (a), we can unfold the adjacent triangles (b), to obtain a local distortion-free planar parameterization at the edge mid-point. This property differentiates our method from vertex based sampling. Note that we oriented the edge from i to j assuming $i < j$.

Transport coefficients. Given a coordinate system at every point, we need to relate the coordinate systems of nearby tangent spaces. For two points p, q on the same face, we call the transport function $r_{pq} : T_p\mathcal{M} \to T_q\mathcal{M}$ the function which transports a vector sampled at p to the point q along a straight line. Namely, r_{pq} takes a vector expressed in the $\vec{X}_p$ coordinate system and expresses it in the $\vec{X}_q$ coordinate system. Since $\vec{X}_p$ and $\vec{X}_q$ have unit norm, r_{pq} only needs to account for the rotation of the coordinate system and can be represented with a single unit norm complex number $r_{pq} \in \mathbb{C}$, which we denote as the *transport coefficient*.

Interpolation. Given the transport coefficients, we can now interpolate values. Let $p \in T_{123}$ be a point in the triangle in Figure 2.2 (a). Given complex values z_1, z_2, z_3 at the edge mid-points q_1, q_2 and q_3, our vector field representation is given by:

$$\psi(p) = \psi_1(p)\, z_1 + \psi_2(p)\, z_2 + \psi_3(p)\, z_3, \tag{2.1}$$

where $\psi_i(p) = \phi_i(p)\, r_{q_i p}$ is our vector field interpolation basis and ϕ_i is the hat basis function for the mid-edges: $\phi_i(q_j) = \delta_{ij}$ [dGDT15] (see Figure 2.2 (b)). To simplify notations, we will denote r_{ij} for $r_{q_i q_j}$. Figure 2.3 illustrates our construction on the generic triangle T_{123} of figure 2.2. Note that our basis is not continuous across edges, but at the edge mid-points. This does not pose any practical problem as all our constructions rely on integrals over $\mathcal{M}$ which are computed as a sum of integrals over triangles.

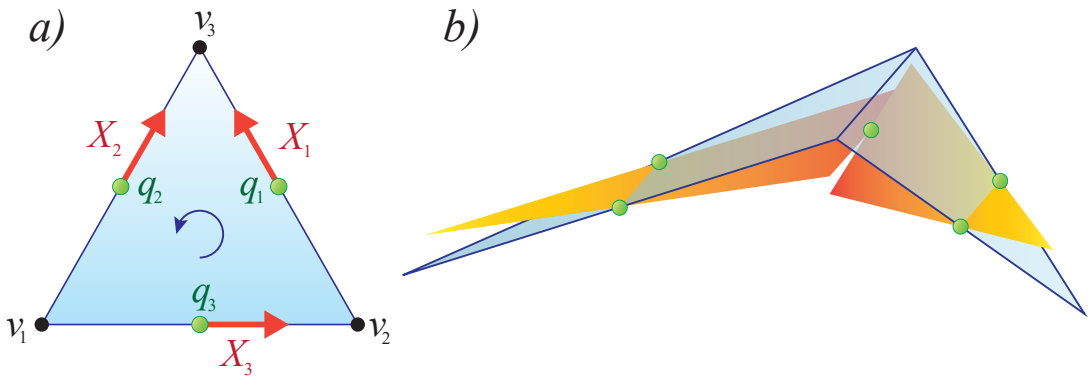

Figure 2.2: a) Notations for a triangle T_{123}. b) Edge mid-point hat basis function (see[dGDT15]).

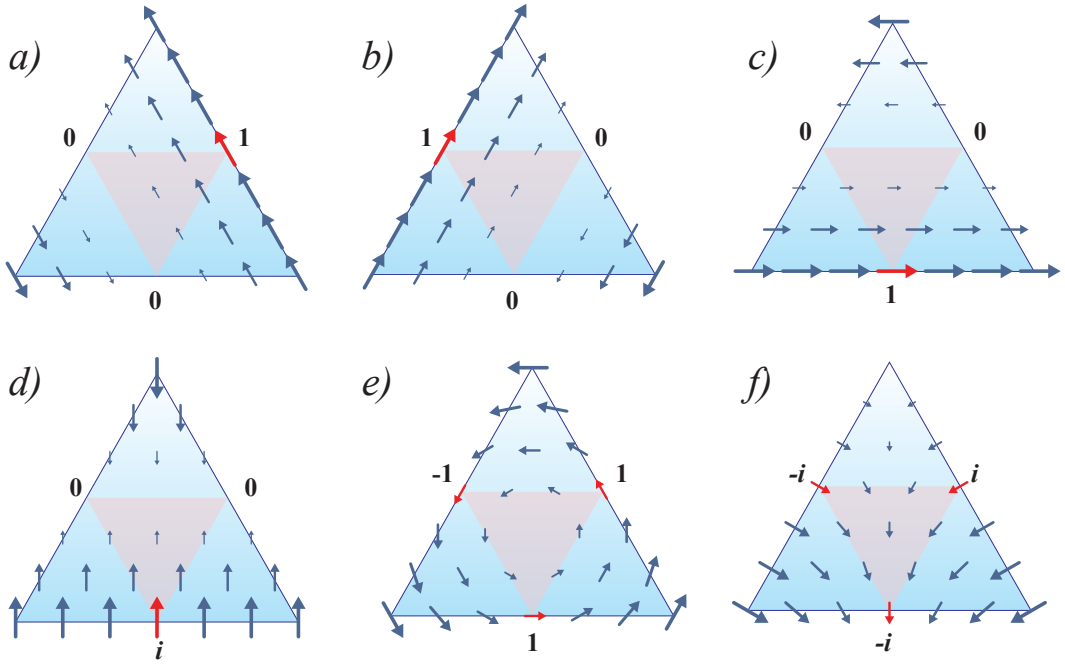

Figure 2.3: The three piecewise linear edge basis vector fields $\{\psi_i\}$ (top) and some interpolated vector fields (bottom) on the triangle from Figure 2.2 a). The pink triangle highlights the region where all the barycentric interpolation coefficients $\phi_i(p)$ are positive.

***n*-vector fields** Following [KCPS13], we represent n-vector fields by their complex *angle-based n-th power*. In other words, if $u = ae^{\imath\theta}$ is one component of our n-vector field (with $a = |u|$), we work with $z = ae^{\imath n\theta}$ which we call the *representation vector*. To get the n-vector field back from z, we take the *angle-based n-th root* of z: $\tilde{u} = |z|\, e^{\imath \arg(z)/n}$ which can differ from u by an integer number of rotations of angle $2\pi/n$ (see figure 2.4). Note that our construction is slightly different from the one of [KCPS13] where the authors used the standard power and angle root, thus leading to different norms for a n-vector and its representation vector, which we believe is less desirable.

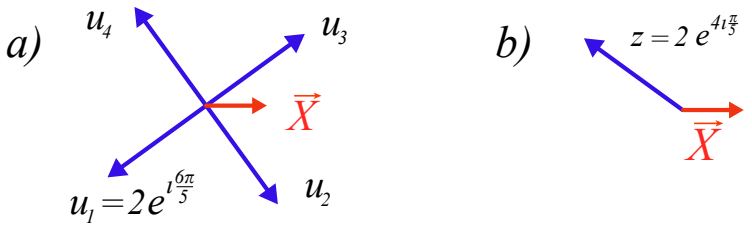

Figure 2.4: a) Original cross field represented by the u_i complex numbers. b) The representation vector z as the nth angle power of u. The first nth angle root of z is u_3.

2.3 Operators and Notation

Unless specified otherwise, for every operator, we compute the stiffness and mass matrices using a variational formulation derived from the problem. Whenever integration by part is required, we implicitly assume the mesh is closed. Since all computations are done by integrating over triangles, we only provide the contribution of a single triangle to the matrix entries. The general assembly into the global matrices are then made by looping over triangles and indices. Please see Figure 2.2 for an illustration of the notations, and the appendix A for the derivation details.

Interpolation Bases. We use φ_i to denote the standard hat basis functions for the vertex v_i, and $\phi_i = 1 - 2\varphi_i$ for the hat basis function of the edge mid-point q_i. We further denote by ψ_i the basis vector field of the edge e_i. The complex value of vector field ψ at q_i is denoted by z_i .

Geometry. The positive angle at the vertex v_i is given by α_i, and the triangle area by A_T. We additionally require the orientation of the edges, and thus denote by δ_i the coefficient matching the orientation of X_i with its corresponding triangle (e.g. in Figure 2.2, $\delta_1 = 1, \delta_2 = -1$ and $\delta_3 = 1$). The transport coefficient in an oriented triangle T_{ijk} from q_i to q_j is denoted by $r_{ij} = -\delta_i\delta_j e^{i\alpha_k}$ (note that $r_{ji} = \bar{r}_{ij}$).

Inner products. We denote by $\langle\langle\cdot,\cdot\rangle\rangle$ and $\|\cdot\|$ the Hermitian dot product and its associated norm on our discrete surface $\mathcal{M}$, i.e.: $\langle\langle\xi,\psi\rangle\rangle = \int_{\mathcal{M}} \overline{\xi}\cdot\psi$ and $\|\psi\|^2 = \langle\langle\psi,\psi\rangle\rangle$.

We additionally use $|\cdot|$ for the pointwise norm and for the absolute value of a complex number, where required.

We will now provide the details for the computation of various vector field operators in our framework, in the context of different applications.

Chapter 3

Vector field design

We describe various possible formulations of tangent vector field design. We start from the simplest setup of finding the *smoothest* tangent vector field using the *connection Laplacian*, then demonstrate how to additionally minimize the *divergence* or *curl* of the vector field, and finally we show how to apply various alignment constraints.

3.1 The Smoothest vector fields

We first provide the setup, following [KCPS13], and then discuss the benefits of our formulation.

Energy. Given a surface $\mathcal{M}$, the smoothest vector fields on $\mathcal{M}$ are defined as the vector fields ψ with $\|\psi\|^2 = \int_{\mathcal{M}} |\psi|^2 = 1$ which minimize the Dirichlet energy:

$$E_{_D}(\psi) = \int_{\mathcal{M}} |\nabla\psi|^2 , \tag{3.1}$$

where $\nabla\psi$ is the Jacobian matrix of ψ. The positive definite quadratic form associated with the Dirichlet energy is the *connection Laplacian* Δ_c. Thus, we can find discrete minimizers of (3.1) by solving an eigenvector problem for a discretization of Δ_c.

The connection Laplacian. Using our interpolation basis, we define a vector field as the linear combination $\psi = \sum z_i\psi_i$, and find the smoothest vector fields as the eigenvectors corresponding to the smallest eigenvalue of the eigenvalue problem:

$$L_c z = \lambda M_z z,$$

with $(L_c)_{ij} = \langle\langle \nabla\psi_i, \nabla\psi_j \rangle\rangle$ and $(M_z)_{ij} = \langle\langle \psi_i, \psi_j \rangle\rangle$. The simplicity of our linear basis makes the matrix entries straightforward to compute, and the resulting formulas are

concise and easy to implement. For the stiffness matrix we have:

$$(L_c)_{ii} = \frac{|e_i|^2}{A_T}, \qquad (L_c)_{ij} = -\frac{|e_i|\,|e_j|}{A_T}\cos(\alpha_k)\; r_{ji},$$

and the mass matrix M_z is diagonal and real: $(M_z)_{ii} = \frac{1}{3}A_T$. Note that no lumping was involved for the mass matrix and M_z is therefore entirely consistent. We refer the reader to the appendix A for the derivation, and show some examples in Figure 3.1.

Smoothest *n*-vector field To design the smoothest n-vector field, we first design the smoothest representation vector field and then take the n-th angle root of the result. As noted in [KCPS13], we only need to slightly modify our stiffness matrix L_c to act on representation vector fields, by replacing r_{ji} with $(r_{ji})^n$. See figure 3.1 (b-d) for an illustration.

