HABILITATION A DIRIGER DES RECHERCHES

A Functional View of Geometry Processing

Operator-based Techniques for Shape Analysis

Author: Maks Ovsjanikov

Université Paris-Sud
Orsay, France

August 2016

Abstract

In this document, we present a set of tools and algorithms for analyzing and processing shapes and their relations. Our main observation is that a very productive way of looking at many operations in geometry processing, both in theory and in practice, is by representing them as linear operators acting on real-valued functions defined on the shapes. Although this point of view has been common in some areas of mathematics, such as dynamical systems, representation theory or parts of differential geometry, it has only recently been adopted in digital geometry processing, where it has led to novel insights and efficient algorithms for a wide variety of problems including shape matching, tangent vector field analysis and shape comparison to name a few. In this document, we will give an overview of these and related techniques and demonstrate, in particular, how the operator point of view can be helpful in a wide variety of practical settings, both by providing a common language in which many operations can be expressed and by enabling the use of classical linear-algebraic tools in novel, and sometimes unexpected scenarios.

Acknowledgments

This work would not have been possible without a very large number of people, starting from the many collaborators that I have had the privilege of working with during the past four years, including (in alphabetical order): Omri Azencot, Mirela Ben-Chen, Thomas Bonis, Adrian Butscher, Mathieu Carrière, Luca Castelli-Aleardi, Antonin Chambolle, Frederic Chazal, Étienne Corman, Leonidas Guibas, Qixing Huang, Moos Hueting, Chunyuan Li, Quentin Mérigot, Niloy Mitra, Alexandre Nolin, Steve Oudot, Viorica Pătrăucean, Raif Rustamov, Justin Solomon, Fan Wang, Max Wardetzky and Steffen Weißmann.

I am also extremely grateful for the support of my colleagues at the the LIX research laboratory of Ecole Polytechnique and at the Geometrica/Datashape teams of INRIA, including Catherine Bensoussan, Christine Biard, Evelyne Rayssac, Sylvie Jabinet, Olivier Bournez, Benjamin Werner, Gilles Schaeffer and Leo Liberti among many others.

Finally, my family has been tremendously supportive over the years, and I would like to thank them, and especially my wife Ayşegül, for their boundless patience, understanding and a constant source of inspiration.

Parts of this work have been supported by the CNRS chaire d'excellence, Marie Curie Career Integration Grant HRGP-334283, the FUI project "TANDEM 2", a grant from the French Direction Générale de l'Armement (DGA), the chaire Jean Marjoulet from Ecole Polytechnique, a Google Focused Research Award, a Qualcomm post-doctoral grant and a Google Faculty Award.

Contents

1 Introduction

The work presented in this dissertation falls within the general field of "geometry processing," which is a relatively young discipline, whose primary objective is to develop novel theoretical tools and practical algorithms for the analysis and processing of geometric data, whose scope and variability have been increasing during the past several decades[1]. The tools developed within this field deal with a wide variety of geometric data, but are traditionally dedicated to surface-based representations, where "shapes" are considered as discrete samplings of smooth surfaces, embedded in 3D. Such discrete representations most often come in two forms: either as point clouds (i.e., collections of points sampled from the surface of the shape), or triangle meshes, where in addition to the coordinates of the points one is also given some topological structure, which represents the connectivity across points. This latter mesh-based representation has its roots in Computer-Aided Design (CAD) and Computer Graphics (especially interactive and special effects) applications, in which one of the primary goals is to create photo-realistic images (renderings) of 3D objects, in addition to being able to manipulate these objects efficiently. In this context, the main focus of geometry processing has been to provide techniques that would help to analyze and process such geometric data, often with the goal of enabling higher order operations, such as shape comparison, deformation transfer or shape parameterization to name a few.

For most of the methods presented below, we will assume that the input data is given as a manifold triangle mesh, or as a collection of such meshes. This means, in particular, that we will assume that each shape is represented as a collection of faces (triangles), such that each edge belongs to at most two faces. We will often assume that the shapes do not contain any boundaries, although this assumption can often be replaced with the appropriate

[1] A great introduction to geometry processing and some of the problems it considers can be found in [BKP+10]. Also see [PLB12] for an overview of 3D imaging and acquisition.

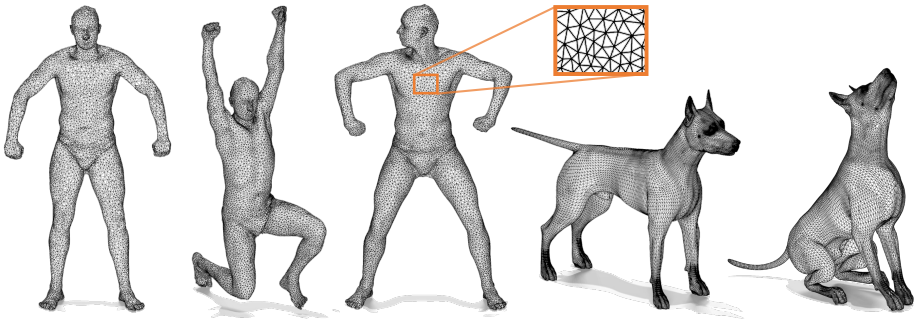


Figure 1.1: Examples of the shape structure and complexity that we consider in our work. The shapes (taken from SCAPE [ASK+05] and TOSCA [BBK08] datasets) are all represented as triangle meshes embedded in $\mathbb{R}^3$, containing several thousand (here, between 12 and 25 thousand) points.

handling of boundary conditions. See Figure 1.1 for some typical examples of such shapes.

One of the main challenges associated with developing efficient techniques in geometry processing is the sheer size of the input data, since even a shape with moderate complexity often contains tens or even hundreds of thousands of points or triangles. This implies that techniques that require complex data processing, and that even have quadratic complexity in the input size are, in most cases not feasible. At the same time, a lot of the data that we will consider, such as for example, the adjacency matrix of the graph associated with the input triangle mesh, is *sparse*, meaning that even though the size of such matrix is quadratic in the number of input points, the number of non-zero entries is linear, and it is important to develop techniques that exploit such structure.

Another important consideration, which is often present in geometry processing applications is robustness with respect to changes in the input. In the types of problems that we will consider below, this means two separate phenomena: one is the resilience with respect to the changes in the coordinates of the points on the shape, and with respect to the changes in triangulation, and the other is with respect to the change in the orientation or "pose" of the object. This latter consideration is especially important in the problem of finding correspondences or matching across different shapes, which is the focus of the first part of this dissertation. Therefore, successful techniques have to be at the same time efficient and robust in the presence of noise and shape changes.

Structure and Overview

In this document, we will concentrate on three different problems: shape matching, tangent vector field design and shape exploration. In all three of these areas, we will show that it is possible to obtain very efficient and robust algorithms by considering shapes as functional spaces and by representing various geometric operations as linear operators acting on appropriate real-valued functions. This "functional" point of view is classical in many areas of pure and applied mathematics, including dynamical systems (where the notion of functional maps that we introduce below is closely related to the composition or Koopman operator [Koo31, SM93]) differential geometry (where vector fields are often *defined* by their action on real-valued functions [Spi99, Mor01]) and representation theory among others. However, its utility has only been recently been observed in computer graphics (e.g., [PMT+11] in the context of fluid simulation), and is only now beginning to be appreciated within geometry processing.

As we show below, in addition to the efficiency and the robustness of methods obtained by considering this linear operator point of view of geometry processing, another very significant advantage of these techniques is that they allow to express many different geometric operations in a common language. This means, for example, that it makes it easy to define the push-forward of a vector field with respect to a map, which can be cumbersome in the discrete setting, by simply considering a composition of appropriate operators (as shown in Chapter 6). Similarly, computing near-isometric maps can be reduced to solving a linear system of equations (Chapter 3) and determining the distortion associated with a correspondence to commutativity with the Laplace-Beltrami operator (Chapter 9). Thus, although some examples of these interactions are provided below, I also strongly believe that this is only a small part of the space of possibilities that this functional point of view provides. Some particularly promising future directions are mentioned in the concluding Chapter 11.

More specifically, the rest of the document is organized in three separate parts, each corresponding to a particular problem. In each part we will first present material based on a particular article, and then mention some extensions and follow-up works.

- Part I: Functional Maps for shape correspondence. Material based on [OBCS+12]. Extensions based on [COC14, OMPG13].

- Part II: Operator-based representation of tangent vector fields. Material based on [ABCCO13]. Extensions based on [AOCBC15, COC15].
- Part III: Map-based shape exploration. Material based on [ROA+13]. Extensions based on [CSBC+17].

Finally, I would like to note that although this material represents a large portion of my work during the past 4 years, there is nevertheless a number of articles that I have published that will not be mentioned below, either due to the lack of space, or due to the difference in content. The articles, written after I started my position at Ecole Polytechnique and *not discussed* in this document include:

1. "Informative Descriptor Preservation via Commutativity for Shape Matching," D. Nogneng and M. O., *Computer Graphics Forum (Proc. Eurographics)*, 2017.
2. "Scene Structure Inference through Scene Map Estimation," M. Hueting, V. Pătrăucean, M. O., N. Mitra, *Proc. VMV - Vision, Modeling and Visualization,* 2016.
3. "Persistence-based Pooling for Shape Pose Recognition," T. Bonis, M. O., S. Oudot, F. Chazal, *International Workshop on Computational Topology in Image Context*, 2016.
4. "Stable Region Correspondences Between Non-Isometric Shapes," V. Ganapathi-Subramanian, B. Thibert, M. O., L. Guibas, Computer Graphics Forum *Proc. SGP*, 2016.
5. "CROSSLINK: Joint Understanding of Image and 3D Model Collections through Shape and Camera Pose Variations," M. Hueting, N. Mitra, and M. O., *Proc. SIGGRAPH Asia*, 2015.
6. "Stable Topological Signatures for Points on 3D Shapes," M. Carrière, Steve Y. Oudot, and M. O., *Proc. SGP*, 2015.
7. "Efficient and practical tree preconditioning for solving Laplacian systems," L. Castelli-Aleardi, A. Nolin, and M. O., *Proc. SEA*, 2015.
8. "Affine Invariant Visual Phrases for Object Instance Recognition", V. Patraucean, and M. O., *Proc. MVA*, 2015.
9. "Functional Fluids on Surfaces," O. Azencot, S. Weissmann, M. O., Max Wardetzky, and M. Ben-Chen, *Proc. SGP*, 2014
10. "Unsupervised Multi-Class Joint Image Segmentation," F. Wang, Q. Huang, M. O. and L. Guibas, *Proc. CVPR*, 2014
11. "Persistence-based Structural Recognition," C. Li, M. O. and F. Chazal, *Proc. CVPR*, 2014
12. "Analysis and visualization of maps between shapes," M. O., M. Ben-Chen, F. Chazal, and L. Guibas, *Comp. Graph. Forum (CGF)*, 2013.
13. "Detection of Mirror-Symmetric Image Patches," V. Pătrăucean, R. Grompone

von Gioi, and M. O., *Proc. CVPR Workshop on Symmetry Detection from Real World Images*, 2013.

14. "Feature-based methods in 3d shape analysis," A. Bronstein, M. Bronstein, and M. O., Book Chapter In *3D Imaging, Analysis and Applications,* Springer-Verlag, London, 2012.

Part I

Mappings as Functional Operators

2
Overview

In this part, we present a representation of maps between pairs of shapes that will serve as the basic building block for the rest of the material presented in this document. Key to our approach is a generalization of the notion of map that puts in correspondence real-valued functions rather than points on the shapes. By choosing a multi-scale basis for the function space on each shape, we show how to obtain a representation of a map that is very compact, yet fully suitable for global inference. We also demonstrate that most natural constraints on a map, such as descriptor preservation, landmark correspondences, part preservation and operator commutativity become linear in this formulation. Moreover, the representation naturally supports certain algebraic operations such as map sum, difference and composition, and enables a number of applications, such as function or annotation transfer without establishing point-to-point correspondences. We exploit these properties to devise an efficient shape matching method, at the core of which is a single linear solve. We also show how this representation can be used in segmentation transfer and other applications.

The material in this part is based on the article:

- "Functional maps: a flexible representation of maps between shapes," by M. O., M. Ben-Chen, J. Solomon, A. Butscher and L. Guibas. In *Proc. SIGGRAPH*, 2012.

While the extensions mentioned at the end are based on the articles:

- "Shape matching via quotient spaces," by M. O., Q. Merigot, V. Patraucean, and L. Guibas. In *Proc. SGP (SGP)*, 2013.

- "Supervised Descriptor Learning for Non-Rigid Shape Matching," by E. Corman, M. O., and A. Chambolle. In *Proc. ECCV NORDIA Workshop*, 2014.

3

Functional Maps

3.1 Introduction

Shape matching lies at the core of many operations in geometry processing. While several solutions to rigid matching are well established, non-rigid shape matching remains difficult even when the space of deformations is limited to e.g. approximate isometries. Part of the difficulty in devising a robust and efficient non-rigid shape matching method is that unlike the rigid case, where the deformation can be represented compactly as a rotation and translation, non-rigid shape matchings are most frequently represented as pairings (correspondences) of points or regions on the two shapes. This representation makes direct map estimation and inference intractable, since the space of possible point correspondences is exponential in size. For example, isometric matching techniques try to find correspondences that preserve geodesic distances as well as possible, but such optimization problems can be shown to be an NP-hard subclass of the quadratic assignment problem [Ç98]. Perhaps more importantly, this representation does not naturally support constraints such as map continuity or global consistency.

Additionally, in many practical situations, it is neither possible nor necessary to establish point-to-point correspondences between a pair of shapes, because of inherent shape ambiguities or because the user may only be interested in approximate alignment. Such ambiguous or approximate map inference is difficult to phrase in terms of point-to-point correspondences.

The majority of existing methods try to tackle these challenges by limiting their search for correspondences between a small set of landmark points and extending those to a dense set of correspondences on entire shapes during final post-processing ([BBK06, HAWG08, LF09, KTCO$^+$10, OMMG10, KLF11, TBW$^+$11, SY11] among many others). This strategy has also been justified theoretically, since under general conditions a small set of landmark

correspondences is known to be sufficient to obtain a unique dense mapping between isometric surfaces ([LF09, OMMG10]). Nevertheless, although this landmark-based approach reduces the complexity of the solution space it still relies on representing shape maps as point-to-point correspondences, making it difficult to incorporate global constraints or return meaningful results when establishing point correspondences is not possible due to the presence of only coarse similarities or symmetry ambiguities.

