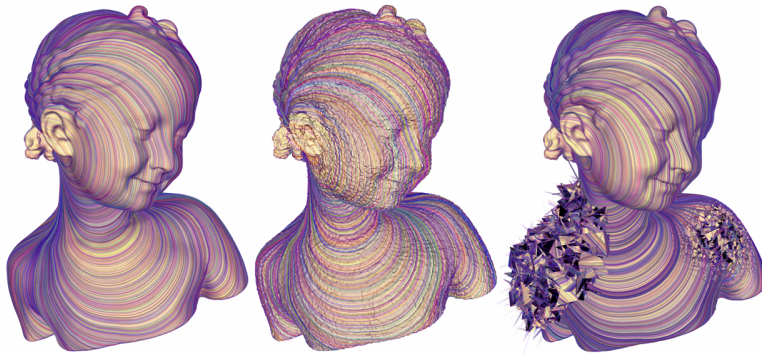
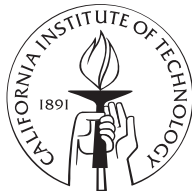


Discrete Connections for Geometry Processing

Thesis by
Keenan Crane



In Partial Fulfillment of the Requirements
for the Degree of
Master of Science



California Institute of Technology
Pasadena, California

May 28, 2010

© 2010

Keenan Crane

All rights Reserved

Acknowledgements

The meshes used to test our algorithm are courtesy of the AIM@Shape Project, the Stanford 3D Scanning Repository, Jotero, and Hugues Hoppe. Thanks to Felix Kälberer, Matthias Nieser, and Konrad Polthier for processing the Aphrodite model using QuadCover [11]. Last but not least, thanks to Peter and Mathieu for letting me wander...

Discrete Connections for Geometry Processing

by

Keenan Crane

In Partial Fulfillment of the
Requirements for the Degree of
Master of Science

Abstract

Connections provide a way to compare local quantities defined at different points of a geometric space. This thesis develops a discrete theory of connections that naturally leads to practical, efficient numerical algorithms for geometry processing. Our formulation is motivated by real-world applications where meshes may be noisy or coarsely discretized. Further, because our discrete framework closely parallels the smooth theory, we can draw upon a huge wealth of existing knowledge to develop and interpret mesh processing algorithms.

The main contribution of this thesis is a new algorithm for computing *trivial* connections on discrete surfaces that are as smooth as possible everywhere but on a set of isolated singularities of given index. A connection is represented via an angle associated with each dual edge, *i.e.*, a discrete angle-valued *1-form*. These angles are determined by the solution to a linear system, and are globally optimal in the sense that they describe the trivial connection closest to Levi-Civita among all solutions with the prescribed set of singularities. Relative to previous methods our algorithm is surprisingly simple, and can be implemented using standard operations from mesh processing and linear algebra. The solution can be used to construct rotationally symmetric direction fields with a prescribed set of singularities and directional constraints, which are essential in applications such as quadrilateral remeshing and texture synthesis.

Contents

1	Introduction	1
1.1	Motivation	1
1.2	Previous Work	3
2	Discrete Connections	5
2.1	Connections	5
2.2	Holonomy	8
2.3	Discrete Connections	9
2.4	Discrete Holonomy	10
3	Algorithm	12
3.1	Setup	12
3.2	Basis Cycles	13
3.3	Angle Defects	15
3.4	Singularities	15
3.5	Optimization	17
3.6	Area Weights	18
3.7	Surfaces with Boundary	18
3.8	Direction Fields	19
3.9	Directional Constraints	19
4	Results	21
4.1	Performance	21
4.2	Robustness	23

5	Discussion	30
5.1	Connections on Surfaces	30
5.2	Trivial Connections	31
5.3	Singularities	35
5.4	Summary	36
6	Conclusion	38

Chapter 1

Introduction

1.1 Motivation

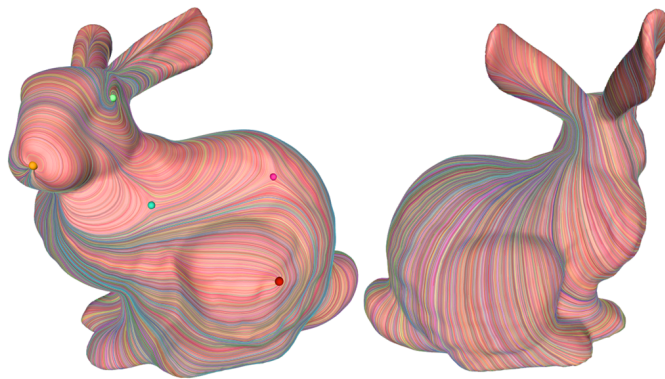


Figure 1.1: Discrete connections can be used to compute a field with singularities precisely where desired (left) and nowhere else (right).

Our framework for discrete connections is motivated by the following question: how does one construct a direction field on a discrete surface that is in some sense as smooth as possible? In general we cannot find a field that is smooth everywhere – in particular, the Poincaré index theorem requires that any direction field on a given surface satisfies

$$\sum_i \text{index}(v_i) = \chi,$$

i.e., the winding numbers or *indices* of the field around singular points v_i must add up to the Euler characteristic χ . We therefore seek a direction field where

1. singularities have the prescribed index and location, and
2. integral curves are as close to straight (geodesic) as possible.

Figure 1.1 illustrates this idea: given some prescribed set of singularities (represented by colored dots), we would like to construct a field that has singularities at these points and nowhere else. Although there are many ways to think about this problem, the perspective provided by *connections* leads to algorithms that are both highly efficient and simple to formulate.

Classically, there are two closely related ways to think about connections: in terms of *differentiation* and in terms of *transport*. The differentiation perspective is captured by the *covariant derivative* $\nabla_X Y$, which can be thought of as the directional derivative of one vector field along another. The transport perspective is captured by a *connection 1-form* ω , which tells us how to modify an object as it moves along a curve. In the smooth setting, these notions are largely equivalent, *i.e.*, a covariant derivative can be used to define transport and vice versa. However, there is a point at which these two perspectives diverge: an *Ehresmann connection* sacrifices the notion of differentiation to provide a more generic notion of transport.

In the discrete setting, the perspective we choose has a significant impact on our computational setup. The covariant derivative takes us down the standard path of discretizing differential operators (using, *e.g.*, finite differences or finite elements). This approach is essential when solving partial differential equations since we need to retain the ability to differentiate along arbitrary directions. However, the additional structure comes at a cost: because differentiation is approximated numerically, it may be very difficult to exactly satisfy global relationships from the smooth theory that depend on curvature and holonomy. Alternatively, we can take the “Ehresmann” approach and abandon differentiation so that holonomy and curvature can play the principal role. In this thesis we adopt the latter perspective since the *trivial connections* we wish to compute are defined in terms of a global

condition on holonomy. The outcome is a discrete connection with an “algebraic” flavor in the sense that curvature and holonomy are expressed purely in terms of simplicial chains and group operations.

1.2 Previous Work

Our approach is closest in spirit to the work of Leok et al [14], which describes a discretization of principal connections via a Lie group-valued discrete 1-form. Their work focuses primarily on the discrete configuration space $Q \times Q$ used in integrators for mechanical systems and does not develop numerical applications on discrete meshes. In computer graphics, early work on smooth, consistent transitions between tangent spaces was motivated by decoration of surfaces with consistently oriented textures and curvature-aligned strokes [19, 10, 23]. While these algorithms were framed in terms of smoothly varying direction fields, we view them as some of the first which constructed *connections* on discrete surfaces. Later motivation for this type of algorithm came from the requirements of quadrilateral remeshing [22, 11, 3], where directions are specified only up to rotations by $\pi/2$ (*a.k.a.*, “cross fields”). These applications led to the development of tools for the controlled design of direction fields [17, 21, 13]. Discrete connections have also appeared in the context of mesh deformation [15, 12] as a natural way to encode the relationship between adjacent frames on a mesh. While these approaches discretize the *Christoffel symbols*, we instead focus on an intrinsic, coordinate-free discretization of connections.

The main application presented in this thesis is the computation of globally consistent direction fields on discrete surfaces, or in other words, *parallel sections* of the unit tangent bundle. A major tension in the computation of direction fields is between simplicity of the formulation and total control over all aspects of the field. Efficient methods for *vector* field design have been proposed (*e.g.*, in [24, 9]), but unintended additional singularities often arise. At the other extreme, methods which offer full control over singularities (location and index) require sophisticated

non-linear solvers (*e.g.*, in [13]). Several approaches provide a trade-off between efficiency and partial control over singularities by applying repeated linear solves (*e.g.*, [20, 3]).