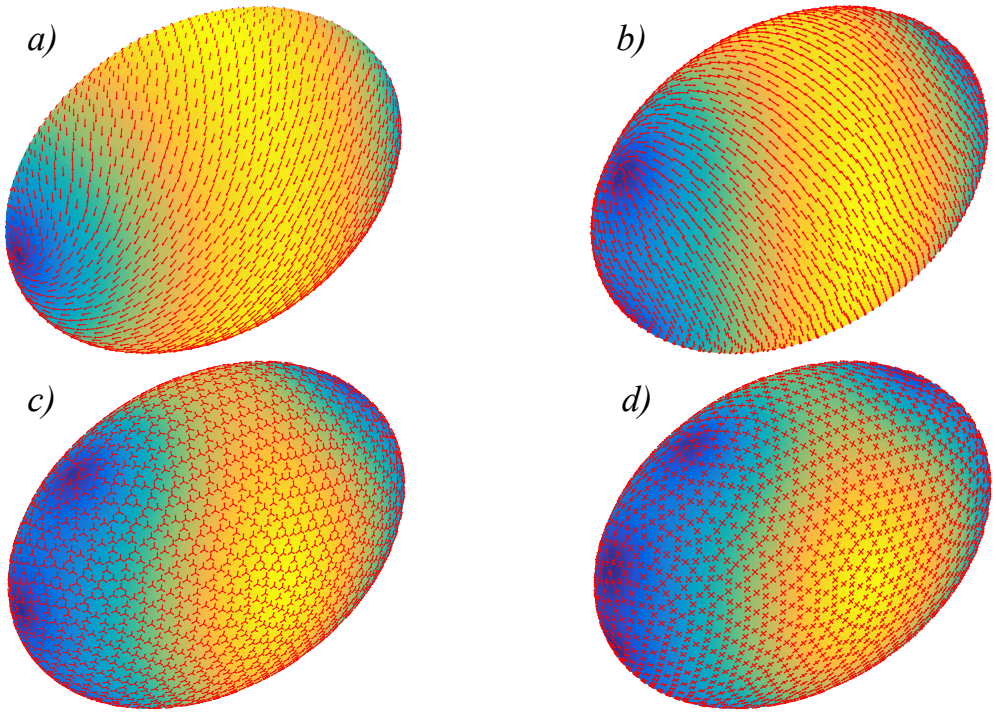

Figure 3.1: Smoothest n-vector fields on an ellipsoid. The color is the norm of the vector field. a), b), c) and d) are respectively the smoothest 1, 2, 3 and 4-vector fields with 2, 4, 6 and 8 singularities.

3.1.1 Comparison to [KCPS13]

Our results are visually very similar to the ones obtained by [KCPS13], however our construction improves on the previous method both in simplicity and in accuracy while preserving some desirable properties such as resistance to noise.

Simplicity. First, our matrix entries are considerably simpler (see e.g. Appendices D.2 and D.3 in [KCPS13]), which is beneficial in practice for implementation and analysis. Furthermore, our mass matrix M_z is real and diagonal, which is not the case for the vertex-based representation. This allows us to compute its inverse, and use it for composing operators (Section 4.2) and for using exponential integrators for PDEs (Sections 3.2 and 5.1).

Accuracy. As illustrated in figure 3.2, the eigenvalues of both discretizations converge at the same rate both for regular and grid-like meshes. As our matrices are bigger (we compute $n_e \approx 3\ n_v$ complex values), we are, for a fixed mesh, on average 2 times slower yet 5 times more accurate. As we can see in figure 3.2 (c), our method indeed provides the best ratio of time-to-accuracy.

Resistance to noise. Similarly to vertex-based sampling, our method is robust to normal noise on vertices (see Figure 3.3).

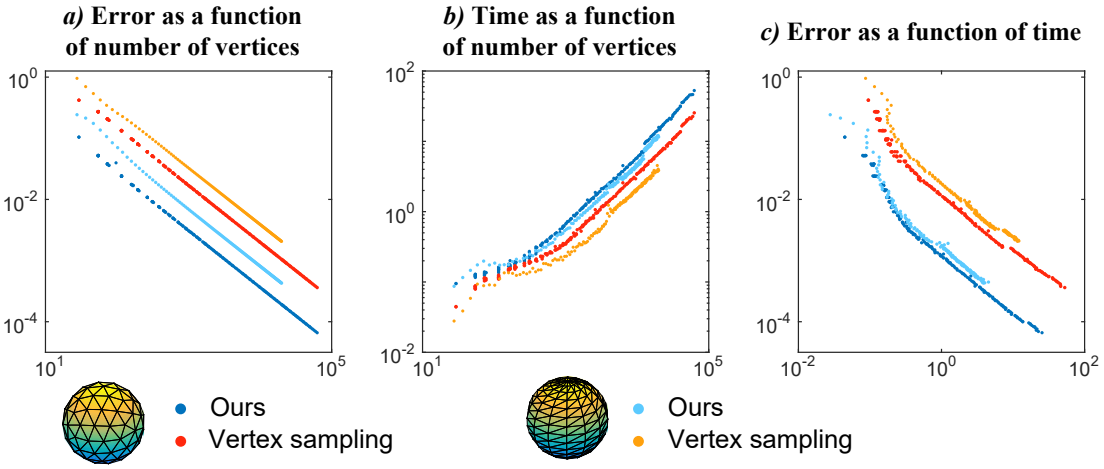

Figure 3.2: Convergence of the first 48 eigenvalues of L_c on a sphere (using two triangulations). Results are compared to the ground truth given in [SW12].

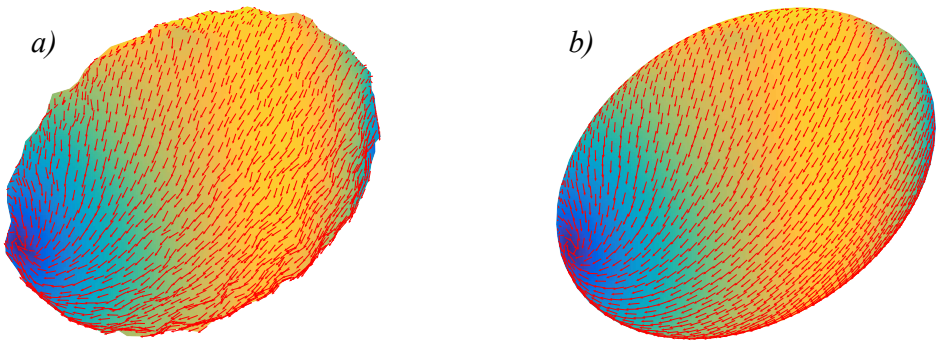

Figure 3.3: Robustness to noise. a) The mesh with added normal noise, on which we compute the smoothest vector field. b) The original mesh with the vector field of a). The result is very close to the one in figure 3.1 (a).

3.2 Vector Field Smoothing by Heat Diffusion

The Laplace-Beltrami operator is used both for computing smooth functions using an eigenvector problem, as well as for function *smoothing*. Similarly, in addition to the computation of the smoothest eigenvectors, the connection Laplacian can be used for *vector field smoothing*. Thus, given an initial, potentially noisy, n-vector field ψ_0, the heat-diffused vector field $\psi(t)$ satisfies:

$$\frac{\partial}{\partial t}\psi(p,t) + \Delta_c\,\psi(p,t) = 0, \quad \psi(p,0) = \psi_0.$$

In the space-discrete setting this becomes $\frac{\partial}{\partial t}z(t) = -M_z^{-1}L_c z(t)$, which we can solve numerically using an exponential integrator [HL97] as $z(t) = \exp(-tM_z^{-1}L_c)z_0$, where exp is the matrix exponent. Note, that there exist efficient methods for computing the multiplication between the exponent of a large sparse matrix and a vector [AMH11]. Results are shown in Figure 3.4 and the attached video. We additionally used heat diffusion for curvature computations in section 3.4.

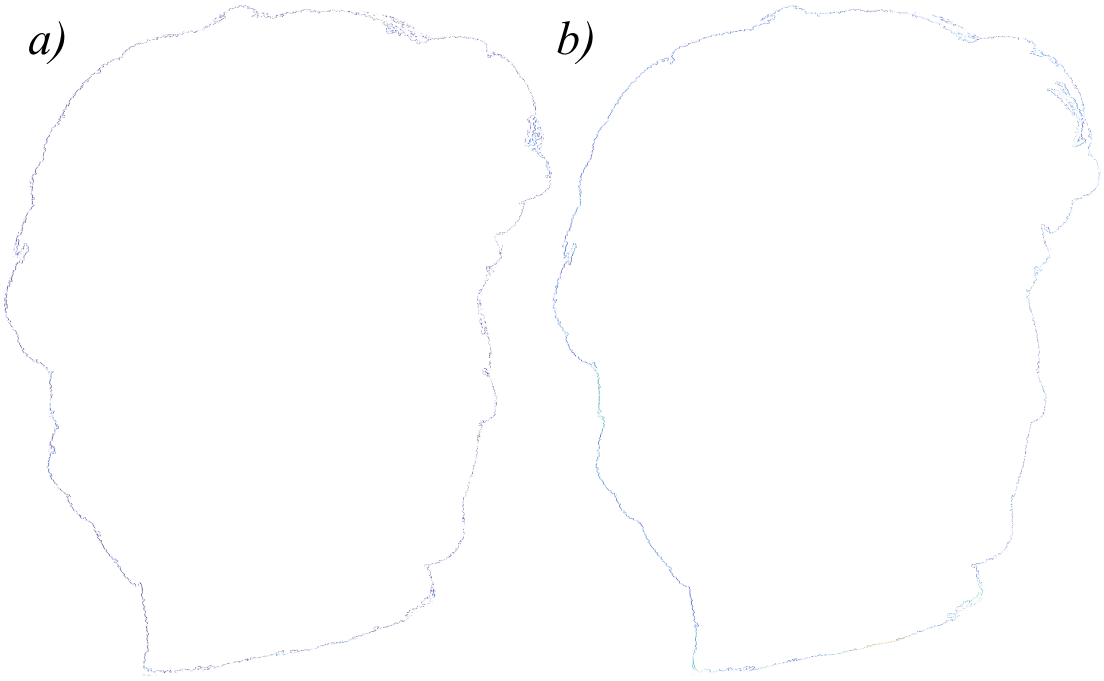

Figure 3.4: Heat diffusion. (a) Very noisy curvature computed as an orthogonal direction to every edge. (b) Heat diffused version of (a) with $t = h/2$ with h the mean edge length of the mesh.

3.3 Prescribe Divergence and Curl

The problem (3.1) has a global degree of freedom, as the global rotation of all the vectors by a constant angle does not modify the energy. We will therefore add another term to the energy to get, for instance, the smoothest divergence free vector field. This will remove the extra degree of freedom, and make the solution unique. As our vector fields are piecewise linear, divergence and curl are available directly as piecewise constant functions per face. Their computation is then trivial, either by explicitly computing derivatives, or using the circulation theorem, as follows.

Integral property. In order to compute the circulation of a vector field, we need to integrate its value along edges. From the median symmetry of the mid-edge hat basis functions ϕ_i, we get the following interesting property for our basis (App. B.1):

$$\int_{e_i} \psi_i(p) = |e_i| \, \vec{X}_i, \qquad \int_{e_j} \psi_i(p) = \int_{e_k} \psi_i(p) = 0. \tag{3.2}$$

In other words, the integral of our vector field along the edge e_i is simply its value z_i at the mid-edge multiplied by the edge length.

Computation. With the help of (3.2), we can now directly compute the circulations of our vector field. Let ψ be a vector field. On a triangle T, we know that $\int_T \operatorname{curl} \psi = \int_{\partial T} \psi \cdot dl$. From the integral property we get:

$$\int_T \operatorname{curl} \psi = \sum_{e \in T} \delta_e \, |e| \operatorname{Re}(z_e).$$

Similar results are achieved for the divergence by noting that $\operatorname{div} \psi = \operatorname{curl} \imath\psi$. If we denote by curl_T and div_T the discrete face valued functions, we have:

$$M_t \operatorname{curl}_T \psi = K \operatorname{Re}(z), \quad M_t \operatorname{div}_T \psi = -K \operatorname{Im}(z)$$

where $K = G_{e \to t} M_e$, $G_{e \to t}$ is the signed graph adjacency matrix from edges to faces, and M_t, M_e are the diagonal mass matrix of triangle area and edge lengths respectively.