In this chapter we present a novel approach for inference and manipulation of maps between shapes that tries to resolve the issues above in a fundamentally different way. Rather than putting in correspondence points on the shapes, we propose to consider mappings between *functions* defined on the shapes. This notion of correspondence generalizes the standard point-to-point map since every pointwise correspondence induces a mapping between function spaces, while the opposite is, in general, not true. However, this generalized representation is: 1) flexible, since it allows choosing a basis for the function space on each shape and representing the mapping as a change of basis matrix and 2) well-suited for shape-matching, since many natural constraints on the map become *linear* constraints on the functional map. As we show in the rest of this paper, our representation works especially well when combined with the eigenfunctions of the Laplace-Beltrami operator, by benefiting from their multi-scale, "geometry-aware" nature. This allows us, in particular, to devise a simple algorithm that achieves state-of-the art results on an isometric shape matching benchmark and at the heart of which is a single linear solve. We also demonstrate the usefulness of this representation on a number of tasks including improving existing maps, segmentation transfer and joint analysis of shape collections without establishing point-to-point correspondences.

3.2 Related Work

Both shape matching in general and non-rigid shape matching in particular are relatively well-established fields with several recent books (e.g. [BBK08]) and surveys [vKZHCO11] dedicated exclusively to this subject. Below we concentrate on reviewing the various classes of underlying *representations* for maps between pairs of shapes and indicate ways in which they are optimized for in the literature.

As mentioned in the introduction, the vast majority of existing shape matching methods represent a map between a pair of shapes as a point-to-point correspondence. Since it is infeasible to optimize over such correspondences directly, most methods aim to obtain a sparse set of point correspondences and extend them to dense mappings [BBK06, HAWG08, LF09, KTCO+10, OMMG10, KLF11, TBW+11, SY11]. Because sparse point correspondences are inherently discrete, common ways to enforce global consistency include preservation of various quantities between pairs or sets of points, including geodesic distances [BBK06, HAWG08, SY11], various spectral quantities [JZvK07, MHK+08, SH10, OMMG10], or embedding shapes into into canonical domains [LF09] based on landmark correspondences, or a combination of multiple geometric and topological tests [DK11, KTCO+10].

A related set of techniques aims to establish shape part or segment correspondences rather than reliable point-to-point matches, e.g. [GF09, XLZ+10, PBB11, HKG11, vKTS+11]. Such techniques either pre-segment the shape and try to establish part correspondences, or more recently phrase the segmentation and correspondence (and possibly labelling) in a joint optimization framework [KHS10, PBB11, HKG11] which generally avoids the need to establish reliable pointwise correspondences. In this chapter we show how segment correspondences can be used as constraints to establish high quality point matches.

Finally, some methods optimize the deformation of one shape to align it with another, rather than optimizing the correspondences directly [ZSCO+08, YLSL10]. In the majority of cases, however, such methods still rely on point correspondences either during alignment or pre-processing as feature matches.

We also note that some recent methods have concentrated on measuring and optimizing consistency of *sets* of maps [NBCW+11, KLF11] and showed superior performance to optimizing individual correspondences. These applications show the importance of algebraic operations on maps (averages, differences), which are challenging to do in the point-to-point correspondence domain.

Our use of spectral quantities is also closely related to spectral embeddings [Rus07] and their application in shape matching [JZvK07, MHK+08, OSG08]. However, unlike such methods our framework does not assume one-to-one correspondences between eigenfunctions of the Laplace-Beltrami

operator. This difference is crucial for both removing the combinatorial complexity present in these methods (e.g. sign ambiguities, order switching) and achieving superior results in practice.

One common characteristic of all existing non-rigid shape matching methods is that they lead to difficult, non-convex, non-linear optimization problems. In this chapter, we argue that this is primarily because maps between shapes are represented as point or segment correspondences, making it inherently difficult to devise map inference methods using global constraints. On the other hand, we show that by generalizing the notion of a map to include pairings of real-valued functions instead of points, map inference can be phrased as a linear system of equations. One danger of this generalization is that the solution may not correspond to a point-to-point map. We show simple regularization techniques that help avoid this possibility.

Note that analyzing mappings through their effect on function spaces is a common theme used in various fields of mathematics. A famous example can be found in the field of Representation Theory (see for instance [Wey46]). Here, one relates the different ways in which a compact Lie group (a continuous group of transformations, e.g. the group of rotations) can act on itself to the induced *linear* action of the group on functions.

3.3 Contributions

Our key contribution is a representation for maps between pairs of shapes as linear transformations between the corresponding function spaces. We show how this notion of a map generalizes the standard point-to-point representation and yet has the following key advantages:

- By using the Laplace-Beltrami basis for the function space on each shape, the map can be well-approximated using a small number of basis functions and expressed simply as a matrix.
- Most natural constraints on maps, such as descriptor preservation, landmark correspondences, part preservation and operator commutativity become linear in the functional formulation, enabling extremely efficient inference.
- Maps in this representation can be manipulated via standard algebraic operations e.g. addition, subtraction, composition.

Last but not least, functional maps can be useful even when they do not cor-

respond to point-to-point maps for information or attribute transfer between shapes, shape collection analysis, and other shape processing tasks.

3.4 Functional Map Representation

To set the stage for functional mappings as a generalization of classical point-to-point mappings, let $T : M \to N$ be a bijective mapping between manifolds M and N (either continuous or discrete). Then, T induces a natural transformation of derived quantities, such as functions on M. To be precise, if we are given a scalar function $f : M \to \mathbb{R}$ then we obtain a corresponding function $g : N \to \mathbb{R}$ by composition, as in $g = f \circ T^{-1}$. Let us denote this induced transformation by $T_F : \mathcal{F}(M, \mathbb{R}) \to \mathcal{F}(N, \mathbb{R})$, where we use $\mathcal{F}(\cdot, \mathbb{R})$ to denote a generic space of real-valued functions. We call T_F the *functional representation* of the mapping T[1]. We now make the following two simple remarks:

Remark 3.4.1. *The original mapping T can be recovered from T_F.*

Indeed, to recover the image $T(a)$ of any point a on M, construct an indicator function $f : M \to \mathbb{R}$, s.t. $f(a) = 1$ and $f(x) = 0 \; \forall \; x \neq a \in M$. By construction if $g = T_F(f)$, then $g(y) = f \circ T^{-1}(y) = 0$ whenever $T^{-1}(y) \neq a$ and 1 otherwise. Since T is assumed to be invertible, there is a unique point y s.t. $T(a) = y$. Thus, g must be an indicator function of $T(a)$ and $T(a)$ is the unique point $y \in N$ s.t. $g(y) = 1$.

Remark 3.4.2. *For any fixed bijective map $T : M \to N$, T_F is a* linear *map between function spaces.*

To see this, note $T_F(\alpha_1 f_1 + \alpha_2 f_2) = (\alpha_1 f_1 + \alpha_2 f_2) \circ T^{-1} = \alpha_1 f_1 \circ T^{-1} + \alpha_2 f_2 \circ T^{-1} = \alpha_1 T_F(f_1) + \alpha_2 T_F(f_2)$.

We may paraphrase these remarks to say that knowledge of T_F is *equivalent* to knowledge of T. And while T may be a complicated mapping between surfaces, T_F acts linearly between function spaces.

Now suppose that the function space of M is equipped with a basis so that any function $f : M \to \mathbb{R}$ can be represented as a linear combination of basis functions $f = \sum_i a_i \phi_i^M$, with scalar coefficients a_i. Then, we have:

[1]Note that it would perhaps be more natural to define T_F as via pull-back with respect to T rather than T^{-1}, so that T_F would map functions from N to M. We follow this approach in the following chapters, but for simplicity of presentation keep T_F and T to be maps in the same direction here.

$$T_F(f) = T_F\left(\sum_i a_i \phi_i^M\right) = \sum_i a_i T_F\left(\phi_i^M\right) .$$

In addition, if N is equipped with a set of basis functions $\{\phi_j^N\}$, then $T_F\left(\phi_i^M\right) = \sum_j c_{ji}\phi_j^N$ for some $\{c_{ji}\}$, and we obtain:

$$T_F(f) = \sum_i a_i \sum_j c_{ji}\phi_j^N = \sum_j \sum_i a_i c_{ji}\phi_j^N . \tag{3.1}$$

Therefore if we represent f as a vector of coefficients $\mathbf{a} = (a_0, a_1, ...a_i, ...)$ and $g = T_F(f)$ as a vector $\mathbf{b} = (b_0, b_1, ..., b_i, ...)$, then Eq. 3.1 simply says: $b_j = \sum_i c_{ji}a_i$, where c_{ji} is independent of f and is completely determined by the bases and the map T. In particular c_{ji} is the j^{th} coefficient of $T_F(\phi_i^M)$ in the basis $\{\phi_j^N\}$. Note that C has a particularly simple representation if the basis functions $\{\phi_i^N\}$ are orthonormal with respect to some inner product $\langle\cdot,\cdot\rangle$, namely $c_{ji} = \langle T_F(\phi_i^M), \phi_j^N\rangle$.

We conclude with the following key observation:

Remark 3.4.3. *The map T_F can be represented as a (possibly infinite) matrix C s.t. for any function f represented as a vector of coefficients* $\mathbf{a}$ *then $T_F(\mathbf{a}) = C\mathbf{a}$.*

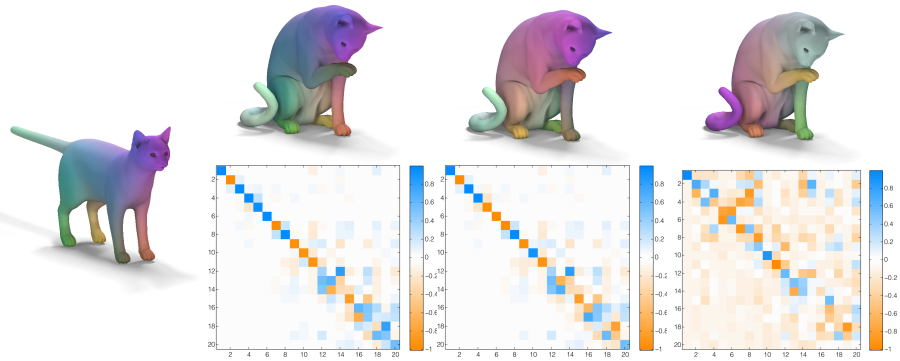
This remark in combination with the previous two remarks shows that the matrix C fully encodes the original map T.

Motivated by this discussion, we now turn towards the definition of *linear functional mappings* that are strictly more general than functional representations of classical point-to-point mappings. The point of view that we take is to downplay the mapping T and focus our attention on the matrix C. We thus define:

Definition 1. *Let $\{\phi_i^M\}$ and $\{\phi_j^N\}$ be bases for $\mathcal{F}(M, \mathbb{R})$ and $\mathcal{F}(N, \mathbb{R})$, respectively. A generalized linear functional mapping $T_F : \mathcal{F}(M, \mathbb{R}) \to \mathcal{F}(N, \mathbb{R})$ with respect to these bases is the operator defined by*

$$T_F\left(\sum_i a_i \phi_i^M\right) = \sum_j \sum_i a_i c_{ji}\phi_j^N ,$$

where c_{ji} is a possibly infinite matrix of real coefficients (subject to conditions that guarantee convergence of the sums above).



(a) source (b) ground-truth map (c) left to right map (d) head to tail map

Figure 3.1: Two shapes with three maps between them, each rendered as a point-to-point mapping through color correspondence (top) and its functional representation (bottom) with colors proportional to matrix values. Note that the least isometric map in (d) leads to a more dense matrix.

Example. As an example, consider a pair of shapes in Figure 3.1 with three bijective maps between then: two approximate isometries (the "natural" map that associates the points of the source with their counterparts of the target, and the left-right mirror symmetric map) and one map that puts the head and tail in correspondence. For each map, the point-to-point representation is shown as color correspondence while the functional representation is shown as a heat map of the matrix $C_{0..20 \times 0..20}$, where we used the Laplace-Beltrami eigenfunctions as the basis for the function space on each shape. Note that the functional representations of the near-isometric maps are close to being sparse and diagonally dominant, whereas the representation of the map that associates the head with the tail is not. Also note that none of the maps is diagonal, an assumption made by previous algorithms [JZvK07, MHK+08, OSG08].

3.5 Functional Representation Properties

As we have noted above, the functional representation of a pointwise bijection can be used to recover its representation as a correspondence, and is thus equivalent. Note, however, that this does not imply that the space of bijections coincides with the space of linear maps between function spaces, as the latter may include functional mappings not associated with any point-

to-point correspondence.

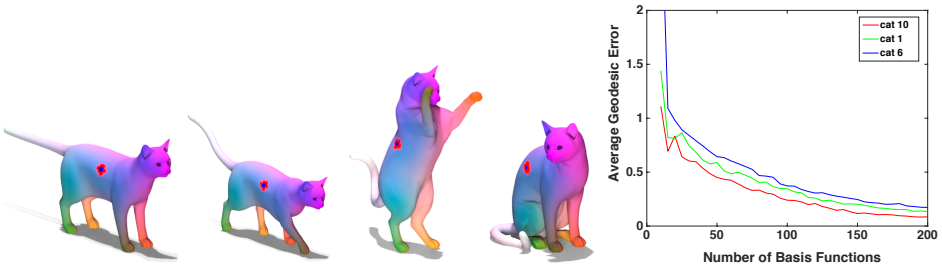
Perhaps the simplest example of this is a functional map D that maps every function on one shape to the constant zero function on the other. Such a map D clearly cannot be associated with any pointwise correspondences since all such functional maps must, by definition, preserve the set of values of each function. Nevertheless, by going to this richer space of correspondences, we obtain a representation that has several key properties making it more suitable for manipulation and inference.

Intuitively, functional maps are easy to manipulate because they can be represented as matrices and thus can benefit from standard linear algebra techniques. To make this intuition practical, however, the size of the matrices must be moderate (i.e., independent of the number of points on the shapes), and furthermore map inference should be phrased in terms of linear constraints in this representation. In the following sections we will show how to achieve these goals first by choosing the appropriate basis for the function space on each shape (Section 3.5.1) and then by showing how many natural constraints on the map can be phrased as linear constraints on the functional map (Section 3.5.3), reducing shape matching to a moderately-sized system of linear equations (Section 3.6).

3.5.1 Choice of basis

As noted above, the functional map representation is *flexible* in the sense that it gives us the freedom to choose the basis functions for the functional spaces of M and N. Indeed, if we choose the basis functions to be indicator functions at the vertices of the shapes, then C is simply the permutation matrix which corresponds to the original mapping. However, other choices of bases are possible, which can lead to significant reductions in representation complexity and are much better suited for near-isometric mappings between shapes, which is desired behavior in many practical applications.

Perhaps the two most important characteristics for choosing a basis for functional maps are compactness and stability. Compactness means that most natural functions on a shape should be well approximated by using a small number of basis elements, while stability means that the *space* of functions spanned by all linear combinations of basis functions must be stable under small shape deformations. These two properties together ensure that we can represent the action T_F using a small and robust subset of basis



(a) source (b) Cat10 (c) Cat1 (d) Cat6 (e) errors

Figure 3.2: Average error vs. number of basis functions used in the representation. For each shape with a known ground-truth point-to-point correspondence (shown as a color correspondence), we computed its functional representation and measured its accuracy in reconstructing the original pointwise map. A geodesic disk of radius 1 is shown on each shape for scale.

