Discrete Killing Fields for Pattern Synthesis and Symmetry Detection

Justin Solomon
Stanford University

Satisfying Undergraduate Honors Thesis Requirements for

Department of Computer Science
Advisor: Prof. Leonidas Guibas

and

Department of Mathematics
Advisor: Prof. Richard Schoen

Contents

1 Introduction

Tyger! Tyger! burning bright
In the forests of the night,
What immortal hand or eye
Could frame thy fearful symmetry?
from "The Tyger," in *Songs of Experience* by William Blake [2]

Numerous studies in philosophy, mathematics, and biology have affirmed the ubiquity of symmetry in nature and humans' basic instincts to prefer symmetric patterns over more chaotic configurations. This instinct to seek regularity in a world of complex surfaces has inspired inquiry into effective definitions of symmetry, the evolutionary mechanisms leading to our affinity for symmetry, and tools that use symmetry to enhance aesthetics or understandability.

While the philosophical or biological underpinnings of symmetry might be better explored by other researchers, computer scientists and in particular computational geometers and vision experts have found a number of compelling applications for the detection and exploitation of symmetry. Surfaces can be "symmetrized" to make them more attractive or correct for scanning errors [16]. Repeating structures can be detected and used to extend surfaces, generating new compartments in a shell and additional columns in a coliseum model [20]. In computer vision, symmetry transforms such as [22] are used to guide robotic grasping and other practical tasks. Symmetry also can assist in the detection of cars, faces, and other biological and man-made shapes [4, 23].

Exploratory studies in symmetry detection including those cited above show great promise for revolutionizing geometry analysis. Robust and efficient symmetry detection could be applied to any number of applications in storage, modeling, and surface generation. For instance, starting with some basic pieces of geometry, an approximation of a surface could be built up using simple descriptions of how the geometry repeats and changes along a base layer. Such a structure could be used to reduce the complexity of geometry storage, enabling software in rendering, topography, and similar applications to process much more data with less redundancy and required space; rather than storing several copies of a repeated element, a single copy can be stored along with a list of ways in which it repeats. In modeling, a description of a surface's repeating elements could help fabricate new geometry in the same way that texture synthesis algorithms use small swatches to generate larger patches with a similar look. Additionally, modifying the data describing a shape's composition could generate new shapes with varying global structure but similar local detail or vice versa.

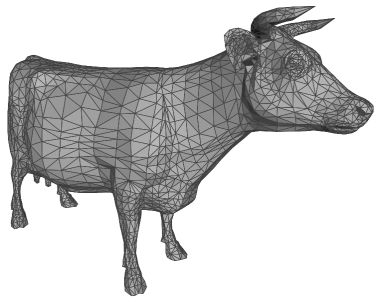
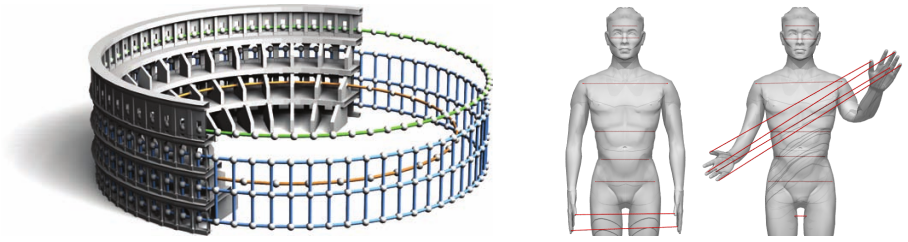


Figure 1: A simple triangle mesh.

More generally, the construction of algorithms for symmetry analysis suggests an important transition from localized to semantic expressions of shape and form. The vast majority of methods for geometry processing involves low-level structures such as points, lines, triangles, and splines. By far the most common structure for storing surfaces is the *polygonal mesh*, in which a smooth surface is approximated by a large number of flat polygonal facets; Figure 1 shows one example



(a) (b)

Figure 2: (a) Symmetries detected by [14] (Figure 1); (b) symmetries detected by [19] (Figure 1).

of a triangle mesh. Even relatively simple shapes other than basic geometric primitives can take thousands of flat faces to express effectively; for instance, the cow in Figure 1 contains 5,804 triangular faces and 4,583 vertices. Perceptually, however, we hardly see 5,804 individual components. Instead, we can observe a number of symmetries and relationships between different parts of the model, including a reflective symmetry from head to back, translational symmetries between the legs, and rotational symmetries on the horns and eyes. Of course, all these relationships are approximate; superimposing one of the cow's legs on top of another reveals that they do not have exactly the same geometry. Even so, any competent artist could reproduce similar geometry given a paragraph or two of description rather than a list of over 4,500 points in $\mathbb{R}^3$ and the polygonal topology linking them together.

While geometry analysis has not yet reached the level of sophistication required to give as intuitive descriptions as we might hope for the model in Figure 1, some recent papers show promising first steps toward this valuable end. For instance, a figure from [14] is reproduced in Figure 2(a), showing one result from the symmetry detection algorithm developed by the authors; we see that the algorithm is able to detect several rows of repeating columns despite drawing from a model that is not a complete circle. Similarly, Figure 2(b) shows that [19] is able to detect several symmetries of the human model, even when its arm is bent in different positions on the left and right sides.

The positive results described in these and other papers serve as promising signs that symmetry detection is a feasible and valuable line of inquiry. Still, no single algorithm, pipeline, or system has proven effective in all or even most practical cases for even a single application of computational symmetry. There are several computational challenges that suggest the difficulty of geometric symmetry detection and may explain why it remains such an open problem:

Defining symmetry: Writing a rigorous mathematical definition of "symmetry" can be a fairly vague task. While humans have a strong intuition for detecting structure and regular patterns, differential geometers, topologists, computer scientists, and other researchers all have different notions of "strong" and "weak" symmetry that are useful for varying applications. Since no single definition of symmetry will satisfy all end users, it is likely that no single general-purpose method for its analysis will suffice. This overarching problem likely is the principal point of difficulty for all other challenges listed below.

Intrinsic versus extrinsic geometry: Early attempts at symmetry analysis focused on the detection and use of *extrinsic* symmetries. In extrinsic geometry, a shape's embedding in an

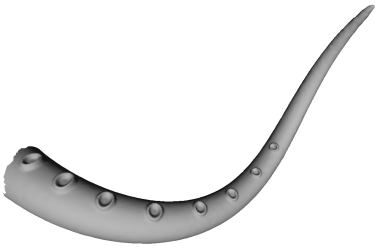
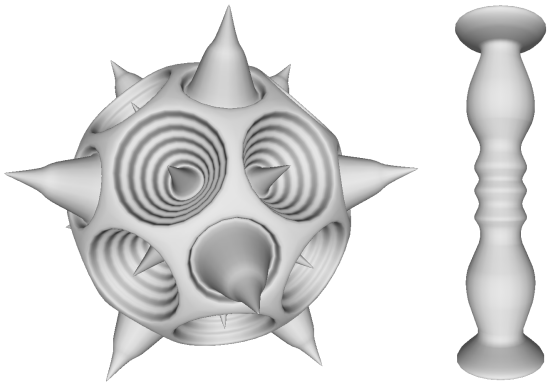


Figure 3: The suckers on this tentacle model have a clear intrinsic symmetry.



(a) (b)

Figure 4: Discrete (a) and continuous (b) symmetry.

ambient space is used to understand its structure. Such a viewpoint makes it easy to detect repeating windows on a building or circular indentations on a can model, but the structure of the suckers on the tentacle model of Figure 3 would not be detected because they are not on a straight line. Instead, more recent papers take an *intrinsic* approach, in which symmetry is analyzed in such a way that it is preserved when a surface is deformed. This approach is more flexible, but it requires relatively sophisticated mathematical structures even to define symmetry rigorously from such a generic viewpoint.

Structure detection: Pairwise symmetry detection can be an unwieldy tool. For instance, a naïve system for detecting rotational symmetries might find $45 = \binom{10}{2}$ rotational symmetries in a pinwheel with 10 spokes, one for each pair of spokes that could be rotated into one another. In reality, these all should be instances of the same rotational symmetry. Detecting that two features are related and describing the relationship between a larger series of features, however, are two related but contrasting tasks. The latter involves not only grouping similar features but also finding a compact expression showing how they related to each other.

Continuous versus discrete symmetry: There are abundant examples of both discrete and continuous symmetry patterns. For instance, the model in Figure 4(a) has discrete symmetry

because the cone-shaped features must be translated a positive distance before they align with each other, while the model in Figure 4(b) has a continuous set of rotational symmetries. Discrete problems carry over naturally to the computational setting, while continuous ones often require a different class of approximate solutions. This difference manifests itself particularly strongly in geometry, where even the mathematical theories dealing with discrete and continuous symmetries differ in several fundamental ways before introducing computational approximations.

Such challenges will lead symmetry detection to be a fertile area of research in geometry processing for the long term.

This paper describes several new approaches for handling an important case of geometric symmetry analysis, that of *continuous intrinsic symmetries and approximate symmetries*. This case has been neglected in recent literature involving symmetry analysis due to the lack of a reliable computational framework for dealing with flows and derivatives of vector fields on manifold surfaces, the building blocks of smooth Riemannian geometry. In particular, we provide methods for computing Killing vector fields on triangular meshes, whose flows represent infinitesimal local isometries for surfaces admitting global continuous symmetries. For surfaces that are nearly symmetric, we develop in parallel the concept of an "approximate Killing field," which reliably finds flows that help classify the local structure of a surface. With a stable Killing field computation technique, we can integrate to find paths of vertices along a mesh. These paths explicitly illustrate surface symmetries and can form the basis for straightforward approaches to texture repetition, geometry synthesis, and pattern analysis techniques that make use of local structure to guide computation.

In an effort to maintain a rigorous approach to discrete Riemannian geometry, we proceed in Section 2 with a review of the relevant mathematics applied in our and others' methods. We also discuss the difficulty of applying some common theoretical techniques such as parallel transport to the particular application and setting at hand. We continue in Section 3 with a review of more recent papers using approaches from differential geometry to understand discrete surfaces. We introduce algorithms for computing Killing fields on triangle meshes in Section 4 and analyze the accuracy of these methods in Section 5; Section 6 outlines one potential application of these methods. Finally, we conclude with a short listing of topics for future study in Section 8.

2 Mathematical Background

The construction of our algorithms for surface analysis and symmetry detection is motivated by and in several cases derived from the case of smooth manifolds. While some of the constructions and indirect proofs of existence used in the smooth case might not be amenable to discretization, they provide important intuition and suggest the proper geometric tools to adapt to triangle meshes.

2.1 Preliminaries

Although many of the concepts discussed here can be extended to n-dimensional manifolds with little to no additional justification, in the discrete setting we will consider only two-dimensional

submanifolds M of $\mathbb{R}^3$, or surfaces. Specifically, we will be dealing with "classical" surfaces satisfying the following simple definition:[1]

Definition 1 (Surface)**.** *A surface is a subset $M \subset \mathbb{R}^3$ such that for all $p \in M$ there exists open $U \subseteq \mathbb{R}^3$ such that $p \in U$ and $M \cap U$ is diffeomorphic to $\mathbb{R}^2$.*

Fortunately, many of the most abstract aspects of a more general Riemannian approach are avoided by considering surfaces rather than manifolds. Most importantly, such an embedding immediately provides us with a metric induced by the inner or "dot" product in $\mathbb{R}^3$. Thus, discussions of quantities such as angles, lengths, and so forth are concrete descriptions of measurable quantities rather than intuitive notions suggesting a more abstract and general theory. Also, note that our definition of a surface above precludes the existence of a boundary; eventually we will need to consider slightly more general surfaces for which a neighborhood around each point p is diffeomorphic either to $\mathbb{R}^2$ or to $\mathbb{R} \times [0, \infty)$.

While we assume the reader is familiar the notion of a (differentiable) map between manifolds as well as the more specific case of a single-valued function on a surface, we are particularly concerned with vector fields on surfaces and thus need to define the tangent space at a point:

Definition 2 (Tangent space)**.** *The tangent space of M at $p \in M$, denoted T_pM, is the set of vectors $\gamma'(0)$ for all curves $\gamma : (-\varepsilon, \varepsilon) \to M$ with $\gamma(0) = p$.*

Of course, our intuition for the tangent space of a surface M at a point p generally involves an affine plane through p tangent to M, and fortunately our intuition is reasonable:

Theorem 1. *For all $p \in M$, T_pM is a two-dimensional subspace of $\mathbb{R}^3$.*

Proof. Take any curve γ as defined above, and take a coordinate chart $\phi : U \to M$ with $\phi(0, 0) = p$; assume ε is sufficiently small so that $\gamma((-\varepsilon, \varepsilon)) \subset \phi(U)$. Then, we can define $\psi = \phi^{-1} \circ \gamma : (-\varepsilon, \varepsilon) \to \mathbb{R}^2$, such that $\gamma'(0) = (\phi \circ \psi)'(0) = D\phi|_{(0,0)} \cdot \psi(0)$. Thus, we have shown that $\gamma'(0)$ is in the column space of matrix $D\phi|_{(0,0)}$, or more generally that $T_pM \subseteq \text{col } D\phi|_{(0,0)}$. Conversely, any vector $\vec{v} \in \text{col } D\phi|_{(0,0)}$ also is in T_pM since $\vec{v} = \frac{d}{dt}\phi(t(D\phi|_{(0,0)})^{-1}\vec{v})|_{t=0}$. So, $T_pM = \text{col } D\phi|_{(0,0)}$, which is a two-dimensional subspace of $\mathbb{R}^3$ since $\phi : \mathbb{R}^2 \to M \subset \mathbb{R}^3$ is a diffeomorphism. □

Occasionally, it will be useful to deal with the set of tangent spaces rather than the space at a single point $p \in M$. For this purpose, we define the *tangent bundle* as the disjoint union of tangent spaces:

$$\begin{aligned} TM &= \coprod_{p \in M} T_pM \\ &= \bigcup_{p \in M} \{p\} \times T_pM \\ &= \{(p, \vec{v}) : p \in M, \vec{v} \in T_pM\} \end{aligned} \tag{1}$$

With these definitions and a basic understanding of tangent planes in place, we finally can define a vector field on a surface:

[1] In this paper, we will assume that all functions, maps, vector fields, and other similar objects are sufficiently differentiable or C^∞ to satisfy concerns about existence or convergence.