Fundamentally, the difference between our approach and previous methods is that we work with a connection 1-form (*i.e.*, angles on dual edges) instead of adopting the more traditional *metric* perspective (*i.e.*, lengths on primal edges). This representation allows us to perform computations that typically require a conformally equivalent flat metric (such as constructing orthogonal curve networks on surfaces) without explicitly determining edge lengths. In addition to increased efficiency, this representation has some interesting benefits – for instance, consider the usual discretization of Gaussian curvature at a vertex, given by

$$K = 2\pi - \sum_i \theta_i,$$

where θ_i are the tip angles. This representation makes it difficult to encode curvatures greater than 2π , since we would require *negative* tip angles. However, we can easily encode large curvature using a discrete connection since there is no constraint on the range of connection angles (see Sections 2.3 and Chapter 3 for more details).

Chapter 2

Discrete Connections

This chapter describes our formulation of discrete connections. Sections 2.1 and 2.2 give some intuition for the smooth objects we discretize; readers seeking a more formal presentation can consult a reference such as Abraham *et al.* [1]. Sections 2.3 and 2.4 develop the discretization we use as our computational framework.

2.1 Connections

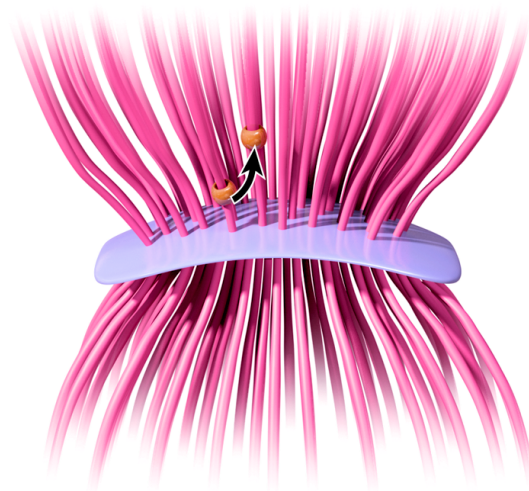


Figure 2.1: A fiber bundle associates an identical space or *fiber* (pink) to every point of a manifold (blue). A connection describes how to move from one fiber to another (illustrated by orange beads).

Roughly speaking, a connection tells us how a quantity associated with a manifold changes as we move from one point to another – it “connects” neighboring spaces (Figure 2.1). In the most generic setting, connections are defined in terms of *fiber bundles*. A fiber bundle $F \rightarrow E \xrightarrow{\pi} B$ consists of a *base space* B , *fiber space* F , and *total space* E , together with a *projection map* $\pi : E \rightarrow B$. The basic idea behind a fiber bundle is that E locally looks like the product of the base and fiber spaces in the sense that every point $x \in B$ is contained in an open set $U \subset B$ such that there is a homeomorphism $\phi : \pi^{-1}(U) \rightarrow U \times F$. Further, $\pi \circ \phi^{-1}$ gives the projection onto the first factor of $U \times F$. In terms of fiber bundles, a connection tells us how movement in the total space induces change along the fiber.

Given this setup, we can define an *Ehresmann connection* on any fiber bundle where B and F are differentiable manifolds. Specifically, consider tangent vectors of the total space that lie “along” fibers, *i.e.*, all the vectors in the kernel of $d\pi$ – this space is the *vertical subbundle* V of TE . An Ehresmann connection ω is a *vertical-valued 1-form* $\omega_x : T_x E \rightarrow V_x$ which leaves vertical vectors fixed, *i.e.*, $\omega(v) = v$ for all $v \in V$ [2]. The only other requirement is that this map is linear, *i.e.*, if $\omega_x : T_x E \rightarrow T_x F$ is a connection 1-form then for any scalar values a, b and tangent vectors $u, v \in T_x E$ we must have $\omega_x(au + bv) = a\omega_x(u) + b\omega_x(v)$ at every point $x \in E$. In many applications we are concerned only with *principal bundles* – in this case F is a Lie group and ω takes values in the corresponding Lie algebra. We say that a 1-form is “angle-valued,” “ \mathfrak{g} -valued,” etc., to indicate the fiber space.

A simple physical example helps to motivate Ehresmann connections and make the idea more concrete [2]. Consider a rolling coin whose configuration q is given by a position $s = (x, y)$, rolling angle θ , and heading φ (Figure 2.1). For a small change \dot{q} in the overall configuration, we can express the resulting change \dot{s} in position via

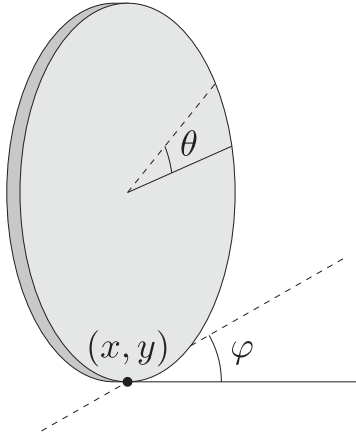


Figure 2.2: Rolling coin.

$$\underbrace{\left[\begin{array}{cc|cc} 1 & 0 & 0 & \cos \varphi \\ 0 & 1 & 0 & \sin \varphi \end{array} \right]}_{\omega: T_q E \rightarrow V_q} \underbrace{\left[\begin{array}{c} \dot{q}_x \\ \dot{q}_y \\ \dot{q}_\theta \\ \dot{q}_\varphi \end{array} \right]}_{\dot{q} \in T_q E} = \underbrace{\left[\begin{array}{c} \dot{q}_x + \cos \varphi \dot{q}_\theta \\ \dot{q}_y + \sin \varphi \dot{q}_\theta \end{array} \right]}_{\dot{s} \in V_q},$$

i.e., the coin slips along the direction of linear velocity, and rolls forward in the direction of the heading – the quantity $\dot{s} = \omega(\dot{q})$ describes the change in position (x, y) induced by our current linear *and* angular velocity. The geometric interpretation is that the base space $B = S^1 \times S^1$ encodes the rotation of the coin, the fiber space $F = \mathbb{R}^2$ encodes the position, and the connection maps a velocity \dot{q} tangent to the total space to the induced change in position \dot{s} tangent to the fiber. If the angles θ and φ are prescribed as functions of time, then we can integrate the change along the fiber to get the full dynamics of the coin.

2.2 Holonomy

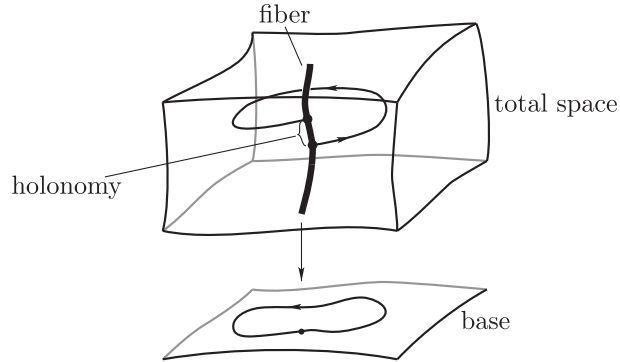


Figure 2.3: A connection maps tangents in the total space to tangents along the fiber. By integrating these tangents as we walk around a closed loop, we get the total change or *holonomy*. (Figure modified from [2], used with permission.)

The dynamics of the rolling coin provide one example of *parallel transport*. More generally, given a curve $\gamma(t)$ in the total space of any fiber bundle, we can evaluate transport by integrating $\omega(\dot{\gamma}(t))$ over the length of the curve, *i.e.*, at each point we take the tangent to the curve and “plug it in” to the connection to get the change along the fiber. Hence, the way quantities are transported is defined by our choice of connection ω .

In general, a quantity transported around a closed loop ℓ will not return to its original location. The difference between the initial and final quantity is called the *holonomy* of ω around ℓ (Figure 2.3). In the most general case, holonomy also depends on the basepoint of our loop, *i.e.*, it depends on a choice of initial point $p \in \ell$. For a *principal* bundle, however, picking a different starting p along the same fiber will not change the holonomy; for a principal bundle with an *abelian* fiber, the choice of basepoint does not affect holonomy at all. In particular, for direction field design we work with the bundle $SO(2) \rightarrow E \xrightarrow{\pi} \mathcal{M}$ where $SO(2)$ is the (abelian) group of rotations in the plane and \mathcal{M} is a surface. Hence, we do not have to worry about a choice of basepoint – in this case, holonomy is simply the

difference in angle between an initial and final vector transported around a loop ℓ (Figure 5.1).

Finally, every connection has an associated *curvature*. In particular, holonomy around *infinitesimal* loops gives the *sectional curvature* of the connection. The most familiar example is perhaps the Levi-Civita connection, whose curvature is the standard Riemannian curvature. On a surface, the curvature of the Levi-Civita connection is the Gaussian curvature.