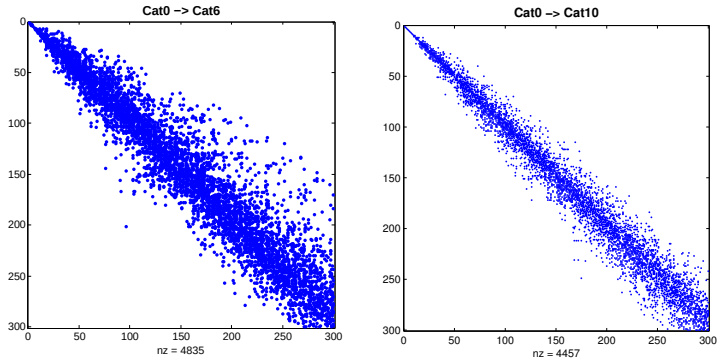
functions and we need only consider a finite *submatrix* $C_{0..m \times 0...n}$, for some moderate values of m and n, of the infinite matrix C (Definition 1). In other words, for a given function f, represented as a vector of coefficients $\mathbf{a} = (a_0, a_1, ...a_i, ...)$, we would like $\sum_j \sum_i a_i c_{ji} \phi_j^N \approx \sum_{j=0}^{n} \sum_{i=0}^{m} a_i c_{ji} \phi_j^N$, for some fixed small values of m and n.

In this work, we will concentrate on shapes undergoing near-isometric deformations, for which we will use the first n Laplace-Beltrami eigenfunctions as the basis for their functional representations (where $n = 100$ throughout all of our experiments, independent of the number of points on the shape). This choice of basis is natural, since eigenfunctions of the Laplace-Beltrami operator are ordered from "low frequency" to "higher frequency," meaning that they provide a natural multi-scale way to approximate functions, and as a result functional mappings, between shapes. Moreover, although individual eigenfunctions are known to be unstable under perturbations, suffering from well-known phenomena such as sign flipping and eigenfunction order changes, the *space* of functions spanned by the first n eigenfunctions of the Laplace-Beltrami operator can be shown to be stable under near-isometries as long as the n^{th} and $(n+1)^{\text{st}}$ eigenvalues are well separated, as shown for example in the work of [Kat95].

To illustrate the role of the size of the basis on the functional representation, we measure the ability of a functional map to capture a ground-truth point-to-point correspondence using a fixed number n of basis functions.

In particular, we consider the eigenfunctions of the standard cotangent weight discretization of the Laplace-Beltrami operator [PP93, MDSB02]. Figure 3.2 shows the average error induced by the functional representation for a set of pairs of deformed versions of the cat shape provided in the TOSCA [BBK08] dataset. Each of these shapes contains 27.8K points, with a known ground-truth correspondence. We represented this pointwise correspondence between the cat0 shape and the others using an increasing number of eigenvectors, and for each point x computed its image as: $T(x) = \arg\min_y ||\delta_y - T_F(\delta_x)||$ where δ_x and δ_y are the projections of the indicator functions at the points x and y onto the corresponding basis (See Section 3.6.1 for details). The error is measured in average geodesic error units, and we plot a geodesic disk of a unit radius around a single (corresponding) point on each shape for reference. Note that the eigenfunctions of the Laplace-Beltrami operator provide a compact representation of the map and that that only $30-40$ eigenfunctions are sufficient to represent the ground truth map to a quality that is extremely close to the ground-truth point-to-point map.

Sparsity. In addition to the multi-scale property of the functional representation with the Laplace-Beltrami eigenfunctions, we also point out that near-isometric maps induce matrices C that are nearly sparse and thus can be stored efficiently. Indeed, if the shapes M and N are isometric and T is an isometry, it is easy to see that the matrix C_{ij} can be non-zero only if ϕ_j^M and ϕ_i^N correspond to the same eigenvalue. In particular, if all eigenvalues are non-repeating, C is a diagonal matrix. In practice, we observe that if T is only approximately an isometry, the matrix C is still close to being sparse, or funnel-shaped. Figure 3.3 shows the sparsity patterns of the matrices C corresponding to two of the maps shown in Figure 3.2. In particular, note that over 94% of the values of these matrices are below 0.1. Let us stress, however, that the functional matrix C stops being diagonal very quickly under even mild non-isometric deformations, and this effect is especially pronounced for high-frequency eigenfunctions (Figure 3.1 illustrates the same effect). While this poses fundamental challenges to previous spectral methods [JZvK07, MHK+08, OSG08], the functional representation naturally encodes such changes.



(a) cat6 (b) cat10

Figure 3.3: Sparsity pattern of the matrices C corresponding to two out of 4 maps shown in Figure 3.2. Only cells where $|C| > 0.11$ are shown. Note that more than 94% are not. Note also that the functional matrix for the more deformed shape cat6 is also farther from being diagonal.

3.5.2 Continuity

Another major advantage of using the functional representation of the mapping is that it naturally handles map continuity unlike the point-to-point or segment-to-segment bijection which is inherently discrete. Here continuity means three distinct phenomena:

Continuity under changes of the input function. This means that the image of a function $T_F(f) = C\mathbf{a}$ varies continuously under changes of the vector of coefficients $\mathbf{a}$ and thus under the changes of the function for a fixed mapping C. This property is useful since in most natural settings the desired mapping is continuous.

Continuity of the image function. The Laplace-Beltrami operator is inherently well-suited for representing *smooth* functions on the shapes. Thus, for any fixed number n, and any set of coefficients $\mathbf{a}$, the function $f = \sum_{i=0}^{n} a_i \phi_i^M$ will be smooth and if we use a truncated matrix $C_{0..n \times 0...n}$ then the image $C\mathbf{a}$ of any function f will be smooth.

Continuity of the representation. Finally, we also note that the functional representation is more amenable to numerical optimization since it is inherently continuous. That is, the matrix C can be modified continuously and still produce meaningful results. Note that there are no inherent

(a) $\alpha = 0$ (b) $\alpha = 0.25$ (c) $\alpha = 0.5$ (d) $\alpha = 0.75$ (e) $\alpha = 1$

Figure 3.4: Mapping of the three coordinate functions from the source shape shown in Figure 3.2a onto the target shape using an interpolation between two maps $C = \alpha C_1 + (1 - \alpha)C_2$. Note that the mapped function varies continuously under changes of the parameter α.

restrictions on the matrix C to be able to establish functional correspondences. Thus, given *any* matrix C and any vector of coefficients $\mathbf{a}$, we can interpret $C\mathbf{a}$ as a functional mapping. To illustrate this, in Figure 3.4 we show the image of a set of three functions from a fixed source shape (shown in Figure 3.2) onto a target shape under a functional correspondence, interpolated between two mappings corresponding to the direct and symmetric shape matching. Note that each mapping is both meaningful and produces intuitive results.

3.5.3 Linearity of constraints

Perhaps even more importantly, the functional representation is particularly well-suited for map inference (i.e., constrained optimization) for the following reason: when the underlying map T (and by extension the matrix C) are unknown, many natural constraints on the map become *linear* constraints in its functional representation. Below we describe the most common scenarios.

Function preservation. Given a pair of functions $f : M \to \mathbb{R}$ and $g : N \to \mathbb{R}$, the correspondence between f and g can be written simply as $C\mathbf{a} = \mathbf{b}$ where C is the functional representation of the map, while $\mathbf{a}$ and $\mathbf{b}$ are the representation of f and g in the chosen bases of M and N. Note that the function preservation constraint can be phrased entirely in terms of the matrix C without knowledge of the underlying correspondence T, since $\mathbf{a}$ and $\mathbf{b}$ do not depend on the map C. This is especially useful for shape matching applications where C is unknown, but could possibly be recovered by phrasing enough constraints of type $C\mathbf{a}_i = \mathbf{b}_i$. The function preservation constraint is quite general and includes the following as special cases.

Descriptor preservation. If f and g are functions corresponding to point descriptors (e.g. $f(x) = \kappa(x)$ where $\kappa(x)$ is Gauss curvature of M at x), then the function preservation constraint simply says that descriptors are approximately preserved by the mapping. Furthermore if the point descriptors are multidimensional so that $f(x) \in \mathbb{R}^k$ for each x then we can phrase k scalar function preservation constraints, one for each dimension of the descriptor.

Landmark point correspondences. If we are given landmark point correspondences $T(x) = y$ for some known $x \in M$ and $y \in N$ (e.g., specified by the user or obtained automatically), we can phrase this knowledge as functional constraints by considering functions f and g that are, for example, distance functions to the landmarks or normally distributed functions around x and y. Indeed, the confidence with which the landmark correspondence is known can be encoded in the functional constraints very naturally (e.g., if it is only known within a certain radius).

Segment correspondences. Similarly, if we are given correspondences between parts of shapes rather than individual points we can phrase such correspondences as functional correspondences again by either considering the indicator functions on the segments or using more robust derived quantities such as the distance function.

In our implementation for finding functional maps between shapes, we impose a variety of functional constraints as described above.

3.5.4 Operator Commutativity

In addition to the function preservation constraint, another class of constraints on the map that induce linear constraints on its functional representation is commutativity with respect to linear operators on M and N. That is, often M and N can be endowed with linear functional operators that we may want to preserve. A first example is a symmetry operator $S_F : \mathcal{F}(M, \mathbb{R}) \to \mathcal{F}(M, \mathbb{R})$ which associates with every function $f : M \to \mathbb{R}$ another function $S_F(f) : M \to \mathbb{R}$ obtained as $S_F(f)(x) = f(S^{-1}(x))$, where $S : M \to M$ is some symmetry of M. A second example is the Laplace-Beltrami operator and derived operators (e.g. the heat operator), which are preserved under isometries. The operators on M and N can be quite general,

however, and can represent any association of functions on the manifold. In any case, given functional operators S_F^M and S_F^N on M and N respectively, it may be natural to require that the functional map C commute with these operators. In particular: $S_F^N C = C S_F^M$ or $\|S_F^M C - C S_F^N\| = 0$. Note that this constraint also leads to linear equations in the elements of C.

3.5.5 Regularization Constraints

Note that although we mentioned in Section 3.5.2 that there are no inherent constraints on the matrix C to be a functional map, this does not mean that *any* matrix C is associated with a point-to-point map. Indeed, while every bijective map T has a functional representation through the matrix C, the converse is not necessarily true. Thus, there may be constraints on the functional representation *if it is known to be associated with a point-to-point map*. Although finding such constraints is difficult in general, a very useful observation is the following (See [OBCS+12] for a proof):

Theorem 3.5.1. *(1) If the basis functions are discrete and orthonormal with respect to the standard inner product, i.e.* $\sum_x \phi_i(x)\phi_j(x) = \delta_{ij}$*, or if the underlying map T (discrete or continuous) is volume preserving, i.e.* $\mu^M(x) = \mu^N(T(x))$ *where μ^M and μ^N are volume elements on M and N respectively, then the matrix C associated with the functional representation of T must be* orthonormal. *(2) If the underlying map T is an isometry then T commutes with the Laplace-Beltrami operator.*

It follows that in most natural settings, e.g. when one expects isometries between shapes, if one is using the functional representation to obtain a point-to-point map it is most meaningful to consider orthonormal or nearly-orthonormal functional map matrices. Furthermore, it makes sense to incorporate commutativity with the Laplace-Beltrami operators into the regularization.

3.5.6 Map Inversion and Composition

A challenging task when considering point-to-point mappings between shapes is map inversion, i.e. given a map $T : M \to N$ that is not necessarily bijective, one is required to find a meaningful version of $T^{-1} : N \to M$. In the functional representation finding an inverse can be done simply by finding

an inverse of the mapping matrix C. Moreover, because for near-isometric maps we expect this matrix to be close to diagonal (or "funnel" shaped as shown in Figure 3.3) it is reasonable to take the inverse of the approximating submatrix of C. Finally, in light of Theorem 3.5.1 this can be done by simply taking the transpose of C or its approximation. We note that similarly, map composition becomes simple matrix multiplication in the functional representation, which has been exploited when we use our representation for joint map inference on a collection of shapes [OBCS+12].

3.6 Functional Map Inference

As mentioned in Section 3.5, functional shape maps are well-suited for inference because of their continuous nature and because a large number of constraints become linear in this representation. In this section we discuss how such inference can be done in practice. For this suppose we are given a pair of discrete shapes represented as meshes, with the corresponding Laplace-Beltrami eigenfunctions. Our goal is to find the underlying functional map represented as a matrix C. The simplest way to do so is to construct a large system of linear equations, where each equation corresponds to one of the constraints mentioned above (either a functional constraint or the operator commutativity constraint) and find the best functional map by finding the matrix C that best satisfies the constraints in the least squares sense.

3.6.1 Efficient Conversion to Point-to-Point

As mentioned in Section 3.4, given a bijection T between two discrete shapes, and the basis vectors of their function spaces, the functional representation C of the map T can be obtained by solving a linear system.

To reconstruct the original mapping from the functional representation, however, is more challenging. The simplest method alluded to in Remark 3.4.1 to find a corresponding point $y \in N$ to a given point $x \in M$ would require constructing a function $f : M \to \mathbb{R}$ (either the indicator function, or a highly peaked Gaussian around x) obtaining its image $g = T_F(f)$ using C and declaring y to be the point at which $g(y)$ obtains the maximum. Such a method, however, would require $O(V_N V_M)$ operations for a pair of meshes with V_N and V_M vertices. Such complexity may be prohibitively expensive in practice for large meshes. To obtain a more efficient method,

Algorithm 1 Functional Map Inference For Matching

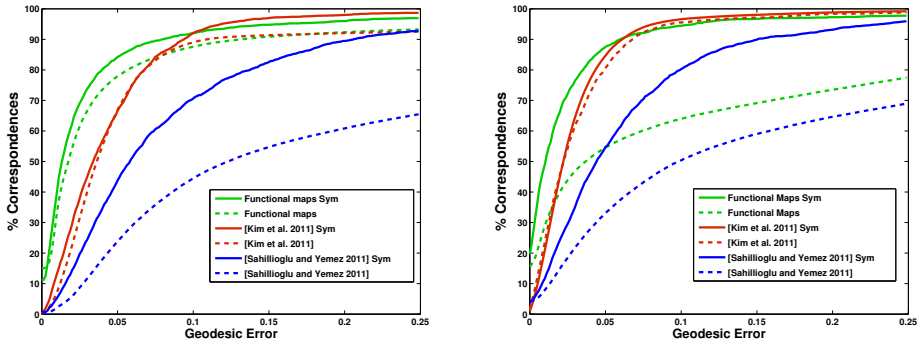
1. Compute a set of descriptors for each point on M and N, and create function preservation constraints.
2. If landmark correspondences or part decomposition constraints are known, compute the function preservation constraints using those.
3. Include operator commutativity constraints for relevant linear operators on M and N (e.g. Laplace-Beltrami or symmetry).
4. Incorporate the constraints into a linear system and solve it in the least squares sense to compute the optimal C.
5. Refine the initial solution C with the iterative method of Section 3.6.2.
6. If point correspondences are required, obtain them using the method in Section 3.6.1.

note that in the Laplace-Beltrami basis δ_x, the delta function around a point $x \in M$, has the coefficients: $a_i = \phi_i^M(x)$. This can be seen for example, since $\delta_x = \lim_{t\to 0^+} k_t^M(x,\cdot) = \lim_{t\to 0^+} \sum_{i=0}^{\infty} e^{-t\lambda_i}\phi_i^M(x)\phi_i^M(\cdot)$, where $k_t^M(\cdot,\cdot)$ is the heat kernel at time t on the shape M.