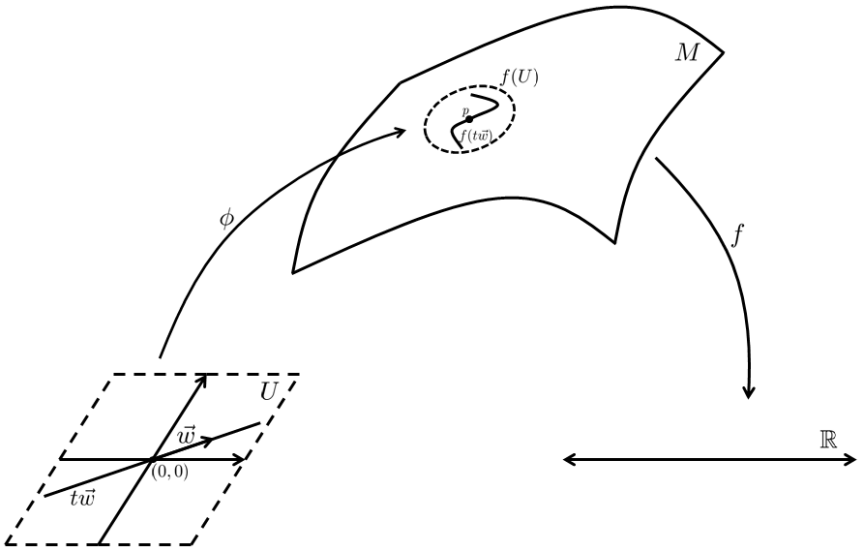


Figure 5: Setup for the Definition 4.

Definition 3 (Vector field)**.** *A vector field X on M is a smooth map from M to TM such that $X(p) \in \{p\} \times T_pM$ for all $p \in M$.*

For practical purposes, we might want to isolate vectors in a field from their accompanying base point; such an operation can be accomplished simply by the projection $\pi : TM \to \mathbb{R}^3$ taking $(p, \vec{v})$ to $\vec{v}$.

2.2 Derivatives and Rates of Change on Surfaces

For a smooth function $f : M \to \mathbb{R}$, we define the differential df_p at a point $p \in M$ the usual way:[2]

Definition 4 (Differential)**.** *Take $\phi : U \subseteq \mathbb{R}^2 \to M$ to be a coordinate chart with $\phi(0, 0) = p$. Take $\vec{v} \in T_pM$ and define $\vec{w} = (D\phi)^{-1}|_{(0,0)}\vec{v}$. Then, the differential $df_p : T_pM \to \mathbb{R}$ is given by*

$$df_p(\vec{v}) = \frac{d}{dt}(f \circ \phi(t\vec{w}))|_{t=0} \tag{2}$$

Figure 5 illustrates the definition above; it is straightforward to see that $df_p(\vec{v})$ defines the directional derivative of f in direction $\vec{v}$. We refer the reader to any basic differential geometry text for proofs that df_p is linear and independent of our choice of ϕ [17]. The differential of a map between surfaces has a similar definition and is no more complicated.

Unlike functions on Euclidean domains, the definition of "the derivative" of a function f on M rather than at a single point p is fairly complicated, involving differential forms in a construction

[2]The development in this and the next section roughly follows the outline of an introductory lecture by Adrian Butscher.

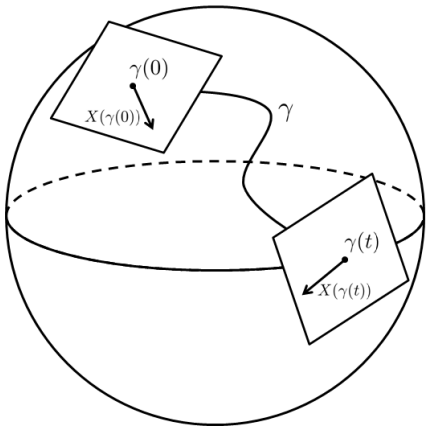


Figure 6: Vectors at $\gamma(0)$ and $\gamma(t)$ are on different tangent planes and cannot be compared directly.

that amounts to describing formally a functional giving df_p simultaneously at each point $p \in M$. Given a vector field X on M, however, we can pose a much more straightforward problem: What is the derivative of f in the direction given by X at each point on M? In some sense, the answer to this question could be considered the action of X on f and thus is notated $X(f)$. In the notation developed above, we write:

$$[X(f)](p) = df_p(X(p)) \tag{3}$$

Note that we identify $X(p)$ and $\pi \circ X(p)$ for ease of notation. So, as expected for $f : M \to \mathbb{R}$ we also have $X(f) : M \to \mathbb{R}$.

Generalizing slightly, we might ask for the rate of change of one vector field Y in the direction of another vector field X. Initially, such a problem might seem straightforward: after all, we could differentiate each of the component functions $\pi \circ Y_i : M \to \mathbb{R}$ using the differential operator defined above to obtain some idea of such a derivative. This naïve definition, however, is not useful from a geometric standpoint. To see why, suppose we are considering the slightly simpler problem of differentiating a vector field X along a curve γ. Then, the definition above amounts to computing the following limit:

$$\lim_{t \to 0} \frac{X(\gamma(t)) - X(\gamma(0))}{t}$$

The difference $X(\gamma(t)) - X(\gamma(0))$, however, is ill-defined: $X(\gamma(t)) \in T_{\gamma(t)}M$ but $X(\gamma(0)) \in T_{\gamma(0)}M$, as shown in Figure 6. In other words, our only notion of a "vector" on M is in an individual tangent space, and we have not yet developed a way of identifying tangent spaces at different points in a geometrically reasonable fashion.[3]

Obviously, we must be more careful in our definition of such a derivative. Rather than constructing an operator with only a vague idea of how it should behave, we take an axiomatic approach, describing desired properties of the operator and proving that it exists and is unique.

[3]Note a potential source of confusion: Such a limit is computable in the numerical sense, since for simplicity we identified T_pM with a subspace of $\mathbb{R}^3$. This identification, however, is not "intrinsic" in the sense that it depends on the embedding of M in $\mathbb{R}^3$.

Specifically, we desire an operator ∇ giving derivative $\nabla_X Y$ of Y in direction X with the following properties (for vector fields X, Y, and Z and functions f and g all on surface M):

Linearity: $\nabla_{X+fY} Z = \nabla_X Z + f\nabla_Y Z$ – This requirement enforces an analog of the fact that df_p is linear in its argument. Geometrically, this property expresses the fact that the derivative of Z in a direction given by the sum of two vector fields or scaling a vector field should be the sum or scale of the derivatives.

Additivity: $\nabla_X(Y+Z) = \nabla_X Y + \nabla_X Z$ – Our remaining two rules are motivated by corresponding identities in basic calculus. Ensuring additivity just reflects that the derivative of the sum of two functions is the sum of the functions' derivatives.

Leibniz (Product) Rule: $\nabla_X(fY) = X(f)Y + f\nabla_Y X$ – This rule reflects the product rule from calculus on $\mathbb{R}^n$. Initially, it may seem to be at odds with the linearity condition above. To understand the distinction, we observe that we are interested in the variation of fY along M, while the variation of X along M is less important. In other words, the value of X already gives the desired derivative direction independent of values nearby, and thus scaling X by f should have little effect (hence $\nabla_{fX} Y = f\nabla_X Y$), while scaling Y can have considerable effects on its directional derivatives on M. In the simplest case, consider scaling a constant vector field Y on $\mathbb{R}^2$ by a non-constant f; such an operation clearly introduces spatial variation in Y that was not there before.

An operator satisfying the constraints above is known as a *covariant derivative* and is one of the basic building blocks of Riemannian geometry. Still, while such an operator would be convenient, we have yet to prove that it exists.

In a sufficiently small neighborhood U of a given point $p \in M$, we can define vector fields E_1 and E_2 spanning $T_{\tilde{p}}M$ for all $\tilde{p} \in U$ using coordinate-wise derivatives of a chart $\phi : V \subseteq \mathbb{R}^2 \to U$. Then, for a given pair of vector fields there exist smooth functions $a^i, b^i : U \to \mathbb{R}$ satisfying:

$$X = \sum_i a^i E_i \tag{4}$$

$$Y = \sum_i b^i E_i \tag{5}$$

Applying the properties of ∇ defined above, we find:

$$\begin{aligned}
\nabla_X Y &= \nabla_{\sum_i a^i E_i} \left[\sum_j b^j E_j \right] \text{ by definition of } X \text{ and } Y \\
&= \sum_i a^i \nabla_{E_i} \left[\sum_j b^j E_j \right] \text{ by linearity} \\
&= \sum_{i,j} a^i \nabla_{E_i} (b^j E_j) \text{ by additivity} \\
&= \sum_{i,j} \left[a^i E_i(b^j) E_j + a^i b^j \nabla_{E_i} E_j \right] \text{ by the Leibniz rule}
\end{aligned}$$

Note that in $\mathbb{R}^2$ we can use the constant canonical basis for E_1 and E_2, making the $\nabla_{E_i} E_j$ term vanish and leaving the standard directional derivative formula. More generally, since ∇ outputs vectors in the tangent plane, we can write the directional derivatives $\nabla_{E_i} E_j$ again in the basis $\{E_1, E_2\}$ as follows:

$$\nabla_{E_i} E_j = \sum_{k=1}^{2} \Gamma_{ij}^k E_k \tag{6}$$

The functions Γ_{ij}^k are known as the *Christoffel symbols* and provide a convenient local description of vector fields.

The local formulation above shows that our concerns about the existence of ∇ were unfounded. Indeed, any consistent choice of functions Γ_{ij}^k will satisfy the definition of a covariant derivative, so we are free to specify *more* desired properties of our vectorized directional derivative operator. Thus, we introduce two more conditions useful for our type of geometric analysis; note that these are by no means the only properties one might desire, and different choice give rise to different covariant derivatives:

Metric Preservation: $X(\langle Y, Z\rangle) = \langle \nabla_X Y, Z\rangle + \langle Y, \nabla_X Z\rangle$ – Here, we use $\langle \cdot, \cdot \rangle$ to denote the usual inner product on $\mathbb{R}^3$. This property is the first we have introduced to specify the interaction between inner products and vector fields.[4] It can be understood as an additional Leibniz rule, forcing inner products to act similarly to products of functions.

Symmetry: $\nabla_X Y - \nabla_Y X = [X, Y]$ – This condition is sometimes referred to as the "torsion-free" condition [7]. First, we must define the vector field $[X, Y]$, known as the "Lie bracket" of X and Y. Most simply, we can express $[X, Y]$ as the (unique) vector field Z for which the action on a function f satisfies $Z(f) = (XY - YX)f$. In local coordinates, it is straightforward to show:

$$[X, Y] = \sum_{i=1}^{2} E_i \sum_{j=1}^{2} \left(a^j \frac{\partial b^i}{\partial x^j} - b^j \frac{\partial a^i}{\partial x^j} \right) \tag{7}$$

We will not prove it here, but the torsion-free condition ensures that if we use ∇ to construct geodesics, or locally length-minimizing curves, the resulting curves will have no torsion, or twist.

One of the basic theorems of Riemannian Geometry is that the five conditions we have chosen define a unique operator on any n-dimensional manifold; this unique symmetric and metric-preserving covariant derivative ∇ is known as the Levi-Civita Connection. We will prove the theorem in the two-dimensional case, although the more general proof is no more difficult. We begin with two lemmas:

Lemma 1. *For functions f and g and vector fields X and Y, we have*

$$[fX, gY] = fg[X, Y] + fX(g)Y - gY(f)X \tag{8}$$

[4]Note that inner products are taken within the same tangent space, and thus our earlier concerns about incompatibility are irrelevant.

Proof. Since we defined the Lie bracket in terms of its action on a function, our proof simply verifies that the action of the left- and right-hand sides of (8) on a function h is the same:

$$\begin{aligned}[fX, gY](h) &= fX(gY(h)) - gY(fX(h)) \text{ by definition of } [\cdot,\cdot] \\ &= f(X(g)Y(h) + gXY(h)) - g(Y(f)X(h) + fYX(h)) \text{ by the product rule} \\ &= fg[X,Y](h) + fX(g)Y(h) - gY(f)X(h) \text{ as desired.}\end{aligned}$$

□

The second lemma is accompanied by a definition:

Definition 5. *(Tensorial functional) A linear functional (or one-form) ω mapping vector fields to functions on surfaces is* tensorial *if it satisfies $\omega(fX) = f\omega(X)$ for all functions f and vector fields X.*

With this definition in place, we state the second lemma:

Lemma 2. *All tensorial one-forms ω can be written*

$$\omega(X) = \langle Y, X\rangle \tag{9}$$

for some vector field Y.

Proof. At a particular point p, take E_1, E_2 to form a local unit-length orthogonal coordinate basis (which exists by the Gram-Schmidt process) and define $Y(p) = \omega(E_1)E_1 + \omega(E_2)E_2$.

We show that our choice of basis is irrelevant. Suppose $\tilde{E}_1, \tilde{E}_2$ is another such basis. Then for some smooth functions a_{ij} we can write $\tilde{E}_1 = a_{11}E_1 + a_{12}E_2$ and $\tilde{E}_2 = a_{21}E_1 + a_{22}E_2$, yielding:

$$\begin{aligned}\omega(\tilde{E}_1)\tilde{E}_1 + \omega(\tilde{E}_2)\tilde{E}_2 = \omega(a_{11}E_1 + a_{12}E_2)(a_{11}E_1 + a_{12}E_2) + \\ \omega(a_{21}E_1 + a_{22}E_2)(a_{21}E_1 + a_{22}E_2)\end{aligned}$$

Applying the definition of tensorial one-forms and collecting terms, we find

$$\begin{aligned}\omega(\tilde{E}_1)\tilde{E}_1 + \omega(\tilde{E}_2)\tilde{E}_2 &= E_1[\omega(E_1)(a_{11}^2 + a_{21}^2) + \omega(E_2)(a_{11}a_{12} + a_{21}a_{22})] + \\ &\quad E_2[\omega(E_1)(a_{11}a_{12} + a_{22}a_{21}) + \omega(E_2)(a_{12}^2 + a_{22}^2)] \\ &= Y(p)\end{aligned}$$

since the matrix (a_{ij}) is orthogonal. Thus, our choice of $Y(p)$ does not depend on coordinate basis and is well-defined.