2.3 Discrete Connections

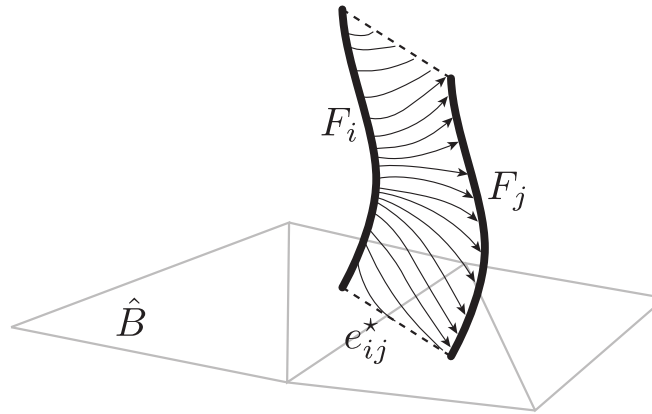


Figure 2.4: A discrete connection is expressed via a map between fibers associated with adjacent k -simplices.

In this work we take the following approach to discrete connections. Let a *semi-discrete fiber bundle* $F \rightarrow \hat{E} \xrightarrow{\pi} \hat{B}$ consist of a triangulated k -manifold \hat{B} where we associate a copy of F with each k -simplex in \hat{B} (Figure 2.4). We call this structure *semi-discrete* because we do not discretize the fiber space. A *discrete connection* is then given by a collection of fiber automorphisms $\hat{\omega}_{ij} : F \rightarrow F$ associated with each ordered pair of k -simplices (σ_i, σ_j) in \hat{B} that share a $k - 1$ -dimensional face. Conceptually, we have a map associated with each *dual edge* e_{ij}^* that encodes parallel transport between adjacent fibers – in fact, we can project

a smooth connection ω onto a discrete connection $\hat{\omega}$ via integration along dual edges. For this reason, we require that $\hat{\omega}_{ji} = \hat{\omega}_{ij}^{-1}$. Note that nothing prevents us from discretizing the base via a more general cell complex; we use triangulated manifolds only for simplicity.

In the case where F is given by a Lie group G , we have a *discrete principal bundle*, and the maps $\hat{\omega}_{ij}$ can be expressed via group action of F on F . In other words, we can explicitly represent the automorphisms by storing a group element $g \in G$ on each dual edge. Alternatively, we can store elements $\xi \in \mathfrak{g}$ – in this case the map between fibers is expressed by $\exp(\xi)$, where $\exp : \mathfrak{g} \rightarrow G$ is the usual exponential map. The latter representation may be preferable for two reasons. First, if G is compact and connected then \exp is surjective, so we can encode *at least* as much information in the algebra as we can in the group. In fact, we can often encode more: consider the case where $G = \text{SO}(n)$ – an element in the algebra can encode, say, *multiple* rotations by 2π around a given axis, whereas all such elements are identified with the identity in the group. Second, if G is abelian then expressions of the form $g_1 g_2 \cdots g_n$ in the group can be represented by linear expressions $\xi_1 + \xi_2 + \cdots + \xi_n$ in the algebra (where $\exp(\xi_i) = g_i$). Both of these considerations will come into play when developing our algorithm for direction field design (Chapter 3).

2.4 Discrete Holonomy

Since the maps $\hat{\omega}_{ij}$ encode transport along dual edges, we can define *discrete parallel transport* along any sequence of consecutive dual edges $e_{i_0 i_1}^*, e_{i_1 i_2}^*, \dots$ by simply composing the corresponding maps $\hat{\omega}_{i_0 i_1}, \hat{\omega}_{i_1 i_2}, \dots$ (Note that we need to be careful about orientation here since $\hat{\omega}_{ji} = \hat{\omega}_{ij}^{-1}$.) Or, in the case of a principle bundle, we simply concatenate the appropriate group elements. The definition of *discrete holonomy* is thus the same as the smooth definition: it is the difference found along the fiber after transporting a quantity around a closed loop (expressed as a *cycle* of dual edges). Since the base \hat{B} does not discretize infinitesimal loops, we do not

have a pointwise notion of the sectional curvature of a discrete connection. For certain bundles, however, discrete holonomy tells us about *integrated* curvature – see Section 5.1.

Chapter 3

Algorithm

This chapter describes an algorithm for computing *trivial connections*, *i.e.*, connections with globally vanishing holonomy. Although we describe this algorithm in terms of the unit tangent bundle of a surface, in principle it can be applied to any semi-discrete fiber bundle whose fiber is an abelian Lie group. Here we give a pragmatic description in terms of familiar operations on meshes – Chapter 5 gives an interpretation of our algorithm in terms of the discrete connections developed in Chapter 2. Chapter 4 provides numerical experiments and timings.

3.1 Setup

We work with a triangulated 2-manifold $K = \{V, E, F\}$ and its dual (Figure 3.1)–note, however, that we do not need to explicitly construct a dual mesh since we can simply store dual quantities on the corresponding primal elements. Most of the tools we need are standard operations from discrete exterior calculus (DEC). Although we review the essential concepts, a more general overview can be found in [7]. Ultimately, we need to solve for a set of *adjustment angles* that tell us how to rotate a vector whenever it moves across an edge. Our algorithm for computing these angles consists of a few simple steps:

1. Find a set of basis cycles.
2. Compute the angle defect around each basis cycle.

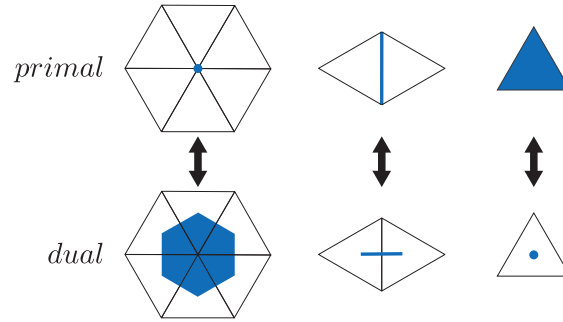


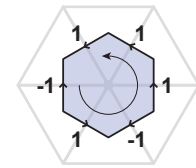
Figure 3.1: The natural setting for a discrete connection is on the dual edges (bottom, center) of a triangulated surface (top).

3. Specify singular vertices and their indices.
4. Solve a linear system for the adjustment angles.

These angles can then be used for various mesh processing tasks; we use them to construct direction fields with user-specified singularities (Section 3.8).

3.2 Basis Cycles

In the context of our algorithm, a *cycle* is a sequence of consistently oriented dual edges that form a loop. More explicitly, a cycle is represented by a vector $c \in \mathbb{Z}^{|E|}$ that has nonzero entries only for dual edges in that cycle. The sign of these entries is determined by the orientation of each dual edge relative to some canonical orientation: positive if it agrees, negative otherwise. A cycle around the boundary of a dual cell is a *boundary cycle*.



Given this representation, it is straightforward to construct a basis for all possible cycles on the surface. Note that any particular cycle is either *contractible*, meaning that it can be continuously deformed to a point, or *noncontractible*, meaning that it cannot (Figure 3.2). We first construct a matrix $d_0 \in \mathbb{R}^{|E| \times |V|}$ whose columns span the contractible cycles:

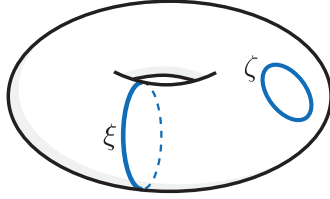


Figure 3.2: Loops on a surface can be contractible (ζ) or noncontractible (ξ).

$$(d_0)_{ij} = \begin{cases} \pm 1, & \text{dual edge } i \text{ is contained in dual cell } j \\ 0, & \text{otherwise.} \end{cases}$$

Here, each column is the boundary cycle of some dual cell (we use d_0 to denote this matrix since it is the discrete exterior derivative on 0-forms [5]). Technically, this matrix defines a *spanning set* since only $|V| - 1$ columns are independent. (This degeneracy is accounted for by a condition on singular indices; see Section 3.4.)

We compute a basis for the noncontractible cycles using the *tree-cotree decomposition* of Eppstein [8]:

- compute a spanning tree T of primal edges;
- compute a spanning tree T^* of dual edges that do not cross edges of T ;
- for any dual edge not contained in T^* and not crossed by T , follow both of its vertices to the root, completing a cycle.

On a surface of genus g , we get exactly $2g$ independent noncontractible cycles or *generators*. This basis can again be represented by the columns of a matrix $H \in \mathbb{R}^{|E| \times 2g}$ given by

$$H_{ij} = \begin{cases} \pm 1, & \text{if dual edge } i \text{ is in generator } j \\ 0, & \text{otherwise.} \end{cases}$$

We combine all basis cycles into a single matrix

$$A = \begin{bmatrix} d_0^T \\ H^T \end{bmatrix}.$$

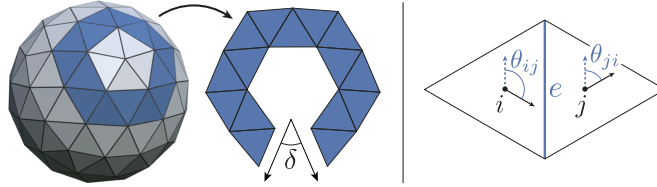


Figure 3.3: Left: In the discrete setting, the holonomy of the Levi-Civita connection on a surface is given by the usual angle defect δ —this defect is found by simply translating a vector across each unfolded pair of triangles in sequence (right figure).