Therefore, given a matrix Φ^M of the Laplace-Beltrami eigenfunctions of M, where each column corresponds to a point and each row to an eigenfunction, one can find the image of *all* of the delta functions centered at points of M simply as $C\Phi^M$. Now recall that by Plancherel's theorem, given two functions g_1 and g_2 both defined on N, with spectral coefficients $\mathbf{b}_1$ and $\mathbf{b}_2$, $\sum_i (b_{1i} - b_{2i})^2 = \int_N (g_1(y) - g_2(y))^2 \mu(y)$. That is, the distances between the coefficient vectors is equal to the L^2 difference between the functions themselves. Therefore an efficient way to find correspondences between points is to consider for every point of $C\Phi^M$ its nearest neighbor in Φ^N. Using an efficient proximity search structure, such as a kd-tree, this procedure will require only $O(V_N \log V_N + V_M \log V_N)$ operations, giving a significant efficiency increase in practice.

3.6.2 Post-Processing Iterative Refinement

The observation made in Section 3.6.1 can also be used to refine a given matrix C to make it closer to a point-to-point map. Suppose we are given an initial estimate matrix C_0 that we believe comes from a point-to-point



(a) SCAPE (b) TOSCA

Figure 3.5: Comparison of our method with the state-of-the-art methods of Kim et al. [KLF11] and Sahillioglu and Yemez [SY11] on two datasets: SCAPE [ASK+05] and TOSCA [BBK08] with and without symmetric maps allowed (solid and dashed lines respectively). Note that since our method is intrinsic only symmetric (solid line) evaluation is meaningful.

map T. As noted in Section 3.6.1, theoretically C_0 must be such that each column of $C_0\Phi^M$ coincides with some column of Φ^N. If we treat Φ^M and Φ^N as two point clouds with dimensionality equal to the number of eigenvalues used in the functional representation C_0 then this means that C_0 must *align* Φ^M and Φ^N. Moreover, since by Theorem 3.5.1 we expect the mapping matrix C_0 to be orthonormal, we can phrase the problem of finding the optimal mapping matrix C as rigid alignment between Φ^M and Φ^N. Thus an iterative refinement of C_0 can be obtained via:

1. For each column x of $C_0\Phi^M$ find the closest $\tilde{x}$ in Φ^N.
2. Find the optimal orthonormal C minimizing $\sum \|Cx - \tilde{x}\|$.
3. Set $C_0 = C$ and iterate until convergence.

Note that this technique is identical to the standard Iterative Closest Point algorithm of Besl & McKay, [BM92], except that it is done in the embedded functional space, rather than the standard Euclidean space. Note also that this method *cannot* be used on its own to obtain the optimal functional matrix C because the embedding Φ^M and Φ^N are only defined up to a sign change (or more generally an orthonormal transformation within an eigenspace). Therefore, it is essential to have a good initial estimate matrix

C_0. Finally, note that the output of this procedure is not only a functional matrix C but also a point-to-point correspondence given by nearest neighbor assignment between points on M and N. We will use this method to obtain good point-to-point maps when we apply these observations to devise an efficient shape matching method in Section 3.7.

Relation to Existing Methods. We remark that this refinement step is similar to existing spectral matching methods such as [JZvK07, MHK+08, SH10]. However, in addition to having a good initial estimate C_0, our method is different since we allow "mixing" across eigenvectors corresponding to different eigenvalues. In addition, the functional representation allows to formulate other constraints such as operator commutativity (Section 3.5.4) and *represent* the map itself compactly.

3.7 Shape Matching

In this section we describe a simple yet very effective method for non-rigid shape matching based on the functional map representation.

The simplest version of the shape matching algorithm is summarized in Algorithm 1. Namely, suppose we are given two shapes M and N in their discrete (e.g. mesh) representation, and the Laplace-Beltrami eigendecomposition. Then, we simply compute functional constraints that correspond to descriptor and segment preservation constraints together with the operator commutativity, form a linear system of equations and solve it in the least squares sense. If necessary, we refine the solution using the method in Section 3.6.2 and compute the point-to-point map using the method in Section 3.6.1.

3.7.1 Implementation

The key ingredients necessary to implement this method in practice are the computation of the eigendecomposition of the Laplace-Beltrami operator, the descriptors used in the function preservation constraints, and a method to obtain landmark or segment correspondences. Note that our framework allows great flexibility for the choice of descriptors and correspondence constraints since they all fit into a general function preservation framework. In our implementation we have used the cotangent scheme [MDSB02] for

the Laplace-Beltrami operator on meshed surfaces. We also used the Wave Kernel Signature (WKS) and Heat Kernel Signature descriptors of [ASC11] and [SOG09]. Because the method described above is fully intrinsic and does not distinguish between left and right symmetries, it is also important to resolve ambiguities using correspondence constraints. However, since point-to-point correspondences (e.g. landmark) are generally unstable and difficult to obtain without manual intervention, we have used segment correspondences instead. Towards this goal, we first pre-segment every shape using the persistence-based segmentation technique of [SOCG10] with the WKS at a fixed energy value of the underlying function (we used $e = 5$ in all examples below). This gives a relatively stable segmentation with a small number of segments (between 3 and 7 in the shapes we examined). Given a pair of shapes, we first compute the segment correspondence constraints. For this, we first compute the set of candidate pairs of segments from the two shapes by computing segment descriptors and finding the ones likely to match. For segment descriptors we use the sum of the WKS values of the points in the segment. Given a pair of candidate segment matches $s1, s2$ on M and N respectively, we construct a set of functional constraints using the Heat Kernel Map [OMMG10] based on segment correspondences. We combine these together with the Laplace-Beltrami commutativity constraint and the WKS constraints into a single linear system and solve it to find the optimal functional mapping matrix C. Finally, we refine the solution using the iterative method described in Section 3.6.2 and find the final dense point-to-point correspondences using the method in 3.6.1.

3.7.2 Results

We have evaluated our basic method for computing point-to-point correspondences on the shape matching benchmark of Kim et al. [KLF11] in which the authors showed state-of-the art results using their Blended Intrinsic Maps (BIM) approach. Using the correspondence evaluation, Figure 3.5 shows the results of our automated shape matching on two standard datasets used in the benchmark of Kim et al. [KLF11] on which their method reported significant improvement over prior work. In addition, we evaluated a recent shape matching method by Sahillioglu and Yemez [SY11] which did not appear in the benchmark. The graphs show the percent of correspondences which have geodesic error smaller than a threshold. Note that our

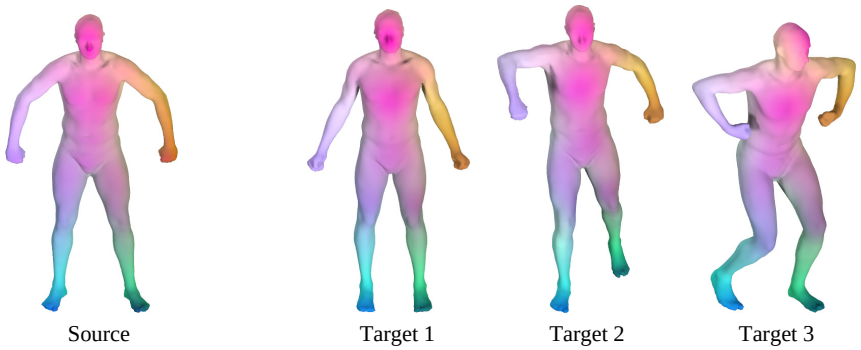


Figure 3.6: Maps between remeshed versions of shapes from the SCAPE collection, mapping the coordinate functions from the source to the three target shapes using an inferred functional map.

method shows quality improvement over Blended Intrinsic Maps on both datasets. Note also that all of the parameters for our system were fixed before running the benchmark evaluation and were therefore not optimized for benchmark performance in any way.

Although the shapes in both SCAPE and TOSCA datasets have the same connectivity structure, this information is not used by our method, and is not needed for applying our algorithm. To demonstrate this, Figure 3.6 shows three maps computed by our method between a source and three target shapes from the SCAPE collection, all remeshed with uniform remeshing. We show the map by transferring the XYZ coordinate functions to the target shapes using the inferred functional maps. These functions are then rendered as RGB channels on the source and target shapes.

3.8 Other Applications

3.8.1 Function (Segmentation) Transfer

As mentioned earlier, one of the advantages of the functional representation is that it reduces the transfer of functions across shapes to a matrix product, without resorting to establishing point-to-point correspondences. This is particularly useful since function transfer is one of the key applications of maps between shapes and obtaining point-to-point correspondences is often challenging. We illustrate the performance of this idea on the task of segmentation transfer across shapes. Here we are given a pair of shapes

Figure 3.7: Segmentation transfer using the functional map representation. For each pair of shapes we show 3 figures: the user-provided source segmentation of the first shape, the image of one of the indicator functions of a segment using the functional map computed with our method, and the final segmentation transfer onto the target shape. Note that point correspondences were not computed at any point during this procedure.

where one of the shapes is pre-segmented and the goal is to find a compatible segmentation of the second shape. To achieve this task we simply construct an indicator function for each of the segments on the source shape and use the functional map to transfer this indicator function. Then each point on the target map will have a set of values for each of the transferred segments. Finally, if "hard" clustering is required, we associate with the point the segment that produced the maximum of these values.

Figure 3.7 shows this idea applied to several shapes from the TOSCA and SCAPE datasets. For each pair of shapes we show the image of the the indicator function of one of the segments as well as the final "hard" segmentation. Note that throughout this procedure no point-to-point correspondences were used.

4 Extensions

The ideas presented in the previous chapter provide a general framework for shape matching using the functional map representation. In particular, Algorithm 1, mentioned in Section 3.7, is very modular and each of its parts can be adapted in different scenarios. Below we outline two extensions of this basic approach, which can be used in the supervised setting (i.e., when some example correspondences are given) and in the presence of ambiguities resulting from shape symmetries respectively.

4.1 Supervised Descriptor Learning

One of limitations of the shape matching algorithm presented in the previous chapter is that it assumes that the functional constraints, including, e.g., the descriptor correspondences are given as input. This means that in practice, the user needs to specify the best descriptors by hand, and the specific descriptor choice can have a significant impact on the resulting functional maps. For the near-isometric shapes present in the TOSCA and SCAPE datasets that we considered, we used the diffusion-based descriptors such as the HKS [SOG09] and WKS [ASC11]. However, in general, finding the best descriptors can be a tedious domain-specific task.

On the other hand, it is possible to formulate the problem of choosing the optimal functional constraints, including descriptor preservation constraints as a supervised learning problem. Namely, suppose we *are given* some example shapes with known (functional, not necessarily point-to-point) correspondences. Then, we can find the constraints such that the resulting functional maps would be as close as possible to the given ground-truth ones, and ultimately use those constraints on new test data.

In [COC14] we presented an approach that for supervised descriptor learning, specifically adapted to the functional maps framework. In particular, it uses the fact that given the functional constraints, the resulting optimization

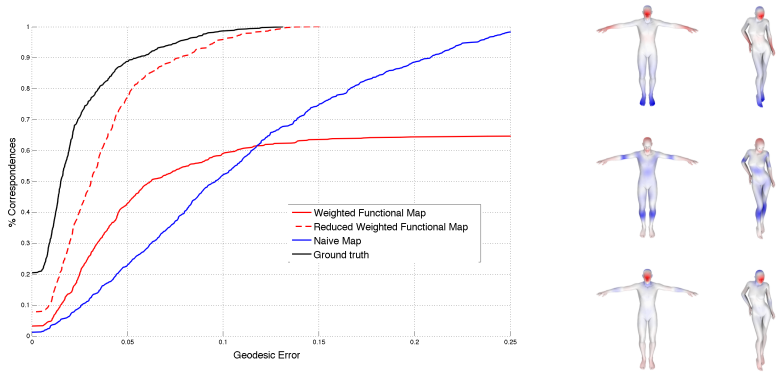


Figure 4.1: Left: quantitative correspondence improvement with learning (dashed red) and without (solid blue). Right: detected stable parts.

problem described in Algorithm 1 is simply a linear least squares system. Thus, if we are given a set of pre-computed descriptors, and if we let D represent a set of scalar weights associated with the descriptors, then the optimal functional map : $C_{\text{opt}}(D)$ is obtained by solving a linear system of equations. In [COC14], we proposed to find the optimal descriptor weights by solving:

$$D^* = \arg\min_D \sum \|C_{\text{opt}}(D) - C_{\text{ground truth}}\|,$$

where the sum is over all the given training shapes. By exploiting the fact that $C(D)$ is simply the solution to a linear system, which is a consequence of the functional map representation, we proposed an efficient method for descriptor computing optimal descriptor weights given example correspondences. The choice of the norm for comparing the functional maps turns out to be very important, as for example Frobenius norm, penalizes the entire functional space uniformly. Finally, after computing the optimal weights, we were also able to extract an optimal *shape parts*, where the computed maps are more reliable, and the optimal descriptors are consistently informative. Figure 4.1 shows both quantitative improvement of using the learning procedure on a shape benchmark, and some detected stable parts.