All that remains is to verify (9). In local coordinates described above, write $X = x^1E_1 + x^2E_2$. Then,

$$\begin{aligned}\omega(X) &= \omega(x^1E_1 + x^2E_2) \\ &= x^1\omega(E_1) + x^2\omega(E_2) \\ &= x^1\langle Y, E_1\rangle + x^2\langle Y, E_2\rangle \\ &= \langle Y, X\rangle\end{aligned}$$

as desired. □

Now, we proceed with a proof of the Fundamental Theorem:

Theorem 2 (Fundamental Theorem of Riemannian Geometry)**.** *There exists a unique metric-preserving, torsion-free covariant derivative operator ∇ on vector fields on M.*

Proof. We follow the more general proof presented in [11]. We first prove uniqueness. Consider three vector fields X, Y, and Z. Differentiating the inner product of any two in the direction of a third yields the following three formulae using the the metric preservation condition:

$$X(\langle Y, Z\rangle) = \langle \nabla_X Y, Z\rangle + \langle Y, \nabla_X Z\rangle \tag{10}$$
$$Y(\langle Z, X\rangle) = \langle \nabla_Y Z, X\rangle + \langle Z, \nabla_Y X\rangle \tag{11}$$
$$Z(\langle X, Y\rangle) = \langle \nabla_Z X, Y\rangle + \langle X, \nabla_Z Y\rangle \tag{12}$$

Adding (10) and (11) and subtracting (12) yields the following expression:

$$\begin{aligned} &X(\langle Y, Z\rangle) + Y(\langle Z, X\rangle) - Z(\langle X, Y\rangle) \\ &\quad = \langle Z, \nabla_X Y + \nabla_Y X\rangle + \langle Y, \nabla_X Z - \nabla_Z X\rangle + \langle X, \nabla_Y Z - \nabla_Z Y\rangle \end{aligned} \tag{13}$$

With the factorization suggested above, we immediately apply symmetry twice to obtain:

$$X(\langle Y, Z\rangle) + Y(\langle Z, X\rangle) - Z(\langle X, Y\rangle) = \langle Z, \nabla_X Y + \nabla_Y X\rangle + \langle Y, [X, Z]\rangle + \langle X, [Y, Z]\rangle \tag{14}$$

Additionally, symmetry implies that $\nabla_Y X = \nabla_X Y - [X, Y]$, so

$$\begin{aligned} &X(\langle Y, Z\rangle) + Y(\langle Z, X\rangle) - Z(\langle X, Y\rangle) \\ &\quad = \langle Z, 2\nabla_X Y - [X, Y]\rangle + \langle Y, [X, Z]\rangle + \langle X, [Y, Z]\rangle \end{aligned} \tag{15}$$

Finally, we reorganize terms to obtain the relationship:

$$\begin{aligned} &\langle Z, \nabla_X Y\rangle = \\ &\quad \frac{1}{2}\left\{X(\langle Y, Z\rangle) + Y(\langle Z, X\rangle) - Z(\langle X, Y\rangle) + \langle Z, [X, Y]\rangle - \langle Y, [X, Z]\rangle - \langle X, [Y, Z]\rangle\right\} \end{aligned} \tag{16}$$

Equation 16 is the well-known Koszul formula for $\nabla_X Y$. It gives an explicit form for the inner product of $\nabla_X Y$ and any other vector field Z; substituting basis vector fields E_1 and E_2 completely determines $\nabla_X Y$. Since the right hand side of (16) is in terms of inner products, directional derivatives, and Lie brackets–all of which are well-defined and understood–we have established uniqueness of $\nabla_X Y$. We have *not*, however, established existence of $\nabla_X Y$ satisfying our criteria: (16) only establishes that if such an operator were to exist, it would behave in a certain way with respect to the inner product operator $\langle \cdot, \cdot \rangle$.

To establish existence, fix two vector fields X and Y and take ω to be the linear functional mapping input field Z to the right-hand side of (16). Applying Lemma 1 and the product rule, we obtain the following relationship; intermediate steps are omitted since they are notationally cumbersome and straightforward:

$$\begin{aligned} \omega(fZ) &= f\omega(Z) + \frac{1}{2}\left\{X(f)\langle Y, Z\rangle + Y(f)\langle Z, X\rangle - X(f)\langle Y, Z\rangle - Y(f)\langle X, Z\rangle\right\} \\ &= f\omega(Z) \end{aligned}$$

Thus, ω is tensorial, and by Lemma 2 there is a unique field A such that $\omega(Z) = \langle A, Z\rangle$ for all Z.

We show $A = \nabla_X Y$. Clearly our choice of A is additive and linear since ω is tensorial and additive and (16) is additive in X and Y. The Leibniz rule can be derived directly from (16) by applying the standard product rule from calculus and substituting into the construction in Lemma 2. Our construction ensures that (16) holds for $\nabla_X Y$, and adding the right hand side of this formula for $\langle Z, \nabla_X Y\rangle$ and $\langle Z, \nabla_Y X\rangle$ immediately the metric preservation property upon simplification. Similarly, subtracting these two expression yields symmetry, completing the proof. □

We conclude this section with a characterization of covariant differentiation for surfaces, or– in the language of Riemannian geometry–two-dimensional submanifolds of $\mathbb{R}^3$. Take M to be a surface and X to be a tangent vector field. Take $U \subseteq \mathbb{R}^3$ to be an open set containing M, and suppose we can extend X to a vector field $X_{ext} : U \to \mathbb{R}^3$; note that simple examples of constructing X_{ext} can be found in sufficiently small neighborhoods U of M by writing $X_{ext}(p + tN_p) = X(p)$ for $p \in M$, $t \in (-\varepsilon_p, \varepsilon_p)$, and normal vector N_p at p. Since X_{ext} is a function on an open subset of $\mathbb{R}^3$, we can define its Jacobian DX_{ext} in the usual way. In particular, we can find the derivative of X_{ext} in direction Y using matrix multiplication:

$$D_Y X_{ext} = DX_{ext} \cdot Y \tag{17}$$

With this construction in hand, we apply the Fundamental Theorem of Riemannian Geometry to make the following claim:

Corollary 1. *The covariant derivative at $p \in M$ is given by*

$$\nabla_Y X(p) = \mathrm{proj}_{T_pM} D_Y X_{ext}(p) \tag{18}$$

Proof. By the Fundamental Theorem of Riemannian Geometry, we simply need to check that ∇ is a metric-preserving, torsion-free covariant derivative operator. Linearity, additivity, the Leibniz rule, and metric preservation all follow from linearity of the projection operator and the corresponding rules from multivariable calculus. By Equation 7, since X and Y are tangent vectors to M, so is $[X, Y]$ when computed using the ambient space and thus ∇ is symmetric, completing the proof. □

Equation 18 provides an elegant intuition for the covariant derivative on surfaces rather than more general n-dimensional manifolds. It shows that the unique covariant derivative of a vector field on M is simply the tangential component of its derivative in $\mathbb{R}^3$. Indeed, this description not only is more intuitive but also more direct; Equation 18 gives a straightforward way to compute $\nabla_Y X$ on surfaces. This observation will be useful in our formulation of discrete Killing fields in Section 4.

2.3 Parallel Transport

The Levi-Civita Connection provides a straightforward mechanism for moving vectors along a curve on a surface so that they maintain length and minimize twisting. Take a curve $\gamma(t)$ on M and vector $X \in T_pM$, where $\gamma(0) = p$. Then, we can define $X(t)$ by the ordinary differential equation

$$\nabla_{\dot\gamma} X = 0 \tag{19}$$

The function $X(t)$ is known as the *parallel transport* of X along γ. In general, we denote the parallel transport of a vector X along a curve γ by $PT_\gamma(X)$.

Thankfully, as shown in [24], by the Fundamental Existence and Uniqueness Theorem for ODEs such an $X(t)$ must exist for all t for which $\gamma(t)$ is defined. Intuitively, Equation 19 requires that $X(t)$ not accelerate tangentially to the curve γ. Note that on $\mathbb{R}^2$ the solution to the geodesic equation is constant; this result corresponds to the notion in elementary physics that vectors represent a constant displacements regardless of their location on the plane.

Initially, it may seem likely that parallel transport would be a useful tool in defining symmetries on surfaces. For instance, translational symmetries in $\mathbb{R}^n$ could be mimicked by translating features along curves using orientations computed using parallel transport. Unfortunately, simplistic attempts at such a formulation fail for the following important reason:

For two curves γ_1 and γ_2 with identical beginning and end points and vector $Y \in T_pM$, it is possible and indeed likely that $PT_{\gamma_1}(Y) \neq PT_{\gamma_2}(Y)$.

In fact, for a given point p one can define the *holonomy group* Hol_p as the set of isometries of T_pM obtained by parallel transporting tangent vectors along loops beginning and ending at p [21]. Holonomy groups are central objects of study in geometry and topology and unsurprisingly generally are non-trivial; for instance, the holonomy group of any point on the sphere S^2 is $SO(2)$. In fact, in some sense the Riemann curvature tensor measures the amount that parallel transport locally will not return tangent vectors to their starting direction.

While this "failing" of parallel transport to be self-consistent may lead to useful definitions of curvature, it makes the use of parallel transport to detect non-trivial symmetries on manifolds very difficult. For instance, parallel transport could not be used to generate consistent orientations for a feature repeated at regular intervals along a closed curve on a surface, since there is no guarantee that the orientation of the last feature of the curve will be aligned in any way with that of the first feature. Thus, the application of parallel transport to pattern synthesis and symmetry detection remains a topic for future research.

2.4 Local Isometries and Killing Vector Fields

While it may not be clear how parallel transport may be applicable to geometric pattern analysis, other notions from differential geometry stand out as potential alternative tools for understanding how surface features relate to one another. One such notion is that of a Killing field, named after German mathematician Wilhelm Karl Joseph Killing (1847-1923).

The easiest way to understand Killing fields is in terms of flows. Just as vector fields on $\mathbb{R}^2$ lead to solution curves that exist at least for short time by the Fundamental Existence and Uniqueness Theorem for ODEs, we can make a similar statement for vector fields on surfaces. For instance, the following theorem is stated in [24] (we refer the reader to the text for a proof):

Theorem 3 ([24], Theorems 5.5 and 5.6, page 149)**.** *Let X be a C^∞ vector field on M, and let $p \in M$. Then there is an open set V containing p and an $\varepsilon > 0$ such that there is a unique collection of diffeomorphisms $\phi_t : V \to \phi_t(V) \subset M$ for $|t| < \varepsilon$ with the following properties:*

1. $\phi : (-\varepsilon, \varepsilon) \times V \to M$, *defined by* $\phi(t, p) = \phi_t(p)$, *is* C^∞.

2. *If* $|s|, |t|, |s + t| < \varepsilon$ *and* $q, \phi_t(q) \in V$, *then*

$$\phi_{s+t}(q) = \phi_s \circ \phi_t(q).$$

3. *If* $q \in V$, *then* X_q *is the tangent vector at* $t = 0$ *of the curve* $t \mapsto \phi_t(q)$.

If X *has compact support (in particular, if* M *is compact), then there are diffeomorphisms* $\phi_t : M \to M$ *for all* $t \in \mathbb{R}$ *with properties 1, 2, and 3.*

Thus, vector fields on surfaces give rise to flows that move points along the surface, and in the case of compact surfaces the flows exist for all time.

Killing fields can be defined in terms of their flows. We adapt the following definition from [21]:

Definition 6 (Killing field)**.** *A Killing field* X *is a vector field whose flows* ϕ_t *defined in Theorem 3 are isometries. That is,* $d\phi_t$ *preserves inner products for all* $t \in (-\varepsilon, \varepsilon)$.

It immediately clear that surfaces with Killing fields are fairly rare, since surfaces admitting any sort of isometric deformation satisfy fairly strict criteria. We will return to this idea later when developing notions of "near-Killing fields."

The main definition of Killing fields is hard to work with from a discrete standpoint. After all, explicitly computing the flow of a vector field along a surface can be a much more involved process than proving its existence. Before we can formulate a more usable definition of Killing fields, however, we need to put into place more sophisticated mathematical machinery for dealing with derivatives of vector fields along flows.

We start by defining the *Lie derivative* of a vector field and an inner product:[5]

Definition 7 (Lie derivative of a vector field)**.** *Take* ϕ_t *to be the flow associated with vector field* X. *Then, the Lie derivative of field* Y *in the* X *direction at* $p \in M$ *is given by*

$$L_X Y = \lim_{t \to 0} \frac{Y_{\phi_t(p)} - d\phi_t(Y_p)}{t} \tag{20}$$

where Y_a *is the value of* Y *at* $a \in M$.

Definition 8 (Lie derivative of an inner product)**.** *The Lie derivative of* $\langle Y, Z \rangle$ *in direction* X *is given by*

$$L_X \langle Y, Z \rangle = \frac{d}{dt} \langle d\phi_t(Y), d\phi_t(Z) \rangle|_{t=0} \tag{21}$$

We proceed with two key properties of Lie derivatives:

[5]The Lie derivative of an inner product is a specific instance of the more general Lie derivative of a two-tensor. The latter requires more complicated development than what we have developed in this paper and thus is omitted since only the Lie derivative of an inner product is needed.

Lemma 3. *The Lie derivative in direction X of a vector field Y satisfies the identity*

$$L_X Y = [X, Y]. \tag{22}$$

Proof. Our proof follows that of [24] using more elementary notation. We verify that the derivatives of a function $f : M \to \mathbb{R}$ in the direction $L_X Y$ are identical to those from $[X, Y]$ for all f; since the action of the two vector fields on all scalar functions is identical, they must be the same field.

