

# Trivial Connections on Discrete Surfaces

Keenan Crane<sup>1</sup> and Mathieu Desbrun<sup>1</sup> and Peter Schröder<sup>1,2</sup>

<sup>1</sup>Computing and Mathematical Sciences, Caltech

<sup>2</sup>Institute for Advanced Study, TU München

---

## Abstract

*This paper presents a straightforward algorithm for constructing connections on discrete surfaces that are as smooth as possible everywhere but on a set of isolated singularities with given index. We compute these connections by solving a single linear system built from standard operators. The solution can be used to design rotationally symmetric direction fields with user-specified singularities and directional constraints.*

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Picture/Image Generation—Line and curve generation

---

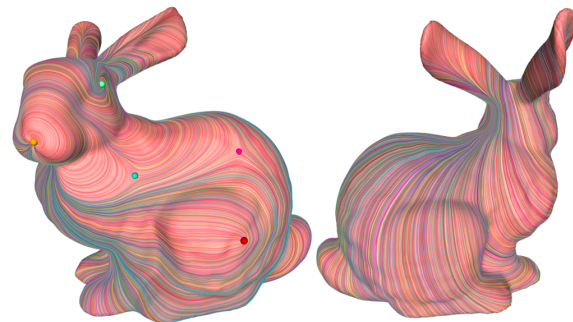
## 1 Introduction

Numerous applications in computer graphics need to compare tangent directions that originate at different points on a surface (consider smoothing a vector field). A natural question to ask, then, is “how do we map one tangent space to another?” One answer is provided by a *connection*, which describes how any tangent vector changes as it moves along the surface. Ideally, though, we want to map vectors from one point to another *consistently*, *i.e.*, independent of the path taken to get there. In this paper, we demonstrate that connections satisfying this fundamental requirement can be computed easily and efficiently.

### 1.1 Previous work

Early work on smooth, consistent transitions between tangent spaces was motivated by decoration of surfaces with consistently oriented textures and curvature-aligned strokes [PFH00, HZ00, Tur01]. 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 [TACSD06, KNP07, BZK09], where directions are specified only up to rotations by  $\pi/2$  (*a.k.a.*, “cross fields”). These applications led to tools for designing fields with more general rotational identifications [PZ07, RVLL08, LJX\*10].

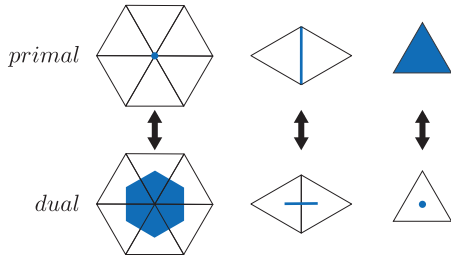
A major tension in the computation of direction fields is between simplicity of the formulation and total control over all aspects of the field. Efficient methods for *vector* field



**Figure 1:** Fields produced by our method have singularities precisely where desired (left) and nowhere else (right).

design have been proposed (*e.g.*, [ZMT06, FSDH07]), but unintended singularities often arise. At the other extreme, methods which offer full control over singularities (location and index) require sophisticated non-linear solvers (*e.g.*, [LJX\*10]). Several approaches provide a trade-off between efficiency and control by applying repeated linear solves (*e.g.*, [RVAL09, BZK09]).

Discrete connections have also appeared in the context of mesh deformation [LSLCO05, KG08] as a way to encode the relationship between adjacent frames on a mesh. While these approaches discretize the *Christoffel symbols*, we will instead focus on an intrinsic, scalar discretization of connections.



**Figure 2:** The natural setting for a discrete connection is on the dual edges of a triangulated surface (bottom, center).

### 1.2 Contributions

We give an algorithm for the computation of *trivial* connections on orientable simplicial surfaces of arbitrary genus with and without boundary. The primary variables are angles attached to dual edges (an angle-valued *dual 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 connections with a prescribed set of singularities. These singularities can be placed at arbitrary vertices with any indices that satisfy a global topological constraint. The method is guaranteed not to produce spurious singularities and comes with a rigorous theoretical foundation. Relative to previous methods our algorithm is surprisingly simple, and can be implemented using standard operations from mesh processing and linear algebra.

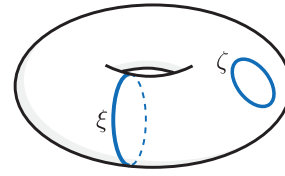
The next section gives a pragmatic description of our algorithm in terms of familiar operations on meshes. Section (3) justifies the decisions made in our algorithm and explores its relationship with smooth differential geometry. Section (4) demonstrates results and provides a more direct comparison with previous work.