4.2 Subspace (quotient) matching

At the same time, we have also used the functional map representation to find correspondences between shapes in the presence of ambiguities resulting from shape symmetries. Shape symmetries are a common problem for

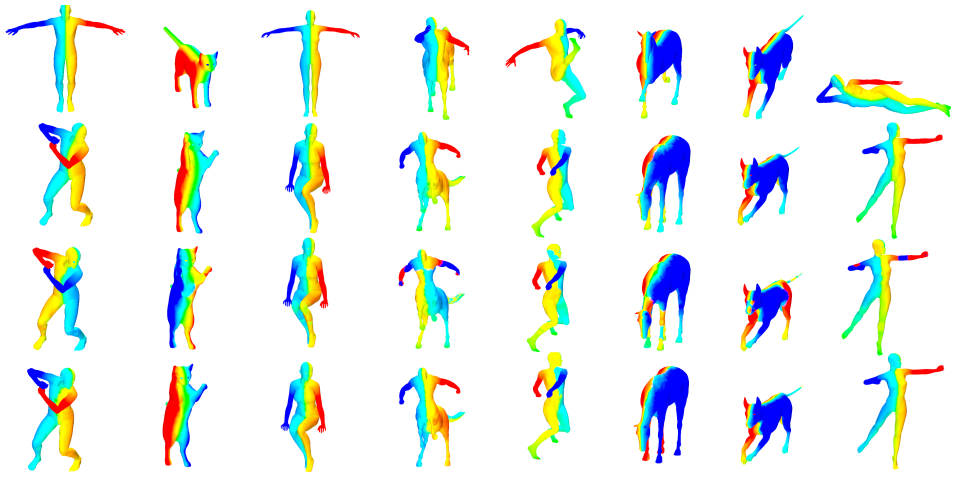


Figure 4.2: Correspondences obtained on the TOSCA dataset [BBK08] between a target shape (first row) and a source shape, using BIM [KLF11] (second row), the original functional maps approach [OBCS+12] (third row), and our semi-quotient method (last row). Corresponding points are indicated through the same color.

isometric shape matching methods, they result in multiple equally good solutions for any method based on purely intrinsic quantities such as geodesic distances or diffusion-based descriptors. This can be a particularly significant issue when solving optimization problems, as the presence of multiple optima can lead to instabilities in the final solution [OHG11].

In [OMPG13] we proposed a method specifically meant to deal with shape correspondences in the presence of symmetry ambiguities by exploiting the fact that the space of functions can easily be factorised into symmetric and non-symmetric ones, without explicitly decomposing the shape geometry. Namely, we proposed a method such that allows to find a functional correspondence between two shapes, when one of them is known to be symmetric with a given symmetry self-map. By decomposing the space of functions into appropriate subspaces, we were able to obtain a method that results in a stable shape matching optimisation problem while being able output multiple direct and symmetric correspondences. Figure 4.2 shows an example of correspondences obtained using this method.

Part II

Tangent Vector Fields as Functional Operators

5 Overview

In this part, we show that the operator-based approach for representing maps, described in the previous part, can also be successfully applied to representing and manipulating tangent vector fields on surfaces. In particular, unlike the commonly used representations of a vector field as an assignment of vectors to the faces of the mesh, or as real values on edges, we argue that, similarly to mappings, vector fields can also be naturally viewed as *linear operators* whose domain and range are functions defined on the mesh. Although this point of view is common in differential geometry it has so far not been adopted in geometry processing applications. We recall the theoretical properties of vector fields represented as operators, and show that *composition* of vector fields with other functional operators is natural in this setup. This leads to the characterization of vector field properties through commutativity with other operators such as the Laplace-Beltrami and symmetry operators, as well as to a straightforward definition of differential properties such as the Lie derivative. Finally, we demonstrate a range of applications, such as Killing vector field design, symmetric vector field estimation and joint design on multiple surfaces.

The material in this part is based on the article:

- "An operator approach to tangent vector field processing," by O. Azencot, M. Ben-Chen, F. Chazal and M. O. In *Proc. SGP*, 2013.

While the extensions mentioned at the end are based on the articles:

- "Continuous Matching via Vector Field Flow," by E. Corman, M. O., and A. Chambolle. In *Proc. SGP*, 2015.
- "Discrete Derivatives of Vector Fields on Surfaces – An Operator Approach," by O. Azencot, M. O., F. Chazal, and M. Ben-Chen. In *Transactions on Graphics (TOG)*, 2015.

6

Vector Fields as Operators

6.1 Introduction

Manipulating and designing tangent vector fields on discrete domains is a fundamental operation in areas as diverse as dynamical systems, finite elements and geometry processing. The first question that needs to be addressed before designing a vector field processing toolbox, is how will the vector fields be represented in the discrete setting? In this chapter we propose a representation, which is inspired by the point of view of vector fields in differential geometry as *operators* or *derivations*.

In the continuous setting, there are a few common ways of defining a tangent vector field on a surface. The first, is to consider a smooth assignment of a vector in the tangent space at each point on the surface. This is, perhaps, the most intuitive way to extend the definition of vector fields from the Euclidean space to manifolds. However, it comes at a price, since on a curved surface one must keep track of the relation between the tangent spaces at different points. A natural discretization corresponding to this point of view (used e.g. in [PP03]) is to assign a single Euclidean vector to each simplex of a polygonal mesh (either a vertex or a face), and to extend them through interpolation. While this representation is clearly useful in many applications, the non-trivial relationships between the tangent spaces complicate tasks such as vector field design and manipulation.

An alternative approach in the continuous case, is to work with *differential forms* (see e.g. [Mor01]) which are linear operators taking tangent vector fields to scalar functions. In the discrete setting this point of view leads to the famous Discrete Exterior Calculus [Hir03, FSDH07], where discrete 1-forms are represented as real-valued functions defined over the edges of the mesh. While this approach is coordinate-free (as no basis for the tangent space needs to be defined), and has many advantages over the previous

method, there are still some operations which are natural in the continuous setting, and not easily representable in DEC. For example, the flow of a tangent vector field is a one parameter set of self-maps and various vector field properties can be defined by composition with its flow, an operation which is somewhat challenging to perform using DEC.

Finally, another point of view of tangent vector fields in the continuous case is to consider their *action* on scalar functions. Namely, for a given vector field, its *covariant derivative* is an operator that associates to any smooth function f on the smooth surface another function which equals the derivative of f in the direction given by the vector field. It is well known that a vector field can be recovered from its covariant derivative operator, and thus any vector field can be uniquely represented as a functional operator. We will refer to these operators as functional vector fields (FVFs). Note, that while this point of view is classical in differential geometry, it has so far received limited attention in geometry processing.

In this chapter, we argue that the operator point of view yields a useful coordinate-free representation of vector fields on discrete surfaces that is complementary to existing representations and that can facilitate a number of novel applications. For example, we show that constructing a Killing vector field [Pet97] on a surface can be done by simply finding a functional vector field that commutes with the Laplace-Beltrami operator. Furthermore, we show that it is possible to transport vector fields across surfaces, find symmetric vector fields and even compute the flow of a vector field easily by employing the natural relationship between FVFs and functional maps described in the previous part. Finally, the Lie derivative of two vector fields is given by the commutator of the two respective operators, and as a result the covariant derivative of a tangent vector field with respect to another can be computed through the Koszul formula [Pet97].

To employ this representation in practice, we show that for a suitable choice of basis, a functional vector field can be represented as a (possibly infinite) matrix. As not all such matrices represent vector fields, we show how to parameterize the space of vector fields using a *basis for the operators*. With these tools in hand, we propose a Finite Element-based discretization for FVFs, and demonstrate its consistency and empirical convergence. Finally, we apply our framework to various vector field processing tasks in vector field design and analysis, which were challenging so far.

6.1.1 Related Work

The body of literature devoted to vector fields in graphics, visualization and geometry processing is vast and a full overview is beyond our scope. Thus, we will focus on the *representation* and discretization of vector fields, as these aspects of vector field processing are most related to our work.

One approach to discretization (e.g. [PP03, TLHD03]) is to use piecewise constant vector fields, where vectors are defined per face and represented in the standard basis in $\mathbb{R}^3$. This approach is simple and allows to define discrete versions of standard operators such as *div* and *curl*, which are consistent with their continuous counterparts (e.g. one can define a discrete Hodge decomposition [PP03]). However, since the representation is based on coordinate frames, it makes vector field design challenging as the relationship between tangent spaces is non-trivial, leading to difficult optimization problems.

An alternative discretization of vector fields was suggested as part of the formalism of Discrete Exterior Calculus (DEC) [Hir03], where vector fields are identified with discrete 1-forms, represented as a single scalar per edge. This approach is inherently coordinate-free, allowing to formulate vector field design as a linear system [FSDH07]. Unfortunately, computing the Lie derivative of vector fields remains a complex task using DEC (as shown in [MMP+11]).

Vector field design and processing applications are also tightly connected to the analysis of rotationally symmetric (RoSy) fields, as described in e.g. [PZ07, RVAL09, CDS10]. In the latter work, for example, a vector field (or a symmetric direction field) is represented using an angle per edge (an angle-valued dual 1-form), which represents how the vector changes between neighboring triangles. While these approaches are also coordinate-free and lead to linear optimization problems for direction field design, it is not clear how vector field valued operators can be represented in such a setup.

We argue that in addition to the existing discretization methods, it is often useful to represent vector fields through their covariant derivatives as linear functional operators. This representation is coordinate-free and, in addition, elucidates the intimate connection between vector fields and self maps of the surface, allowing us to extend the basic vector field processing toolbox to computational tasks which are challenging using existing discretization tools.

Note that the operator representation of vector fields has been used in

the context of fluid simulation by Pavlov et al. [PMT+11]. However, in that work, the authors were primarily interested in representing divergence-free vector fields and did not use this representation for tangent vector field analysis and design. On the other hand, we consider general vector fields, demonstrate how this representation can be used for vector field processing, and show a deep connection with the functional map framework.

6.1.2 Contributions

Our main observation is that tangent vector fields can be represented in a coordinate-free way as functional operators. While this view is classical in differential geometry [Mor01], it has so far received limited attention in geometry processing. Using this perspective we:

- Show how functional vector fields can be naturally composed with other operators, and thus relate vector fields to functional maps and the Laplace-Beltrami operator (Section 6.2) among others.
- Provide a novel coordinate-free discretization of tangent vector fields (Section 6.4).
- Describe various applications for vector field processing including Killing vector field design, design of symmetric vector fields and joint vector field design on multiple shapes, which are all easily solvable as linear systems in our framework (Section 6.5).

6.2 Vector Fields as Operators

In this section we define the coordinate-free view of vector fields as abstract *derivations* of functions in the continuous setting. This point of view is well-known in differential geometry (see e.g. [Mor01] for an excellent reference). Thus, we only recall the standard definition and its main properties.

6.2.1 The Covariant Derivative of Functions

We first assume that we are given a compact smooth Riemannian manifold M and a tangent vector field V, which can be thought of as a smooth assignment of a tangent vector $V(p)$ to each point $p \in M$. The vector field defines a one-parameter family of maps, $\Phi_V^t : M \to M$ for $t \in \mathbb{R}$, called

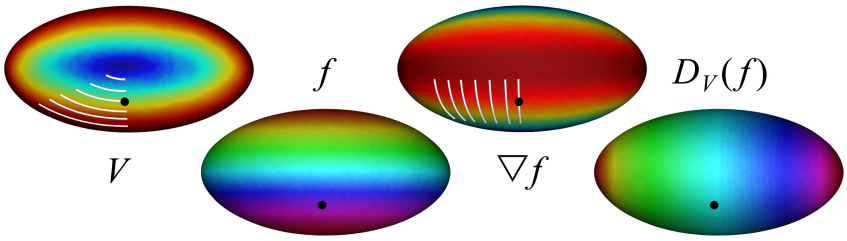


Figure 6.1: Given a vector field V (left) and a function f (center left), the inner product of ∇f (center right) with V is the covariant derivative $D_V(f)$ (right). For the marked point, for example, V is orthogonal to ∇f, yielding 0 for $D_V(f)$. Vector fields are visualized by color coding their norm, and showing a few flow lines for a fixed time t.

the *flow* of V. The flow is formally defined as the unique solution to the differential equation:

$$\frac{d}{dt}\Phi_V^t(p) = V(\Phi_V^t(p)), \qquad \Phi_V^0(p) = p. \tag{6.1}$$

Then, for a given function $f \in C^\infty(M)$, the *covariant derivative* $D_V(f)$ of f with respect to V is a function g, which intuitively measures the change in f with respect to the flow under V. Formally,

$$g(p) = D_V(f)(p) = \lim_{t\to 0} \frac{f(\Phi_V^t(p)) - f(p)}{t}.$$

A classical result in Riemannian geometry (See [Mor01], p. 148) is that the covariant derivative can also be computed as:

$$D_V(f)(p) = g(p) = \langle (\nabla f)(p), V(p) \rangle_p , \tag{6.2}$$

where $\langle , \rangle_p$ denotes the inner product in the tangent space of p, and ∇f is the gradient of f (see Figure 6.1).

6.2.2 The Covariant Derivative as a Functional Operator

We stress that D_V is best viewed as an *operator*, which maps smooth functions on M to smooth functions on M. Moreover, one can show that D_V encodes V so that if V_1 and V_2 are vector fields such that $D_{V_1} f = D_{V_2} f$ for any $f \in C^\infty(M)$, then $V_1 = V_2$ (see [Mor01], p. 38). Said differently, there is no loss of information when moving from V to D_V.

The covariant derivative (viewed as a functional operator, i.e. an FVF) satisfies the following two key properties:

Linearity:

$$D(\alpha f + \beta g) = \alpha D(f) + \beta D(g), \tag{6.3}$$

and Leibnitz (product) rule:

$$D(fg) = fD(g) + gD(f). \tag{6.4}$$

Conversely, a functional operator D corresponds to a vector field, if and only if it satisfies (6.3) and (6.4) (see [Spi99] p. 79).

Why are these the necessary properties for operators that represent vector fields? Intuitively, this is because vector fields can be thought of as first order directional derivatives, which have two essential properties. First, that *constant functions are mapped to the zero function.* And second, that $D_V(f)$ depends on f *only to first order.*

One of the advantages of considering vector fields as abstract derivations is that this point of view can be generalized to settings where differential quantities are not well-defined. For example, on a discrete surface there is no well-defined normal direction at vertices and edges. By working with purely algebraic constructs, such as linear operators, we can define differentiation without requiring the concept of a limit, which is useful when the underlying surface is not continuous and such a limit does not exist. Moreover, as we will see, the operator point of view makes it easier to manipulate vector fields and relate them to other functional operators.

6.2.3 Properties

While the operator point of view is equivalent to the standard notion of a vector field as a smooth assignment of tangent vectors, certain operations are more natural in this representation. Below we list such operations, which we will use in our applications in Section 6.5. For the proofs of all lemmas, please see [ABCCO13].

Operator composition. By using the operator point of view of vector fields, it becomes easy to define their *composition* both with other vector fields and other more general functional operators. Unfortunately, given two vector fields D_{V_1} and D_{V_2}, the operator $D_{V_1} \circ D_{V_2}$ does not necessarily correspond to a vector field. However, one can modify this operator to obtain a fundamental notion:

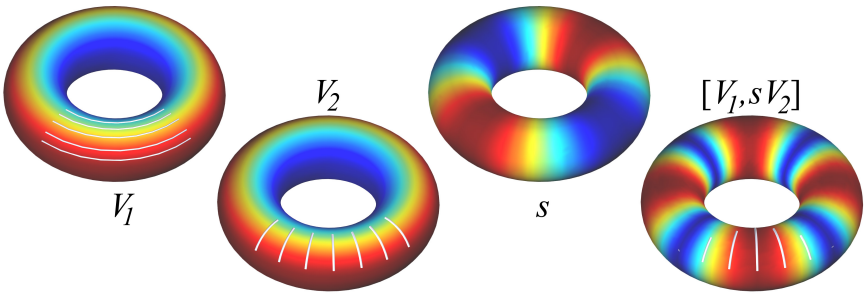


Figure 6.2: Two orthogonal vector fields on the torus V_1, V_2, whose Lie derivative is 0. Modifying the norm of V_2 using a function s yields a Lie derivative which is parallel to V_2.