3.3 Angle Defects

Each cycle in our basis specifies a sequence of dual edges, or equivalently, a sequence of primal triangles. The *angle defect* δ of a cycle is simply the angle between initial and final edges when these triangles are unfolded in the plane (Figure 3.3, left). More explicitly, given an initial angle α_i in face i , we compute a new angle α_j in neighboring face j as

$$\alpha_j = \alpha_i - \theta_{ij} + \theta_{ji}, \quad (3.1)$$

where θ_{ij} and θ_{ji} are the angles between the shared edge e and an arbitrary but fixed reference direction in triangles i and j , respectively (Figure 3.3, right). Repeating this procedure for n consecutive dual edges in a cycle gives us a sequence of angles $\alpha_0, \dots, \alpha_n$, and the angle defect is given by $\delta = \alpha_n - \alpha_0$. In the case of contractible basis cycles, this procedure yields the usual discretization of Gaussian curvature. We hence use $K \in \mathbb{R}^{|V|}$ to denote the vector of defects around contractible cycles; we use $z \in \mathbb{R}^{2g}$ to denote defects around noncontractible cycles.

3.4 Singularities

To control the placement and behavior of singularities, we specify an *index* for each primal vertex. The index determines the number of full rotations experienced by a vector transported along a small loop around the vertex (Figure 3.4); most vertices

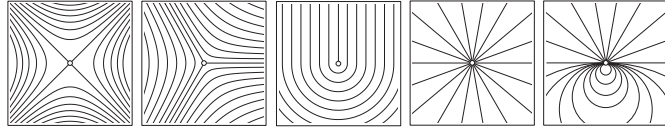


Figure 3.4: On most surfaces, a direction field must have at least one *singularity*. Here we see a few examples (left to right): saddle (-1), tripod (-1/2), thorn (+1/2), focus (+1), apple (+3/2) [16].

will have index zero. We can also specify the number of rotations experienced by vectors transported around generators (Figure 3.5). In our algorithm, we simply specify a vector $k \in \mathbb{Z}^{|V|+2g}$ of indices corresponding to the cycles in our basis. The only requirement is that $\sum_i k_i = \chi$ over vertices and boundary loops (Section 3.7), where $\chi = |V| - |E| + |F|$ is the *Euler characteristic*—indices of the remaining generators may be assigned arbitrarily. These indices are used to modify angle defects around basis cycles: $\tilde{K}_i = K_i - 2k_i\pi$, and $\tilde{z}_i = z_i - 2k_i\pi$. We then concatenate these values into a single vector $b \in \mathbb{R}^{|V|+2g}$ of modified defects $b = [\tilde{K} \tilde{z}]^T$.

So far, direction fields have been considered consistent only if directions are mapped to themselves modulo 2π by parallel transport. More flexibility is achieved by allowing directions to be mapped to themselves modulo $2\pi/N$ for some fixed $N \in \mathbb{N}$ (e.g., $N = 4$ for cross fields—see Figure 4.3). This is achieved by simply setting *fractional* singular indices $k_i = n_i/N$, $n_i \in \mathbb{Z}$ and proceeding as before. Singularities can be placed by hand or determined by an automatic method such as [20].

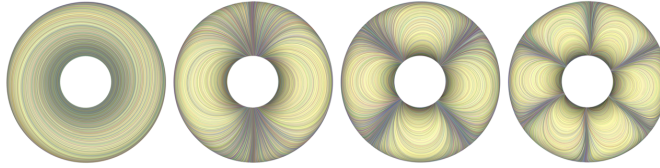


Figure 3.5: Our method gives control over the holonomy around generators – note that there are no singularities as the field direction “spins” along one of the generators (left to right: no turn, one turn, two turns, three turns).

3.5 Optimization

Finally, to compute the adjustment angles we solve the convex problem

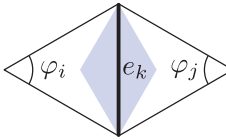
$$\operatorname{argmin}_x \|x\|_2 \quad \text{s.t.} \quad Ax = -b, \quad (3.2)$$

whose only local minimum is the unique global minimizer. Further, the constraints encode the index prescribed at each vertex (see Section 5.3), so we cannot end up with more singularities than we asked for.

At this point, standard algorithms for convex problems (*e.g.*, equality-constrained Newton’s method) could be applied to obtain the minimizer. However, the simple structure of this problem permits a more efficient approach. Since the system of constraints is underdetermined, the minimizer x^* of (3.2) is the unique solution to $Ax = -b$ that has no component in the kernel of A – all other solutions have larger ℓ_2 norm. One way to compute x^* is to first find *any* solution \tilde{x} to the constraint equation $Ax = -b$ and then project out its null space component. Since the null space is spanned by the columns of d_1 (*i.e.*, the discrete exterior derivative on 1-forms [7]), the optimal solution is given by $x^* = \tilde{x} - d_1^T (d_1 d_1^T)^{-1} d_1 \tilde{x}$, which entails an additional linear solve. However, a number of efficient linear solvers directly compute solutions with no nullspace component – in practice, we use the multifrontal sparse QR factorization method implemented in SuiteSparseQR [4].

3.6 Area Weights

We can easily include a diagonal matrix $D \in \mathbb{R}^{|E| \times |E|}$ in our objective to control the importance of smoothness over the mesh. In particular, we use the standard *cotangent weights*

$$D_{kk} = \sqrt{2(\cot \varphi_i + \cot \varphi_j)^{-1}},$$


to get proper area weighting over the *diamond areas* associated with each dual edge (see [5]). Here φ_i and φ_j are the angles opposing edge k . To solve the augmented problem, we apply the change of variables $y = Dx$ and solve for y exactly as before, recovering the final solution via $x^* = D^{-1}y^*$. (Note that in this case we never have to explicitly evaluate the reciprocal of $\cot \varphi_i + \cot \varphi_j$, which avoids potential instability.)

3.7 Surfaces with Boundary

For surfaces with boundary our constraint matrix A needs to include boundary loops and omit cycles around boundary vertices. This requirement entails only three simple modifications to our algorithm:

- Skip dual cells along the boundary when building the basis for contractible cycles;
- Skip boundary vertices when constructing the primal spanning tree T ;
- Skip dual edges that cross the boundary when extracting loops from the tree-cotree decomposition.

The (modified) tree-cotree decomposition will now yield a generator from *every* class of noncontractible cycles, including boundary loops.

At this point there are a number of ways one could modify the vector b to control behavior at the boundary. Perhaps the simplest is to require only that the

sum of the indices of singular vertices equals zero – Figure 4.7 demonstrates the resulting effect on parallel transport.

3.8 Direction Fields

Once we have a vector x of connection angles, constructing a global direction field is straightforward: starting at an arbitrary face f_0 and initial direction β_0 , traverse the primal faces in any order. Across each edge e_k , compute the angle in the next triangle via

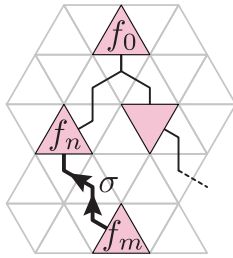
$$\beta_j = \beta_i - \theta_{ij} + \theta_{ji} - x_k. \quad (3.3)$$

Note that Eq. (3.3) is just the operation used to compute angle defects (Eq. (3.1)), augmented with the adjustment angles x . Because of the way we compute x , the resulting direction field is independent of traversal order, and is only a function of the choice of β_0 (see Section 5.2).

3.9 Directional Constraints

We can specify a set of faces where the field direction is fixed by prescribing the angle γ in each of these faces (Figure 4.8). To accommodate these constraints we build an additional spanning tree T_c of the primal faces rooted at one of the constrained faces f_0 . Each time we encounter a constrained face f_m , we follow the tree back towards the root until we encounter another constrained face f_n (possibly the root f_0).

The sequence σ of dual edges between f_m and f_n in T_c forms an additional row in our constraint matrix A . We then transport the constraint angle γ_m along σ using Equation (3.1) to get γ'_m , and store the difference $\gamma_n - \gamma'_m$ in the corresponding entry of b . Finally, we make sure to compute our direction field starting at f_0 using the initial angle γ_0 . This way, all directional constraints are satisfied by construction. Note that constraints on holonomy and directional constraints are linearly independent since no collection



of paths in T_c can be combined to form a cycle.