### 2 Algorithm

We work with a triangulated 2-manifold  $K = \{V, E, F\}$  and its dual (Figure 2) – note that we do not need to explicitly construct a dual mesh since we can 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 [ES06]. 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.
3. Specify singular vertices and their indices.
4. Solve a linear system for the adjustment angles.

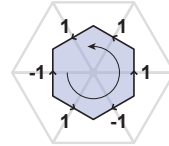
These angles can be used for various mesh processing tasks; we use them to construct direction fields with user-specified singularities (Section 2.7).



**Figure 3:** Loops on a surface can be contractible ( $\zeta$ ) or noncontractible ( $\xi$ ).

### 2.1 Finding 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 collection of dual cells 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). We first construct a matrix  $d_0 \in \mathbb{R}^{|E| \times |V|}$  whose columns span the contractible cycles:

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

Here each column is the boundary cycle of a single dual cell, hence sign is given by relative orientation (we use  $d_0$  to denote this matrix since it is the discrete exterior derivative on 0-forms [DKT08]). Technically, this matrix defines a *spanning set* since only  $|V| - 1$  columns are independent.

We compute a basis for the noncontractible cycles using the *tree-cotree decomposition* of Eppstein [Epp03]:

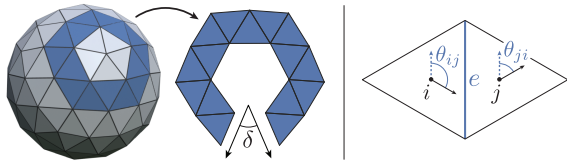
- compute a spanning tree  $T$  of primal edges;
- compute a spanning tree  $T^*$  of dual edges that do not cross edges of  $T$ ;
- for any dual edge not contained in  $T^*$  and not crossed by  $T$ , follow both of its vertices to the root, completing a cycle.

On a surface of genus  $g$ , we get exactly  $2g$  independent noncontractible cycles or *generators*. This basis is again 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 4:** The holonomy of the discrete Levi-Civita connection is the usual angle defect  $\delta$  (left) found by translating a vector across a sequence of unfolded triangles (right).

### 2.2 Computing angle defects

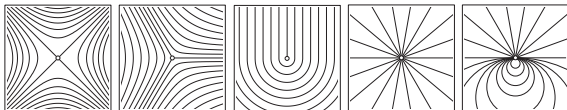
Each cycle in our basis specifies a sequence of dual edges, or equivalently, a sequence of primal triangles. The *angle defect*  $\delta$  of a cycle is simply the angle between initial and final edges when these triangles are unfolded isometrically in the plane (Figure 4, left). More explicitly, given any initial angle  $\alpha_i$  in face  $i$ , we compute a new angle  $\alpha_j$  in neighboring face  $j$  as

$$\alpha_j = \alpha_i - \theta_{ij} + \theta_{ji}, \tag{1}$$

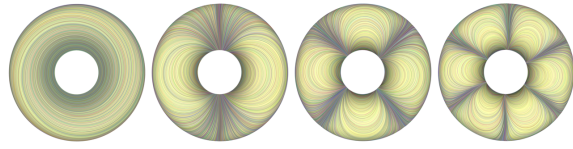
where  $\theta_{ij}$  and  $\theta_{ji}$  are the angles between the shared edge  $e$  and an arbitrary but fixed reference direction in triangles  $i$  and  $j$ , respectively (Figure 4, right). Repeating this procedure for  $n$  consecutive dual edges in a cycle gives us a sequence of angles  $\alpha_0, \dots, \alpha_n$ , and the resulting 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.

### 2.3 Setting 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 5); most vertices will have index zero. (In the discrete case the index is simply the discrete holonomy divided by  $2\pi$  – see Section 3.2.) We can also control the number of rotations experienced by vectors transported around generators (Figure 6). 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 2.6), 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



**Figure 5:** On most surfaces, direction fields must have at least one singularity. Left to right: saddle (-1), tripod (-1/2), thorn (+1/2), focus (+1), apple (+3/2) [Nik01].



**Figure 6:** Constraining the holonomy around generators causes the field direction to “spin” along any noncontractible cycle (generator holonomy from left to right:  $0, 2\pi, 4\pi, 6\pi$ ).

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  $b = [\tilde{K} \tilde{z}]^T$ .

More flexibility is achieved by permitting identification of directions by rotations of  $2\pi/N$  for some fixed  $N \in \mathbb{N}$  (e.g.,  $N = 4$  for cross fields – see Figure 7). 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 [RVAL09].