Lie derivative of a vector field. Given two vector fields V_1 and V_2, the Lie derivative (or Lie bracket) of V_2 with respect to V_1 is a vector field V_3 defined as:

$$\mathcal{L}_{V_1}(V_2) = [V_1, V_2] = D_{V_3} = D_{V_1} \circ D_{V_2} - D_{V_2} \circ D_{V_1}. \tag{6.5}$$

It is easy to see that D_{V_3} thus defined is both linear and satisfies the product rule. Hence, D_{V_3} corresponds to a unique vector field V_3. Intuitively, the Lie derivative captures the commutativity of the flows of V_1 and V_2. In particular, the Lie derivative is zero if and only if the flows defined by V_1 and V_2 commute (see [Spi99], p.157):

$$\Phi_{V_2}^{-s} \circ \Phi_{V_1}^{-t} \circ \Phi_{V_2}^{s} \circ \Phi_{V_1}^{t} = Id \quad \forall s, t \in \mathbb{R} \tag{6.6}$$

Figure 6.2 illustrates the computation of the Lie derivative on a torus. We consider two vector fields V_1 and V_2 whose flows commute. The average norm of $[V_1, V_2]$ computed using the discrete operators we describe in Section 6.4 is on the order of 1e-8, close to 0 as expected. In general, if $[V_1, V_2] = 0$, it can be shown that for any scalar function $s : M \to \mathbb{R}$, $[V_1, sV_2]$ must be parallel to V_2. In Figure 6.2, we show a scaling function s, and the computed vector field $V_3 = [V_1, sV_2]$, which is parallel to V_2, as expected.

Composition with other operators. Of course, it is possible to consider the composition of the FVF operator D_V with other functional operators. Interestingly, the commutativity of D_V with a differential operator D is closely related to the commutativity of its flow with D.

Lemma 6.2.1. *Let T_F^t, $t \in \mathbb{R}$ be the functional operator representations of the flow diffeomorphisms $\Phi_V^t : M \to M$ of V, defined by $T_F^t(f) = f \circ \Phi_V^t$ for*

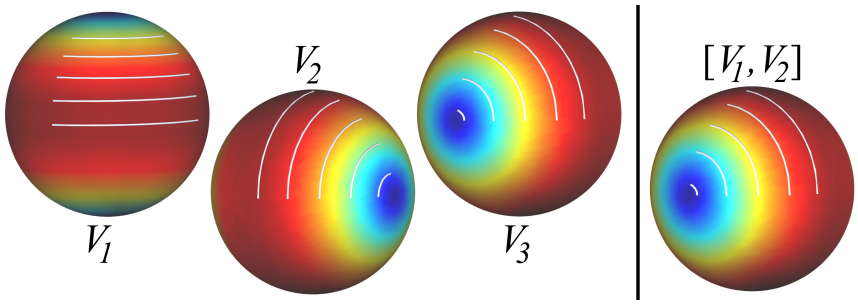


Figure 6.3: Using commutativity with L, we compute the KVFs on the sphere (V_1, V_2, V_3). Alternatively, we compute $V_4 = [V_1, V_2]$, note the similarity of V_3 and V_4.

any function $f \in C^\infty(M)$. If D is a linear partial differential operator then $D_V \circ D = D \circ D_V$ if and only if for any $t \in \mathbb{R}$, $T_F^t \circ D = D \circ T_F^t$.

For example, on a Riemannian manifold, we can consider composition with the Laplace-Beltrami operator L. The commutativity of D_V with L is then closely related to the metric distortion imposed by the flow of V. In particular, recall that a vector field is called a Killing vector field (KVF) if Φ_V^t is an isometry for all t (see [Pet97], Chapter 7). Then:

Lemma 6.2.2. *A vector field V is a Killing vector field if and only if $D_V \circ L = L \circ D_V$.*

From this lemma, it is easy to see that KVFs form a group under the Lie derivative. Indeed, the following lemma, which follows directly from the definition of the Lie derivative, is useful in general:

Lemma 6.2.3. *Given two vector fields D_{V_1} and D_{V_2} that both commute with some operator D, the Lie derivative $\mathcal{L}_{V_1}(V_2)$ will also commute with D.*

Figure 6.3 demonstrates these properties on the sphere. On the left, we show V_1, V_2, V_3, the three KVFs of the sphere, computed using Lemma 6.2.2 by constructing a linear system (as explained in Section 6.5). On the right, we show $V_4 = [V_1, V_2]$, which was computed as the Lie bracket of the first two KVFs. Note the similarity between V_3 and V_4. We will use these results for designing approximate KVFs in Section 6.5.

Composition with mappings. In many settings we are also interested in the relation between vector fields on multiple surfaces related by

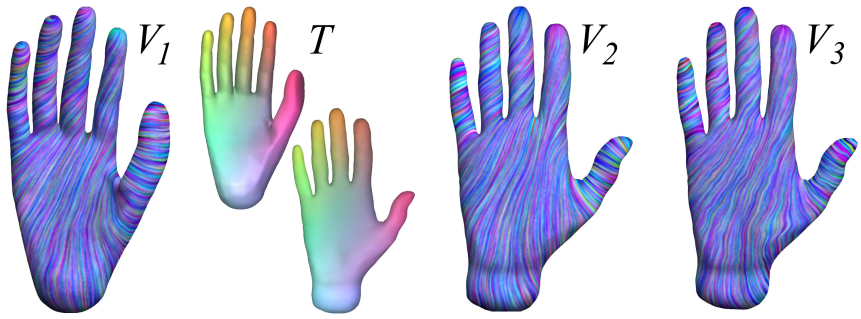


Figure 6.4: Given a vector field V_1 on M and a map $T : M \to N$, we generate a vector field V_2 on N using Lemma 6.2.4. Compare with directly transporting V_1 using the differential of the map, yielding V_3. Note the ringing artifacts in V_3.

mappings. In particular, given a vector field V_1 on surface M and a *diffeomorphism* $T : M \to N$, one can define the vector field V_2 on surface N via the push forward: $V_2(q) = dT(V_1(T^{-1}(q)))$. Note that in the discrete case, it is often difficult to compute the differential dT of a map T between shapes with different discretizations. At the same time, one can equivalently define the vector field V_2 using the operator approach, without relying on dT, by using the functional representation of the map T.

As mentioned in Part I (Chapter 3), the functional representation T_F of a map T is a linear operator on the space of functions, taking functions on N to functions on M defined by $T_F(g) = g \circ T$ for any function $g \in C^\infty(N)$. This means that the functional vector field D_{V_2}, and thus V_2 itself can be obtained by simple composition of three linear functional operators without the need to estimate the differential dT, using:

Lemma 6.2.4. $D_{V_2} = (T_F)^{-1} \circ D_{V_1} \circ T_F$.

Figure 6.4 illustrates vector field transportation using this approach (vector fields are visualized using [PZ11]). Given V_1 on M, and a map $T : M \to N$, we generate V_2 on N using Lemma 6.2.4. V_3 is computed using the differential of the map, given by the affine map between corresponding triangles. Note that V_3 tends to be noisy, due to the locality of the transport procedure, as opposed to the smoother V_2. Furthermore, this approach is applicable to shapes with different connectivity, where computing dT is challenging. In Section 6.5 we use a similar approach to *design* vector fields which are consistent with the map $T : M \to N$.

Vector field flow. The FVF D_V representing a vector field is also closely related to the functional representation of the flow Φ_V^t. In particular:

Lemma 6.2.5. *Let $T^t = \Phi_V^t$ be the self-map associated with the flow of V at time t. Then if T_F^t is the functional representation of T^t, for any real analytic function f (see [DFN92], p. 210):*

$$T_F^t f = \exp(t\ D_V) f = \sum_{k=0}^{\infty} \frac{(tD_V)^k f}{k!}.$$

This lemma is particularly useful since it allows to avoid the potentially costly solution of the system of differential equations (6.1) and directly estimate the functional representation of the map Φ_V^t through operator exponentiation. Note that D_V is a moderately sized matrix when represented in a basis, and therefore its exponent can be computed efficiently. Figure 6.5 shows an example of function flow using this method.

Covariant derivative of a tangent vector field. Some PDEs can be described using the *covariant derivative* [Mor01] of a vector field V_1 with respect to another vector field V_2, denoted $\nabla_{V_2} V_1$. For planar vector fields, for example, $\nabla_{V_2} V_1 = J(V_1)V_2$, where $J(V_1)$ is the Jacobian matrix of V_1.

On a surface, however, this representation requires a basis for the tangent space at every point, and a suitable *connection* that allows to transport a vector $V(p)$ to a neighboring point q, which makes $\nabla_{V_2} V_1$ elusive to compute in a coordinate-free way. Fortunately, there is an intimate connection between the Lie and covariant derivatives of vector fields, through the Koszul formula, ([Pet97], p. 25):

$$\begin{aligned} 2g(\nabla_{V_1} V_2, Z) = {} & D_{V_1}(g(V_2, Z)) - g(V_1, [V_2, Z]) \\ & + D_{V_2}(g(V_1, Z)) - g(V_2, [V_1, Z]) \\ & - D_Z(g(V_1, V_2)) + g(Z, [V_1, V_2]). \end{aligned} \tag{6.7}$$

Here, Z is an arbitrary vector field, $g(\cdot,\cdot) = \langle \cdot, \cdot \rangle_p$ is the inner product in the tangent space of p, and $[\cdot,\cdot]$ is the Lie bracket (Eq. 6.5). Hence, given an operator representation of D_{V_1} and D_{V_2}, we can use Equation (6.7) to compute $\nabla_{V_1} V_2$. We leave further investigation of this direction, and possible applications for future work.

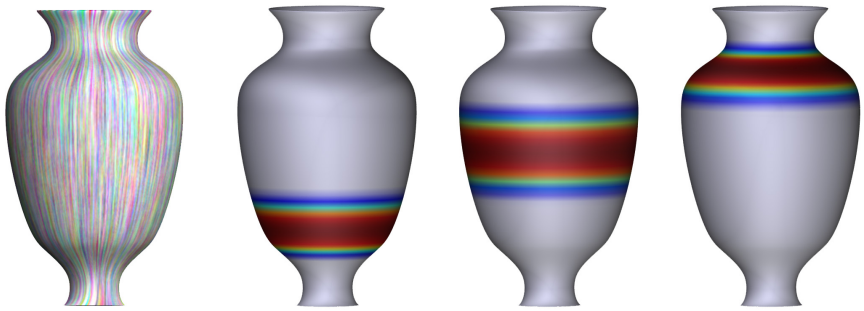


Figure 6.5: Applying the flow of a vector field (left) to a function (center left) using Lemma 6.2.5. (center right, right) The function after the flow, for two sample t values.

6.3 Representation in a Basis

The properties mentioned above suggest that representing and analyzing tangent vector fields through their functional representation can enable a number of applications which are challenging using standard methods. Our goal, therefore, will be to represent this operator such that it can be easily analyzed and manipulated in practice.

6.3.1 Basis for the Function Space

As mentioned in Section 6.2.2, an FVF is a linear operator acting on smooth functions defined on the manifold. In practice, we will assume that the functional space of interest can be endowed with a (possibly infinite) basis, so that any function can be represented as a linear combination of some basis functions $\{\phi_i\}$. Then, for any given function $f = \sum_i a_i \phi_i$, we have that $g = D_V(f) = D_V(\sum_i a_i \phi_i) = \sum_i a_i D_V(\phi_i)$. Since $D_V(\phi_i)$ is also a function, it can be represented in the basis as $D_V(\phi_i) = \sum_j D_{ji}\phi_j$. Therefore, $g = \sum_j (\sum_i D_{ji} a_i)\phi_j = \sum_j b_j \phi_j$. In other words, if one thinks of the coefficients a_i, b_i as vectors $\mathbf{a}, \mathbf{b}$ and D as a matrix, then the transformation between the basis representations of f and $g = D_V(f)$ is given by: $\mathbf{b} = D\mathbf{a}$.

When the basis functions ϕ_i are orthonormal with respect to the standard functional inner product on M, i.e. $\int_M \phi_i \phi_j d\mu = 1$ if $i = j$ and 0 otherwise, then the $(j,i)^{\text{th}}$ element D_{ji} of the FVF corresponding to V is given by:

$$D_{ji} = \int_M \phi_j D_V(\phi_i) d\mu(p) = \int_M \phi_j(p) \left\langle V(p), \nabla \phi_i \right\rangle_p d\mu(p), \tag{6.8}$$

where $\langle , \rangle_p$ denotes the inner product in the tangent space of the point p, and $d\mu(p)$ represents the volume measure at p.

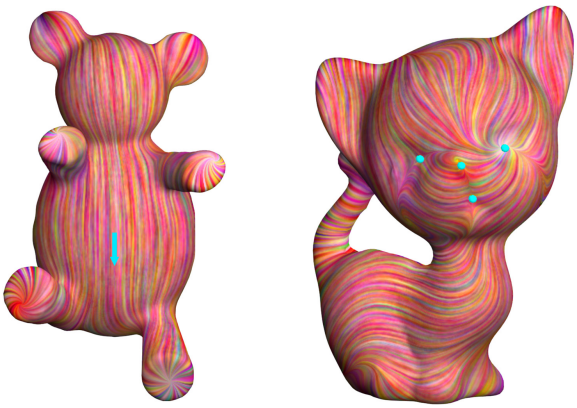


Figure 6.6: Prescribing directional constraints (left) or singularities (right).

The Laplace-Beltrami basis. A useful basis for the space of smooth functions on a compact manifold, which we will often use in practice, is the basis given by the eigenfunctions of the Laplace-Beltrami operator (note that on a compact manifold the space $L^2(M)$ is strictly larger than the space of smooth functions). Since each eigenfunction of the Laplace-Beltrami operator is smooth, Equation (6.8) is well defined. One advantage of this basis is that the basis functions are ordered and can be attributed a notion of *scale*, given by the corresponding eigenvalue. This has been exploited in a number of scenarios including the work on functional maps in Part I, where a mapping between two shapes is compactly encoded using a sub-matrix of a possibly infinite functional map matrix. This choice of basis yields a compact representation of the FVF operator as an $N_f \times N_f$ matrix, where N_f is the number of basis functions we use.

6.3.2 Parameterization with Basis Operators

As mentioned in Section 6.2.2, the space of linear functional operators is strictly larger than the space of vector fields. Therefore, in order to work with this representation in practice, it is useful to have a *parametrization* of the space of FVFs, which is easy to manipulate.