Divergence and Curl. We can now assemble our curl (resp. divergence) energy. The global energy becomes $E_\mu = E_D + \mu E_{\text{curl}}$ where μ balances how much the curl energy influences the smoothness energy, and where $E_{\text{curl}}(\psi) = \int_\Omega (\operatorname{curl} \psi)^2$. We now need to compute the matrix L_{curl}. However, as the curl operator is not "complex linear" on complex vector fields, because $\operatorname{Re}(\imath z) \neq \imath \operatorname{Re}(z)$, we need to transform our matrices to act on $\mathbb{R}^{2n}$:

$$z \leftarrow \begin{pmatrix} \operatorname{Re}(z) \\ \operatorname{Im}(z) \end{pmatrix} \quad , \quad L_c \leftarrow \begin{pmatrix} \operatorname{Re}(L_c) & -\operatorname{Im}(L_c) \\ \operatorname{Im}(L_c) & \operatorname{Re}(L_c) \end{pmatrix}.$$

The matrices L_{curl} and L_{div} can then be written as:

$$L_{\text{curl}} = \begin{pmatrix} K^T M_t^{-1} K & 0 \\ 0 & 0 \end{pmatrix} \quad , \quad L_{\text{div}} = \begin{pmatrix} 0 & 0 \\ 0 & K^T M_t^{-1} K \end{pmatrix}.$$

We illustrate some results in figure 3.5. As expected, the energy E_μ with μ small only lifts the global rotation degree of freedom, without affecting the smoothness.

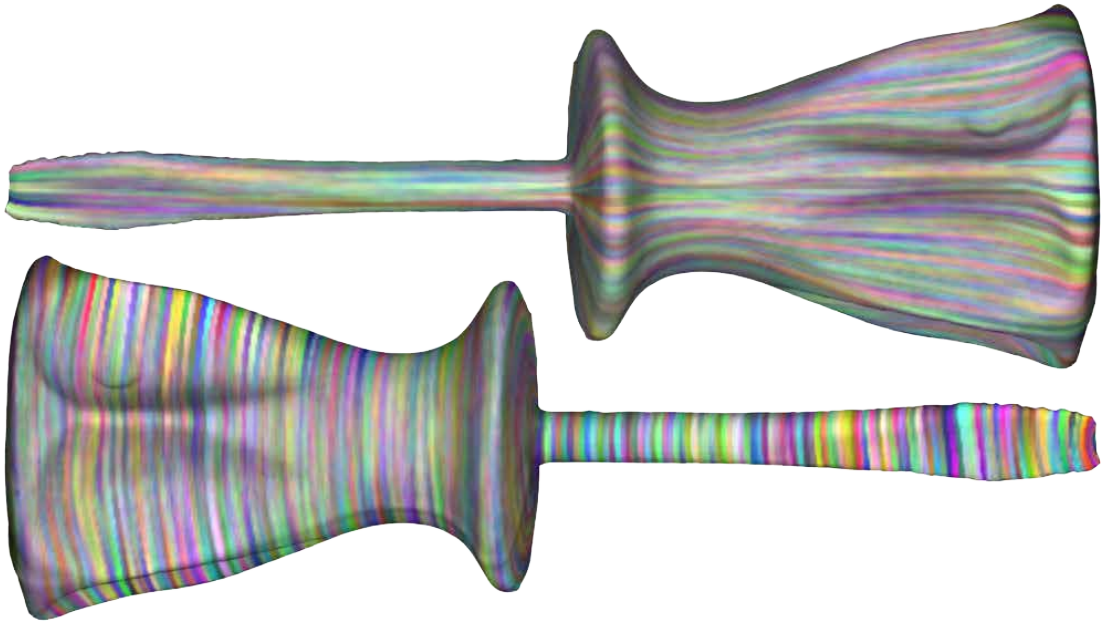

Figure 3.5: Smoothest curl free (top) and div free (bottom) vector fields.

3.4 Alignment

Sparse hard constraints. Sparse hard constraints given as prescribed value at prescribed locations are modeled by minimizing the smoothness energy under linear equality constraints. To prescribe only the direction and not the norm, we add instead an orthogonality constraint as follows. Let $w_i \in \mathbb{C}$ be the direction constraint we impose at q_i. The value z_i of the vector field ψ at (q_i) must satisfy: $\mathrm{Re}(z_i)\,\mathrm{Im}(w_i) - \mathrm{Im}(z_i)\,\mathrm{Re}(w_i) = 0$. Figure 3.6 shows an example where hard constraints (a) and direction constraints (b) are imposed.

Soft global alignment. For soft alignment, we use the alignment energy proposed in [KCPS13]: given an alignment field $\omega = \sum w_i \psi_i$, the alignment energy is: $E_{\text{align}} = \mathrm{Re}\,\langle\langle \omega, \psi \rangle\rangle$. When combined with the smoothness energy using $E_t = (1-t)E_D - tE_{\text{align}}$ where $t \in [0,1]$ balances the tradeoff between alignment and smoothness, E_t has the advantage of promoting alignment where $|\omega(p)|$ is high and promoting smoothness when $|\omega(p)|$ is small. The discrete minimizer to E_t is solution of the system: $(L_c - \lambda_t M_z)z = M_z w$, where choosing $\lambda_t = 0$ usually provides a good trade-off.

Curvature computation In some applications alignment to the curvature directions is required. We compute the curvature 2-vector field ω using the 2 angle power of $w_j = \imath\beta_j / |e_j|$, where β_j is the signed dihedral angle between the two adjacent triangles to the edge e_j [BKP+10]. The curvature direction is therefore orthogonal to every edge. This simple discretization is enough for most meshes, however it is more robust to slightly heat diffuse the curvature directions prior feeding it to the alignment energy. We use $t = h/10$ with h being the mean edge length of our mesh. Figure 3.7 shows a few examples of direction fields aligned with curvature directions.

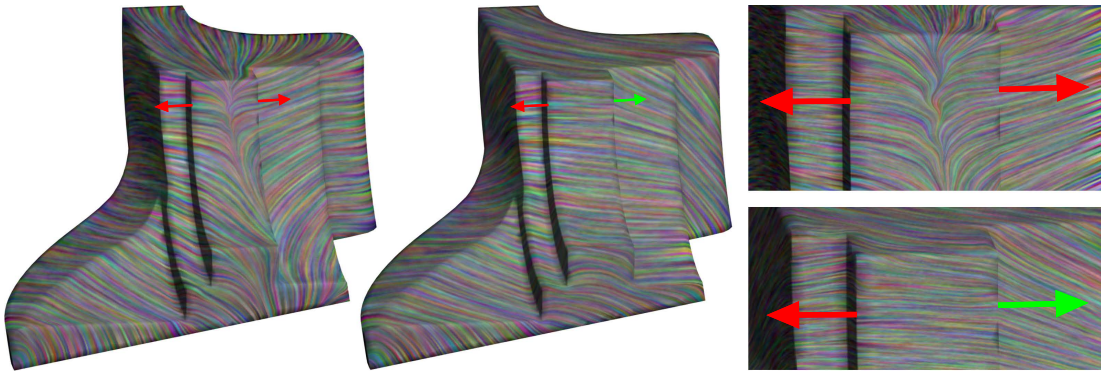

Figure 3.6: The two types of hard constraints. The red arrows enforce norm and direction whereas the green arrow only forces the direction.

Figure 3.7: Direction fields aligned with curvature directions: a) cross field, b) and c) 2-direction field.

Chapter 4

Quadrangulation

To demonstrate the applicability of our vector field representation to quadrangular remeshing, we follow [BZK09], and show how the relevant steps can be performed in our framework.

The quadrangulation algorithm. For the purpose of this paper, we compute the u, v coordinate functions, and refer to [EBCK13] for the quad extraction. The general algorithm is:

1. Design a smooth curvature aligned cross field.
2. Identify the locations of the singularities.
3. Cut the mesh into topological disk patches, such that the singularities of the cross field are on the cut, and propagate on each patch a consistent orientation of one of the cross field component to generate two vector fields U, V.
4. Minimize $E(u, v) = \int_{\mathcal{M}} |\nabla u - U|^2 + |\nabla v - V|^2$ to find the coordinate functions, with additional integer constraints.

The first step is performed as explained in the previous section, and the third step is a topological operation independent of the representation of the vector field. We will therefore first describe how to do the second step, and then address the projection of a given vector field onto the space of gradients of functions, for the last step.

4.1 Singularity detection.

The notion of singularities is closely linked to the one of *turning number* (see [RVLL08]). The turning number $T_\psi(\gamma)$ of the vector field ψ along the loop γ is the integrated angle of the vector field with the tangent to the curve: $T_\psi(\gamma) = \frac{1}{2\pi} \oint_\gamma d\theta(\vec{t_\gamma}, \psi)$, where $\theta(\vec{t_\gamma}, \psi)$ is the oriented angle between ψ and the tangent vector field $\vec{t_\gamma}$ to γ. S is a singularity if for any loop γ enclosing S, $T_\psi(\gamma) \neq 1$. We can then define the index of a singularity as the turning number of an arbitrary small loop enclosing S minus one.

To detect a singularity, we partition the surface into non overlapping "vertex regions" and "face regions" and then compute the turning number on every region's boundary (see figure 4.1 a)). The index formula for the region inside a face T_{123} is $I_f(T) = \frac{1}{2\pi}(\omega_{12} + \omega_{23} + \omega_{31})$, where $\omega_{ij} = \arg\left(\frac{z_j}{z_i\, r_{ij}}\right) \in [-\pi, +\pi]$ represents how much the vector field rotates from q_i to q_j. Similarly, the index formula for the region around the vertex v_i is $I_v(v_i) = \frac{1}{2\pi}\left(\Omega_i + \sum_{T_{ijk}} \omega_{jk}\right)$, where T_{ijk} are the adjacent triangles to v_i and Ω_i is the angle defect at v_i. Note that the computed indexes can only 0, 1 or -1. Figure 4.1 (b,c) and our video show singularities detected on several models.

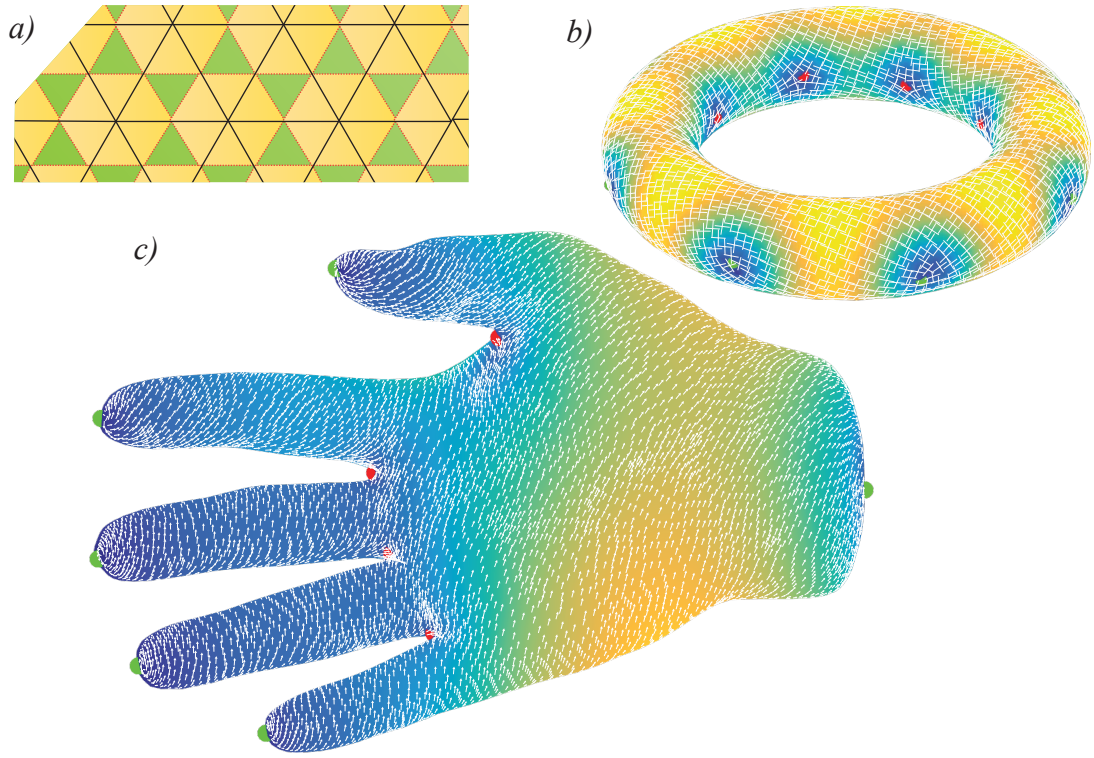

Figure 4.1: Singularity computation. Positive and negative singularities are plotted in green and red respectively. (a) The partition to vertex regions and triangle regions. (b) singularities on the torus for a cross field. (c) Singularities on the hand for a vector field. Note that the singularities are located in natural places.