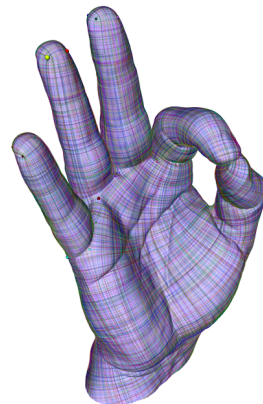
### 2.4 Solving for adjustment angles

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

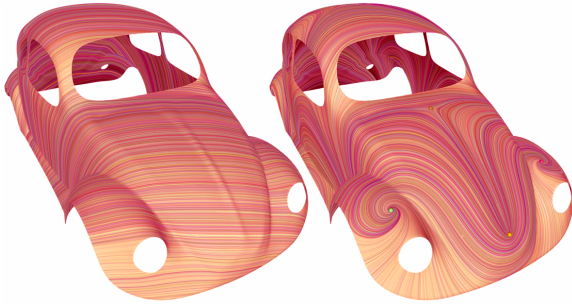
$$\min_x \|x\|_2 \quad \text{s.t.} \quad Ax = -b, \tag{2}$$

which has a unique global minimum at  $x^*$ . Further, the constraints encode the index prescribed at each vertex (see Section 3.4), so we cannot end up with more singularities than we asked for. Note that the system  $Ax = b$  always has solutions – see Appendix A.

At this point, standard algorithms for convex problems (e.g., equality-constrained Newton’s method) could be applied to obtain the minimizer  $x^*$ . However, the simple structure of this problem permits a more efficient approach. Since the system of constraints is underdetermined,  $x^*$  is the unique solution to  $Ax = -b$  that has no component in the kernel of  $A$  – all other solutions have larger  $\ell_2$  norm. One way to compute



**Figure 7:** Our algorithm generates direction fields smooth up to local rotations by multiples of  $2\pi/N$  (here  $N = 4$ ).

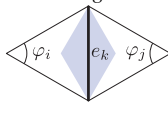


**Figure 8:** Fields on surfaces with boundary do not require singularities (left), but we can easily add singularities and still get natural boundary behavior (right).

$x^*$  is to first find *any* solution  $\tilde{x}$  to the constraint equation  $Ax = -b$  and then project out its null space component. For this operation we need an explicit representation of the null space, which is spanned by the rows of  $d_1$ , the discrete exterior derivative on 1-forms [ES06]. The optimal solution is thus 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 [Dav08].

### 2.5 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 edge (see [DKT08]). Here  $\varphi_i$  and  $\varphi_j$  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.)

### 2.6 Surfaces with boundary

For surfaces with boundary our constraint matrix  $A$  needs to include *boundary loops* and omit cycles around the dual cells associated with boundary vertices. This entails three simple modifications to our algorithm:

- Skip dual cells incident on 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 8 demonstrates the resulting effect.

### 2.7 Constructing direction fields

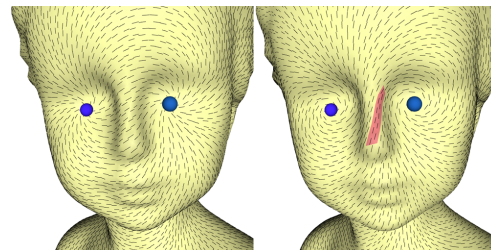
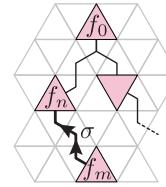
Once we have a vector  $x$  of connection angles, constructing a global direction field is straightforward: starting with an arbitrary face  $f_0$  and initial direction  $\beta_0$ , traverse the primal faces in any order. Across each edge  $e_k$ , compute the angle in the next triangle via

$$\beta_j = \beta_i - \theta_{ij} + \theta_{ji} - x_k. \quad (3)$$

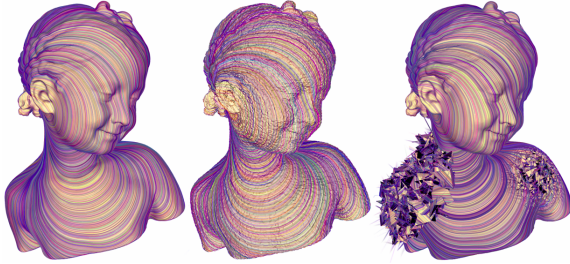
Note that Eq. (3) is just the operation used to compute angle defects (Eq. (1)), augmented with the adjustment angles  $x$ . Because of the way we compute  $x$ , the resulting direction field is independent of traversal order, and is only a function of the choice of  $\beta_0$  (see Section 3.3).