Chapter 4

Results

This chapter examines how our algorithm compares to existing methods in terms of performance and robustness. Results are shown in figures at the end of the chapter; in all examples we were able to achieve exactly the prescribed field topology (Figure 1.1). Figure 4.5 demonstrates that fields produced by our method can be used to drive quadrilateral parameterization algorithms such as *QuadCover* [11], which maps a cross field to a vector field on a multiple covering of the input surface. One benefit of our approach is that it provides exact matchings between different sheets of the covering, even near singular vertices of large index. Figure 4.12 shows two artistic applications of our method.

4.1 Performance

We tested performance on a number of standard meshes with varying size and element quality. Since singularities and constraint directions depend only on the data vector b , we can prefactor our constraint matrix A and edit direction fields in real time (Figure 4.1). Adding faces to the constraint set entails updating A ; factorization took no more than 9 seconds on our largest model (lion, 400k faces – see Figure 4.2). As described in Section 3.5, our solutions are globally optimal since they are simply the minimum-norm solution to an underconstrained linear system. Overall we observed very consistent performance, even on fields with many singularities (Figure 4.10). A large number of directional constraints could

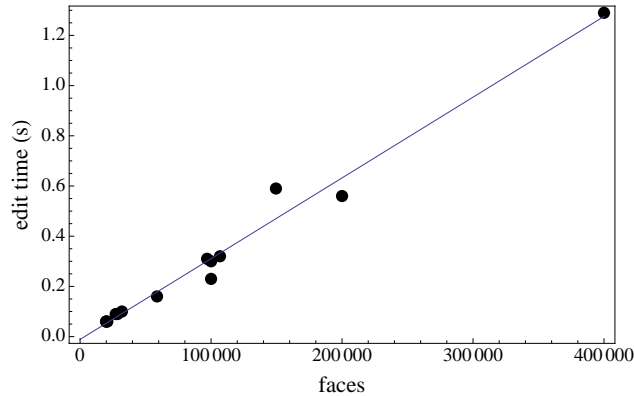


Figure 4.1: Timings of our implementation for all meshes shown in figures (2.4 GHz Core 2 Duo laptop, single thread). On the largest mesh (lion, 400k faces), singularities and constraint directions can be edited in roughly 1.3s after 8.2s of setup time.

considerably increase the size of the system, though by no more than a factor of two: at worst we have one constraint for each edge in a spanning tree on $|V|$ vertices.

Relative to the method described in Ray *et al.* [20] we can edit a mesh with 100k faces roughly 15-48x times faster, depending on the convergence rate of their nonlinear solver. Note that their method cannot guarantee optimality since it relies on iterative reprojection onto a nonconvex constraint set. The method in Lai *et al.* [13] computes a globally optimal solution via discrete Ricci flow, but is nonlinear. Hence we can edit singularities about 25-30x faster (using the same meshes and comparable hardware), and we can additionally edit directional constraints at roughly the same rate. Fisher *et al.* [9] also compute a solution via a single linear solve, but cannot guarantee the global topology of the resulting field, nor can they deal with fractional indices.

4.2 Robustness

As depicted in Figure 4.11, our results are consistent across different discretizations of the same surface. More remarkably, fields retain the same qualitative behavior even after significant noise or distortion has been applied to the mesh (Figure 4.4), a consequence of the intrinsic, variational nature of our formulation. Note that some triangles may have negative cotangents; in this case we simply clamp cotangents to zero when computing area weights (Section 3.6)—alternatively, we can simply use unit weights on all edges ($D = I$). In practice these options produce very similar results; we did not encounter *any* meshes where bad triangles resulted in a visible problem.

Finally, our method had no difficulty dealing with singularities of large index (see Figure 4.9) – even on extremely coarse meshes – since we can encode an arbitrarily large amount of “turning” across a single edge (as discussed in Section 2.3). In comparison, methods that store absolute angles per face [20] or vertex [13] may need to refine the mesh or cut out a boundary region near such singularities, since (as noted earlier) the angle defect around a single vertex can only encode so much curvature.



Figure 4.2: Even meshes with a large number of faces (lion, 400k faces) can be edited in about a second on a standard laptop.

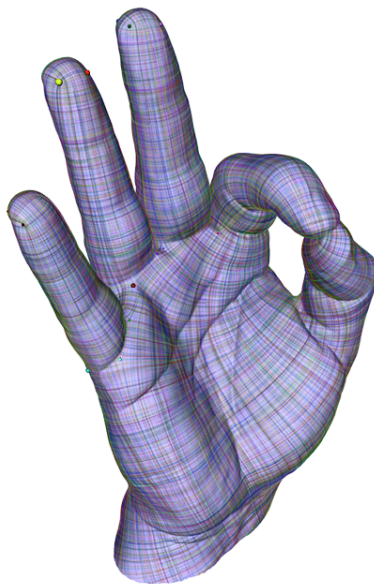


Figure 4.3: Our algorithm generates direction fields that are smooth up to local rotations by multiples of $2\pi/N$.

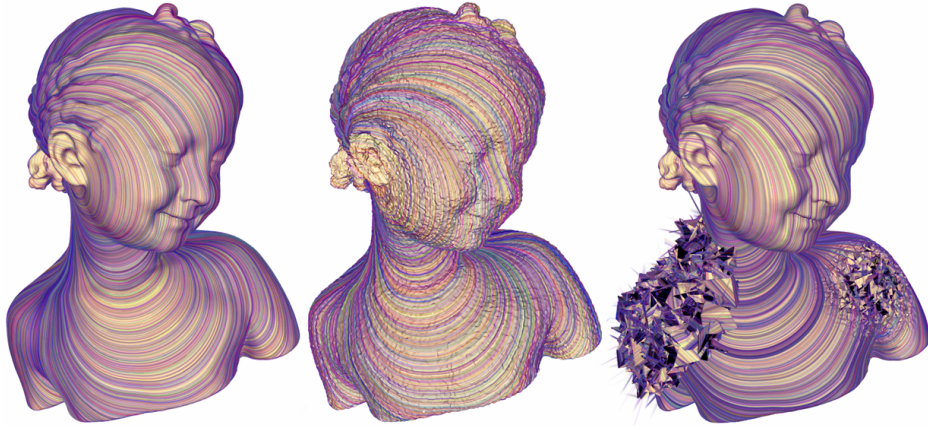


Figure 4.4: Because our method is purely intrinsic, it is robust to noise (center) and extreme perversions of the input mesh (right).

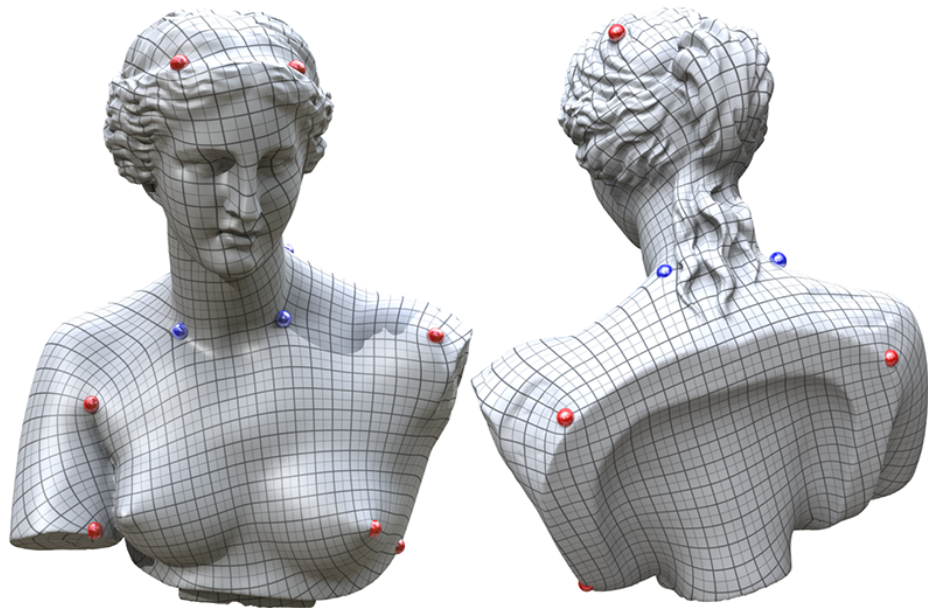


Figure 4.5: The fields we generate can be used as input to *QuadCover* [11]. Here a small set of hand-picked singularities of index $\pm 1/4$ yields a parameterization with very little distortion.