Singularity remeshing In step 3 of the algorithm, the singularities are required to be on vertices, which is not guaranteed by our cross field. If our cross field has singularity on a triangle, we locate the exact location of the singularity inside the triangle and create a new vertex at the singularity location. We solve for a and b (see App B.2) such that $M(a,\ b)^T = B$, with:

$$M = \begin{pmatrix} \mathrm{Re}\,(z_1 - z_{21}) & \mathrm{Re}\,(z_1 - z_{31}) \\ \mathrm{Im}\,(z_1 - z_{21}) & \mathrm{Im}\,(z_1 - z_{31}) \end{pmatrix}, B = \frac{1}{2}\begin{pmatrix} \mathrm{Re}\,(z_1 - z_{21} - z_{31}) \\ \mathrm{Im}\,(z_1 - z_{21} - z_{31}) \end{pmatrix},$$

where $z_{ij} = z_i r_{ij}$ is the transported value z_i from q_i to q_j. Then, the singularity S is located at $S = v_1 + a\,(v_2 - v_1) + b\,(v_3 - v_1)$.

4.2 Projection on Gradient Vector Fields

We will now address the problem of projecting a given input vector field on the space of gradients of functions, effectively minimizing $E(u) = \int_{\mathcal{M}} |\nabla u - U|^2$, to find the coordinate function u.

The Gradient.

Let f be a function defined the vertices of $\mathcal{M}$, the variational formulation yields:

$$\psi = \nabla f \quad \Leftrightarrow \quad M_z\, z = G\, f,$$

where G is a $n_e \times n_v$ matrix whose entries are $G_{ij} = \langle\langle \psi_i, \nabla \varphi_j \rangle\rangle$. The contribution of a triangle T_{ijk} to G is:

$$G_{ii} = \imath\delta_i \frac{|e_i|}{6}, \qquad G_{ij} = \imath\delta_j \frac{|e_j|}{6} r_{ji}.$$

Graphical analysis Figure 4.2 provides an illustration of the gradient of the hat function. Note that only the red vectors are computed, the others are the result of the interpolation in the face. The 1D analogy of figure 4.2 should not be understood as exact, as the gradient field of figure b) is not even continuous on the common edge, but should serve more as an intuition to explain the improved results for the Laplace-Beltrami operator below.

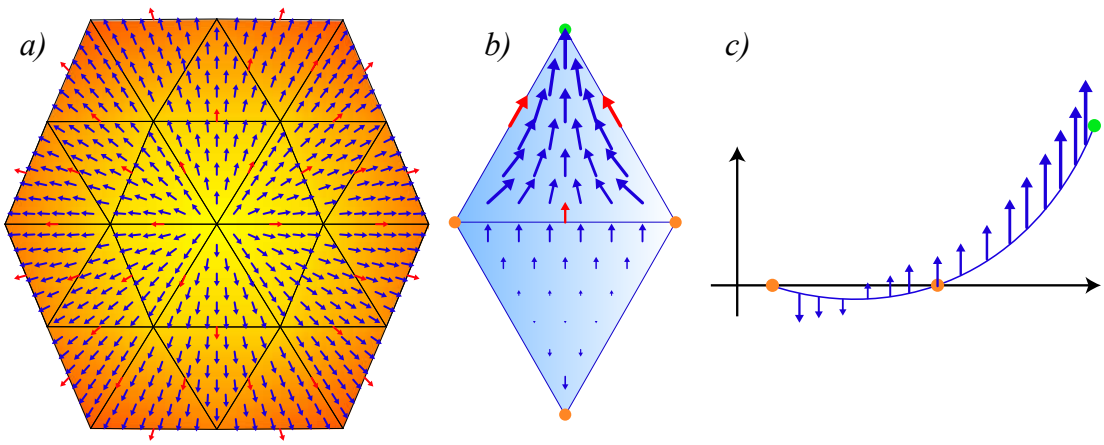

Figure 4.2: Gradients computed with our method. a) Gradient of the height function $f(x, y, z) = z$) at the south pole of the sphere. For plotting purposes we normalized the vectors lengths. In figure a) and b), we illustrate how our gradient can be understood as an approximation of a smooth interpolation of the orange and green points. b) Gradient of the hat function of the green vertex. c) 1D analog figure b).

Tangential component We can show (App. B.3) that the tangential component of the gradient corresponds to a natural finite difference approximation. Let $f = \sum f_i \varphi_i$ be

a function and $z = M_z^{-1} G f$ its discrete gradient. For the edge e_{ij} between the vertices v_i and v_j, we have:

$$\mathrm{Re}(z_{e_{ij}}) = \frac{f_j - f_i}{|e_{ij}|}. \tag{4.1}$$

With (3.2) and (4.1), it is then straightforward to prove that the discrete version of $\mathrm{curl} \circ \nabla = 0$ holds.

The Gradient's Adjoint In the continuous setting, it is well known that the gradient is the L^2 adjoint of the divergence operator. Interestingly, in our complex discrete framework, we get (see App. B.4):

$$M_v(\mathrm{div}_v(\psi) + \imath\, \mathrm{curl}_v(\psi)) = -G^* z,$$

thus, an alternative formulation for the divergence operator, is given by minus the real part of the adjoint of the gradient.

Laplace-Beltrami operator

To project a vector field on the space of gradients, we effectively need to solve a Poisson equation, and thus require a *Laplace-Beltrami operator*. Using our gradient and its adjoint, we can define a new discretization which has a better accuracy in terms of convergence than the standard cotangent weight Laplacian, at virtually the same computational cost. For a function f on $\mathcal{M}$, the classical variational formulation for the Laplace Beltrami operator is

$$u = -\Delta f \quad \Leftrightarrow \quad \forall v, \quad \int_{\mathcal{M}} v \cdot u = \int_{\mathcal{M}} \nabla v \cdot \nabla f.$$

In our framework, this yields the discrete integrated operator:

$$L = G^* M_z^{-1} G. \tag{4.2}$$

As Figure 4.3 shows, this discretization of the Laplace-Beltrami operator is about 3 times more accurate than the standard cotangent weight discretization, and still does not use any higher order lattice subdivision contrary to quadratic or cubic finite elements. However, it is also less sparse than the cotangent weight formulation as it acts on the 2 ring of each vertex, so it is approximately 1.3 times slower. Similarly to the connection Laplacian's convergence (figure 3.2), our discretization has the best ratio of accuracy-to-computation time, compared to all standard discretizations including quadratic and cubic finite elements discretization.

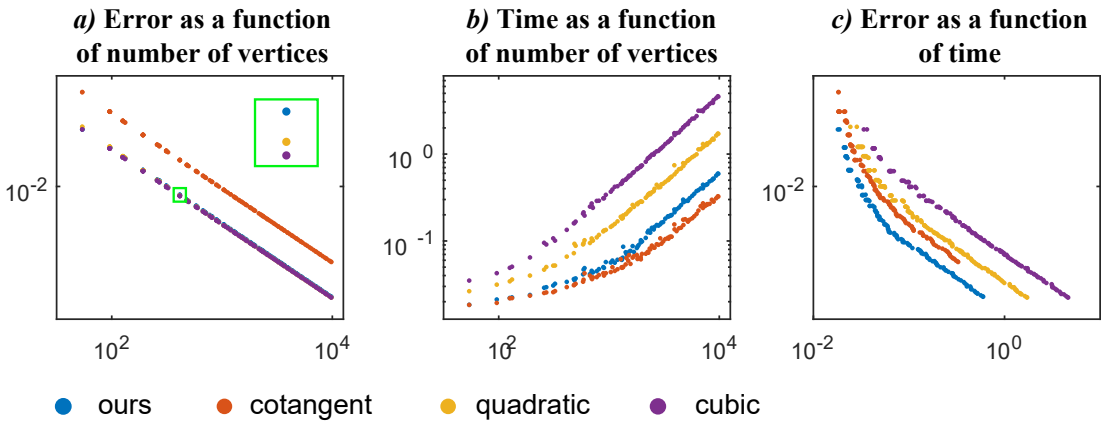

Figure 4.3: Convergence of the first 16 eigen-values of the Laplace-Beltrami operator on the sphere, on a regular triangulation with non-lumped mass matrices. The a') graph is a zoom on the green rectangle of graph a). Note in c) that our new discretization has the best ratio accuracy-to-complexity compared to all previous standard discretizations including quadratic and cubic finite elements.

Results Once we completed the steps 1-3 of the algorithm, we minimize the energy $E(u, v) = E_z(u) + E_{iz}(v)$ using

$$E_z(u) = u^T L\, u + 2\,\mathrm{Re}(G^* z)^T u + z^* M_z\, z$$

where z represents the propagated direction U of the cross field. Figure 4.4 shows results we obtained this framework using the solver [GO15]. Note, that our goal is not to claim better results than state-of-the-art quad meshing methods, but simply to demonstrate that our piecewise-linear vector field representation is indeed feasible for use for quad-meshing applications, which were so far limited to piecewise constant representations.

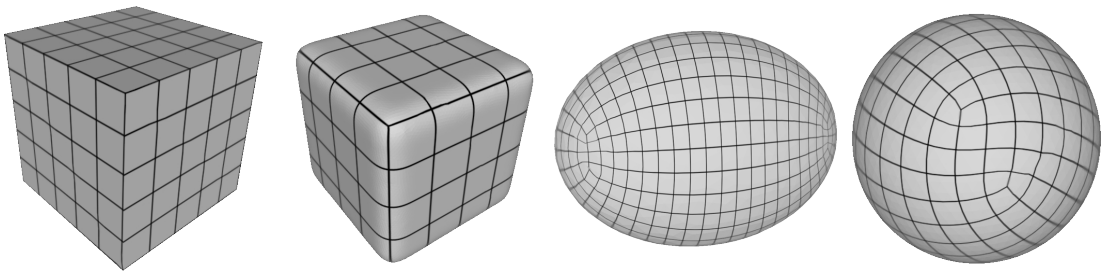

Figure 4.4: Quadrangulation on several simple meshes.

Chapter 5

Transport

Finally, for our last application, we show how our representation can be used in conjunction with the recently proposed operator representation of tangent vector fields [ABCCO13]. Specifically, we provide a discretization of the *covariant derivative* operator, and use it to simulate the parallel transport of a vector field, similarly to what was done in [AOCBC15].

5.1 Covariant derivative

If we denote z, w amd x the discrete complex values of ψ, ω and χ, we have:

$$\chi = \nabla_\omega \psi \quad \Leftrightarrow \quad M_z\, x = D_\omega z$$

with

$$(D_\omega)_{ij} = \langle\langle \psi_i, \nabla_\omega \psi_j \rangle\rangle\,,$$

where the contribution of the triangle T_{ijk} is

$$(D_\omega)_{ii} = 0 \quad , \quad (D_\omega)_{i\neq j} = -\delta_j \frac{|e_j|}{3} \operatorname{Im}(\omega_i r_{ij}) r_{ji}$$

Following [AOCBC15], we compute the *backward parallel transport* of the vector field ψ_0 along ω, using:

$$\frac{\partial}{\partial t} M_z \psi(p,t) + D_w\, \psi(p,t) = 0, \quad \psi(p,0) = \psi_0,$$

whose closed-form solution is then given by:

$$\psi(t,p) = \exp(t M_z^{-1} D_\omega)\psi_0(p).$$

We implemented this equation on the sphere with a simple turning vector field transported along itself. To add some regularization, as the solution quickly diverges, we smooth the resulting vector field at every step, using the connection Laplacian (similarly

to [AOCBC15]). Figure 5.1 and the accompanying video shows the resulting transported vector field. Note, that we get results comparable to [AOCBC15], without the need to transport the norm of the vector field separately as has been done there.