### 2.8 Directional constraints

We can specify a set of faces where the field direction is fixed by prescribing the angle  $\gamma$  in each of these faces (Figure 9). 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  $\sigma$  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  $\gamma_m$  along  $\sigma$  using Equation (1) to get  $\gamma'_m$ , 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  $\gamma_0$ . This way, all directional constraints are satisfied by construction.



**Figure 9:** 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 the specified singularities.



**Figure 10:** Because our method is intrinsic, it is robust to noise (center) and extreme perversions of the input (right).

### 3 Interpretation

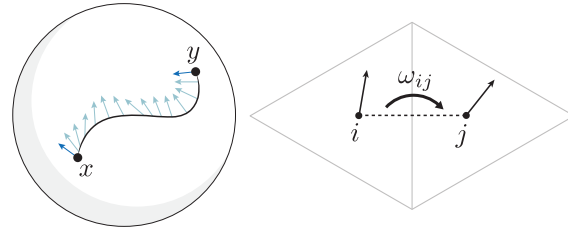
Our algorithm provides a nice link between the smooth geometric perspective and discrete differential geometry [BSSZ08]: quantities computed by our algorithm can be put into direct correspondence with objects from classical Riemannian geometry. In this section we review these concepts and discuss their relationship with our discrete setup.

#### 3.1 Connections

On a smooth surface, a *connection*  $\omega$  describes how a tangent vector changes when moved an infinitesimal distance along a given direction – it “connects” neighboring tangent spaces. Often, we care only about the *direction* of vectors (*i.e.*, not their magnitude). In this case we can think of a connection as associating an *infinitesimal rotation* with each possible direction of motion. (Formally, this describes a *principal connection* on the frame bundle of a smooth surface, which is encoded by an  $\mathfrak{so}(2)$ -valued 1-form.) By integrating these infinitesimal changes along a curve, we can *parallel transport* a vector from one point to another (Figure 11, left).

When developing a discrete theory of connections, the first question is: how should we represent tangent vectors? Storing vectors at vertices is a popular choice [LSLCO05, LJX\*10], but requires that we define approximate tangent planes in terms of extrinsic geometry (*i.e.*, directions in  $\mathbb{R}^3$ ), which can be delicate. Putting vectors on *faces* (as in [KG08, RVAL09]) is perhaps more natural because here tangent directions can be described *intrinsically*, making them well-defined even for poorly discretized surfaces (see especially Figure 10). This setup also leads to a discretization where the curvature associated with the *Levi-Civita* connection is just the usual discrete Gaussian curvature – see Section 3.3.

Given this choice of tangent planes, a discrete connection  $\hat{\omega}$  is easy to define: for each dual edge  $e_k^*$  we store a single angle  $\hat{\omega}_k$  that represents the total (*i.e.*, integrated) rotation of a vector as we travel from one face to the next (see Figure 11, 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 [DKT08], which in our case is *angle* valued. Hence, this interpretation puts us in direct correspondence with the smooth theory: integrating infinitesimal



**Figure 11:** Left: in the continuous setting, a connection determines how tangent vectors change along a curve. Right: a discrete connection can be represented by a rotation angle  $\omega_{ij} = -\omega_{ji}$  associated with each oriented dual edge  $e_{ij}^*$ .

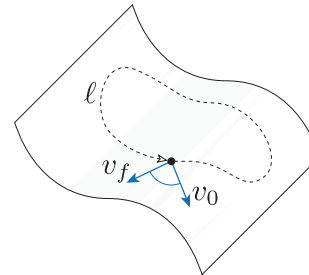
rotations along a path connecting two neighboring faces produces a finite rotation.

Discrete parallel transport also becomes easy to define at this point: starting with an initial direction  $\alpha_0$ , we add consecutive angles  $\hat{\omega}_k$  along a sequence of dual edges. Since each value  $\hat{\omega}_k$  is the integral of infinitesimal rotations along a path from one face to the next, this sum corresponds to piecewise integration of a smooth connection. We use this procedure to construct our direction field in Equation (3). (Note that the discussion of discrete parallel transport in several recent papers [PS98, LJX\*10] refers specifically to transport by the Levi-Civita connection.)

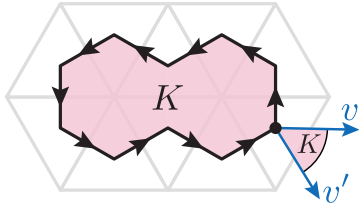
#### 3.2 Holonomy

In general, a vector transported around a closed loop  $\ell$  by a connection  $\omega$  will not return to its original orientation. The difference in angle between the initial and final vector is called the *holonomy* of  $\omega$  around  $\ell$  (Figure 12). Since we have given discrete objects the same names as their smooth counterparts, the definition of *discrete holonomy* is identical: it is the difference in angle found after a vector has been (discretely) parallel transported around a cycle by a (discrete) connection.