One possible such parameterization, is to consider a basis for the space of tangent vector fields ψ_i, and to represent an operator D_V as a linear combination of the functional vector field operators D_{ψ_i}. In our work, we consider the eigenfunctions of the 1-form Laplace-de Rham operator to generate a basis for the 1-forms on a surface, and use these as a basis for the tangent

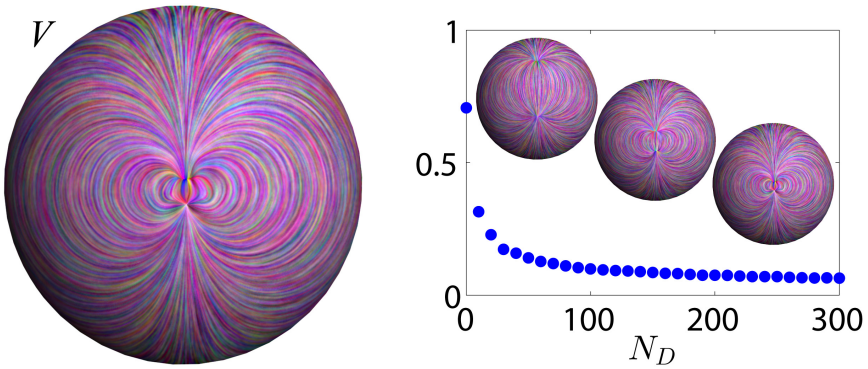


Figure 6.7: Given a vector field (left), we reconstruct it with growing accuracy by increasing the number of basis operators N_D (right). Note that the index 2 singularity is accurately reconstructed given enough basis operators.

vectors, by duality [Mor01].

Given such basis operators D_{ψ_i}, the FVF operator D_V that represents a vector field $V = \sum_i a_i \psi_i$ is given by: $D_V = \sum_i a_i D_{\psi_i}$. Note, that this basis is also ordered, so that smoother vector fields can be represented using fewer basis operators. In practice, we truncate the basis, and limit the number of basis operators to a fixed value N_D.

With this parameterization, it is straightforward to use the properties we mentioned in Section 6.2 to design a vector field that has various desirable characteristics, simply by solving a linear system for the coefficients a_i. Figure 6.6 shows a vector field designed by posing a small number of directional constraints (one direction for the teddy (left) and 4 zero valued vectors for the kitten (right)), and solving for the coefficients as explained in Section 6.5.

Figure 6.7 demonstrates the effect of using a varying number of basis operators. Given a direction field (left), we project it on a growing number of basis operators and show the reconstruction error as a function of N_D (right). We additionally show the reconstructed vector field, for a few choices of N_D. Note, that although the direction field is smooth, due to the jump from unit length norm to zero norm at the singular point, it is difficult to reconstruct this vector field exactly. However, using a growing number of basis operators we can approximate better this discontinuity in scale.

6.4 Discretization

So far we have described the properties of tangent vector fields as functional operators in the continuous case. In this section we will focus on the discretization of these concepts to surfaces which are represented as triangle meshes. We propose a finite-element based discretization, and discuss its consistency and experimental convergence properties.

6.4.1 Representation

We will first address the following problem: given a triangle mesh $M = (X, F, N)$, where X are the vertices, F the faces and N the normals to the faces, and a piecewise constant tangent vector field $V = \{v_r \in \mathbb{R}^3 | r \in F, v_r \perp N_r\}$, how do we represent the functional vector field D_V?

The answer is in fact straightforward, when we consider the representation of D_V in the functional basis given by the standard hat functions. On a triangle mesh we can represent functions in a piecewise linear basis, namely $f(p) = \sum_{j=1}^{|X|} b_j \gamma_j(p)$, where γ_j are the standard hat functions (valued 1 at vertex i and 0 at all other vertices), and $b_j \in \mathbb{R}$ are the coefficients. Now, given the function $f(p) = \sum_j b_j \gamma_j(p)$, and a piecewise constant vector field V, we wish to compute $g = D_V(f)$. We set $g(p) = \sum_j a_j \gamma_j(p)$, and solve (6.2) in the weak sense, as is standard in Finite Element Analysis (see [AFW06] for a complete discussion of this approach):

$$\int_M \gamma_i g d\mu = \int_M \gamma_i D_V(f) d\mu, \quad \forall i.$$

Plugging in the expressions for f, g and D_V we get $\forall i$:

$$\sum_j a_j \int_M \gamma_i \gamma_j d\mu = \sum_j b_j \int_M \gamma_i \langle \nabla \gamma_j, V \rangle d\mu. \tag{6.9}$$

The integrands in (6.9) vanish everywhere, except on the set of triangles $R_{ij} \subset F$, for which both γ_i and γ_j are non-zero. For $i = j$, these are the triangles neighboring the vertex i. For $i \neq j$, we have that (i, j) must be an edge, and R_{ij} contains only the two triangles which share that edge.

This leads to $\sum_j a_j B_{ij} = \sum_j b_j S_{ij}$, where:

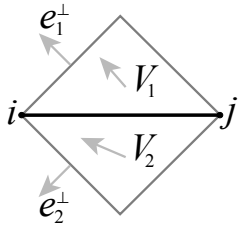
$$B_{ij} = \sum_{t_r \in R_{ij}} \int_{t_r} \gamma_i \gamma_j d\mu, \quad S_{ij} = \sum_{t_r \in R_{ij}} \int_{t_r} \gamma_i \langle \nabla \gamma_j, V \rangle d\mu.$$

Computing the elements B_{ij} yields the standard mass matrix used in the solution of Laplacian systems, whereas S_{ij} is given by (see the inset figure

for the notations):

$$S_{ij} = \frac{1}{6}\left(\left\langle V_1, e_1^{\perp}\right\rangle + \left\langle V_2, e_2^{\perp}\right\rangle\right)$$

$$S_{ii} = -\sum_{j \in N(i)} S_{ij}.$$



Here, r_1 and r_2 are the two faces that share the edge (i, j), V_1 is the value of V on the face r_1, $e_1^{\perp}$ is the rotation by $\pi/2$ of the edge opposite to the vertex j in the face r_1 (similarly for V_2 and $e_2^{\perp}$), and $N(i)$ are the neighboring vertices of vertex i. See [ABCCO13] for the derivation.

We further replace B with a diagonal lumped mass matrix W of the Voronoi areas w_i of the vertices [BKP+10], and get:

$$\mathbf{a} = \hat{D}_V \mathbf{b}, \quad \hat{D}_V = W^{-1} S. \tag{6.10}$$

Note, that the size of $\hat{D}_V$ is $|X| \times |X|$, but it is sparse, as only the diagonal and entries of adjacent vertices are non-zero.

It is sometimes useful to decompose $\hat{D}_V$ as a product of two operators: $\hat{D}_V = P_{|X|\times|F|}(\hat{D}_V^F)_{|F|\times|X|}$, where P is independent of V and depends only on the mesh. We take:

$$(P)_{ir} = \frac{1}{3w_i}\mathcal{A}_r, \quad (\hat{D}_V^F)_{ri} = \langle \nabla\gamma_i, V\rangle_r\,, \tag{6.11}$$

where $\mathcal{A}_r$ is the area of the triangle t_r. In fact, the operator $\hat{D}_V^F$ is simply the smooth operator D_V per triangle, where V is fixed. Therefore, it preserves most of the properties of its smooth counterpart. However, to get an operator which commutes with other operators, we need to apply P, averaging values from faces to vertices. This introduces a discretization error into our formulation, due to the discontinuity of the vector field near the vertices.

Alternatively, we can use the first N_f eigenvectors $\hat{\phi}_i$ of the discrete Laplace-Beltrami operator as the basis for the function space, and then D_V will be represented using an $N_f \times N_f$ matrix, which we will denote by $\hat{D}_V^{LB}$. We compute $\hat{D}_V^{LB}$ by using a change of basis:

$$\hat{D}_V^{LB} = B^{+}\hat{D}_V B, \tag{6.12}$$

where B is a matrix whose columns are $\hat{\phi}_i$ and B^{+} is its pseudo-inverse.

This representation introduces some additional error, due to the truncation of the basis, and there exists a trade-off between the complexity of the representation (in terms of N_f) and the amount of detail the functions we work with can represent.

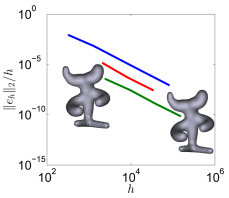
6.4.2 Properties

It is interesting to investigate which properties of D_V are preserved from the smooth case, and which are not but converge under refinement of the mesh.

Constant functions. We have that $D_V(c) = 0$, for any constant function c. It is easy to see this property is preserved in the discrete case, since the rows of $\hat{D}_V$ sum to zero, hence the constant functions are in its kernel.

Product rule. The continuous D_V fulfills two defining properties: linearity (Equation (6.3)) and the Leibnitz product rule (Equation (6.4)). Since $\hat{D}_V$ is a matrix, linearity is clearly satisfied. However, as we work in a limited subspace of functions, the product rule is no longer valid: given two PL functions f, g, their pointwise product fg is no longer PL, and therefore we cannot apply $\hat{D}_V$ to it. However, we can show empirically that when applying increasingly finer discretizations of D_V to increasingly finer discretizations of continuous functions f, g, the product rule error decreases.

Let f_h, g_h, be the two random smooth piecewise linear functions defined on a mesh with h vertices, and take V to be a smooth tangential vector field. Now, for every h, compute the error $e_h = D_V(f_h g_h) - (g_h D_V(f_h) + f_h D_V(g_h))$, where the multiplication is done vertex-wise. The inset figure shows the graph of $\|e_h\|_2/h$ as a function of h, in *loglog* scale, for a few choices of models. Note that the graph is linear, implying exponential convergence under refinement.



Uniqueness. The correspondence between a vector field V and its FVF operator D_V is one-to-one and onto in the continuous case, implying that given an operator D_V we can uniquely reconstruct the vector field V. This property, unfortunately, may not hold in the discrete case. We do, however have the following weaker result:

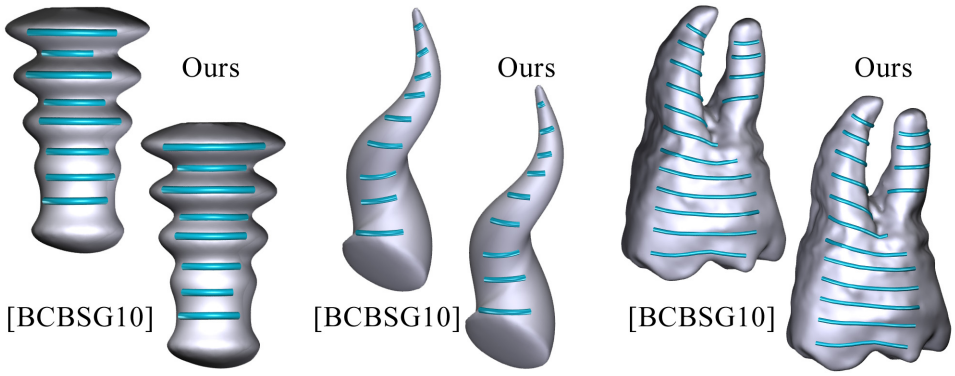


Figure 6.8: Geodesic distances between pairs of starting points are measured before and after the flow. Comparing the normalized average error for the models shown yields (left to right): 0.2,0.96, 2.47 for our method, and 0.23,1.15, 4.5 for [BCBSG10] (units are average edge length).

Lemma 6.4.1. *Let $M = (X, F, N)$ and let V_1, V_2 be two piecewise constant vector fields on M. Then: $\hat{D}^F_{V_1} = \hat{D}^F_{V_2}$ if and only if $V_1 = V_2$.*

In practice, given an operator $\hat{D}_V$ we reconstruct the corresponding vector field V by projecting on the operator basis, as described in Section 6.3.2.

Metric invariance. The continuous functional vector field operator D_V commutes with the pushforward under a map. Specifically, given a bijective diffeomorphism $T : M \to N$, a vector field V_1 on M and a function $f : M \to \mathbb{R}$, we have that $D_{V_1}(f)(p) = D_{V_2}(f \circ T^{-1})(T(p))$, where $V_2 = dT(V_1(p))$, and dT is the differential of T. As a consequence, D_V does not depend on the embedding of the shape M.

As we do not have the uniqueness property, the discrete metric invariance property is also limited to the $\hat{D}^F_V$ operator:

Lemma 6.4.2. *Let $M_1 = (X_1, F, N_1)$ and $M_2 = (X_2, F, N_2)$ be two triangle meshes with the same connectivity but different metric (i.e. different embedding). Additionally, let V_1 be a piecewise constant vector field on M_1, then $\hat{D}^F_{V_1} = \hat{D}^F_{V_2}$.*

Here $(V_2)_r = A(V_1)_r$, where A is the linear transformation that takes the triangle r in M_1 to the corresponding triangle in M_2. Note that $\hat{D}_{V_i}$ is computed using the embedding X_i.

Integration by parts. For a closed surface, we have that $\int_M f(\nabla \mathrm{div}(V)) = \int_M \langle \nabla f, V \rangle = \int_M D_V(f)$, for all $f : M \to \mathbb{R}$. This holds exactly in the dis-

crete case, when using the standard vertex-based discrete divergence, defined as in [PP03]:

Lemma 6.4.3. *Let $M = (X, F, N)$, V a piecewise constant vector field on M, $f = \sum_i f_i \gamma_i$ a PL function on M, and w_i the Voronoi area weights, then:*

$$\sum_{i=1}^{|X|} w_i (\hat{D}_V f)_i = \sum_{i=1}^{|X|} w_i f_i (\mathrm{div}(V))_i.$$

6.5 Applications

In this section, we describe how our representation can be used to compute vector fields which have various desirable properties. While some of the suggested applications have been attempted before (e.g. designing vector fields using direction and singularity constraints [FSDH07, CDS10], computing Killing vector fields [BCBSG10] and symmetric vector fields [PLPZ12], among others), our framework is unique in that it allows to combine any such constraints into a single optimization problem. In addition, we provide a proof-of-concept for more advanced tools, such as jointly designing vector fields on two or more surfaces.

6.5.1 Implementation Details

Given a mesh M, scalars N_f, N_D and a set of desired properties for a vector field, we propose the following algorithm:

1. Compute the first N_f eigenfunctions of the LB operator $\hat{\phi}_i$, using the area normalized cotangent scheme [BKP+10].

2. Compute the first N_D 1-form eigenfunctions of the Laplace-de Rham operator, and convert those to piecewise constant vector fields $\hat{\psi}_i$. We used the definitions from [FSDH07] for both operations.