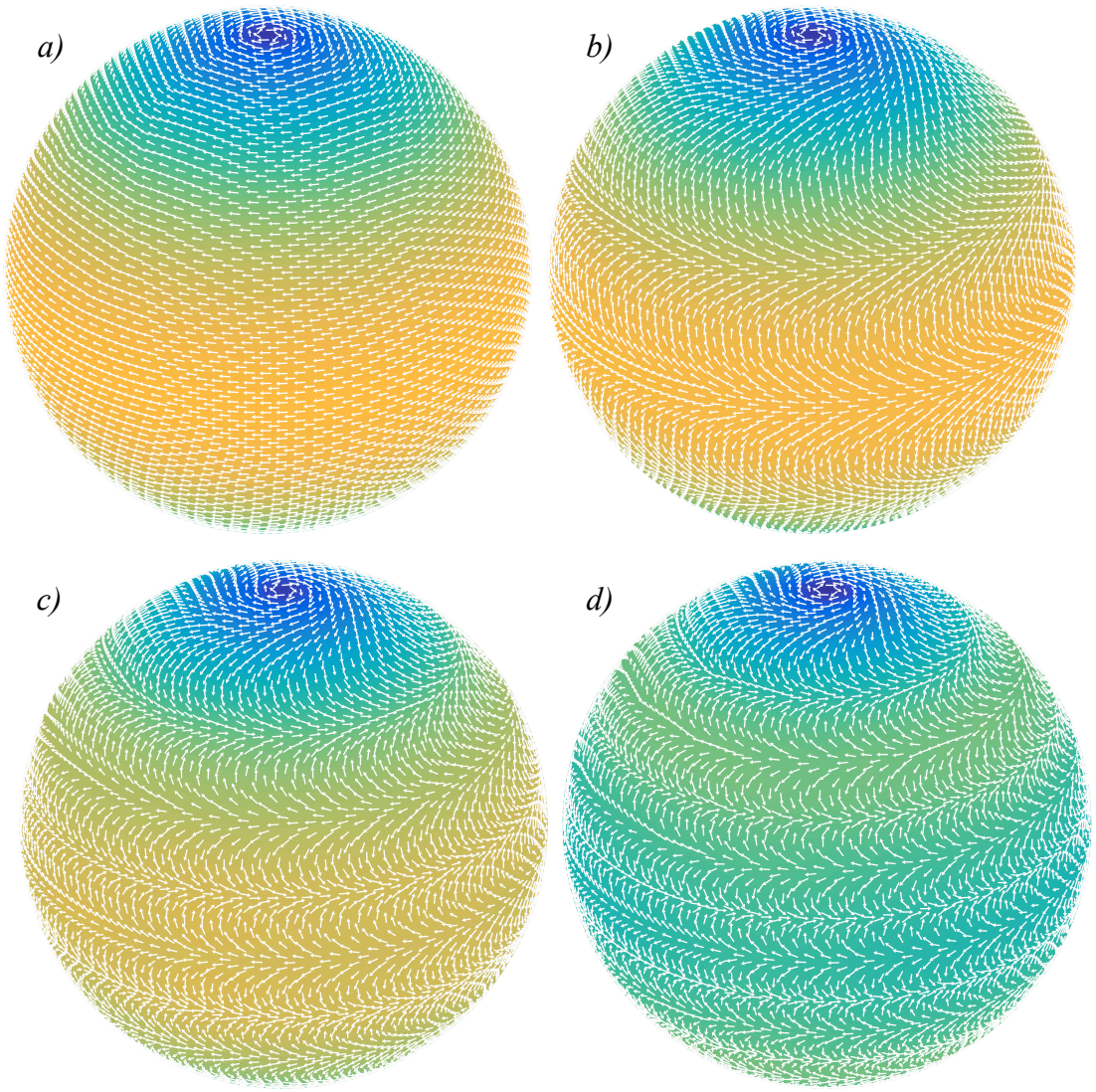

Figure 5.1: Parallel transport using the covariant derivative. We transported the vectors of the first illustration 1, 2 and 3 times around the globe along their circle latitude.

Chapter 6

Relation to other discretizations

In this section we provide a detailed discussion of the main differences between our method and existing representations.For a detailed survey on how to compute the operators in each discretization, we refer the reader to [dGDT15].

6.1 Face-Based Piecewise constant vector fields

As mentioned in the introduction, piecewise constant vector fields probably provide the most popular discretization. Using the same complex construction of section 2.2, it is straightforward to generalize it to n-vector fields. Our main advantage with respect to this approach is the robust computation of derivatives, such as the connection Laplacian and the covariant derivative.

6.2 Vertex-based sampling

Vertex based sampling was first introduced in [ZMT06] and extended by [KCPS13] and [dGDT15]. Sampling tangent vector fields on vertices might seem natural, especially in the plane where the linear basis can be directly constructed. However, on curved surfaces, the tangent plane at a vertex is not well-defined, which complicates the definition of a linear vertex-based vector field basis. The original approach to define the tangent space at the vertex used a geodesic polar map. Based on the idea presented in [Cra13], we propose an alternative, yet equivalent, point of view which we find more intuitive.

The inflation-deflation method. Consider the vertex v of the tetrahedron of figure 6.1. Contrary to what we did in figure 2.1, it is impossible here to directly define a coherent intrinsic notion of tangent space by just unfolding. In order to compensate for the angle deficiency, we need to: (1) Unfold and inflate the mesh around the vertex; (2) Define the basis on the inflated mesh; (3) Deflate and fold the mesh back to its original structure; and (4) Multiply by the hat basis function. This procedure is illustrated in figure 6.2 and the final result on the folded mesh is shown in figure 6.3 (a).

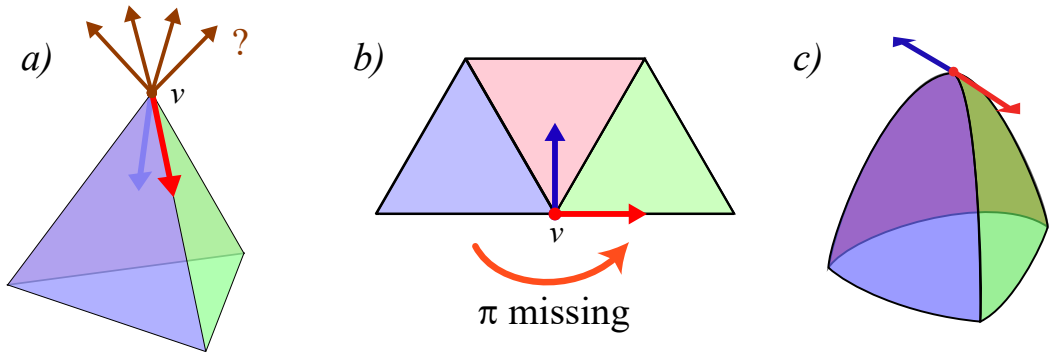

Figure 6.1: Vertex unfolding and inflation of a tetrahedron mesh. a) No clear definition of tangent space. b) Vertex unfolding of the tetrahedron. c) Inflated mesh. Compare to figure 2.1 where the unfolding was done without distortion. Our tangent space should encode the fact that the red and blue vectors are opposite to each others in a), which b) does not encode but c) does.

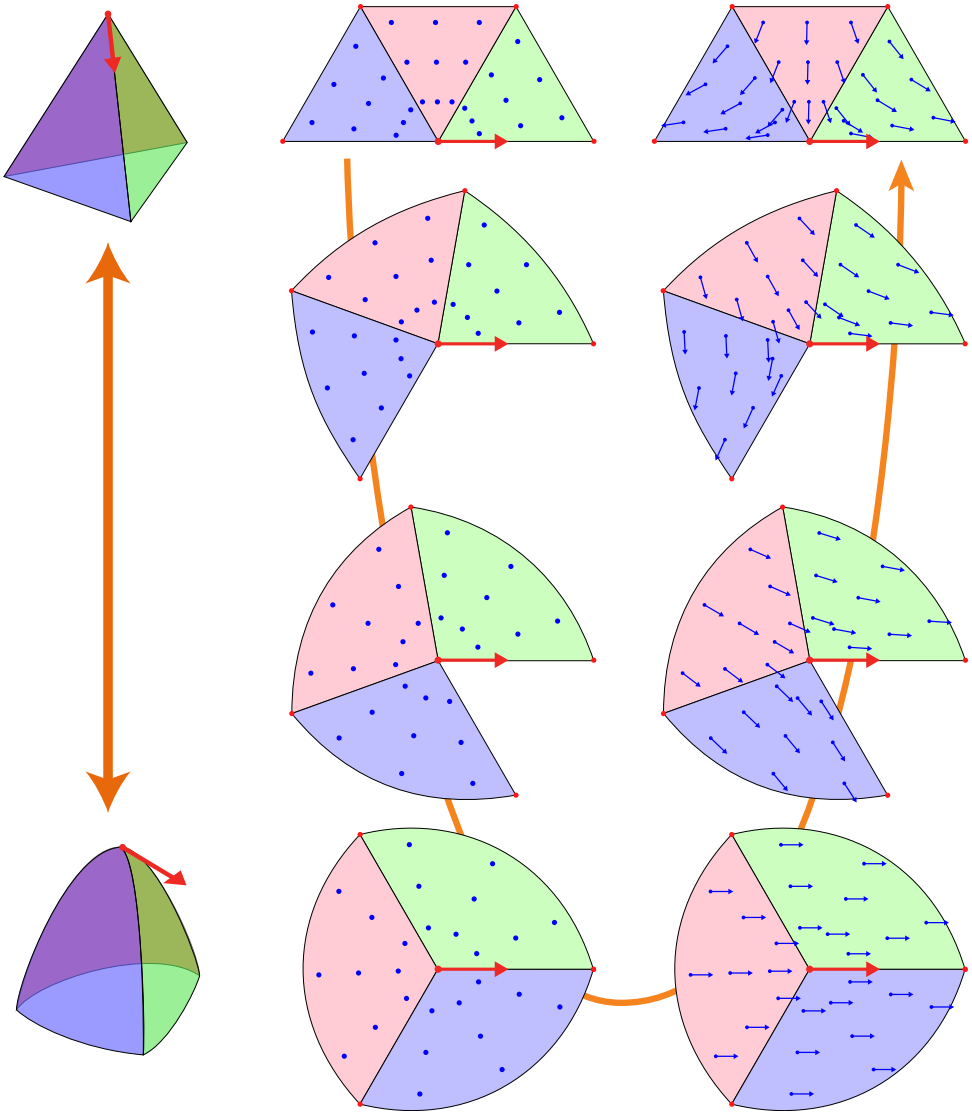

Figure 6.2: The inflation-deflation procedure to define the vertex based basis.

Closed form expression Although neither [ZMT06], [KCPS13] nor [dGDT15] provided a closed form expression of the basis, we propose the following concise expression (App. B.5):

$$\psi_i(p) = \varphi_i(p)\, e^{-\imath \arg(p)(s_i - 1)},$$

where $p \in \mathbb{C}$ is the point on the unfolded mesh of figure 6.1 b), i is the index of the vertex v, and s_i is the *angle deficiency rescaling factor* $s_i = \frac{2\pi}{\sum_{T \ni v_i} \alpha_i}$.

Treating the discontinuity The produced vector field is not linear, and is not even continuous as illustrated in figure 6.3. The discontinuity of the resulting basis on vertices prevents a direct computation of the Dirichlet energy on a triangle, as the derivative diverges near the vertex. [ZMT06] alleviated the problem of the divergent Jacobian by cutting off an arbitrary small triangular region at the problematic vertices (see the dark green area in figure 6.3 b)). [KCPS13] proposed a more rigorous approach by computing the integrals on the curved inflated mesh (see figure 6.3 c)) which drastically complicates the integration formulas.

Conversion to and from If conversion to or from the vertex representation is needed, we provide here a simple method. Because of the complexity of the vertex based basis, instead of proposing the complicated formulas resulting from the variational formulation, we propose a simple average as a conversion: average of adjacent edge values to convert to vertices, and average the 2 vertex values to convert to edges. To do so, we only need to compute r_{ve} the transport coefficient from a vertex v to the midedge point of an adjacent edge e (remember that $r_{ev} = \bar{r}_{ve}$). If we note θ the oriented angle between $\vec{X}_v$ and $\vec{X}_e$ on the unfolded mesh, we have $r_{ve} = e^{-\imath s\theta}$ where s is the angle deficiency rescaling factor of v.