Every connection  $\omega$  also has an associated *curvature*. Namely, the curvature of  $\omega$  at a point  $x$  is the holonomy of an infinitesimal loop around  $x$ , and the curvature integrated over any



**Figure 12:** In general, a vector parallel transported around a loop will not end up where it started – the resulting difference is called the holonomy.



**Figure 13:** Every connection has an associated curvature. In the discrete case, the curvature of a region is just the “angle defect” of a vector transported around the region’s boundary.

region  $C$  equals the holonomy around its boundary  $\partial C$ . Since we do not have infinitesimal loops in the discrete case, the latter formulation gives the full story: the curvature of a (discrete) connection  $\hat{\omega}$  over a region  $C$  is the (discrete) holonomy around its boundary cycle  $\partial C$  (Figure 13). The converse is not true, 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.

### 3.3 Trivial connections

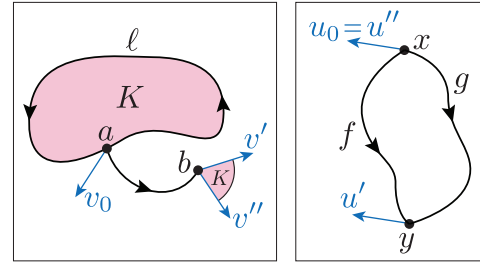
With all this machinery in place, we arrive at the central question: which connection should we use? In other words, how should we transport vectors around the surface in practice?

One answer is given by the canonical *Levi-Civita connection*, whose curvature is the usual Gaussian curvature of the surface [Do 92]. Parallel transport via the *discrete* Levi-Civita connection is computed as in Equation (1), and the resulting holonomy or “angle defect”  $\delta$  around a dual cell corresponds to the standard discretization of Gaussian curvature in terms of vertex tip angles (Figure 4). This choice is popular in computer graphics because it is straightforward to compute and agrees with our usual notion of straightness [PS98].

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 14, 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 14, 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  $g$ .

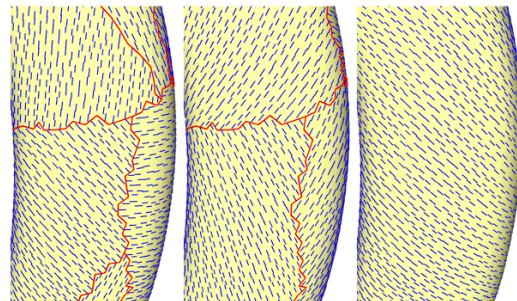
Though not formulated explicitly in terms of connections, this basic premise is the underlying idea in recent work on direction field design [RVAL09, LJX\*10]. Ray *et al.* [RVAL09] effectively compute a connection where *curvature* vanishes and then apply smoothing to obtain a globally consistent result. The reason smoothing is needed here is that curva-



**Figure 14:** Left: transporting a vector  $v_0$  from  $a$  to  $b$  along two different paths may yield different results ( $v'$ ,  $v''$ ) since 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.

ture alone is not sufficient to characterize consistency – as noted earlier, it describes holonomy only around *boundary* cycles (see Figure 15). More recently, Lai *et al.* [LJX\*10] acknowledge the importance of the holonomy around generators, but are concerned that constraining the holonomy around all loops is computationally infeasible. Like Ray *et al.*, 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 over any region 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 2.1). 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 around any cycle should exactly *cancel* the holonomy we find via Levi-Civita (Section 2.2). Implicitly, we are constructing a connection for a surface with a



**Figure 15:** Left: transport via Levi-Civita is not globally consistent, and yields discontinuities in direction (in red). Center: a flat metric improves the situation, but inconsistencies remain. Right: a trivial connection achieves global consistency by constraining all cycles – including noncontractible ones.

flat metric, but expressing this connection with respect to the given embedding. This way we do not need to explicitly construct a flat metric. Notably, however, a *trivial connection* is more specific than a *flat metric* since a trivial connection also has zero holonomy around any noncontractible cycle.

We can now give an interpretation of our objective as well:  $\|Dx\|_2$  is the  $\ell_2$  distance between our trivial connection and the discrete Levi-Civita connection. The diagonal factor  $D$  – or  $\star_1^{1/2}$  in the language of DEC – simply gives the appropriate area-weighted norm on  $x$  (Section 2.5). Overall, then, our optimization problem seeks a globally consistent way to transport vectors that agrees with our usual notion of “straight” (i.e., geodesic) as much as possible.