Take any such f, and take ϕ_t to be the flow associated with X. Define $g(t, p) = f \circ \phi_t - f$, and take $h = \frac{\partial g}{\partial t}$. We define $\psi : (-\varepsilon, \varepsilon) \times M \to \mathbb{R}$ as

$$\psi(t, p) = \int_0^1 h(st, p) ds \tag{23}$$

It is easy to verify that $g(0, p) = 0$, $t\psi = g$, and $\psi(0, p) = h(0, p) = Xf$. Then,

$$\begin{aligned}
[d\phi_t(Y)]_p(f) &= d\phi_t(Y_{\phi_{-t}(p)})(f) \text{ by definition} \\
&= Y_{\phi_{-t}(p)}(f \circ \phi_t) \text{ by the chain rule} \\
&= Y_{\phi_{-t}(p)}(f + g) \text{ by definition of } g \\
&= Y_{\phi_{-t}(p)}(f + t\psi)
\end{aligned} \tag{24}$$

With this identity, we can complete the proof:

$$\begin{aligned}
L_X Y(f) &= \lim_{t \to 0} \frac{1}{t} [Y_p - (d\phi_t(Y))_p](f) \\
&= \lim_{t \to 0} \frac{1}{t} [(Yf)(p) - (Yf)(\phi_{-t}(p))] - \lim_{t \to 0} (Y\psi)(\phi_{-t}(p)) \text{ by Equation 24} \\
&= XY(f) - YX(f) \\
&= [X, Y](f) \text{ as needed.}
\end{aligned}$$

□

Lemma 4. *The directional derivative of an inner product can be written*

$$X(\langle Y, Z \rangle) = \langle L_X Y, Z \rangle + \langle Y, L_X Z \rangle + L_X \langle Y, Z \rangle \tag{25}$$

Proof. We work with limit forms of directional and Lie derivatives that are easily derived from their definitions:

$$X(\langle Y, Z \rangle) = \lim_{t \to 0} \frac{1}{t} \left[\langle Y_{\phi_t(p)}, Z_{\phi_t(p)} \rangle - \langle Y_p, Z_p \rangle \right] \tag{26}$$

$$\langle L_X Y, Z \rangle = \lim_{t \to 0} \frac{1}{t} \left[\langle Y_{\phi_t(p)}, Z_{\phi_t(p)} \rangle - \langle d\phi_t(Y_p), Z_{\phi_t(p)} \rangle \right] \tag{27}$$

$$L_X \langle Y, Z \rangle = \lim_{t \to 0} \frac{1}{t} \left[\langle d\phi_t(Y_p), d\phi_t(Z_p) \rangle - \langle Y_p, Z_p \rangle \right] \tag{28}$$

Starting with Equation 26, we simplify as follows:

$$\begin{aligned}
X(\langle Y, Z\rangle) =& \langle L_X Y, Z\rangle + \lim_{t\to 0} \frac{1}{t}\left[\langle d\phi_t(Y_p), Z_{\phi_t(p)}\rangle - \langle Y_p, Z_p\rangle\right] \text{ by Equation 27} \\
=& \langle L_X Y, Z\rangle + L_X\langle Y, Z\rangle + \lim_{t\to 0} \frac{1}{t}\left[\langle d\phi_t(Y_p), Z_{\phi_t(p)} - d\phi_t(Z_p)\rangle\right] \\
& \text{by Equation 28} \\
=& \langle L_X Y, Z\rangle + L_X\langle Y, Z\rangle + \left\langle Y, \lim_{t\to 0} \frac{1}{t}\left[Z_{\phi_t(p)} - d\phi_t(Z_p)\right]\right\rangle \\
& \text{by simple properties of limits} \\
=& \langle L_X Y, Z\rangle + L_X\langle Y, Z\rangle + \langle Y, L_X Z\rangle \text{ by definition, as desired.}
\end{aligned}$$

□

With these two lemmas, we can prove a symmetry identity that characterizes Killing fields:

Lemma 5. *A vector field X is a Killing field if and only if for all other fields Y and Z it satisfies the identity*

$$0 = \langle Z, \nabla_Y X\rangle + \langle Y, \nabla_Z X\rangle \tag{29}$$

Proof. The proof below shows that if X is a Killing field it must satisfy Equation 29; all the steps trivially are reversible, however, so the explicit proof of equivalence is omitted.

Consider two vectors Y and Z at a point $p \in M$, where M admits a Killing vector field X. By Definition 6, the flow ϕ_t of X must preserve inner products and thus there exists a constant c such that:

$$c \equiv \langle d\phi_t(Y), d\phi_t(Z)\rangle \tag{30}$$

Differentiating both sides with respect to t and substituting $t = 0$ yields

$$\begin{aligned}
0 =& \frac{d}{dt}\langle d\phi_t(Y), d\phi_t(Z)\rangle|_{t=0} \\
=& L_X\langle Y, Z\rangle \text{ by definition} \\
=& X(\langle Y, Z\rangle) - \langle L_X Y, Z\rangle - \langle Y, L_X Z\rangle \text{ by Lemma 4} \\
=& X(\langle Y, Z\rangle) - \langle [X, Y], Z\rangle - \langle Y, [X, Z]\rangle \text{ by Lemma 3} \\
=& (\langle \nabla_X Y, Z\rangle + \langle Y, \nabla_X Z\rangle) - (\langle \nabla_X Y, Z\rangle - \langle \nabla_Y X, Z\rangle) - (\langle Y, \nabla_X Z\rangle - \langle Y, \nabla_Z X\rangle) \\
& \text{by metric preservation and symmetry} \\
=& \langle \nabla_Y X, Z\rangle + \langle Y, \nabla_Z X\rangle \text{ as desired.}
\end{aligned}$$

□

Lemma 5 suggests an alternative way to express the Killing field condition. Given Killing field X, we could define a tensor T taking input fields Y and Z given by:

$$T(Y, Z) = \langle Z, \nabla_Y X\rangle \tag{31}$$

Then, Equation 29 shows $T(Y, Z) = -T(Z, Y)$. All of our proof steps above again are reversible, and thus we have proved the following theorem:

Theorem 4. *X is a Killing field if and only if T is an antisymmetric tensor.*

Equivalently, we can state the following corollary by simple algebraic manipulation:

Corollary 2. *X is a Killing field if and only if for any vector field Y we have $\langle \nabla_Y X, Y \rangle \equiv 0$.*

Using the notation of the theorem, if we take E_1 and E_2 to be a local coordinate basis, then for $i, j \in \{1, 2\}$ we must have $\nabla_{E_i} X^j + \nabla_{E_j} X^i = 0$ where $X = X^1 E_1 + X^2 E_2$. This formulation provides three distinct equations defining Killing fields, corresponding to $(i, j) \in \{(1, 1), (1, 2), (2, 2)\}$; note $(i, j) = (2, 1)$ yields the same equation as $(i, j) = (1, 2)$. Thus, it also is unsurprising from an analytical standpoint that Killing fields would be relatively uncommon, as they have two degrees of freedom (corresponding to their restriction to the tangent plane) and three independent defining equations.

In the rest of this section, we develop two characterizations of Killing fields that will be useful in the development of discrete algorithms for their computation on triangle meshes.

2.4.1 Variational Approach

We define the symmetric part of a two-tensor S as follows:

$$\mathrm{Sym}(S)(Y, Z) = \frac{1}{2}(S(Y, Z) + S(Z, Y)) \tag{32}$$

Also, we define the inner product of two-tensors S and T. Take a local coordinate basis $\{E_1, E_2\}$ and define $S_{ij} = S(E_i, E_j)$ and $T_{ij} = T(E_i, E_j)$. Then, we write

$$\langle S, T \rangle = \sum_{i,j,k,l} \langle E_i, E_k \rangle \langle E_j, E_l \rangle S_{ij} T_{kl} \tag{33}$$

Then, the norm of such a tensor can be written with the usual inner product identity $||T||^2 = \langle T, T \rangle$. A simple analytical argument shows that this definition of norms and inner products is independent of our choice of $\{E_1, E_2\}$.

Obviously, by direct application of Theorem 4 we have $\mathrm{Sym}(T) \equiv 0$. Thus, a Killing vector field X on an open set $\Omega \subseteq M$ is a (non-unique) minimizer of the integral $\int_\Omega ||\mathrm{Sym}(T)||^2 d\mathrm{Vol}_M$. In fact, Killing vector fields (scaled by some constant) are solutions to the following more constrained optimization problem, which ensures that X is non-trivial:

Minimize $\mathcal{F}(X) \equiv \int_\Omega ||\mathrm{Sym}(T)||^2 d\mathrm{Vol}_M$

subject to $\int_\Omega ||X||^2 d\mathrm{Vol}_M = 1$.

Note that regions Ω not admitting Killing vector fields will have non-zero minimizing values. When nonzero Killing fields exist, however, by our above derivation they must be the only minimizers of this more constrained system by Theorem 4. Regardless, minimizers of the functional $\mathcal{F}$ with the given constraint can be deemed *approximate Killing fields*.[6]

[6]This notion and the subsequent development were proposed by Adrian Butscher in 2009.

As discussed in [26], we can apply an analogous approach to that of "Lagrange multipliers" from multivariable calculus to search for a minimum of the functional $\mathcal{F}$. We define a modified functional $\tilde{\mathcal{F}}$ taking the integral constraint into account:

$$\tilde{\mathcal{F}}(X) \equiv \int_\Omega \left(||\mathrm{Sym}(T)||^2 - \lambda ||X||^2 \right) d\mathrm{Vol}_M \tag{34}$$

Then, approximate Killing fields occur as critical points of the "Gâteaux derivative" of $\tilde{\mathcal{F}}$ given by $\delta\tilde{\mathcal{F}}(X;U) = \frac{d}{dt}\tilde{\mathcal{F}}(X+tU)|_{t=0}$. Differentiating under the integral, it is easy to verify in this case that

$$\delta\tilde{\mathcal{F}}(X;U) = \int_\Omega \left(\langle P(X), P(U) \rangle - \lambda \langle X, U \rangle \right) d\mathrm{Vol}_M \tag{35}$$

where $P(V)(Y,Z) = \frac{1}{2}\left(\langle Z, \nabla_Y V \rangle + \langle Y, \nabla_Z V \rangle \right)$ is the symmetric part of the tensor arising from vector field V given by Equation 31 for field X. If we take P^* to be the formal adjoint of P, then applying integration by parts to the first term yields the following expression:

$$\delta\tilde{\mathcal{F}}(X;U) = \int_\Omega \left(\langle P^*P(X) - \lambda X, U \rangle \right) d\mathrm{Vol}_M + \int_{\partial\Omega} P(X)(\nu, U) d\nu \tag{36}$$

For our choice of X to be a critical point of $\tilde{\mathcal{F}}$, we must have $\delta\tilde{\mathcal{F}}(X;U) = 0$ for *all* vector fields U. Thus, Killing vector fields are characterized by the partial differential equation

$$P^*P(X) - \lambda X = 0 \text{ in } \Omega \tag{37}$$

with boundary conditions

$$P(X)(\nu, U) = 0 \text{ on } \partial\Omega, \tag{38}$$

where ν is any vector normal to $\partial\Omega$. This is the Euler-Lagrange equation corresponding to the variational problem of minimizing $\mathcal{F}$ subject to having non-zero values of X [8]. Since Equation 37 is equivalent to solving $P^*P(X) = \lambda X$, we thus can regard finding Killing fields as an eigenvalue problem.

Since the steps above are reversible, applying basic theorems for the Calculus of Variations regarding the minimization of functionals such as $\mathcal{F}$ yields the following theorem:

Theorem 5. *X is a Killing field on Ω if and only if X solves Equation 37 with $\lambda = 0$ subject to the boundary conditions in Equation 38.*

Additionally, we note that small values of λ indicate that a vector field is in some sense close to being an infinitesimal isometry. Thus, we make the following definition:

Definition 9 (λ-approximate Killing vector field)**.** *A vector field X is a λ-approximate Killing vector field if it satisfies $P^*P(X) = \lambda X$ for P defined above.*

This definition will become particularly useful in our consideration of noisy or slightly deformed surfaces which still have relatively clear (near-)symmetries.

2.4.2 Differential Forms

One of the basic tools of Riemannian geometry is the *differential form*, which provides a means by which integrands can be expressed without local coordinates and give methods for understanding functions and their derivatives without resorting to coordinate-wise expressions. The basic definition of a differential form is as follows:

Definition 10 (Differential form)**.** *A differential form of degree k smoothly defines an alternating multilinear map $\alpha_p : T_pM \times \cdots \times T_pM \to \mathbb{R}$ taking in k vectors at each $p \in M$.*

Of course, this definition is far from motivated or formal, leaving questions regarding the definition of smoothness and the reasons for making use of such an object. A full review of operations and definitions involving differential forms is outside the scope of this paper; the reader can refer to [27] or any other similar differential geometry or manifold theory text for a discussion of these topics. In particular, we assume that the reader is familiar with the definition of a k-form and the basics of exterior calculus including the wedge operator $\wedge$ and differential operator d.

An equivalent development of the theory of vector fields on surfaces can be derived through the use of differential forms and exterior calculus. Rather than manipulating vector fields directly, this approach makes use of their "dual" one-forms, defined as follows:

Definition 11 (Dual of a vector field)**.** *The dual of a vector field X on M is the one-form ω satisfying $\omega(Y) = \langle X, Y \rangle$ for all vector fields Y.*

It is easy to check and intuitively reasonable that there exists exactly one dual form for each vector field than that all one forms are duals of vector fields. In this way, all proofs involving properties of one-forms can be stated as vector field and tensor proofs and vice versa; the succinct language and formal development of forms, however, make them useful tools for a number of geometric problems.

Given our previous treatment of derivatives of vector fields, it is reasonable to attempt to differentiate forms as well. In particular, since one-forms are the duals of vector fields, we should expect there to be an analog to covariant differentiation that processes such objects. Fortunately, rather than developing such a theory from scratch, we can define the covariant derivative of a one-form ω with dual field X in direction Y as the dual of $\nabla_Y X$.

Just as vector fields are dual to one-forms, we locally can represent two-tensors and two-forms as 2×2 matrices. In particular, a two-form T can be written locally using the matrix with entries $T(E_i, E_j)$, which must be sufficient to represent T by linearity; note that for simplicity, here and for the rest of the section we make use of *geodesic normal coordinates* to generate our basis E_1, E_2, allowing our metric $\langle \cdot, \cdot \rangle$ to be Euclidean to first order near p [21]. Using the notation of Section 2.4.1, we thus represent two-tensor $P(\omega)$, where ω is dual to X, as the matrix with entries $\frac{1}{2}(\nabla_{E_j}\omega_i + \nabla_{E_i}\omega_j)$.

Using coordinate-wise expressions for the differential of a one-form $d\omega$ and substituting the definition of a directional derivative, we easily can derive the following identity:

$$d\omega(X, Y) = X(\omega(Y)) - Y(\omega(X)) - \omega([X, Y]) \tag{39}$$

Applying the symmetry of ∇ shows that $d\omega$ can be represented using the matrix $\nabla_{E_j}\omega_i - \nabla_{E_i}\omega_j$. To complete our summary of useful operators for defining Killing vector fields in terms of forms, we define the divergence or co-differential operator applied to a one-form as the zero-form $\delta\omega = -\sum_i \nabla_{E_i}\omega_i$.

Using the inner product norm, we can find an expression for $||P(X)||^2$ as follows:

$$\begin{aligned}
||P(X)||^2 &= \sum_{i,j}\left(\frac{1}{2}(\nabla_{E_j}\omega_i + \nabla_{E_i}\omega_j)\right)^2 \\
&= (\nabla_{E_1}\omega_1)^2 + (\nabla_{E_2}\omega_2)^2 + \frac{1}{2}(\nabla_{E_1}\omega_2)^2 + \nabla_{E_1}\omega_2\nabla_{E_2}\omega_1 + \frac{1}{2}(\nabla_{E_2}\omega_1)^2 \\
&= \frac{1}{2}(\nabla_{E_2}\omega_1 - \nabla_{E_1}\omega_2)^2 + (\nabla_{E_1}\omega_1 + \nabla_{E_2}\omega_2)^2 - 2Q \\
&= \frac{1}{2}||d\omega||^2 + (\delta\omega)^2 - 2Q
\end{aligned} \tag{40}$$

where $Q \equiv \nabla_{E_1}\omega_1\nabla_{E_2}\omega_2 - \nabla_{E_1}\omega_2\nabla_{E_2}\omega_1$.