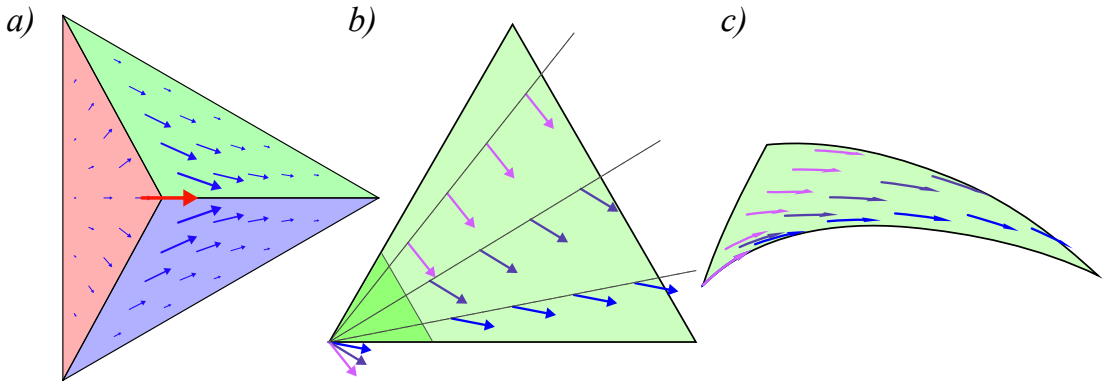

Figure 6.3: a) Top view of the tetrahedron on which we plotted the resulting basis of the inflation-deflation procedure. b) Focus on the green face to highlight the discontinuity of the produced vector field at the vertex. c) The curved green triangle used by [KCPS13] to bypass the discontinuity of (b). The triangle is obtained by pushing the curvature of the 3 adjacent vertices to the triangle.

6.3 Discrete one-forms

Discrete one-forms [Hir03] require storing coefficients on edges, thus several analogies to our method can be drawn. To understand the main differences between the two frameworks, we will show that discrete one-forms can only represent the specific subset of divergence free vector fields.

6.3.1 Analogies between the two constructions.

A discrete one-form provides a proxy to a vector field, by storing on every edge the integral of the dot product of the vector field and the unit edge direction. Let e be an oriented edge of $\mathcal{M}$, the stored coefficient c_e is $c_e = \int_e \langle w(p), \vec{X}_e \rangle dp$, where $\vec{X}_e$ is the unit norm vector in the direction of e. Note the difficulty of handling n-vector fields with this representation, as the technique to raise a complex number to the power n to alleviate the direction uncertainty is no longer available.

The analogy with our construction is now clear: if z represents our complex coefficient, then (3.2) implies $c_e = |e| \operatorname{Re}(z_e)$. Since the curl operation directly uses (3.2), the curl operation from section 3.3 is the exactly the curl operator of DEC as both are computed using the circulation theorem.

Interestingly, one could also work with the orthogonal version of DEC, by storing the integral of the orthogonal component on the edge, to get the similar conjugate results. To summarize, if we denote by $\mathrm{DEC}_\parallel$ and $\mathrm{DEC}_\perp$ the regular DEC framework and its orthogonal conjugate, we get:

$$\begin{aligned} c_\parallel + \imath c_\perp &= |e|\, z \\ |t| \left(\mathrm{curl}_\parallel - \imath\, \mathrm{div}_\perp \right) &= Kz, \end{aligned}$$

where $|e|$, and $|t|$ are the diagonal mass matrices of edge lengths and triangle areas.

6.3.2 Why our construction is not just $\mathrm{DEC}_\parallel + \imath \mathrm{DEC}_\perp$

The previous identities might mislead the reader into thinking that our construction can be assembled by using the two different DEC versions. However, replacing real coefficients by complex ones in $\mathrm{DEC}_\parallel$ does not work, mainly because of the interpolation basis used both in DEC and its orthogonal conjugate. Thus, building a new basis as we have done in section 2 cannot be avoided.

The Whitney basis Let us focus on $\mathrm{DEC}_\parallel$. The Whitney basis ψ_e is the reconstruction basis associated to the coefficient c_e of an edge e. ψ_e can be easily computed with the hat basis functions:

$$\psi_{e_{ij}} = \varphi_i \nabla \varphi_j - \varphi_j \nabla \varphi_i,$$

where v_i and v_j are the 2 vertices of $e = e_{ij}$ (see figure 6.4 for the construction). To get a better intuition, we propose a new formula involving complex numbers. In a triangle T_{ijk} (figure 2.2), we have

$$\psi_k = \delta_k \, \psi_{ij} = \frac{\imath \, \delta_k}{2A_T}(z - v_k), \tag{6.1}$$

where the term $(z - v_k)$ accounts for the singularity at the vertex v_k, $\imath$ for the vortex shape of the singularity, and δ_k for the counterclockwise or clockwise rotation of the vector field.

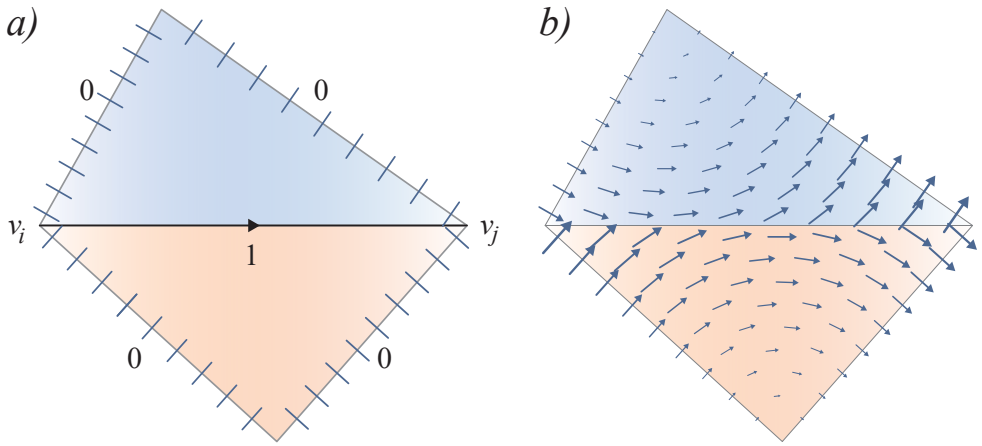

Figure 6.4: Whitney basis 1-forms (exceptionally, vectors are plotted at their mid points). To construct the basis for the edge e_{ij}, we set $c_{ij} = 1$ and $c = 0$ on the others edges. Since the integral of the dot product $\langle u_e, \psi_{ij} \rangle$ is null on the exterior edges, it is natural (although not necessary) to assume that ψ_{ij} is orthogonal to the exterior edges, hence the orthogonal constraints in a) and the resulting Whitney basis in b).

Circular vector fields From equation (6.1) it is straightforward to show that any non constant vector field produced from Whitney 1-forms on a single triangle, can be written as:

$$\psi(z) = \imath\lambda(z - \omega), \tag{6.2}$$

where $\omega \in \mathbb{C}$ is the singularity location of the vector field, and $\lambda \in \mathbb{R}$ is the oriented magnitude of the vector field. Therefore the Whitney basis only produces divergence free vector fields. From (6.1) and (6.2), one can also prove that the singularity of a combination of Whitney 1-forms is the barycenter of the original singularities. In other words, if $\psi = a \, \psi_1 + b \, \psi_2$, with $a, b \in \mathbb{R}$, we get $\psi(z) = \imath\lambda \, (z - \omega)$ with

$$\lambda = \frac{\alpha + \beta}{2A_T} \quad , \quad \omega = \frac{\alpha \, v_1 + \beta \, v_2}{\alpha + \beta}$$

where $\alpha = a \, \delta_1$ and $\beta = b \, \delta_2$ are the re-oriented coefficients. See figure 6.5 for an illustration.

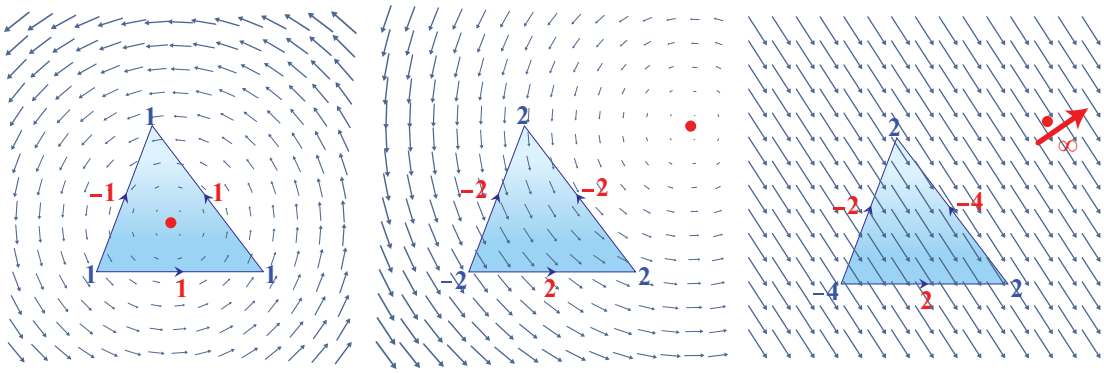

Figure 6.5: The DEC reconstruction vector fields are always circular, with the singularity being the barycenter of the vertices with the re-oriented edge coefficients as weights (blue numbers). Note that the singularity might go to infinity yielding a constant vector field.

Complex Whitney 1-forms ? It is therefore straightforward to show that allowing the discrete one-form coefficient to be complex (which is equivalent to adding the Whitney basis of $\mathrm{DEC}_\perp$ as an imaginary part) would not be enough to recover the whole space of linear vector fields. The general form of such vector fields would still be of the form (6.2), yet with $\lambda \in \mathbb{C}$, which only covers a 4 dimensional space, whereas the space of linear vector fields is of dimension 6.

Chapter 7

Conclusion and further work

We proposed a new discretization of tangent vector fields on triangle meshes, which is both simple to implement and widely applicable. We additionally showed how to derive various operators which are required in the context of two important applications - vector field design, and quadrangular remeshing. We believe that our representation could be a viable alternative to the widely used piecewise-constant vector fields in other applications as well.

One piece which is missing from our discretization, is a discrete Hodge decomposition which is available for the face-based representation and the discrete one-form representation. However, due to the analogies between our framework and DEC, we are hopeful that future research would uncover a Hodge decomposition for our representation as well. Finally, we believe that our representation brings us somewhat closer to the ultimate goal of providing a complete and unified framework for handling discrete functions and discrete vector fields in geometry processing applications.

Appendix A

Computation of matrix indices

Let us note $\langle\langle \psi_i, \psi_j \rangle\rangle_{ijk}$ the hermitian product restricted to the triangle T_{ijk}, and $\langle \cdot, \cdot \rangle_r$ the real part of $\langle \cdot, \cdot \rangle$. Let us also note u_i the complex coefficients of $\vec{X}_i$ in the basis $\vec{X}_T$, h_i the height of the vertex v_i in a triangle T_{ijk} and A_{ijk} the area of T_{ijk}.

A.1 Some general identities

Before starting the computation with specific operators, here are some results which we will use in all the following computations. In a triangle T, we have for all i, j:

$$\int_T \phi_i = \frac{A_T}{3} \qquad \int_T \phi_i^2 = \frac{A_T}{3} \qquad \int_T \phi_i \phi_j = 0 \qquad \nabla \phi_i = -2\imath\delta_i \frac{1}{h_i} \vec{X}_i \qquad \langle u_i, u_j \rangle = r_{ji}$$

Note also that $2A_{ijk} = |e_i|\, h_i$.

A.2 Mass matrix M_z

We want to compute $M_z = \langle\langle \psi_i, \psi_j \rangle\rangle_{ij}$. Note that $(M_z)_{ij} \neq 0$ only if i and j are adjacent. In T_{ijk}, we have

$$\langle\langle \psi_i, \psi_j \rangle\rangle_{ijk} = \int_{T_{ijk}} \langle \phi_i u_i, \phi_j u_j \rangle = \langle u_i, u_j \rangle \int_{T_{ijk}} \phi_i \phi_j = 0$$
$$\langle\langle \psi_i, \psi_i \rangle\rangle_{ijk} = \int_{T_{ijk}} \langle \phi_i u_i, \phi_i u_i \rangle = \int_{T_{ijk}} \phi_i^2 = A_{ijk}/3$$

A.3 Connection Laplacian L_c

Note that $\nabla \psi_i$ is constant on each triangle. In the plane, we have

$$\nabla \psi_i = \nabla(\phi_i u_i) = \begin{pmatrix} \frac{\partial \phi_i}{\partial x} u_{i,x} & \frac{\partial \phi_i}{\partial x} u_{i,y} \\ \frac{\partial \phi_i}{\partial y} u_{i,x} & \frac{\partial \phi_i}{\partial y} u_{i,y} \end{pmatrix} = \begin{pmatrix} \frac{\partial \phi_i}{\partial x} u_i \\ \frac{\partial \phi_i}{\partial y} u_i \end{pmatrix} = \nabla \phi_i \, u_i$$

where the notation $\nabla\phi_i\, u_i \in \mathbb{C}^2$ is maybe abusive, but we leave the reader check that the calculus are the same when dealing with full matrices in $\mathbb{R}^{2\times 2}$.