### 3.4 Singularities

Not every surface admits a trivial connection, however: the Gauss-Bonnet theorem states that the total curvature of a surface is  $2\pi\chi$ , where the Euler characteristic  $\chi$  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 curvature where it will not interfere with the transport of vectors. Since curvature describes the holonomy around region boundaries (Section 3.2), we want the curvature of every region to be an integer multiple of  $2\pi$  so that vectors transported around closed loops are mapped back to themselves – even if they experience several 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  $2\pi$ . In the discrete case, this is equivalent to constraining the holonomy around some small set of vertices (possibly just one) as done in Section 2.3. For surfaces with boundary, we can also concentrate curvature on boundary loops. (Note 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.* [RVLL08] is a straightforward consequence of the Gauss-Bonnet theorem.

### 3.5 Summary

Our computational setup can 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 induces a finite dimensional linear system of sum conditions around discrete cycles of dual edges. The minimum  $\ell_2$  norm solution of this linear system is the minimum  $L_2$  norm solution of the projected energy with respect to the underlying smooth 1-form. The result is a connection with nonzero curvature only at a fixed set of singularities with specified index.



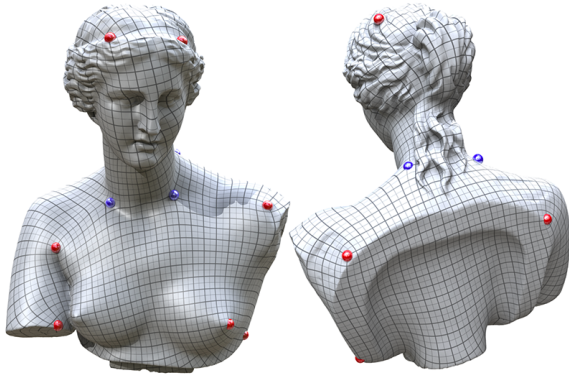
**Figure 16:** Fast direction field editing makes it easy to wrap a T-rex in ribbon (top) or build a horse out of flexible drinking straws (bottom).

## 4 Results

Now that we have discussed some of the theoretical differences between our approach and previous work (Section 3), we focus on performance and robustness. Results are shown in figures throughout; in all examples we were able to achieve exactly the prescribed field topology (Figure 1). Figure 17 shows that our method can be used to drive quadrilateral parameterization algorithms such as *QuadCover* [KNP07], which maps a cross field to a vector field on a multiple covering of the input surface. One nice feature of our approach is that it provides exact matchings between different sheets of the covering, even near singular vertices of large index. Figure 16 shows two artistic applications of our method.

### 4.1 Performance

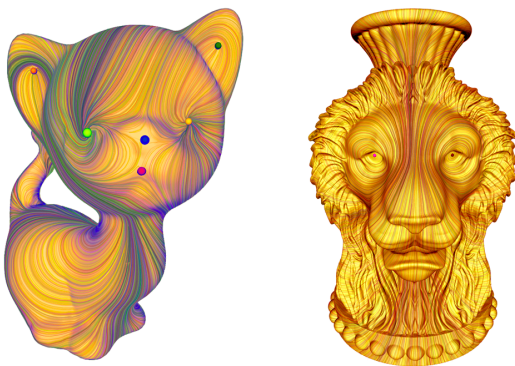
We tested performance on several standard meshes of varying size and element quality. Since singularities and constraint directions depend only on the data vector  $b$ , we can pre-factor our constraint matrix  $A$  and perform edits in real time (Figure 19). Adding directional constraints entails updating  $A$ ; factorization took no more than 9 seconds on our largest model of 400k faces (Figure 18, right). As noted in Section 2.4, our solutions are globally optimal since they are simply the minimum-norm solution to an underconstrained linear system. Overall we observed very consistent perfor-



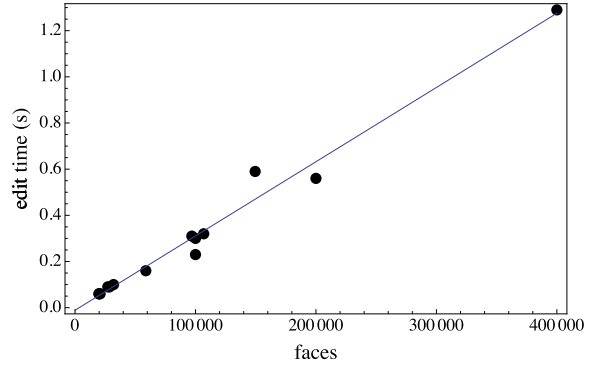
**Figure 17:** The fields we generate can be used as input to QuadCover [KNP07]. Here a small set of hand-picked singularities yields a parameterization with very little distortion.

mance, even on fields with many singularities of large index (Figure 20). A large number of directional constraints can increase the size of the system, though at *most* we will have only  $|E|$  constraints (see Appendix A).