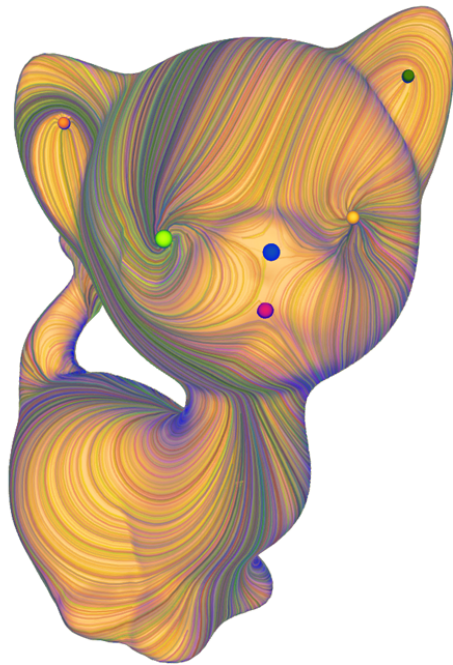


Figure 4.6: Real-time editing makes it easy to place singularities in locations that are geometrically uninteresting but artistically relevant.

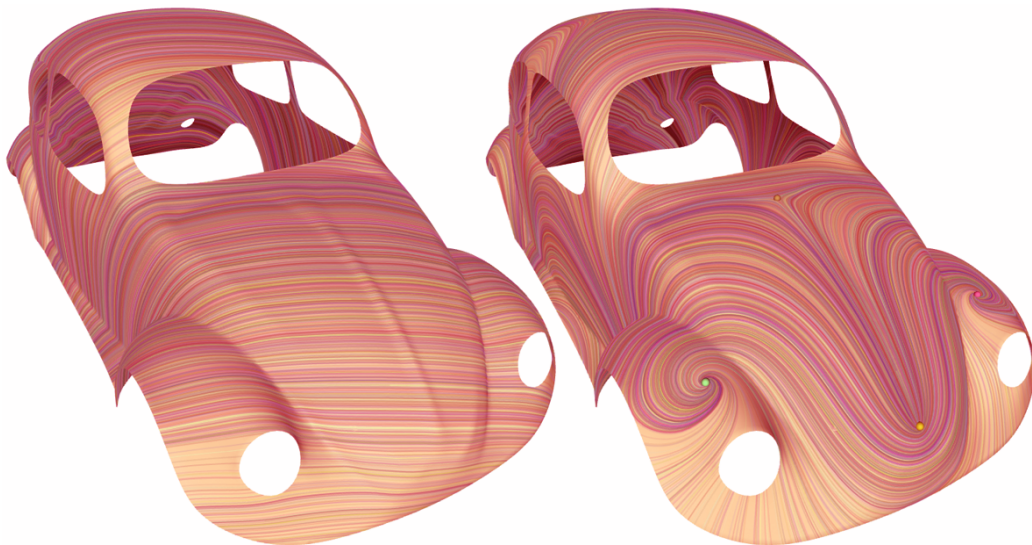


Figure 4.7: Fields on surfaces with boundary do not require singularities (left), but we can easily add singularities and still get natural boundary behavior (right).

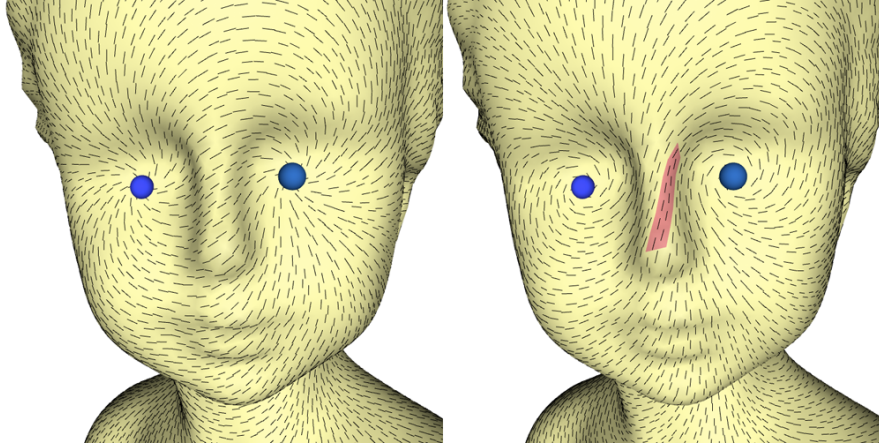


Figure 4.8: We can fix the direction of the field at specified faces by constraining transport between pairs of fixed faces. Notice that we still obtain a smooth field with only specified singularities.

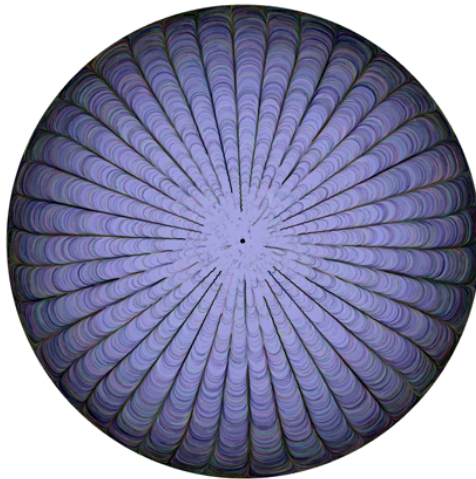


Figure 4.9: Since our method does not need to explicitly compute a metric, we have no trouble handling singularities of arbitrarily large index (above: singularity of positive index 20).

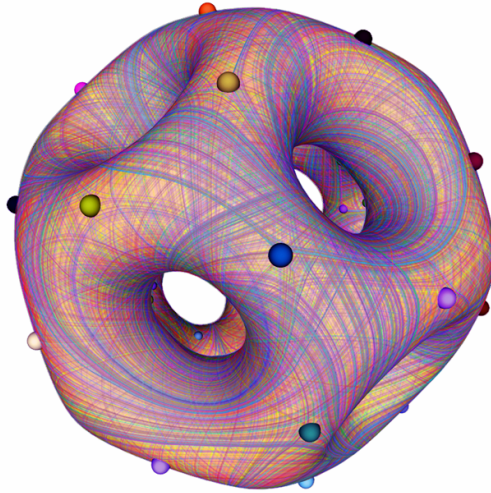


Figure 4.10: Our method has no difficulty with high genus or a large number of singularities – here we see a direction field with 60 singularities on a surface of genus 11.

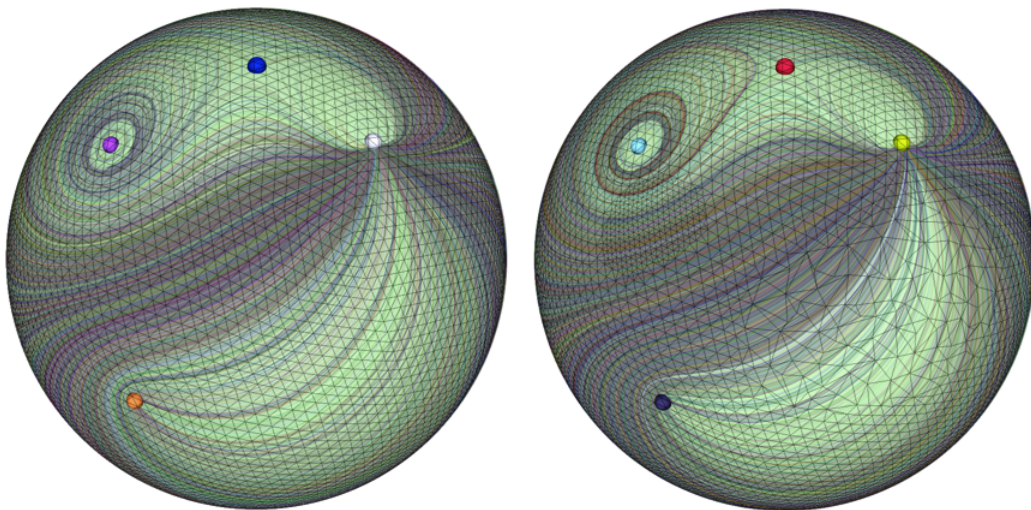


Figure 4.11: Our discretization yields similar results on different meshes of the same surface.



Figure 4.12: Fast direction field editing makes it easy to wrap a T-rex in ribbon (top) or build a horse out of flexible drinking straws (bottom).

Chapter 5

Discussion

This chapter gives an interpretation of the algorithm described in Chapter 3 in terms of the discrete connections defined in Chapter 2.

5.1 Connections on Surfaces

As mentioned earlier, direction fields computed by our algorithm can be viewed as sections of the *unit tangent bundle* $SO(2) \rightarrow E \xrightarrow{\pi} \mathcal{M}$, *i.e.*, an angle at each point of the surface \mathcal{M} giving the direction of the field. A connection ω on this bundle therefore maps each direction of motion to an *infinitesimal rotation*. Formally, ω defines a principal connection on the frame bundle of a smooth surface, which is encoded by an $\mathfrak{so}(2)$ -valued 1-form.