We can now derive the expressions. We have:

$$\begin{aligned}\langle\langle\nabla\psi_i, \nabla\psi_i\rangle\rangle_{ijk} &= A_{ijk}\left\|\nabla\phi_i\, u_i\right\|^2 \\ &= A_{ijk}\left\|\nabla\phi_i\right\|^2 \\ &= \left|e_i\right|^2 / A_{ijk}\end{aligned}$$

and

$$\begin{aligned}\langle\langle\nabla\psi_i, \nabla\psi_j\rangle\rangle_{ijk} &= A_{ijk}\left\langle\nabla\phi_i\, u_i, \nabla\phi_j u_i\right\rangle \\ &= A_{ijk}\,\overline{u_i}\left\langle\nabla\phi_i, \nabla\phi_j\right\rangle_r u_j \\ &= A_{ijk}\frac{2}{h_i}\frac{2}{h_j}\cos(\pi-\alpha_k)\overline{u_i}\, u_j \\ &= -\frac{\left|e_i\right|\left|e_j\right|}{A_{ijk}}\cos(\alpha_k) r_{ji}.\end{aligned}$$

A.4 Gradient operator G

We want to compute $\langle\langle\psi_i, \nabla\varphi_j\rangle\rangle_{ijk}$, but note first that one should not get confused between i which is an index for the complex edge basis ψ_i and j which is an index of the function basis φ_i.

We have:

$$\begin{aligned}\langle\langle\psi_i, \nabla\varphi_j\rangle\rangle_{ijk} &= \int_{T_{ijk}}\left\langle\phi_i u_i, \imath\delta_j\frac{1}{h_j}u_j\right\rangle = \frac{A_{ijk}}{3}\left\langle u_i, \frac{1}{h_j}\imath\delta_j u_j\right\rangle \\ &= \imath\delta_j\frac{\left|e_j\right|}{6} r_{ji} \\ \langle\langle\psi_i, \nabla\varphi_i\rangle\rangle_{ijk} &= \int_{T_{ijk}}\left\langle\phi_i u_i, \imath\delta_i\frac{1}{h_i}u_i\right\rangle = \frac{A_{ijk}}{3}\left\langle u_i, \frac{1}{h_i}\imath\delta_i u_i\right\rangle \\ &= \imath\delta_i\frac{\left|e_i\right|}{6}\end{aligned}$$

A.5 Covariant derivative

We want to compute $(D_\omega)_{ij} = \langle\langle\psi_i, \nabla_\omega\psi_j\rangle\rangle$. Let us first compute $\nabla_\omega\psi_j$ on T_{ijk}. We have

$$\nabla_\omega\psi_j = \nabla_{\omega_i\psi_i+\omega_j\psi_j+\omega_k\psi_k}\psi_j = \nabla_{\omega_i\psi_i}\psi_j + \nabla_{\omega_j\psi_j}\psi_j + \nabla_{\omega_k\psi_k}\psi_j.$$

We have

$$\nabla_{\omega_i\psi_i}\psi_j = \nabla_{\omega_i\phi_i u_i}(\phi_j u_j) = \phi_i\left(\nabla_{\omega_i u_i}\phi_j\right) u_j = \phi_i\left\langle\nabla\phi_j, \omega_i u_i\right\rangle_r u_j.$$

So, putting it together, we get:

$$\begin{aligned}
\langle\langle \psi_i, \nabla_\omega \psi_j \rangle\rangle_{ijk} &= \int_T \left\langle \psi_i, \nabla_{\omega_i \psi_i + \omega_j \psi_j + \omega_k \psi_k} \psi_j \right\rangle \\
&= \int_T \left\langle \psi_i, \sum_k \nabla_{\omega_k \psi_k} \psi_j \right\rangle \\
&= \sum_k \int_T \left\langle \psi_i, \phi_k \left\langle \nabla \phi_j, \omega_k u_k \right\rangle_r u_j \right\rangle \\
&= \sum_k \int_T \phi_i \phi_k \left\langle \nabla \phi_j, \omega_k u_k \right\rangle_r \left\langle u_i, u_j \right\rangle \\
&= \sum_k \left\langle \nabla \phi_j, \omega_k u_k \right\rangle_r \left\langle u_i, u_j \right\rangle \int_T \phi_i \phi_k \\
&= \left\langle \nabla \phi_j, \omega_i u_i \right\rangle_r \left\langle u_i, u_j \right\rangle \int_{T_{ijk}} \phi_i \phi_i + 0 \\
&= \left\langle \nabla \phi_j, \omega_i u_i \right\rangle_r r_{ji} \frac{A_T}{3} \\
&= \frac{A_T}{3} \operatorname{Re}\left(-2\delta_j \frac{1}{h_j} \overline{\imath u_j}\, \omega_i u_i \right) r_{ji} \\
&= -\delta_j \frac{2A_T}{3h_j} \operatorname{Re}\left(-\imath\, \overline{u_j}\, \omega_i u_i \right) r_{ji} \\
&= -\delta_j \frac{|e_j|}{3} \operatorname{Im}(\omega_i r_{ij}) r_{ji}
\end{aligned}$$

If we consider the case $j = i$, we get:

$$\langle\langle \psi_i, \nabla_\omega \psi_j \rangle\rangle_{ijk} = -\delta_i \frac{|e_i|}{3} \operatorname{Im}(\omega_i)$$

Therefore on a closed mesh, if we note T_{ijk} and T_{ijl} the 2 adjacent triangles to e_{ij}, we have:

$$\begin{aligned}
\langle\langle \psi_i, \nabla_\omega \psi_j \rangle\rangle &= \langle\langle \psi_i, \nabla_\omega \psi_j \rangle\rangle_{ijk} + \langle\langle \psi_i, \nabla_\omega \psi_j \rangle\rangle_{ijl} \\
&= -\left(\delta_i^{ijk} + \delta_i^{ijl} \right) \frac{|e_i|}{3} \operatorname{Im}(\omega_i) \\
&= 0
\end{aligned}$$

since an edge has the right orientation with one its adjacent triangle and the wrong orientation with the other one.

Note however that on a non closed mesh, the diagonal term would be non zero for the boundary edges.

Appendix B

Other proofs

B.1 Integral property

Note that $\phi_i = 1$ on e_i. We then have:

$$\int_{e_i} \psi_i(p)dp = \int_{e_i} \phi_i(p)\vec{X}_i dp = \left(\int_{e_i} 1dp\right)\vec{X}_i = |e_i|\,\vec{X}_i$$
$$\int_{e_i} \psi_j(p)dp = \int_{e_i} \phi_j(p)\vec{X}_j dp = \left(\int_{e_i} \phi_j(p)dp\right)\vec{X}_j = 0$$

because ϕ_j is symmetric on e_i.

B.2 Singularity remeshing formula

We are looking for a, b such that $S = v_1 + a\,(v_2 - v_1) + b\,(v_3 - v_1)$. We know that S being a singularity implies

$$\sum_{i=1,2,3} \psi_i(S) = 0$$

We have then

$$0 = \sum_{i=1,2,3} \psi_i(S) = \sum_{i=1,2,3} z_i X_i \phi_i(S) = \sum_{i=1,2,3} z_i r_{i1} \phi_i(a,b)$$

with

$$\phi_1(a,b) = 1 - 2\varphi_i(a,b) = 1 - 2(1-a-b) = 2a + 2b - 1$$
$$\phi_2(a,b) = 1 - 2\varphi_i(a,b) = 1 - 2a$$
$$\phi_3(a,b) = 1 - 2\varphi_i(a,b) = 1 - 2b$$

Note that we arbitrarily decided to convert the vector values in the X_1 basis, thus the r_{i1}.

B.3 Tangential component of our gradients

Let $e = e_{ij}$ be the edge between the vertices v_i and v_j, and to simplify notations. Let f be a function on vertices and let us note t_e the tangential component of its gradient on e. Remember that M_z is real and diagonal, and that f is real.

We have:

$$t_e = \mathrm{Re}((M_z^{-1}Gf)_e) = \frac{1}{(M_z)_e}\sum_{k=1}^{n_v}\mathrm{Re}(G_{e,k})f_k$$

Let us call T_{ijk} and T_{ijl} the 2 adjacent triangles to e_{ij}. We have:

$$\begin{aligned}
\sum_{k=1}^{n_v}\mathrm{Re}(G_{e,k})f_k &= \frac{A_{ijk}}{3}\left(f_i\,\mathrm{Re}\,\langle u_{ij},\nabla\varphi_i\rangle + f_j\,\mathrm{Re}\,\langle u_{ij},\nabla\varphi_j\rangle + f_k\overbrace{\mathrm{Re}\,\langle u_{ij},\nabla\varphi_k\rangle}^{=0}\right) + \frac{A_{ijl}}{3}... \\
&= \frac{A_{ijk}}{3}\left(-f_i\frac{\sin\alpha_j}{h_i} + f_j\frac{\sin\alpha_i}{h_j}\right) + \frac{A_{ijl}}{3}... \\
&= \frac{A_{ijk}}{3}\left(-f_i\frac{h_k/|e_i|}{h_i} + f_j\frac{h_k/|e_j|}{h_j}\right) + \frac{A_{ijl}}{3}... \\
&= \frac{h_k}{6}(-f_i+f_j) + \frac{h_l}{6}(-f_i+f_j)... \\
&= \frac{h_k+h_l}{6}(f_j-f_i)
\end{aligned}$$

and we also have

$$(M_z)_{e_{ij}} = (A_{ijk}+A_{ijl})/3.$$

Combining the two results, we get:

$$\begin{aligned}
t_e &= \frac{3}{A_{ijk}+A_{ijl}}\frac{h_k+h_l}{6}(f_j-f_i) \\
&= \frac{6}{|e_{ij}|\,h_k + |e_{ij}|\,h_l}\frac{h_k+h_l}{6}(f_j-f_i) \\
&= \frac{f_j-f_i}{|e_{ij}|}.
\end{aligned}$$

B.4 Gradient adjoint

Let us focus in the discrete matrix case first. We have:

$$\begin{aligned}
\langle\langle\psi,\nabla f\rangle\rangle &= z^* M_z\left(M_z^{-1}Gf\right) \\
&= z^*Gf
\end{aligned}$$

Now we want to transform this matrix expression to be a dot product of vertex functions, so we artificially insert the vertex mass matrix and develop:

$$\begin{aligned}
\langle\langle \psi, \nabla f \rangle\rangle &= z^* G M_v^{-1} M_v f \\
&= (M_v^{-1} G^* z)^* M_v f \\
&= \mathrm{Re}((M_v^{-1} G^* z)^* M_v f) + \imath \,\mathrm{Im}((M_v^{-1} G^* z)^* M_v f) \\
&= M_v^{-1} \,\mathrm{Re}((G^* z)^*) M_v f + \imath M_v^{-1} \,\mathrm{Im}((G^* z)^*) M_v f \\
&= M_v^{-1} \,\mathrm{Re}(G^* z)^T M_v f - \imath M_v^{-1} \,\mathrm{Im}(G^* z)^T M_v f
\end{aligned}$$

In the smooth case, we know from the Green formula (also called Stokes formula) that

$$\begin{aligned}
\langle\langle \mathrm{div}\, \psi, f \rangle\rangle &= -\langle\langle \psi, \nabla f \rangle\rangle \\
\langle\langle \mathrm{curl}\, \psi, f \rangle\rangle &= \langle\langle J\psi, \nabla f \rangle\rangle
\end{aligned}$$

Therefore, by identification, we get

$$\begin{aligned}
\mathrm{div}\, z &= -M_v^{-1} \,\mathrm{Re}(G^* z) \\
\mathrm{curl}\, z &= -M_v^{-1} \,\mathrm{Im}(G^* z)
\end{aligned}$$

B.5 Inflation-Deflation closed form expression

Let s be the angle defect rescaling factor, and let $z = ae^{\imath\theta}$ be a point inside the unfolded mesh. The inflated point point z_i is

$$z_i = ae^{\imath s\theta}$$

On all the inflated points we then associate the same basis vector $q(z) = q$. When z_i deflates to its deflated version $z_d = z$, the same transformation is applied to $q(z)$ which becomes $q_d(z)$ what we want. Note that the angle between the point z and $q(z)$ stays the same while deflating. The norm being also preserved, we have a constant ratio

$$\frac{z_i}{q_i} = \frac{z_d}{q_d}$$

Therefore

$$q_d = q_i \frac{z_d}{z_i} = q_i \frac{ae^{\imath\theta}}{a\imath s\theta} = q_i e^{\imath\theta(1-s)}$$

Remember that q_i is constant. We can choose $q_i = 1$ for simplicity as the choice of the basis is arbitrary. Multiplying by the hat function yields

$$q_d(z) = \varphi(z) e^{\imath \arg(z)(1-s)}$$

which is our formula.