On a mesh of 100k faces, we can edit direction fields roughly 15-48x times faster than with the method described in Ray *et al.* [RVAL09] depending on the convergence rate of their nonlinear solver. Note that their method cannot guarantee optimality since it relies on iterative reprojection onto a non-convex constraint set. The method in Lai *et al.* [LJX\*10] computes a globally optimal solution via discrete Ricci flow, but is nonlinear. Hence we can edit singularities about 25-30x faster (on comparable hardware using the same meshes), and can edit directional constraints at roughly the same rate. Fisher *et al.* [FSDH07] also compute a solution via a single linear solve, but their method may produce unwanted singularities and cannot deal with fractional indices.



**Figure 18:** Left: real-time editing makes it easy to place singularities in locations that are geometrically uninteresting but artistically relevant. Right: even very large meshes (here: 400k faces) can be edited in about a second on a standard laptop.

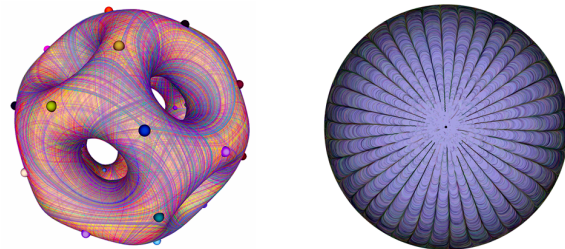


**Figure 19:** Timings of our implementation for all meshes shown in figures (2.4 GHz Core 2 Duo laptop, single thread). On our largest mesh (400k faces), each editing operation took roughly 1.3s after 8.2s of setup time.

#### 4.2 Robustness

As depicted in Figure 21, 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 10), a consequence of the intrinsic, variational nature of our formulation. Note that some angles may have negative cotangents. In this case we simply clamp cotangents to zero when computing area weights (Section 2.5) – 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 20, right) – even on extremely coarse meshes – since we can encode an arbitrarily large amount of “turning” across a single edge. In comparison, methods that store absolute angles per face [RVAL09] or vertex [LJX\*10] may need to refine the mesh or cut out a boundary region near such singularities, since the angle defect around a single vertex can only encode so much curvature.



**Figure 20:** Left: we can easily handle high genus and many singularities (here: 60 singularities,  $g = 11$ ). Right: since we do not need an explicit metric, we have no trouble representing singularities of arbitrarily large index (here: index +20).



## 5 Conclusion

We have provided 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 algorithm a valuable tool for a number of graphics-related applications.

### A Existence of solutions

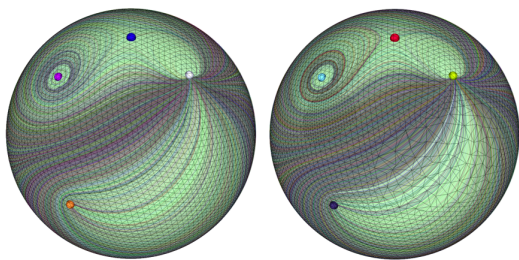
The constraint system  $Ax = -b$  always has at least one solution. For surfaces without boundary, recall that we have  $|V| + 2g$  holonomy constraints: one around each vertex and generator. However, only  $|V| - 1$  of the contractible cycles are independent since the boundary of any dual cell can be expressed as the (negated) sum of all other dual cell boundaries. Directional constraints consist of disjoint subsets of dual edges from a spanning tree ( $T_c$ ) on  $|F|$  faces. Hence, we have at most  $|F| - 1$  mutually independent directional constraints, which are also independent of the holonomy constraints since no collection of paths in  $T_c$  can be combined to form a cycle. In total we have at most  $|V| + |F| + 2g - 2 = |E|$  independent constraints for  $|E|$  degrees of freedom. The matrix  $A$  thus has only one redundant constraint and Stokes' theorem gives us the condition for consistency:  $\sum_i b_i = 0$ , where the sum is over vertices only. This condition holds as long as the indices  $k_i$  sum to  $\chi$  since (by Gauss-Bonnet)

$$\sum_i b_i = \sum_i K_i - 2\pi k_i = 2\pi\chi - 2\pi\chi = 0.$$

For surfaces with boundary, the same argument holds if we imagine that each boundary loop actually bounds a dual cell with the prescribed curvature.

### Acknowledgements

Meshes are courtesy of the AIM@Shape Project, the Stanford 3D Scanning Repository, Jotero GbR, and Hugues Hoppe. Thanks to Felix Kälberer, Matthias Nieser, and Konrad Polthier for parameterizing the Aphrodite model using QuadCover [KNP07]. This research was partially funded by NSF grants (CCF-0635112, CCF-0811373, CMMI-0757106, and CCF-1011944), the Center for the Mathematics of Information at Caltech, and the IAS at TU München.