When developing a discrete representation of ω , the first question is: how should we represent tangent vectors? Storing tangents on *faces* is perhaps most natural because, as pointed out by Kircher and Garland, “it avoids the need to invent tangent planes that lie outside the surface” [12]. In other words, tangent directions in faces are *intrinsic*, which means that they are well-defined even on poorly discretized surfaces (see especially Figure 4.4). This setup leads to a semi-discrete fiber bundle $SO(2) \rightarrow \hat{E} \xrightarrow{\pi} \hat{\mathcal{M}}$, where $\hat{\mathcal{M}}$ is a simplicial surface (Chapter 2.3).

Within this framework, a discrete connection $\hat{\omega}$ has a particularly simple representation: for each dual edge e_k^* we store a single angle $\hat{\omega}_k$ which represents the

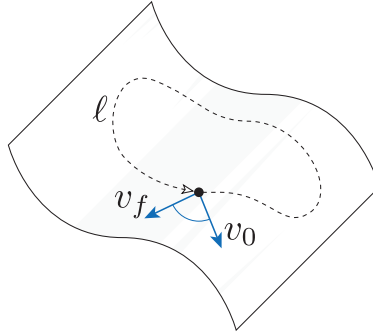


Figure 5.1: On a surface, holonomy is given by the difference in angle after a vector is parallel transported around a closed loop.

total (*i.e.*, integrated) rotation of a vector as we travel from one face to the next (see Figure 5.2, right). In terms of our algorithm, this angle is given explicitly by $\hat{\omega}_k = \theta_{ji} - \theta_{ij} - x_k$, *i.e.*, a change of frame followed by an “adjustment.” In the language of DEC, a value per dual edge is a (dual) discrete 1-form [5], which in our case is *angle* valued. Note that these angles can take any value in \mathbb{R} , and can therefore be thought of as elements of the Lie algebra $\mathfrak{so}(2)$.

Discrete parallel transport via $\hat{\omega}$ is also simple: starting with an initial direction α_0 , we add consecutive angles $\hat{\omega}_k$ along a sequence of dual edges. Again, since each value $\hat{\omega}_k$ represents the integral of infinitesimal rotations along a path from one face to the next, this sum can be thought of as piecewise integration of a smooth connection. The holonomy of $\hat{\omega}$ is thus given by sums of angles around cycles, and the total curvature over a region is given by the holonomy around the region boundary. (Figure 5.3). However, curvature does not tell us everything about holonomy since not every cycle is a boundary—this fact plays a critical role in the formulation of our algorithm.

5.2 Trivial Connections

With all of this machinery in place, we arrive at the central question: which connection should we use to construct direction fields? One answer is given by

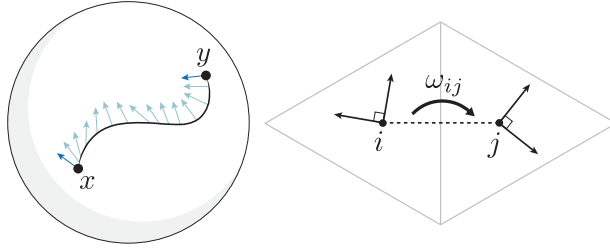


Figure 5.2: Left: in the continuous setting, a connection on the unit tangent bundle determines how tangent directions change along a curve. Right: a discrete connection on this bundle is represented by a rotation angle $\omega_{ij} = -\omega_{ji}$ at each oriented dual edge e_{ij}^* of a triangulated surface.

the canonical *Levi-Civita connection* [6]. Parallel transport via the *discrete* Levi-Civita connection is computed as in Equation (3.1), and the resulting holonomy or “angle defect” δ around a dual cell corresponds to the standard discretization of Gaussian curvature in terms of vertex tip angles (Figure 3.3). One way to see that this procedure corresponds to a proper discretization of the Levi-Civita connection is to consider that Levi-Civita on a surface is given by the pullback under the Gauss map of Levi-Civita on the sphere. Since parallel transport on the sphere maps one tangent plane to another via rotation along a great arc, we can transport a tangent vector from a triangle to one of its neighbors by rotating it

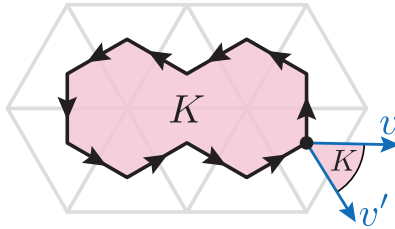


Figure 5.3: Curvature of the unit tangent bundle. In the discrete case, the total curvature of a region is simply the angle “defect” of a unit vector transported around the region’s boundary.

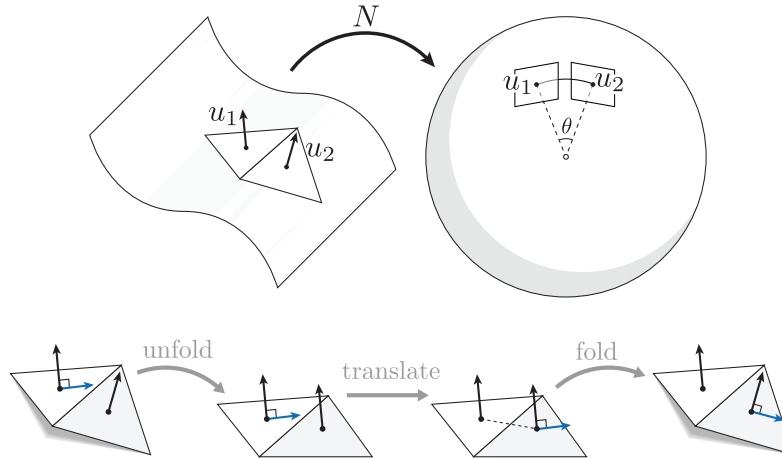


Figure 5.4: The Levi-Civita connection on a discrete surface is induced by the pullback under the Gauss map N of the Levi-Civita connection on the sphere.

around their shared edge (Figure 5.4). This choice is popular in computer graphics because it is easy to compute and agrees with our usual notion of straightness [18].

However, in many practical situations this simple scheme is problematic: since the holonomy of the Levi-Civita connection equals the Gaussian curvature, a vector transported around a closed loop is not mapped back to itself. As a consequence, transport from one point to another will depend on the choice of path, since we can “pick up” additional curvature along the way (see Figure 5.5, left).

Instead, we seek a *trivial connection*, *i.e.*, a connection where the holonomy around every cycle is *zero*. It is easy to see that transport via a trivial connection is path-independent: in particular, consider transport along any two paths f and g from a point x to a point y (Figure 5.5, right) – the only way the total change around the combined loop $f-g$ can be zero is if change along f equals the change along g .

Though not formulated explicitly in terms of connections, this basic premise is the underlying idea in recent work on direction field design [20, 13]. Ray *et al.* [20] effectively compute a connection where *curvature* vanishes and then apply smoothing to obtain a globally consistent result. The reason smoothing is needed here is

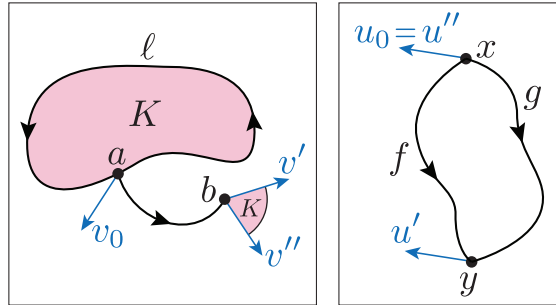


Figure 5.5: Left: transporting a vector v_0 from a to b along two different paths may yield different results (v' resp. v'') because we can pick up different amounts of curvature along the way. Right: a trivial connection guarantees that transport is path-independent since any loop $f - g$ must have zero holonomy.

that curvature alone is not sufficient to characterize consistency – as noted earlier, it describes holonomy only around *boundary* cycles (see Figure 5.6). More recently, Lai *et al.* [13] acknowledge the importance of the holonomy around generators, but are concerned that constraining the holonomy around all loops is computationally infeasible. Like Ray, their solution is to first eliminate curvature (by computing a flat metric with cone singularities), and then account for the generator holonomy with a “rotation compensation” field.

In fact, the holonomy around any cycle can be easily expressed in terms of the curvature and the holonomy around a set of generators. In the discrete case, it is especially straightforward to compute a small set of basis cycles that encode this information, which is the approach we take in our algorithm (Section 3.2). More specifically, the “adjustment angles” in our algorithm (or what Ray *et al.* call the “field curvature”) actually describe the deviation of our discrete connection $\hat{\omega}$ from the (discrete) Levi-Civita connection. Hence, our linear constraint $Ax = -b$ states that the sum of these deviations along any cycle should exactly *cancel* the holonomy we find with Levi-Civita (Section 3.3). Implicitly, we are constructing a connection for a surface with a flat metric, but expressing this connection with respect to the given embedding. This way we do not need to explicitly determine

the edge lengths that define the new metric. Notably, however, a *trivial connection* is more specific than a *flat metric* since a trivial connection also has zero holonomy around generators.