Bibliography

[ABCCO13] Omri Azencot, Mirela Ben-Chen, Frédéric Chazal, and Maks Ovsjanikov. An operator approach to tangent vector field processing. In *Computer Graphics Forum*, volume 32, pages 73–82. Wiley Online Library, 2013.

[AMH11] Awad H Al-Mohy and Nicholas J Higham. Computing the action of the matrix exponential, with an application to exponential integrators. *SIAM journal on scientific computing*, 33(2):488–511, 2011.

[AOCBC15] Omri Azencot, Maks Ovsjanikov, Frédéric Chazal, and Mirela Ben-Chen. Discrete derivatives of vector fields on surfaces–an operator approach. *ACM Transactions on Graphics (TOG)*, 34(3):29, 2015.

[AVW+15] Omri Azencot, Orestis Vantzos, Max Wardetzky, Martin Rumpf, and Mirela Ben-Chen. Functional thin films on surfaces. In *Proceedings of the 14th ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 137–146. ACM, 2015.

[AWO+14] Omri Azencot, Steffen Weißmann, Maks Ovsjanikov, Max Wardetzky, and Mirela Ben-Chen. Functional fluids on surfaces. In *Computer Graphics Forum*, volume 33, pages 237–246. Wiley Online Library, 2014.

[BKP+10] M. Botsch, L. Kobbelt, M. Pauly, P. Alliez, and B. Levy. *Polygon Mesh Processing*. Ak Peters Series. Taylor & Francis, 2010.

[BLP+12] D. Bommes, B. Lévy, N. Pietroni, E. Puppo, C. Silva, M. Tarini, and D. Zorin. State of the art in quad meshing. In *Eurographics STARS*, 2012.

[BZK09] David Bommes, Henrik Zimmer, and Leif Kobbelt. Mixed-integer quadrangulation. *ACM Trans. Graph.*, 28(3):77:1–77:10, July 2009.

[Cra13] Keenan Crane. Youtube presentation on the paper "Globally Optimal Direction Fields", https://youtu.be/uU0iZSN4Hgo, 2013.

[dGDT15] Fernando de Goes, Mathieu Desbrun, and Yiying Tong. Vector field processing on triangle meshes. *SIGGRAPH Asia Course Notes*, 2015.

[dGLB+14] Fernando de Goes, Beibei Liu, Max Budninskiy, Yiying Tong, and Mathieu Desbrun. Discrete 2-tensor fields on triangulations. In *Computer Graphics Forum*, volume 33, pages 13–24. Wiley Online Library, 2014.

[EBCK13] Hans-Christian Ebke, David Bommes, Marcel Campen, and Leif Kobbelt. QEx: Robust quad mesh extraction. *ACM Trans. Graph.*, 32(6):168:1–168:10, November 2013.

[GO15] Inc. Gurobi Optimization. Gurobi optimizer for matlab, http://www.gurobi.com, 2015.

[Hir03] Anil N Hirani. *Discrete exterior calculus*. PhD thesis, California Institute of Technology, 2003.

[HL97] Marlis Hochbruck and Christian Lubich. On Krylov subspace approximations to the matrix exponential operator. *SIAM Journal on Numerical Analysis*, 34(5):1911–1925, 1997.

[HZ00] Aaron Hertzmann and Denis Zorin. Illustrating smooth surfaces. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 517–526. ACM Press/Addison-Wesley Publishing Co., 2000.

[KCPS13] Felix Knöppel, Keenan Crane, Ulrich Pinkall, and Peter Schröder. Globally optimal direction fields. *ACM Transactions on Graphics (TOG)*, 32(4):59, 2013.

[KCPS15] Felix Knöppel, Keenan Crane, Ulrich Pinkall, and Peter Schröder. Stripe patterns on surfaces. *ACM Transactions on Graphics (TOG)*, 34(4):39, 2015.

[PP03] Konrad Polthier and Eike Preuss. Identifying vector field singularities using a discrete Hodge decomposition. In *Visualization and Mathematics III*, pages 113–134. Springer, 2003.

[RVLL08] Nicolas Ray, Bruno Vallet, Wan Chiu Li, and Bruno Lévy. N-symmetry direction field design. *ACM Trans. Graph.*, 27(2):10:1–10:13, May 2008.

[SW12] A. Singer and H.-T. Wu. Vector diffusion maps and the connection laplacian. *Communications on Pure and Applied Mathematics*, 65(8):1067–1144, 2012.

[War06] Max Wardetzky. *Discrete Differential Operators on Polyhedral Surfaces–Convergence and Approximation*. PhD thesis, Freie Universitat Berlin, 2006.

[WLKT09] Li-Yi Wei, Sylvain Lefebvre, Vivek Kwatra, and Greg Turk. State of the art in example-based texture synthesis. In *Eurographics 2009, State of the Art Report, EG-STAR*, pages 93–117. Eurographics Association, 2009.

[ZMT06] Eugene Zhang, Konstantin Mischaikow, and Greg Turk. Vector field design on surfaces. *ACM Transactions on Graphics (TOG)*, 25(4):1294–1326, 2006.

של הטלת השדה הוקטורי על הקשת. הייצוג הזה הינו לינארי, ולכן פשוט, ובנוסף הוא גם מדוייק. למרות זאת, יש לו חיסרון עיקרי: בניגוד לייצוגים הקודמים של שדות וקטוריים, לא ברור כיצד ניתן לייצג בשיטה זו שדות N-Rosy. שדות N-Rosy הם הכללה של שדות וקטוריים כאשר במקום וקטור יחיד בכל נקודה הם מאפשרים לשייך N וקטורים סימטריים מעגלית. למשל, לשדה 4-Rosy יש צורה של צלב. שדות N-Rosy שימושיים למשל עבור שילוש מחדש, ועבור רינדור לא פוטוריאליסטי, בעיקר בגלל שניתן ליישר אותם עם כיווני העקמומיות בקלות, בניגוד לשדות וקטוריים כלליים.

בעבודה זו, אנחנו מציעים שיטת ייצוג פשוטה של שדות וקטוריים, חזקה לא פחות מהשיטות האחרות, שניתן להכליל אותה בקלות לייצוג של שדות N-Rosy. אנחנו מציעים לדגום את השדה הוקטורי על אמצעי הקשתות ולהשתמש באינטרפולציה לינארית כדי לייצג אותו על הפיאות. באופן מעניין, בעיית ההגדרה של המישורים המשיקים ניתנת לפתרון בקלות והשדות הוקטוריים שנוצרים הם לינארים על כל פיאה. לכן, הייצוג שלנו פשוט באותה מידה כמו בשיטת הייצוג על הפיאות, ועם זאת מגיעה לדיוק דומה לשיטת הייצוג של השדה על הקודקודים. בתזה זו, אנחנו מראים שניתן להשתמש בשיטת הייצוג שלנו בשתיים מהאפליקציות הכי פופולריות בתחום של שדות וקטוריים משיקים: עיצוב של שדות וקטורים משיקים ודגימה מחדש של המשטח עם פיאות ריבועיות. בנוסף, השתמשנו בשיטה שלנו עבור יישומים מבוססי אופרטורים שהוצעו לאחרונה. על כן, השיטה שלנו נותנת כלי פשוט וחזק לעיצוב ועיבוד של שדות וקטוריים משיקים.

תקציר

מאז צאת הסרט "טרון" בשנת 1982, תמונות הנוצרות על ידי מחשב תפסו בהדרגה חלק גדול בהפקות של סרטים מודרניים. החל מסרטי אנימציה כמו "רטטוי" ועד לסרטים עתירי אפקטים כמו "מלחמת הכוכבים", ניתן למצוא אותן כמעט בכל סרט. את ההתפתחות העצומה מ"טרון" ל "רטטוי" ניתן לייחס חלקית להתקדמויות בתחום של עיבוד גיאומטרי.

תחום העיבוד הגיאומטרי מכיל מספר תתי תחומים, שניתן למנות ביניהם עיבוד טקסטורות, אנימציה ודגימה מחדש של משטח. ברוב היישומים האלו, האלגוריתמים מסתמכים על היכולת לעצב ולנתח שדות וקטוריים, שבמקרים רבים הם משיקים למשטח. למשל, שדות וקטוריים נמצאים בשימוש עבור יצירת טקסטורות, עיצוב שיער, סימולציות של נוזלים או אפילו כקווים מנחים עבור דגימה מחדש. למרות שמאוד אינטואיטיבי להציג שדות וקטוריים משיקים, דרכים לעבד אותם נחקרו במשך שנים רבות ושיטות חדשות ממשיכות להתפתח גם עכשיו. ההמרה של משטחים למשולשים (דיסקרטיזציה) נעשית באמצעות קירובים שצריכים להיות מטופלים בזהירות. למשל, המישור המשיק לא מוגדר על הצלעות או הקודקודים של השילוש. לכן, השאלה העיקרית שנמצאת ביסוד המחקרים האלו היא: "כיצד דוגמים שדות וקטוריים על משטח בדיד?". טבעי לשאול מהן האפשרויות ומה היתרונות והחסרונות של כל אחת מהן. אנליזה זו הובילה אותנו להגדרה שונה, פשוטה יותר ועם זאת חזקה לא פחות מכל שאר השיטות לדיסקרטיזציה של שדות וקטוריים.

השיטה הראשונה והכי פופולרית היא דגימה של הוקטורים על המשולשים של המשטח. שיטת ייצוג זו כוללת וקטור אחד לכל משולש ומניחה שהשדה הוקטורי אחיד על כל משולש. מאחר והמישור המשיק של משולש מכיל את המשולש עצמו, השיטה הזו מקלה על הבעיה של הגדרת המישור המשיק ולכן כוללת ביטויים פשוטים עבור רוב האופרטורים הנפוצים. עם זאת, היא מובילה לדיוק נמוך יותר ולבעיות בהגדרת נגזרות ואנרגיות שמודדות עד כמה השדה חלק (מאחר והשדה הוקטורי קבוע בחלקים, ההגדרה שתתקבל עבור הנגזרת תהיה מאוד מוגבלת).

אופציה אחרת היא דגימה של השדה הוקטורי על הקודקודים, ושימוש באינטרפולציה לינארית כדי לקבל את הוקטורים על המשולשים. שיטת ייצוג זו מניבה דיוק רב יותר ומאפשרת חישוב ישיר של נגזרת ושל אנרגיית חלקות. לעומת זאת, הביטויים בשיטה זו יותר מורכבים, בגלל הבעיה בהגדרת המישור המשיק בכל קודקוד. מאחר וסכום הזוויות סביב כל קודקוד אינו 360 מעלות, הסימון של זווית צריך להיות מוגדר מחדש כדי להתאים לסכום. לרוע המזל, ההגדרה מחדש היא פעולה לא לינארית שמניבה ביטויים מסובכים שעלולים להרתיע משימוש בהם בהרבה אפליקציות.

האופציה האחרונה בעיבוד גיאומטרי נובעת מנקודת מבט שונה: במקום לדגום את השדה הוקטורי במקומות ספציפיים, בשיטת Discrete Exterior Calculus שומרים על כל קשת את ערך האינטגרל

המחקר בוצע בהנחייתה של פרופסור מירלה בן־חן, בפקולטה למדעי המחשב.

אני מודה לטכניון על התמיכה הכספית הנדיבה בהשתלמותי.

שדות וקטוריים משיקיים על משטחים משולשיים - שיטה מבוססת קשתות

חיבור על מחקר

לשם מילוי חלקי של הדרישות לקבלת התואר
מגיסטר למדעים במדעי המחשב

אלכסנדר ג'רבטיאן

הוגש לסנט הטכניון – מכון טכנולוגי לישראל
בשבט תשע"ו חיפה ינואר 2016

שדות וקטוריים משיקיים על משטחים משולשיים - שיטה מבוססת קשתות

אלכסנדר ג'רבטיאן