**Figure 21:** Our discretization yields similar results on different meshes of the same surface.

## References

- [BSSZ08] BOBENKO A. I., SCHRÖDER P., SULLIVAN J. M., ZIEGLER G. M. (Eds.): *Discrete Differential Geometry*, vol. 38 of *Oberwolfach Seminars*. Birkhäuser Verlag, 2008.
- [BZK09] BOMMES D., ZIMMER H., KOBELT L.: *Mixed-Integer Quadrangulation*. *ACM Trans. Graph.* 28, 3 (2009), art. 77.
- [Dav08] DAVIS T.: A multithreaded multifrontal sparse QR factorization. University of Florida, 2008. <http://www.cise.ufl.edu/research/sparse/SPQR/>.
- [DKT08] DESBRUN M., KANSO E., TONG Y.: *Discrete Differential Forms for Computational Modeling*. In Bobenko et al. [BSSZ08].
- [Do 92] DO CARMO M.: *Riemannian Geometry*. Birkhäuser, 1992.
- [Epp03] EPPSTEIN D.: *Dynamic Generators of Topologically Embedded Graphs*. In *Proc. ACM-SIAM Symp. on Discr. Alg.* (2003), pp. 599–608.
- [ES06] ELCOTT S., SCHRÖDER P.: *Building your own DEC at home*. In *ACM SIGGRAPH Course Notes on Discrete Differential Geometry* (2006), pp. 55–59.
- [FSDH07] FISHER M., SCHRÖDER P., DESBRUN M., HOPPE H.: *Design of Tangent Vector Fields*. *ACM Trans. Graph.* 26, 3 (2007), art. 56.
- [HZ00] HERTZMANN A., ZORIN D.: *Illustrating Smooth Surfaces*. In *Proc. ACM/SIGGRAPH Conf.* (2000), pp. 517–526.
- [KG08] KIRCHER S., GARLAND M.: *Free-Form Motion Processing*. *ACM Trans. Graph.* 27, 2 (2008), 1–13.
- [KNP07] KÄLBERER F., NIESER M., POLTHIER K.: *QuadCover - Surface Parameterization using Branched Coverings*. *Comp. Graph. Forum* 26, 3 (2007), 375–384.
- [LJX\*10] LAI Y.-K., JIN M., XIE X., HE Y., PALACIOS J., ZHANG E., HU S.-M., GU X.: *Metric-Driven RoSy Field Design and Remeshing*. *IEEE Trans. Vis. Comp. Graph.* 16 (2010), 95–108.
- [LSLCO05] LIPMAN Y., SORKINE O., LEVIN D., COHEN-OR D.: *Linear Rotation-Invariant Coordinates for Meshes*. *ACM Trans. Graph.* 24, 3 (2005), 479–487.
- [Nik01] NIKOLAEV I.: *Foliations on Surfaces*. Springer, 2001.
- [PFH00] PRAUN E., FINKELSTEIN A., HOPPE H.: *Lapped Textures*. In *Proc. ACM/SIGGRAPH Conf.* (2000), pp. 465–470.
- [PS98] POLTHIER K., SCHMIES M.: Straightest geodesics on polyhedral surfaces. In *Math. Vis.* (1998), pp. 391–398.
- [PZ07] PALACIOS J., ZHANG E.: *Rotational Symmetry Field Design on Surfaces*. *ACM Trans. Graph.* 26, 3 (2007), art. 55.
- [RVAL09] RAY N., VALLET B., ALONSO L., LÉVY B.: *Geometry-Aware Direction Field Processing*. *ACM Trans. Graph.* 29, 1 (2009), 1–11.
- [RVLL08] RAY N., VALLET B., LI W. C., LÉVY B.: *N-symmetry direction field design*. *ACM Trans. Graph.* 27, 2 (2008), 1–13.
- [TACSD06] TONG Y., ALLIEZ P., COHEN-STEINER D., DESBRUN M.: *Designing Quadrangulations with Discrete Harmonic Forms*. In *Proc. Symp. Geom. Proc.* (2006), pp. 201–210.
- [Tur01] TURK G.: *Texture Synthesis on Surfaces*. In *Proc. ACM/SIGGRAPH Conf.* (2001), pp. 347–354.
- [ZMT06] ZHANG E., MISCHAIKOW K., TURK G.: *Vector Field Design on Surfaces*. *ACM Trans. Graph.* 25, 4 (2006), 1294–1326.