The only term in our expression for $||P(X)||^2$ that is not in terms of well-understood differential operators is that involving Q. Thus, we proceed by continuing to refactor Q. To do so, take $\perp$ to be the 90° rotation operator, such that in our local coordinates $E_1^\perp = -E_2$ and $E_2^\perp = E_1$. We define a one-form F as follows:

$$F(Y) = -\langle \nabla_{Y^\perp} X, X^\perp \rangle \tag{41}$$

In particular, we can define function F_1 as

$$\begin{aligned}
F_1 \equiv F(E_1) &= -\langle \nabla_{E_1^\perp} X, X^\perp \rangle \\
&= -\langle \nabla_{-E_2} X, X^\perp \rangle \\
&= \langle (\nabla_{E_2}\omega_1, \nabla_{E_2}\omega_2), (-\omega_2, \omega_1) \rangle \text{ by definition of } X \text{ in terms of } \omega \\
&= \omega_1\nabla_{E_2}\omega_2 - \omega_2\nabla_{E_2}\omega_1.
\end{aligned} \tag{42}$$

Similarly, we can take $F_2 \equiv F(E_2) = \omega_2\nabla_{E_1}\omega_1 - \omega_1\nabla_{E_1}\omega_2$. So, since our coordinate frame is orthonormal, the components of F are exactly F_1 and F_2.

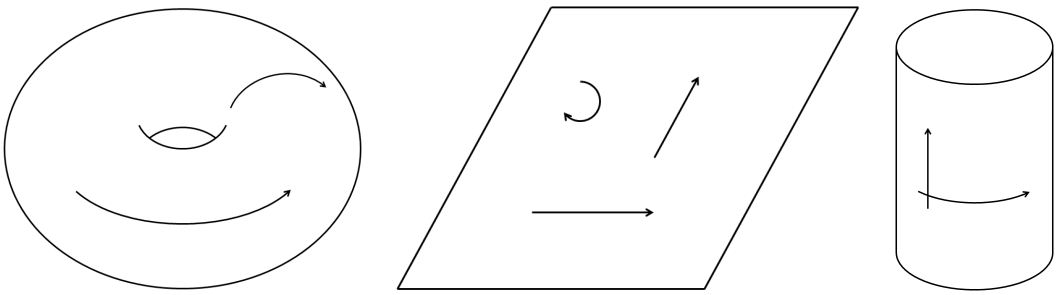
By definition, since geodesic normal coordinates are locally Euclidean, we can write the Gauss curvature K of M using the following "implicit" definition:

$$\nabla_{E_2}\nabla_{E_1}\omega - \nabla_{E_1}\nabla_{E_2}\omega = -K\omega^\perp \tag{43}$$

This implicit definition follows from properties of the first and second fundamental forms from classical differential geometry. Using this identity as well as our expressions for the components of F, we can simplify Equation 40 to obtain:

$$||P(X)||^2 = ||d\omega||^2 + 2(\delta\omega)^2 - 2K||\omega||^2 - 2\delta F \tag{44}$$

We can integrate this expression to find an alternative expression for the "Killing energy" of X that we minimize in Section 2.4.1. Assuming M has no boundary and integrating Equation 44 by parts yields the following theorem:



(a) (b) (c)

Figure 7: Connected symmetries of the (a) torus, (b) plane, and (c) cylinder.

Theorem 6. *The Killing energy of a vector field X is given by:*

$$\begin{aligned}\mathcal{F}(X) &= \int_M ||P(X)||^2 dVol_M \\ &= \int_M \langle \omega, \Delta\omega + d\delta\omega - 2K\omega \rangle \end{aligned} \tag{45}$$

This formula is the main expression used in our formulation of discrete Killing vector fields and approximate Killing vector fields. Its derivation is an example of the well-known "Bochner technique" for simplifying or re-expressing differential properties of forms [21].

2.5 Connected Symmetry Groups

Since the composition of two isometries of a surface M clearly is another isometry, it is clear that the set of isometries forms a group acting on M. Often times, this group can have a "discrete" structure. For instance, there obviously is no smooth flow of the points on a cube (or even a cube with smoothed corners) that preserves angles, but rotating the cube integer multiples of 90° about any of the cube's axes produces an isometry. On the other hand, other subsets of isometries can form "connected" group structures, such as the rotations of a cylinder.

Since Killing fields represent flows on surfaces, we concern ourselves with the "connected isometry group" of a surface, defined as the largest connected subgroup of the isometry group containing the identity map. This group initially may appear somewhat complicated; after all, the group structure $S^1 \times S^1$ of a torus is very different from the Euclidean group $E(2)$ structure of the plane, as illustrated in Figure 7. Fortunately, the following theorem from [18] completely characterizes all connected isometry groups of surfaces as we have defined them:

Theorem 7 (Theorem 5 from [18])**.** *The group of isometries of a two-dimensional Riemannian manifold is discrete, except possibly in the case of surfaces homeomorphic to the sphere, projective plane, plane, cylinder, non-orientable cylinder (Möbius strip), torus, and Klein bottle.*

In other words, there are relatively few *exact* symmetries that exist in the set of surfaces, or two-dimensional Riemannian manifolds, and such symmetries can only be found on surfaces that have similar topologies to one of the seven surfaces listed in the theorem.

(a) (b)

Figure 8: Approximate or near-symmetries.

This theorem highlights several important observations that have appeared in the treatment of Riemannian surfaces from previous sections:

- Exact continuous symmetries are rare if not non-existent in nature or real-world models.
- Killing fields, which can be viewed as differential elements of connected isometry groups, are uncommon.

Thus, our treatment of approximate Killing fields, which leads through simple ODE integration to the notion of an "approximate symmetry," is not only an interesting extension of a classical principle but indeed a necessary addition for dealing with partially-symmetric surfaces. For instance, the surfaces in Figure 8 have clear instances of near or partial symmetry but do not admit Killing fields or global isometries. Seeking approximate symmetries where they exists greatly extends the applicability of symmetry detection techniques and allows some flexibility for imperfect input.

3 Previous Work

There is a vast array of literature that potentially is relevant to continuous symmetry analysis and pattern generation. Most prominently, the concept of symmetry unsurprisingly has inspired research by generations of mathematicians, leading to important developments in differential and algebraic geometry, algebra, and even seemingly less relevant fields like number theory and combinatorics. Other concepts of symmetry from Riemannian and semi-Riemannian geometry including many applications and identities involving Killing fields arose during the invention and development of general relativity, although the metrics used to understand manifolds in general relativity are very different those on surfaces in $\mathbb{R}^3$. For a review of the mathematics relevant to this paper starting from elementary principles, refer to Section 2.

Within computer science, symmetry can be used to assist data analysis and help users generate more aesthetic or understandable artwork and geometry. Almost any problem involving real-world

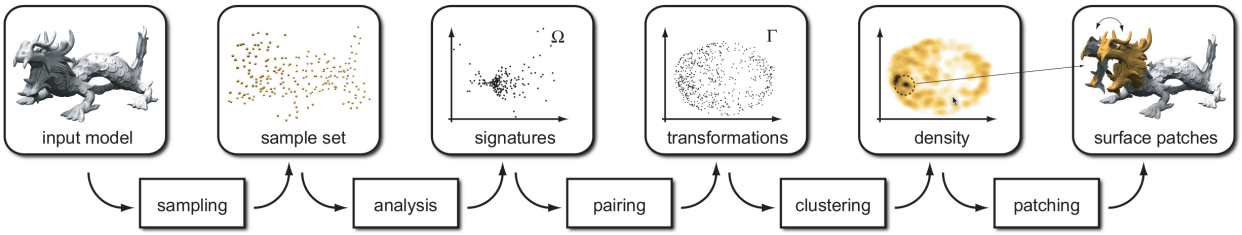


Figure 9: Figure from [15] illustrating the paper's approach to symmetry detection.

data can benefit from some understanding of symmetry and repetition; in some sense, even the vast field of Fourier analysis can be understood as a framework for recognizing certain types of symmetries on a Euclidean or manifold domain. In this section, we will focus on relevant papers in the field of geometry processing, since this is the main application of interest for the research at hand. Papers are divided depending on whether the search for intrinsic or extrinsic shape symmetries to highlight the challenges and research opportunities in both domains.

3.1 Extrinsic Symmetry Detection

The earliest work on symmetry detection for geometry processing involved the detection and manipulation of *extrinsic* symmetries. The existence of these types of symmetries depends on how a surface is embedded in $\mathbb{R}^3$ and thus can appear or disappear after applying even simple transformations to a shape. For instance, a building model may appear to have an axis-aligned grid of windows if the axes of $\mathbb{R}^3$ are aligned with those of the building, and otherwise no such translational symmetry exists. While such a formulation may appear rigid, many shapes including architectural figures and CAD models are likely to have mostly extrinsic symmetries that can be found more accurately and efficiently with dedicated methods.

Probably the best-known approach to extrinsic symmetry detection and extension for geometry processing was provided in a series of papers exploring the discovery and representation of Euclidean shape symmetries starting in 2006 [14, 15, 16]. These papers provide a good overview of previous work on symmetry detection, and the algorithms they discuss are effective examples of the strengths and weakness of an extrinsic approach.

The first of these papers to be released presented a statistic approach to the problem of recognizing surface patches that are similar to each other by some rigid (or inverting) symmetry transformation [15]. Each point p on a surface M is assigned a "signature" describing local curvature, orientation, and other basic geometric information. These signatures define a mapping of the surface or some sampled set of points from the surface to signature space. Then, the algorithm matches points with similar signatures. Each pair defines a unique symmetry; for instance, two points p and q define a reflective symmetry along the plane with normal $p - q$ through point $(p + q)/2$. These symmetries are clustered to find likely symmetries on the surface expressed as large sets of pairs of points that have the similar symmetry transformations. Finally, strong pairs in a likely symmetry cluster are used to grow regions on the surface that represent symmetries. This

Figure 10: Figure from [16] showing the results of symmetrization; the gray dragon is the original model, and the yellow dragon is the symmetrized version.

process is illustrated in Figure 9 (a figure from the original paper).

After laying the foundations of Euclidean symmetry detection, [16] provided a potential use for surface patches identified as similar to one another. In this paper, correspondences between patches were used to identify not only exact symmetries but also deviations from symmetry in a given model. This information is used to "symmetrize" a given model, as in Figure 10, producing minimal displacements to enforce stricter symmetry. This work is of interest not only in graphics but also in data analysis, where enhancing symmetries may be useful for emphasizing certain patterns or trends, and also for accomplishing certain mesh processing and "clean-up" tasks.

While the *detection* process presented in [15] might be sufficient for some applications, it provides little information about how sets of similar patches are structured. For instance, the windows on a simple building model likely have a grid-like structure isomorphic to a subset of $\mathbb{Z} \times \mathbb{Z}$, while compartments on a shell model may have a more complex repetition of rotation and translation. In general, we can use a group structure to express symmetries, in which the action of group elements shifts is a transformation mapping one feature to another. For the Euclidean case, most of the interesting simple orientation-preserving transformations, including rotations, translations, scaling, and combinations thereof, can be expressed as abelian groups of linear transformations with two generators.

The algorithm in [14] is designed to find such instances two-parameter groups within a given model. As in [15], a surface is mapped to "transformation space," in which transformations between similar points are plotted based on their particular parameters. Then, the algorithm searches for regular patterns or grids in transformation space; these grids correspond to the various two-parameter groups that the algorithm seeks.

The weaknesses of [14] highlight the need for intrinsic symmetry detection and processing. While the algorithm can identify symmetries on grid-like structures like buildings with windows, by definition it cannot identify symmetries on more complex models such as that shown in Figure 11, in which translational or other symmetries bend with the "base surface" of the model at hand.

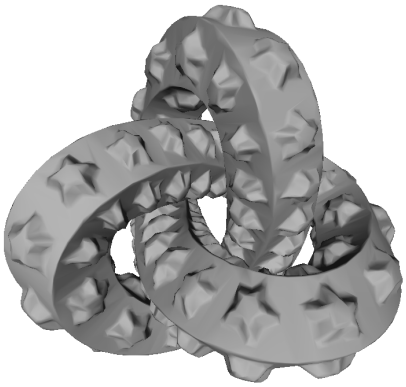


Figure 11: A complex instance of intrinsic translational symmetry; the stars clearly form a pattern isomorphic to $\mathbb{Z}/4 \times \mathbb{Z}/n$ where there are $4n$ stars in the knot loop (n on each of the four flat sides).

3.2 Intrinsic Symmetry Detection

Given the success of extrinsic symmetry detection as well as the failings of extrinsic algorithms to identify certain types of symmetries, more recent research has focused on the formulation of methods for finding intrinsic symmetries of shape. This preliminary research has revealed considerably more flexible approaches to understanding symmetry.

One particularly interesting approach uses intuition from heat diffusion to build up "signatures" for points on a surface [25]. In this line of research, points are identified with graphs of heat over time based on heat spreading from a single point. It is shown that this so-called Heat Kernel Signature (HKS) is preserved under isometry and that any mapping preserving HKS values is an isometry; in particular, it is an intrinsic property of shape. With this powerful descriptor in place, the paper describes how to identify repeated features or in some sense global symmetries by finding points that have similar signatures.

A related line of research is presented in [19], which computes symmetries expressed as isometries using the Laplace-Beltrami operator. The Laplace-Beltrami operator defines a derivative for functions on surface and is the analog of the Laplace operator for functions on a manifold domain; it arises when considering waves or heat diffusion along a surface. Just as the eigenfunctions of the Laplace operator give rise to Fourier analysis on Euclidean domains, the eigenfunctions of the Laplace-Beltrami operator form an L_2 basis for integrable functions on a given surface. The Laplace-Beltrami operator is intrinsic and uniquely determines the metric of a manifold, so it appears to be a reasonable tool for this sort of analysis. Indeed, for a given point $p \in M$, the paper defines its "restricted global point signature (GPS)" to be:

$$s(p) = \left(\frac{\phi_1(p)}{\sqrt{\lambda_1}}, \frac{\phi_2(p)}{\sqrt{\lambda_2}}, \cdots, \frac{\phi_d(p)}{\sqrt{\lambda_d}}, \right) \tag{46}$$

where $\phi_1, \phi_2, \ldots$ are the eigenfunctions of the Laplace-Beltrami operator and $\lambda_1, \lambda_2, \ldots$ are the corresponding eigenvalues. It can be shown under reasonable conditions that under the GPS map, isometries are expressed as Euclidean symmetries that can be detected using a specialized method

developed in the paper. This approach is an effective way to detect discrete symmetries, although it does not detect structure and may be difficult to use for detecting only partial symmetries.