We can now give an interpretation of the objective in our algorithm as well: $\|Dx\|_2$ is the distance from the Levi-Civita connection with respect to the norm induced by the Hodge inner product, *i.e.*, the standard 2-norm on differential forms. The diagonal factor D – or $\star_1^{1/2}$ in the language of DEC – simply gives the appropriate area weighting (Section 3.6). Overall, then, our optimization problem seeks a globally consistent way to transport vectors that agrees with our usual notion of “straight” as much as possible.

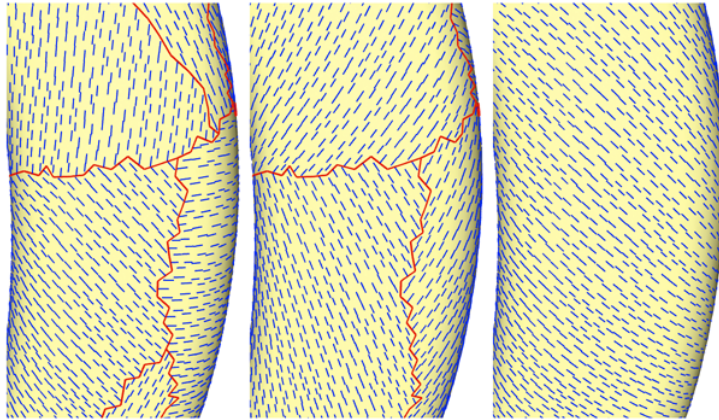


Figure 5.6: Left: parallel transport via the Levi-Civita connection is not globally consistent, and yields discontinuities in direction fields (displayed in red). Center: using a flat metric improves the situation, but inconsistencies remain. Right: a *trivial* connection achieves global consistency by constraining *all* cycles—including generators.

5.3 Singularities

Not every surface admits a trivial connection, however. Consider the Gauss-Bonnet theorem, which states that the total curvature of a surface equals $2\pi\chi$, where the

Euler characteristic χ is a *topological* invariant. In other words, our surface must have curvature *somewhere*, but we get to choose where this curvature goes.

Ideally, we would like to put this curvature where it will not interfere with the transport of vectors. Remembering that Gaussian curvature is given by the holonomy around region boundaries (Section 5.1), this means we want the curvature of every region to be an integer multiple of 2π , so that vectors transported around closed loops are mapped back to themselves—even if they experience a number of full rotations along the way. If we can do this, then transport from one point to another is still consistent up to rotations by $2k\pi$, hence the *vector* we end up with will remain the same.

An easy way to achieve this goal is to concentrate all of our curvature at a set of isolated points or *singularities*, in increments of 2π . In the discrete case, this is equivalent to constraining the holonomy around some small set of vertices (possibly just one) as done in Section 3.4. For surfaces with boundary, we can also concentrate curvature on boundary loops. (Note, however, that these considerations place no restriction on the holonomy around generators.) Further, if we instead use increments of $2\pi/N$, then transport will be consistent up to rotations by $2k\pi/N$ —suitable for line fields, cross fields, *etc.*. Thus, from the perspective of *connections*, the generalization of the Poincaré-Hopf theorem given in Ray *et al.* [21] is a straightforward consequence of the Gauss-Bonnet theorem.

5.4 Summary

On the unit tangent bundle, our computational setup can easily be seen as a projection of the smooth theory onto discrete meshes. Dual edges carry finite angles which equal path integrals of incremental rotations between neighboring faces. A zero-holonomy condition on the space of loops (including noncontractible loops) results in a finite dimensional linear system of sum conditions around discrete cycles of dual edges. The minimum ℓ_2 norm solution of this linear system is the minimum L_2 norm solution of the projected energy on the underlying smooth 1-

form. The result is a trivial connection with curvature that vanishes everywhere except at a fixed set of singularities and boundary loops with specified indices.

Chapter 6

Conclusion

We have described a theory of discrete connections which is suitable for computations where parallel transport, holonomy, and curvature are of primary importance. Our framework for computing *trivial* connections provides a simple, effective foundation for geometry processing tasks that need to compare frames or directions on surfaces. Although our algorithm is quite simple from the perspective of mesh processing, it comes from a solid geometric foundation that links together several aspects of discrete differential geometry. On the practical side of things, we believe that robustness, efficiency, and ease of implementation make our framework a valuable tool for a number of graphics-related applications.

Bibliography

- [1] R. Abraham, J. E. Marsden, and R. Ratiu. *Manifolds, tensor analysis, and applications: 2nd edition*. Springer-Verlag New York, Inc., New York, NY, USA, 1988.
- [2] A. Bloch. *Manifolds, tensor analysis, and applications: 2nd edition*. Springer-Verlag New York, Inc., New York, NY, USA, 2003.
- [3] David Bommes, Henrik Zimmer, and Leif Kobbelt. Mixed-Integer Quadrangulation. *ACM Trans. Graph.*, 28(3):art. 77, 2009.
- [4] Tim Davis. SuiteSparseQR: A multithreaded multifrontal sparse QR factorization. <http://www.cise.ufl.edu/research/sparse/SPQR/>.
- [5] Mathieu Desbrun, Eva Kanso, and Yiyang Tong. Discrete Differential Forms for Computational Modeling.
- [6] Manfredo P. DoCarmo. *Riemannian Geometry*. Birkhäuser, 1992.
- [7] Sharif Elcott and Peter Schröder. Building your own DEC at home. In *ACM SIGGRAPH Course Notes on Discrete Differential Geometry*, pages 55–59, 2006.
- [8] David Eppstein. Dynamic Generators of Topologically Embedded Graphs. In *Proc. ACM-SIAM Symp. on Discr. Alg.*, pages 599–608, 2003.
- [9] Matthew Fisher, Peter Schröder, Mathieu Desbrun, and Hugues Hoppe. Design of Tangent Vector Fields. *ACM Trans. Graph.*, 26(3):art. 56, 2007.

- [10] Aaron Hertzmann and Denis Zorin. Illustrating Smooth Surfaces. In *Proc. ACM/SIGGRAPH Conf.*, pages 517–526, 2000.
- [11] Felix Kälberer, Matthias Nieser, and Konrad Polthier. QuadCover - Surface Parameterization using Branched Coverings. *Comp. Graph. Forum*, 26(3):375–384, 2007.
- [12] Scott Kircher and Michael Garland. Free-Form Motion Processing. *ACM Trans. Graph.*, 27(2):1–13, 2008.
- [13] Yu-Kun Lai, Miao Jin, Xuexiang Xie, Ying He, Jonathan Palacios, Eugene Zhang, Shi-Min Hu, and Xianfeng Gu. Metric-Driven RoSy Field Design and Remeshing. *IEEE Trans. Vis. Comp. Graph.*, 16:95–108, 2010.
- [14] M. Leok, J. E. Marsden, and A. Weinstein. A discrete theory of connections on principal bundles. (preprint, [arXiv:math.DG/0508338](https://arxiv.org/abs/math/0508338)), 2004.
- [15] Yaron Lipman, Olga Sorkine, David Levin, and Daniel Cohen-Or. Linear Rotation-Invariant Coordinates for Meshes. *ACM Trans. Graph.*, 24(3):479–487, 2005.
- [16] Igor. Nikolaev. *Foliations on Surfaces*. Springer, 2001.
- [17] Jonathan Palacios and Eugene Zhang. Rotational Symmetry Field Design on Surfaces. *ACM Trans. Graph.*, 26(3):art. 55, 2007.
- [18] Konrad Polthier and Markus Schmies. Straightest geodesics on polyhedral surfaces. In *Math. Vis.*, pages 391–398, 1998.
- [19] Emil Praun, Adam Finkelstein, and Hugues Hoppe. Lapped Textures. In *Proc. ACM/SIGGRAPH Conf.*, pages 465–470, 2000.
- [20] Nicolas Ray, Bruno Vallet, Laurent Alonso, and Bruno Lévy. Geometry-Aware Direction Field Processing. *ACM Trans. Graph.*, 29(1):1–11, 2009.
- [21] Nicolas Ray, Bruno Vallet, Wan Chiu Li, and Bruno Lévy. N-symmetry direction field design. *ACM Trans. Graph.*, 27(2):1–13, 2008.

- [22] Yiying Tong, Pierre Alliez, David Cohen-Steiner, and Mathieu Desbrun. Designing Quadrangulations with Discrete Harmonic Forms. In *Proc. Symp. Geom. Proc.*, pages 201–210, 2006.
- [23] Greg Turk. Texture Synthesis on Surfaces. In *Proc. ACM/SIGGRAPH Conf.*, pages 347–354, 2001.
- [24] Eugene Zhang, Konstantin Mischaikow, and Greg Turk. Vector Field Design on Surfaces. *ACM Trans. Graph.*, 25(4):1294–1326, 2006.