A contrasting approach to intrinsic symmetry detection is presented in [1]. This paper matches features to find both isometric and non-isometric symmetries, the latter of which may only be approximately intrinsic. This task is accomplished by developing a graph of feature points on the surface annotated with intrinsic properties of shape; correspondences are computed by finding regions with similar graph structure. Then, similar regions are grown out from corresponding points to obtain dense mappings between surface patches, which are used to obtain sets of "symmetric" or repeated regions. Relaxing feature descriptors makes the approach non-intrinsic but more robust in some cases. It achieves promising results on a number of inputs but is limited by the use of several interdependent parameters and by the choice of feature descriptors.

While most approaches to intrinsic symmetry detection have focused on deterministic approaches to the problem, [12] uses a Markov random field model to assign probabilities to potential intrinsic maps from a shape to itself. The method can be used to find partial and approximate symmetries and has a fairly succinct statement, but it is inefficient both in terms of time and space and will need a considerable reformulation to become practical [1].

Obviously, no single method seems to be the most effective approach to symmetry detection in all cases. In particular, few if any papers have focused on connected rather than discrete symmetries, and many discrete methods present ad-hoc approaches with few theoretical guarantees on performance.

4 Connected Symmetries on Discrete Meshes

Our primary new contribution is the development of methods for computing connected symmetries on triangle meshes. Since these types of symmetries are expressed theoretically as flows on surfaces which are completely determined by their corresponding Killing vector fields, we concentrate in this section on the discretization of covariant differentiation and Killing vector field computation. All the methods here are derived from principles from Riemannian geometry described in Section 2. While some assumptions about the computational model are made in discretizing these concepts, no mathematical approximations or statistical assumptions are made in any of the algorithms presented here; thus, various properties and invariants from the continuous case are more likely to carry over to these discrete vector fields.

4.1 Discrete Covariant Differentiation

By Corollary 1, if we can extend a vector field X to a field X_{ext} on some neighborhood of a given surface M, then the projection of its directional derivative in direction Y onto the tangent space gives the covariant derivative $\nabla_Y X$. Indeed, since $\nabla_Y X$ can be stated independently of X_{ext} and since $\nabla_Y X$ depends on the values of X and Y only in an infinitesimally small neighborhood of the point at which it is being evaluated, it is sufficient to find *any* extension field X_{ext} in some open neighborhood of each point of evaluation independently.

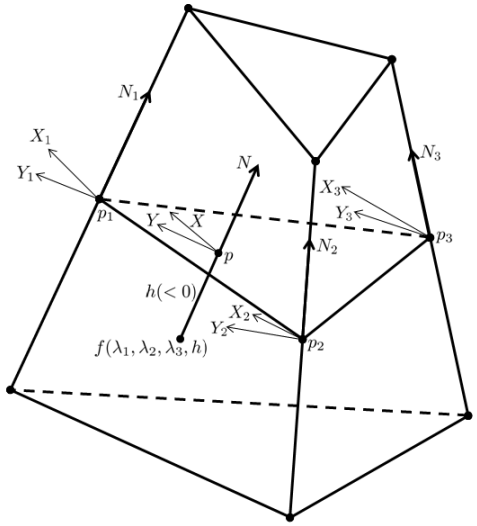


Figure 12: Setup for discrete covariant differentiation.

With these observations in place, one of the most obvious approaches to finding $\nabla_Y X$ on a mesh would be to treat each vertex or face independently and find the tangential component of $D_Y X_{ext}$ at each location for some localized extension field X_{ext}.[7] We use exactly this approach, generating a linear operator for a given field Y mapping X to $\nabla_Y X$.

Before proceeding, we state the basic input/output setup of the proposed algorithm for discrete covariant differentiation:

- Geometry is represented using a triangle mesh M.
- X and Y are given on the vertices of mesh M.
- A field of per-vertex normals N is given; note that simple estimates of per-vertex normals can be found simply by averaging adjacent face normals.
- The algorithm outputs covariant derivatives $\nabla_Y X$ on the faces of M, which can be interpolated at the vertices if desired. This construction mimics the setup in many numerical analysis algorithms involving divided differences in which the divided difference is treated as being evaluated at the midpoint of its two sampling vertices.

For ease of notation, for the rest of this section we will identify vector fields X and X_{ext} since we have $X_{ext}|_M = X$.

Take p_1, p_2, and p_3 to be the vertices of a given triangle face, and denote its per-vertex normals as N_1, N_2, and N_3 and its vector field samples as X_i and Y_i, $i \in \{1, 2, 3\}$; we assume $\langle X_i, N_i \rangle = \langle Y_i, N_i \rangle = 0$ for all i. The setup is illustrated in Figure 12. Suppose we want to evaluate $\nabla_Y X$ at a

[7] This general approach originally was suggested by Adrian Butscher.

point p with barycentric coordinates $(\lambda_1, \lambda_2, \lambda_3)$, so $p = \sum_i \lambda_i p_i$. Then, disregarding the need for unit-length normals we can define an interpolated normal N as follows:

$$N = \sum_{i=1}^{3} \lambda_i N_i \tag{47}$$

We can interpolate X_i in a similar way to write interpolated $\bar{X}$; to ensure that our normal and tangent vector fields are perpendicular, however, we project out the normal component of $\bar{X}$ to write the interpolated field vector X:

$$X = \hat{X} - \frac{\langle \bar{X}, N \rangle}{\langle N, N \rangle} N \tag{48}$$

With this model for interpolating X along a triangle face, we now extend X to a neighborhood of the surface. To do so, we define a function $f : F \times (-\varepsilon, \varepsilon) \to \mathbb{R}^3$ for triangle face F:

$$\begin{aligned} f(\lambda_1, \lambda_2, \lambda_3, h) &= \sum_{i=1}^{3} (\lambda_i p_i + h \lambda_i N_i) \\ &= p(\lambda_1, \lambda_2, \lambda_3) + h \cdot N(\lambda_1, \lambda_2, \lambda_3) \end{aligned} \tag{49}$$

As shown in Figure 12, by varying h, f defines a "fattening" of the triangle in such a way that fattenings of adjacent triangles share polygonal faces and do not overlap; this property ensures that the construction below does not introduce discontinuities in the differentiated functions. The following lemma justifies our use of f:

Lemma 6. *Substituting* $\lambda_3 = 1 - \lambda_1 - \lambda_2$, $f : (0,1)^2 \times (-\varepsilon, \varepsilon) \to \mathbb{R}^3$ *is a local diffeomorphism when* $h = 0$

Proof. By the Inverse Function Theorem, it is sufficient to check that Df is not degenerate. We have (for $i \in \{1, 2\}$):

$$\frac{\partial f}{\partial h} = N \tag{50}$$

$$\frac{\partial f}{\partial \lambda_i} = p_i - p_3 \tag{51}$$

For non-degenerate T, we clearly have that $\{N, p_1 - p_3, p_2 - p_3\}$ is a basis for $\mathbb{R}^3$ since $p_i - p_3$ are non-parallel tangent vectors and N is normal to the triangle. Thus, Df is invertible, as needed. □

Based on the results of this lemma, we can set X at $f(\lambda_1, \lambda_2, \lambda_3, h)$ equal to X at $f(\lambda_1, \lambda_2, \lambda_3, 0)$ with no conflict for small h.

To find $\nabla_Y X$ using the "ambient field" approach, we effectively need the matrix DX with composed of derivatives of X with respect to Euclidean coordinates (a_1, a_2, a_3). We divide this

task into two steps using the chain rule for partial derivatives (again assume we substituted $\lambda_3 = 1 - \lambda_1 - \lambda_2$ before completing this computation):

$$\frac{\partial X}{\partial a_i} = \sum_{k=1}^{2} \frac{\partial X}{\partial \lambda_k} \frac{\partial \lambda_k}{\partial a_i} \tag{52}$$

While the task of finding an explicit expression for $\frac{\partial \lambda_k}{\partial a_i}$ may appear to be difficult given that the definition of f in Equation 49 is hard if not impossible to invert, Lemma 6 trivializes the computation. In particular, the partial derivatives in Equations 50 and 51 form the columns a of $(Df)^{-1}$ at p, which must be invertible by the lemma.

The partial derivative $\frac{\partial X}{\partial \lambda_k}$ can be found directly in a series of steps. First, we give a precise definition of $\hat{X}$ adapted from the interpolation scheme introduced in [6] (expression (4)):

$$\begin{aligned} 2|T|\hat{X} = (c_{31}\lambda_3 - c_{12}\lambda_2)(p_3 - p_2)^{\perp} + (c_{12}\lambda_1 - c_{23}\lambda_3)(p_1 - p_3)^{\perp} \\ + (c_{23}\lambda_2 - c_{31}\lambda_1)(p_2 - p_1)^{\perp} \end{aligned} \tag{53}$$

where $|T|$ is the area of triangle T, the $\perp$ operator rotates 90° in the tangent plane, and c_{ij} is the component of X parallel to edge ij, expressed as a signed scalar with the property $c_{ij} = -c_{ji}$. Substituting $p_{ij} = p_j - p_i$ and $\lambda_3 = 1 - \lambda_1 - \lambda_2$, we have

$$\begin{aligned} \hat{X} = \frac{1}{2|T|}(c_{31}(1 - \lambda_1 - \lambda_2) - c_{12}\lambda_2)p_{23}^{\perp} + (c_{12}\lambda_1 - c_{23}(1 - \lambda_1 - \lambda_2))p_{31}^{\perp} \\ + (c_{23}\lambda_2 - c_{31}\lambda_1)p_{12}^{\perp} \end{aligned} \tag{54}$$

Differentiating,

$$\frac{\partial \hat{X}}{\partial \lambda_1} = \frac{1}{2|T|}\left[-c_{31}p_{23}^{\perp} + (c_{12} + c_{23})p_{31}^{\perp} - c_{31}p_{12}^{\perp}\right] \tag{55}$$

$$\frac{\partial \hat{X}}{\partial \lambda_2} = \frac{1}{2|T|}\left[-(c_{31} + c_{12})p_{23}^{\perp} + c_{23}p_{31}^{\perp} + c_{23}p_{12}^{\perp}\right] \tag{56}$$

Note the asymmetry with respect to sign.

In actuality, we want to differentiate the component X of $\hat{X}$ perpendicular to N, as defined in Equation 48. We write:

$$N = \lambda_1 N_1 + \lambda_2 N_2 + (1 - \lambda_1 - \lambda_2)N_3 \tag{57}$$

Take $\lambda \in \{\lambda_1, \lambda_2\}$. Then, applying the product and quotient rules,

$$\begin{aligned} \frac{\partial X}{\partial \lambda} =& \frac{\partial \hat{X}}{\partial \lambda} - \frac{\langle \bar{X}, N \rangle}{\langle N, N \rangle}\frac{\partial N}{\partial \lambda} - \\ & \left[\langle N, N \rangle \left(\left\langle \frac{\partial \hat{X}}{\partial \lambda}, N \right\rangle + \left\langle \hat{X}, \frac{\partial N}{\partial \lambda} \right\rangle\right) - \langle \hat{X}, N \rangle \frac{\partial ||N||^2}{\partial \lambda}\right] \frac{N}{\langle N, N \rangle^2} \end{aligned} \tag{58}$$

Note that we already have shown how to compute $\frac{\partial \hat{X}}{\partial \lambda}$ and that $\frac{\partial N}{\partial \lambda}$ is easily derived from Equation 57. Thus, we only are missing the derivatives of $||N||^2 = \langle N, N \rangle$. Expanding,

$$\begin{aligned}||N||^2 =& \langle N, N \rangle \\ =& \lambda_1^2 \langle N_1, N_1 \rangle + \lambda_2^2 \langle N_2, N_2 \rangle + (1 - \lambda_1 - \lambda_2)^2 \langle N_3, N_3 \rangle + 2\lambda_1 \lambda_2 \langle N_1, N_2 \rangle + \\ & 2\lambda_1 (1 - \lambda_1 - \lambda_2) \langle N_1, N_3 \rangle + 2\lambda_2 (1 - \lambda_1 - \lambda_2) \langle N_2, N_3 \rangle \end{aligned} \tag{59}$$

For the sake of completeness, we provide the partial derivatives of $||N||^2$ with respect to λ_1 and λ_2, substituting back in λ_3 for simplicity:

$$\begin{aligned}\frac{\partial ||N||^2}{\partial \lambda_1} =& 2\lambda_1 \langle N_1, N_1 \rangle - \lambda_3 \langle N_3, N_3 \rangle + 2\lambda_2 \langle N_1, N_2 \rangle \\ & + 2(\lambda_3 - \lambda_1) \langle N_1, N_3 \rangle - 2\lambda_2 \langle N_2, N_3 \rangle \end{aligned} \tag{60}$$

$$\begin{aligned}\frac{\partial ||N||^2}{\partial \lambda_2} =& 2\lambda_2 \langle N_2, N_2 \rangle - 2\lambda_3 \langle N_3, N_3 \rangle + 2\lambda_1 \langle N_1, N_2 \rangle \\ & - 2\lambda_1 \langle N_1, N_3 \rangle + 2(\lambda_3 - \lambda_2) \langle N_2, N_3 \rangle \end{aligned} \tag{61}$$

These expressions complete the derivation of all terms necessary to find $\frac{\partial X}{\partial \lambda_i}$ and thus to find DX and $\nabla_Y X$. While they may not be particularly "clean," they are relatively easy to compute. In fact, many terms depend only on N and could be computed ahead of time for varying X and Y. It also is interesting to see that Y has little to no effect on the level of complication of the derivation above, since it does not appear until we are ready to compute $D_Y X$. One final important observation is that all terms involving X are linear, as we would hope since X is the field being differentiated.

4.2 Discrete Killing Fields from Differential Forms

With a discrete notion of covariant differentiation in place, it should be possible to compute Killing vector fields directly using the variational approach or by approximating one of the identities characterizing Killing field behavior. Attempts at this type of discretization quickly become complicated, however, since finding a suitable "smooth basis" for the set of vector fields on a given discrete surface is far from trivial.

Instead, we use a setup from discrete exterior calculus (DEC), introduced in [10], given in [6] to discretize the differential form definition of Killing fields in Equation 45.[8] In the DEC formulation, a discrete analog of exterior calculus is developed by associating k-forms with their integral along k-dimensional simplices of the mesh complex. For instance, 0-forms are functions on M and thus are expressed as values assigned to each vertex of a mesh; similarly, 1-forms are expressed as values stored on edges and 2-forms are expressed as values on faces. Applications of Stokes' Theorem and other basic identities allow operators such as the exterior derivative d, co-differential δ, Laplace operator Δ, and Hodge star $\star$. The course notes in [5] provide a straightforward explanation of how to define each operator in the discrete setting and which properties of these operators carries over directly or approximately on a mesh. We approximate Gauss curvature along an edge

[8] This approach was developed jointly with Mirela Ben-Chen and Adrian Butscher.

using a well-known formula involving the cotangents of interior angles of faces adjacent to a given vertex and averaging the vertex Gauss curvatures on either side of the edge with area weights [3].

Using Equation 45 as motivation, we define a matrix R acting on vectors of one-forms (a single value per edge) as follows:

$$R = \Delta + d\delta - 2BG \tag{62}$$

where G is a diagonal matrix with per-edge Gauss curvatures and B is the discrete diagonal Hodge operator. Then, following the variational approach of Section 2.4.1 we define a *discrete λ-approximate Killing vector field* as the field dual to any solution ω of the eigenvector problem:

$$R\omega = \lambda B\omega \tag{63}$$

Finally, [6] provides a method to compute per-face vector fields from discrete one-forms, allowing the Killing field to be expressed directly rather than using integrated quantities in ω.

Thus, while deriving the vector field identities needed to express Killing vector fields and λ-approximate Killing vector fields on discrete meshes took considerable mathematical sophistication, the final method for finding such a vector field is as simple as finding the eigenvectors of a matrix. This reduced problem is well-studied and can be solved efficiently and stably in almost all cases with a prepackaged implementation.

4.3 Vector Field Integration

Since our discretized Killing field representation is constant on triangle faces, it can be integrated *exactly* using line segments to find flows of individual points on the mesh through the field. Thus, the only errors introduced within the applications of Killing fields discussed in this paper occur from discretization rather than integration, assuming numerical errors are minimal. For the sake of completeness, we describe here one strategy for finding the flow of a per-face vector field from a single vertex that is fairly stable and easy to implement. It is fast enough to be repeated for every vertex on a mesh to obtain a complete flow over time. Difficulties arise only in the fact that integration in this setting must be capable of following both edges and faces with various starting points.

The most basic operation in the ODE solver is integration along a single face. Suppose a face F has vertices p_1, p_2, and p_3. Define edge vectors $\vec{a} = p_2 - p_1$ and $\vec{b} = p_3 - p_1$, and take $\vec{c} = \vec{a} \times \vec{b}$. Then, we can define the matrix M to map from the face to the triangle with vertices $(0, 0, 0)$, $(1, 0, 0)$, and $(0, 1, 0)$ implicitly using the relation:

$$M \begin{pmatrix} | & | & | \\ \vec{a} & \vec{b} & \vec{c} \\ | & | & | \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{64}$$

Note that since $\vec{c}$ is normal to the triangle, M maps points on the face subtracted from p_1 to the xy plane and gives the barycentric localized coordinates of the point in question. Of course, M^{-1} never needs to be computed explicitly but can be applied to vectors using Gaussian elimination or a more stable or efficient factorization technique.

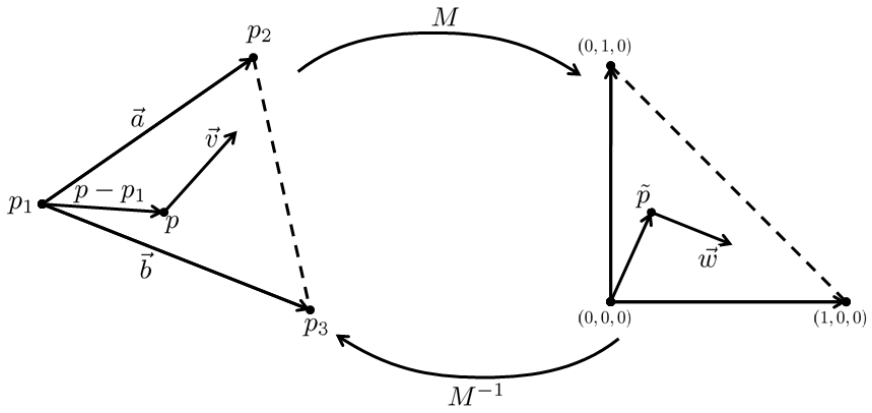


Figure 13: M maps from global to local coordinates.

M provides a mapping between vectors on F and vectors on the triangle $\tilde{T}$ with vertices $(0,0,0)$, $(1,0,0)$, and $(0,1,0)$ on the xy plane, as shown in Figure 13. By convexity of F, a ray starting at any point inside of F mapped to the xy plane can intersect at most a single side, excluding any side on which its starting point resides. Thus, our method for integrating along a vector $\vec{v}$ with starting point p on F is as follows:

1. Compute M or M^{-1} from $\vec{a}$, $\vec{b}$, and $\vec{c}$.
2. Map p and $\vec{v}$ to the xy plane as $\tilde{p} = M(p - p_1)$ and $\vec{w} = M\vec{v}$. To deal with numerical inaccuracies, assign $\vec{w}_z = \tilde{p}_z = 0$.
3. Compute t_1, t_2, and t_3 as the intersection parameters of ray $\tilde{p} + t\vec{w}$ with the three sides of the triangle with vertices $(0,0,0)$, $(1,0,0)$, and $(0,1,0)$.
4. Compute $t = \min\{t \in \{t_1, t_2, t_3\} : t > 0$ and $\tilde{p} + t\vec{w}$ is in $\tilde{T}\}$. If such a t exists, then the path of p is a straight line segment from p to $p + t\vec{v}$, taking time t. Otherwise, the ray points outside of the triangle.

The complete algorithm for computing flows of per-face vector fields is no more complicated. The vector field is followed from one edge to another of each face and chained together to create a series of line segments. Starting at a vertex or dealing with a flow going through a vertex also is straightforward; the flow simply continues whichever face has an outgoing vector from the vertex.

There is only one remaining case, illustrated in Figure 14. Occasionally adjacent faces with nearly parallel vectors may create a "false sink" in which the field points inward toward the edge on both faces. In this case, we simply follow the edge between the faces to the next vertex and restart.

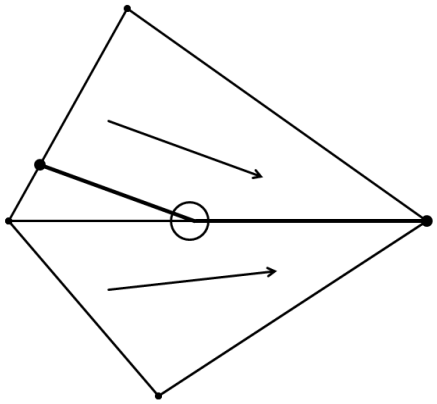


Figure 14: The vector field on the upper and lower triangles points in toward the centerline, so when simulation reaches the circled point it can follow neither adjacent face. Rather than stop, the simulation method follows the edge between the two faces.

5 Results

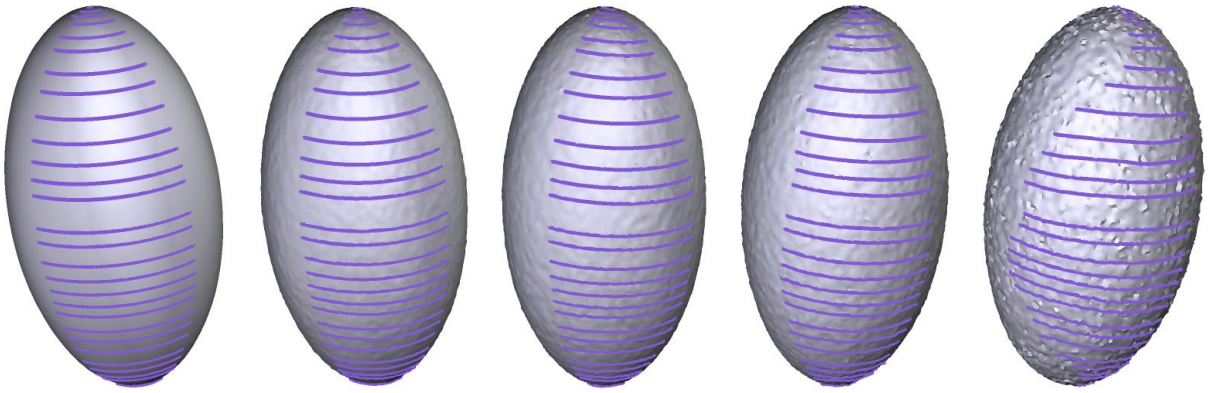
Discrete Killing fields are the main objects of interest for the target application of symmetry analysis. Thus, this section focuses on the effectiveness of the method presented in Section 4.2 for their computation. An implementation of the algorithm for finding discrete covariant derivatives is under development, although at this point its applications mainly are of only theoretical interest and possibly to verify Killing vector field identities to analyze the accuracy of the algorithm discussed below.

Figure 15 shows flows of discrete Killing fields and discrete approximate Killing fields generated using the algorithms presented in this paper.[9] As expected, the flows of the Killing field follow the circular cross-sections of the ellipsoid. Even with a considerable amount of noise, the flows of the approximate Killing vector field illustrate the global symmetry of the ellipsoid, providing an initial sign that the formulation of λ-approximate fields is a reasonable one. Figure 16 shows other examples of approximate Killing vector fields, which all illustrate reasonable symmetries on surfaces that do not admit exact isometries.

It may appear that our formulation of Killing vector fields can identify at most one class of symmetries on a surface. Examining eigenvectors corresponding to higher eigenvalues, however, reveals that they are useful for finding partial symmetries of a given surface. For instance, Figure 17 shows the λ-approximate Killing fields corresponding to the first four eigenvectors of R for a surface admitting no single continuous isometry class. Interestingly, each field describes a different localized continuous symmetry of the surface, indicating that a larger part of the spectrum of R may be interesting rather than its single smallest eigenvalue.

We can devise a number of tests indicating the effectiveness of our approach in approximating actual Killing fields. Perhaps the simplest test is to check that the discrete Killing field represents

[9]Figures illustrating the output of the algorithm were generated by Mirela Ben-Chen. The implementation of the algorithms for Killing field generation and integration was completed jointly by Mirela Ben-Chen and Justin Solomon.



$\sigma = 0$	$\sigma = 0.065$	$\sigma = 0.087$	$\sigma = 0.1145$	$\sigma = 0.2$
$E = 0.0006$	$E = 0.29$	$E = 0.55$	$E = 1.33$	$E = 6.7$

Figure 15: The ellipsoid model on the left with increasing amounts of noise with standard deviation σ times the average edge length in the normal direction; flows of the approximate Killing vector field for fixed time is shown. E is the "Killing energy" integrated over the entire surface.

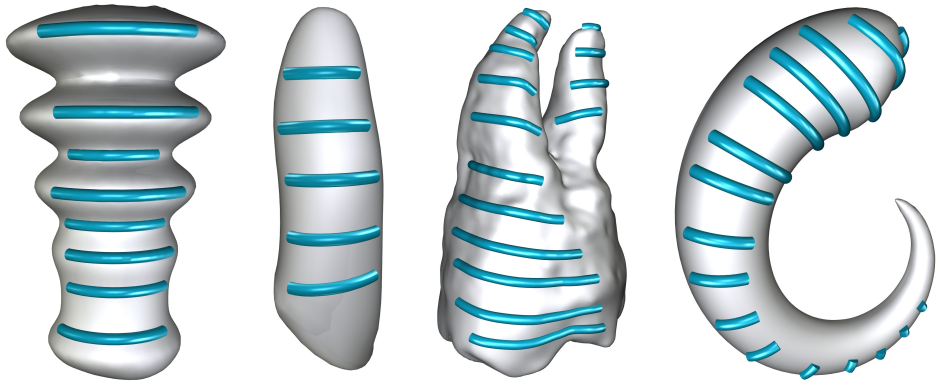


Figure 16: Flows of approximate Killing vector fields on some example surfaces.

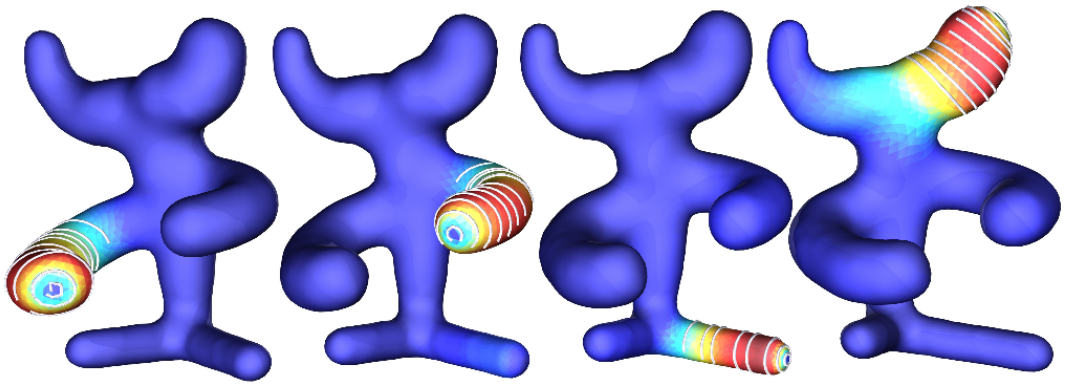


Figure 17: Vector fields corresponding to the four lowest eigenvalues of R; colors indicate the norm of the approximate Killing field displayed.

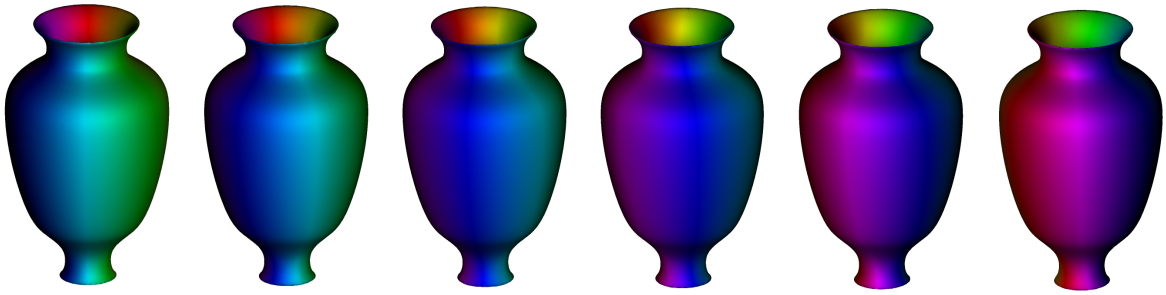


Figure 18: Flows of approximate Killing vector fields on some example surfaces.

a differential isometry. We compute the flows of all vertices on the mesh using the method in Section 4.3 and use these to generate a sequence of meshes representing vertex positions at various times. These meshes should be at most different by isometry from the original mesh. Figure 18 shows such a sequence for a model admitting a rotational symmetry, color coded from the original model. It is clear that the Killing field flows are indistinguishable from rotations, as would be expected. Also, these flows provide effective approximate isometries; edge lengths are preserved to several digits of precision. Only after thousands of time steps do problems become apparent in the rotating model, but this is an unreasonably large time step over which *any* discretization of an ODE is bound to fail.

An additional test to verify the effectiveness of the discrete Killing field formulation is to check the spectrum of the matrix R against analytically-computed spectra of Killing operators from simple surfaces. While we do not show the explicit computation here, we find that the error in computing the first 180 eigenvalues of R for a spherical mesh with 20,000 vertices is:

$$\sqrt{\frac{\sum_i \left(\lambda_i^{mesh} - \lambda_i\right)^2}{\sum_i \lambda_i^2}} \approx 0.0016 \tag{65}$$

Clearly, the formulation of discrete Killing fields not only generates reasonable flows but also is justified theoretically.

6 Applications

Using approximate Killing vector fields to guide repeating textures produces patterns that by construction are as symmetric as possible. This simple potential application of discrete approximate Killing fields reveals their usefulness in expressing surface symmetries accurately and simply in the connected case.

Of course, since textures take positive surface area, they must be spaced out rather than "dragged" along a surface as suggested by the use of vector field flows. If a surface M admits an exact Killing vector field with flow ϕ_t, $t \in \mathbb{R}$, then its connected symmetry group $\{\phi_t : t \in \mathbb{R}\}$ contains discrete subgroups $\{\phi_{k\sigma} : k \in \mathbb{Z}\}$ for a given $\sigma \in \mathbb{R}$. More generally, for any vector field with flow for even partial time $t \in (-\varepsilon, \varepsilon)$ we can define a similar discrete set of isometries. This set can be used to generate repeated textures with even spacing along a surface, since Killing fields admit flows that preserve geodesic distances.

Assume a discrete surface M is a associated with a function $(u(p), v(p))$ mapping each point $p \in M$ to a texture coordinate on some external 2D texture. We make no assumptions about the continuity of (u, v) and instead require that it be compactly supported on M in a small enough region that it could reasonably be repeated. Then, repeating a texture is straightforward; for some set $S \subset \mathbb{Z}$ and some $\sigma \in \mathbb{R}$ we simply assign each point p texture coordinates

$$\sum_{i \in S} (u(\phi_{-\sigma i}(p)), v(\phi_{-\sigma i}(p)). \tag{66}$$

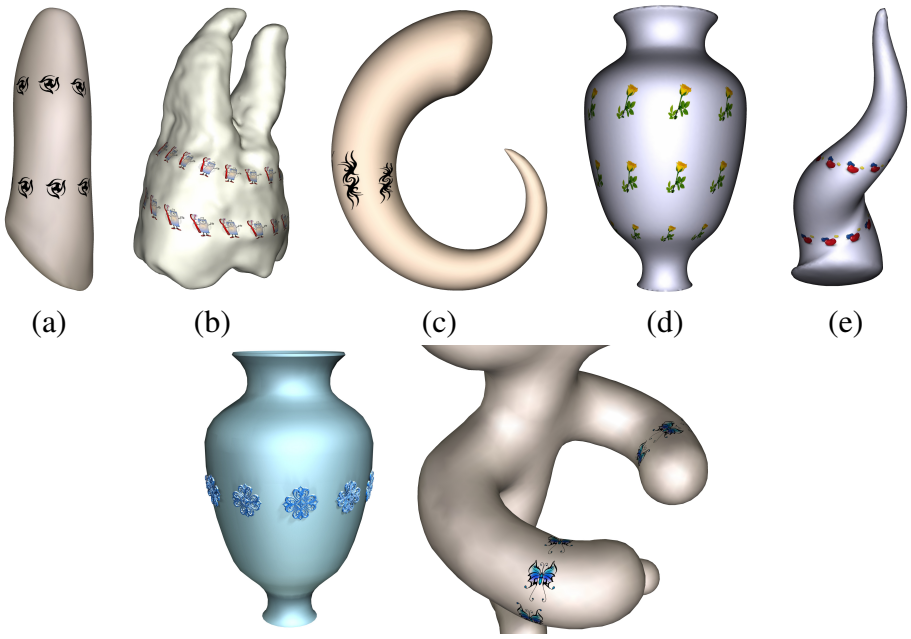


Figure 19: (a)-(e) Example textures repeated on a surface using approximate Killing fields; (f) a geometric pattern repeated using a discrete Killing field guiding [13]; (g) a texture repeated using the sum of two λ-approximate Killing fields, one on each arm of the model.

Assuming the texture is small enough, at most one term in the sum will be nonzero, representing the coordinates of the repeated texture. Note that $\phi_{-\sigma i}$ is likely to map to a point inside of a triangle face, in which case barycentric-weighted texture coordinates from the input surface are sufficient.

Of course, some practicalities should be addressed. After computing flows $\phi_{-\sigma i}$, it is likely that some triangles will have sampled texture coordinates that cross a discontinuity. If this is the case, all three vertices of the triangle should be assigned to a "null" texture rather than allowing for a nonsensical map; this fix is acceptable as long as the repeated texture has some empty space on its boundary. Additionally, this method could produce some texture distortion if the discretized Killing field is not accurate. Although in practice this has not been found to be a considerable problem, it may be possible to replace this "reverse texture lookup" scheme with a method that repeats a single point in a texture and independently places the texture at each point with some decision to be made regarding orientation; the orientation problem may be a hidden difficulty due to the non-commutativity of parallel transport.

Figure 19 shows some sample surfaces with repeated patterns placed using the method described above. Even on surfaces not admitting exact Killing fields, when λ is small enough that the integral curves of the field are approximately closed, the texture repetition scheme is effective. Of course, while this approach is limited to continuous symmetries, the approximation scheme makes it surprisingly applicable to the repetition of patterns on a variety of surfaces.

7 Conclusion

While the theoretical development of vector field theory on surfaces is involved and makes use of sophisticated mathematics, discrete algorithms for computing Killing fields and vector field flows are surprisingly straightforward. Mechanisms of differentiation and integration that require careful treatment and considerable proof become no more complicated than computing sums and divided differences and applying standard algorithms for matrix inversion or eigenvalue problems. This simplicity not only leads to straightforward and efficient algorithms but also begins to reveal the elegant structure of Riemannian geometry that can be lost behind its usual surprisingly analytical treatment. We illustrate one potential application of discrete Killing vector fields to texture generation here and expect there to be several others, as will be discussed in Section 8. Regardless, the discretization of Riemannian geometry in many ways represents a large step toward completing the link and equalizing the levels of sophistication of theoretical and algorithmic geometry and is likely to lead to important tools for modeling, simulation, graphics, and other fundamental computational tasks for artists and scientists.

8 Future Work

While this project is a promising first step toward continuous symmetry detection and the analysis of flows on surfaces, there are several valuable avenues for future research that could yield more practical or widely-applicable algorithms that use symmetry for geometry processing. Such work could make symmetry a more prominent tool integrated in software products used to generate and modify surface models for rendering, physics, and other applications.

Most directly, a better understanding of the relationship between the minimal λ parameter associated with an approximate Killing field and the actual presence of symmetry could yield a stronger classifier describing the likelihood of potential continuous symmetry in a given model. More generally, just as [19] uses the spectrum of the Laplace-Beltrami operator to help detect symmetries, the spectrum of the "Killing operator" R may encode useful information for shape classification or similar tasks.

Of particular interest is the classification of continuous isometries discussed in Section 2.5. Theorem 7 indicates that there are only seven surfaces up to isometry that admit connected symmetry groups. This number is small enough that it seems eminently possible to compute to which isometry class a given surface belongs if it admits a Killing field. Of course, even accounting for numerical error, this case is rare in both the continuous and discrete settings. It may be a considerable challenge theoretically to classify all the possible flows possible by integrating λ-approximate Killing fields, but any sort of classification theorem could narrow the possible classes of observable symmetries and may make it easier to generate algorithmic approaches to finding global semantic descriptions of continuous symmetries rather than localized vector fields.

With or without a better understanding of the theoretical structure of approximate Killing fields, there are a number of potential application areas that could benefit from a better understanding of continuous symmetries and isometry classes. For instance, isometric or near-isometric deformations of surfaces are of interest in designing techniques for surface deformation; isometry provides

some metric that can be used to judge the amount by which a surface is changed by a given deformation. Additionally, in the spirit of [16] it may be possible to use approximate symmetries as indicators that a model should be modified to enhance its symmetry. For instance, techniques could be employed to decrease the minimum eigenvalue λ of the matrix R, thus increasing the likelihood that a surface admits global or local differential isometry. The spectrum of R might also be useful in segmenting a given model. Outside the field of geometry processing, additional applications of Killing fields might be possible in the process of simulating physical interactions due to general relativity; while this paper discusses discrete Killing fields only on surfaces, the more general case on a simplicial complex should not be appreciably more difficult.

On a larger scale, we still are far from a unified algorithmic treatment of symmetry for geometry processing applications. To make symmetry detection, enhancement, and understanding a usable tool for the end user, the various methods involving continuous, discrete, intrinsic, and extrinsic symmetries should be connected in such a way that the client is not forced to do the nontrivial task of identifying which "class" of symmetries a surface exhibits to carry out a meaningful operation or analysis. Similarly, more intuitive controls than vector fields and isometry classes will be needed for interactive applications used by artists, modelers, or scientists who might not be familiar with the language or structure of Riemannian geometry. These concerns aside, the continuous approach presented here combined with previous or new discrete methods hold considerable potential to make geometry processing a more natural and global process which takes advantage of repeated structures to eliminate unnecessary, time- and space-consuming, and often tedious work in generating, texturing, and representing surfaces on the computer.

References

[1] Berner, Alexander et al. “Generalized Intrinsic Symmetry Detection.” *MPI Informatik Tech Report MPI-I-2009-4-005* (2009).

[2] Blake, William. *Songs of Innocence and of Experience*. New York: Chartwell Books, 2009 (orig. 1794).

[3] Bobenko, Alexander, Peter Schröder, and John Sullivan, ed. *Discrete Differential Geometry*. Birkhäuser Verlag: Berlin, 2008.

[4] Choi, Jae-Young and Young-Kyu Yang. “Vehicle Detection from Aerial Images Using Local Shape Information.” In *Advances in Image and Video Technology*. Berlin: Springer, 2009.

[5] Elcott, Sharif and Peter Schröder. “Build Your Own DEC at Home.” *SIGGRAPH 2005 Courses*.

[6] Fisher, Matthew et al. “Design of Tangent Vector Fields.” *International Conference on Computer Graphics and Interactive Techniques 2007* (San Diego): 56:1-56:9.

[7] Gallot, Sylvestre, Dominique Hulin, and Jacques Lafontaine. *Riemannian Geometry*. 3rd ed. New York: Springer, 2004.

[8] Gelfand, I.M. and S.V. Fomin. *Calculus of Variations*. Mineola: Dover, 2000.

[9] Guillemin, Victor and Alan Pollack. *Differential Topology*. Englewood Cliffs: Prentice-Hall, 1974.

[10] Hirani, Anil. “Discrete Exterior Calculus.” Caltech PhD Thesis (2003).

[11] Jost, Jürgen. *Riemannian Geometry and Geometric Analysis*. 5th ed. Berlin: Springer, 2008.

[12] Lasowski, Ruxandra et al. “A Probabilistic Framework for Partial Intrinsic Symmetries in Geometric Data.” *IEEE International Conference on Computer Vision 2009* (Kyoto): 963-970.

[13] Li, Y. et al. “Geometry Synthesis on Surfaces using Field-Guided Shape Grammars.” *IEEE Transactions on Visualization and Computer Graphics*, to appear 2010.

[14] Mitra, Niloy J. et al. “Discovering Structural Regularity in 3D Geometry.” *ACM Transactions on Graphics* 27.3: 43:1-43:11.

[15] Mitra, Niloy J. et al. “Partial and Approximate Symmetry Detection for 3D Geometry.” *ACM Transactions on Graphics* 25.3 (2006): 560-568.

[16] Mitra, Niloy J. et al. “Symmetrization.” *ACM Transactions on Graphics* 26.3: no. 63.

[17] Montiel, Sebastián and Antonio Ros. *Curves and Surfaces*. 2nd ed. Providence: American Mathematical Society, 2005.

[18] Myers, Sumner. "Isometries of 2-Dimensional Riemannian Manifolds into Themselves." *Proceedings of the National Academy of Sciences of the United States of America* 22.5: 297-300.

[19] Ovsjanikov, Maks et al. "Global Intrinsic Symmetries of Shapes." *Computer Graphics Forum* 27.5: 1341-1348.

[20] Pauly, Mark et al. "Discovering Structural Regularity in 3D Geometry." *ACM Transactions on Graphics* 27.3 (2008): 42:1-43:11.

[21] Petersen, Peter. *Riemannian Geometry*. 2nd ed. New York: Springer, 2010.

[22] Reisfeld, Daniel, Haim Wolfson, and Yehezkel Yeshurun. "Context-free Attentional Operators: The Generalized Symmetry Transform." *International Journal of Computer Vision* 14.2: 119-130 (1995).

[23] Saber, E. and A.M. Tekalp. "Face Detection and Facial Feature Extraction Using Color, Shape and Symmetry-Based Cost functions." *13th International Conference on Pattern Recognition* (Vienna): 654-659.

[24] Spivak, Michael. *A Comprehensive Introduction to Differential Geometry*. Houston: Publish or Perish, 1999.

[25] Sun, Jian, Maks Ovsjanikov, and Leonidas Guibas. "A Concise and Provably Informative Multi-Scale Signature Based on Heat Diffusion." *Eurographics Symposium on Geometry Processing* 28.5: 1383–1392.

[26] Troutman, John L. *Variational Calculus and Optimal Control: Optimization with Elementary Convexity*. 2nd ed. New York: Springer, 1995.

[27] Warner, Frank. *Foundations of Differentiable Manifolds and Lie Groups*. New York: Springer-Verlag, 1983.