3. Convert $\hat{\psi}_i$ to $\hat{D}^{LB}_{\hat{\psi}_i}$ using Equation (6.12).

4. Optimize simultaneously for the vector field $V = \sum_i a_i \hat{\psi}_i$ and its functional representation $D_V = \sum_i a_i \hat{D}^{LB}_{\hat{\psi}_i}$, by solving a linear system for a_i. The joint formulation allows us to stack constraints which are best represented using the operator (e.g. commutativity constraints) together with constraints which require the vector field (e.g. prescribed

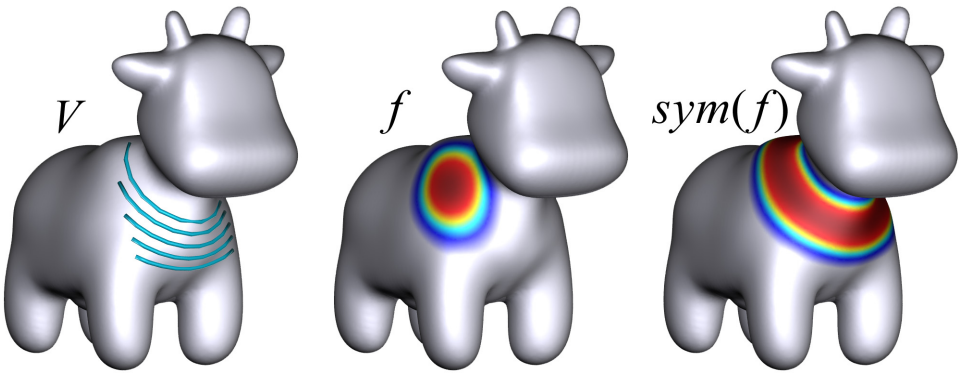


Figure 6.9: An AKVF V (left), an indicator function f (center), and its symmetrization computed by projecting f on the kernel of D_V (right).

directions at specified locations). This yields a linear system $\mathbf{Wa} = \mathbf{c}$, which we solve in the least squares sense.

5. Output the computed vector field $V = \sum_i a_i \hat{\psi}_i$.

Throughout our experiments we used meshes in the range of 5k-200k vertices, with N_f and N_D between 50 and 300, depending on the experiment. The computational time was dominated by the eigen-decompositions and took a few minutes on a standard laptop.

Figures 6.2, 6.3, 6.4 and 6.6 from the previous sections were generated using this framework. In addition, we describe a few examples of potential applications of our framework, related to the properties discussed in Section 6.2.

6.5.2 Approximate Killing Vector Fields

Lemma 6.2.2 provides a linear constraint on the FVF operator, which guarantees that a given vector field is a KVF. We can use this result, and optimize for the best KVF on a given surface, by optimizing for a set of coefficients $\mathbf{a}$ such that the resulting operator D_V will commute with the Laplace-Beltrami operator, i.e. $||D_V \circ L - L \circ D_V|| = 0$. Here we get a homogeneous system $\mathbf{Wa} = 0$, hence the AKVF is the singular vector corresponding to the lowest singular value.

Figure 6.8 shows a comparison of the resulting vector fields with the results of the state-of-the-art algorithm [BCBSG10]. The comparison is done using the same meshes, where on each mesh we pick a few vertices and show the flow lines for a fixed time t starting from these vertices. Note,

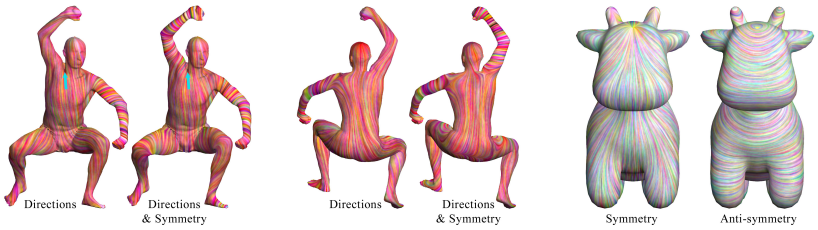


Figure 6.10: On the human model (left and center) we show design results with and without symmetry constraints - note the difference on the right hand. On the spot model (right) we show symmetric and anti-symmetric vector fields.

that we achieve similar results, but in our framework we can easily combine the KVF constraint with other constraints such as commutativity with a symmetry operator.

Interestingly, the spectral decomposition of the functional vector field operator is meaningful and potentially useful in applications. Specifically, functions are in the kernel of D_V if and only if they are fixed points of the flow Φ_V^t for all t (since $D_V f = 0$ if and only if $\exp(tD_V)f = f, \forall t$). Therefore, the kernel of an AKVF operator spans the linear subspace of symmetric functions under the corresponding symmetry. This implies, that given an arbitrary function f, we can symmetrize it by projecting it onto the kernel of such an operator. Figure 6.9 shows an example of an AKVF V, an indicator function f and its symmetrization $sym(f)$.

6.5.3 Composition with Mappings

Given a self-map S, we design a symmetric vector field by posing a constraint of the form $||D_V \circ S - S \circ D_V|| = 0$. Figure 6.10 (left and center) shows an example of a vector field designed with directional constraints and one designed with both directional and symmetry commutativity constraints. Note the difference on the hand of the model, as the symmetric constraints enforce similar behavior on both hands. Additionally, we can define an anti-symmetric vector field, by requiring $V(S(p)) = -V(p)$, where S is the symmetry map. To enforce this requirement, we use the constraint $||D_V \circ S + S \circ D_V|| = 0$. Figure 6.10 (right) shows an example of symmetric and anti-symmetric vector fields.

Given a collection of shapes, a desirable goal when designing vector fields is to have different constraints on each shape, yet generate compatible vector fields across the collection. In Figure 6.11 (right) we achieve this goal using

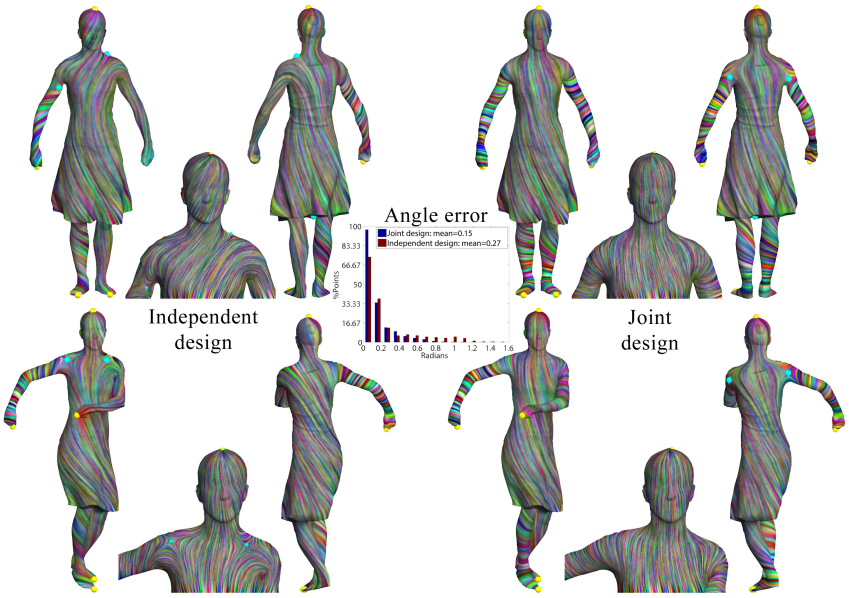


Figure 6.11: (left) Independent design on two shapes which are in correspondence does not yield a consistent vector field, even if compatible constraints are used. (right) Solving jointly using our framework yields consistent vector fields (note the corresponding locations of the singularities on the back of the shape). See the text for details.

the map composition property. We are given two shapes M_1 and M_2 and a functional map T_F between the corresponding function spaces. In addition, on each shape we are given a set of directional constraints c_1, c_2. We wish to generate vector fields V_i on the shapes M_i, such that V_i commute with T_F, and fulfill the constraints. A natural approach would be to transfer the constraints and solve separately for each mesh. However, as shown in Figure 6.11 (left), there is a large difference between the resulting fields - e.g in the locations of the singularities. Figure 6.11 (right), shows the result when solving jointly for both shapes. Note that the singularities on the back of the shape are consistent between the models. For evaluation, we transport V_1 to M_2 and measure the angle difference between the resulting vector field and V_2. Figure 6.11 (center) shows the resulting histogram, emphasizing that our joint design method preserves the directions better.

7 Extensions

As mentioned in the previous chapter, one of the particularly appealing properties of the operator representation of vector fields is that it allows to establish an explicit relation between vector fields and the functional representation of their associated flows, which are simply self-maps resulting from vector field advection. We have exploited this relation in several follow-up works, including two mentioned below.

7.1 Continuous Matching via Vector Field Flow

One limitation of the shape matching technique presented in Chapter 1 and in particular the method for converting functional maps to point-to-point maps mentioned in Section 3.6.1 is that the image of each point is computed independently and thus the resulting correspondences are not necessarily continuous. To alleviate this issue, in [COC15] we proposed to use the connection between the flows of vector fields and their functional representation to find an optimal continuous correspondence associated with a given functional map C. Namely, given a functional map C and an arbitrary continuous point-to-point correspondence T_{cont} a functional representation C_{cont}, between a pair of shapes M and N, we find the optimal vector field V such that:

$$V_{\text{opt}} = \arg\min_V \|\Phi_V \circ C_{\text{cont}} - C\| = \arg\min_V \|\exp(V) C_{\text{cont}} - C\|.$$

Here Φ_V is the functional representation of the flow of V, and we compose maps by simple matrix multiplication.

By solving for the optimal vector field V *in the functional domain* and then performing advection in the primal (shape) domain, we are able to find accurate correspondences, that are as continuous as the given maps T_{cont}.

7.2 Covariant Derivatives as Operators

We have also used the idea of representing vector fields as operators when discretizing their derivatives and in particular the Levi-Civita covariant derivative $\nabla_U V$ of a vector field V with respect to another vector field U [AOCBC15]. The covariant derivative is one of the most fundamental concepts in differential geometry, and is closely related to other notions such as parallel transport and yet it has been very challenging to find a satisfactory discretization for it using classical approaches in the discrete setting, such as Discrete Exterior Calculus. In our work [AOCBC15] we proposed a novel discretization of the covariant derivatives of vector fields by using the covariant derivative of functions and furthermore demostrated that by considering the linear operators $\nabla_U(\cdot)$ and $\nabla_{(\cdot)}(V)$ acting on vector fields, for fixed U and V respectively, one can obtain an accurate discretization of parallel transport on triangle meshes, which we have used in applications such as fluid simulation on curved surfaces, among others.

Part III

Shapes and their Differences as Operators

8 Overview

In this part we demonstrate that the operator-based approach presented in the two previous parts, can also be used to develop a novel formulation for the notion of shape differences, aimed at providing detailed information about the location and nature of the differences or distortions between the two shapes being compared. The difference operator, derived from a functional map, that we obtain is much more informative than just a scalar global shape similarity score, rendering it useful in a variety of applications where more refined shape comparisons are necessary. Similarly to the other two parts, the approach presented here is intrinsic and is based on a linear algebraic framework, allowing the use of many common linear algebra tools (e.g, SVD, PCA) for studying a matrix representation of the operator. Remarkably, the formulation allows us not only to localize shape differences on the shapes involved, but also to *compare shape differences* across pairs of shapes, and to analyze the variability in entire shape collections based on the differences between the shapes. We give a number of applications of shape differences, including parameterizing the intrinsic variability in a shape collection, exploring shape collections using local variability at different scales, performing shape analogies, and aligning shape collections.

The material in this part is based on the article:

- "Map-Based Exploration of Intrinsic Shape Differences and Variability," by R. Rustamov, M. O., O. Azencot, M. Ben-Chen, F. Chazal, and L. Guibas. In *Proc. SIGGRAPH*, 2013.

While the extensions mentioned at the end are based on :

- "Functional characterization of intrinsic and extrinsic geometry", by Etienne Corman, J. Solomon, M. Ben-Chen, L. Guibas, and M.O. In Transactions on Graphics (to appear), 2017.

9

Shape Differences

9.1 Introduction and Rationale

Comparing shapes is a fundamental operation in shape analysis and geometry processing, with many applications to computer graphics, including interactive shape design, shape search, and the organization of shape collections. Most approaches to comparing shapes reduce the comparison to a single number, a shape similarity score or distance. These distances can be computed either by establishing correspondences between the shapes (and therefore being able to compare the geometry at a finer scale) or by computing certain global shape descriptors and then estimating a distance in descriptor space.

In many settings, however, we may desire a more detailed understanding of how two shapes differ that goes beyond a single similarity score. Shapes can be complex objects and the very plethora of shape distances that have been proposed is testimony to the fact that no single scalar metric is able to satisfy all applications. For example, we may be interested in *where* two shapes are different and in *how* they are different. Such finer comparisons have long been important in other fields, such as industrial metrology to assess the quality of manufacturing processes, or in computational anatomy, to separate normal organ variability from disease forms for diagnostic purposes. In computer graphics and geometry processing, as shape collections are getting larger and larger with more objects in each category, these finer and more detailed shape comparisons are becoming important – and difficult to handle by coarse traditional techniques.

When computing maps or correspondences between shapes (including shape parametrization) the minimization of measures of shape distortion has long been used as a key optimization criterion. Yet once the map is computed, the distortion information is not stored, analyzed, or compared

to that of other maps. We propose reverse this process. Starting from a map between two shapes, we propose a novel notion of shape differences as seen by this map, one that provides detailed information about how the shapes differ. Thus our work leverages the recent flurry of activity in algorithms for mapping shapes.

The main contribution of the work described in this part is to give a rigorous formulation of the concept of a shape difference under a map and show how such shape differences can be computed, analyzed, and compared – thus making shape differences concrete, tangible objects that can be manipulated just like the shapes themselves can. Our approach is based on the following insight: in classical Riemannian geometry, local distortions induced by a map are expressed in terms of changes in the metric — which essentially is equivalent to tracking the changes in *inner products of tangent vectors* before and after these vectors are transported by the map from the source to the target shape. In contrast, we track the changes in *inner products of real-valued functions* induced by transporting these functions from the source shape to the target shape via a functional map. Our main observation is that all these changes can be captured by certain linear operators (matrices), which we call shape differences; remarkably, a single such operator works simultaneously for all pairs of functions.

Our approach has several key advantages. First, we exploit the functional map formulation, so our notion of a map can be quite general and incorporate mapping ambiguities due to symmetry, slippage, etc. Second, the approach is intrinsic and is not affected by the embedding of the shape in 3D. Third, when we have point-to-point correspondences, our shape difference can be directly related to classical local notions of geometric distortion, such as area or conformal distortion. Additionally, under a few assumptions and allowing for certain equivalences, the original map can be recovered from the shape difference. Fourth, we define shape difference via a linear operator formulation and discretize it into a matrix or vector form, giving us access to a wealth of linear algebra tools.

Our explicit representation of shape differences facilitates a number of challenging shape analysis tasks. For example, given two pairs of shapes, A, B and C, D, shape differences allow us to quantify how much the *change* from A to B is similar to the change from C to D, regardless of how similar A is to C. We can do these kinds of "shape analogies" only because we

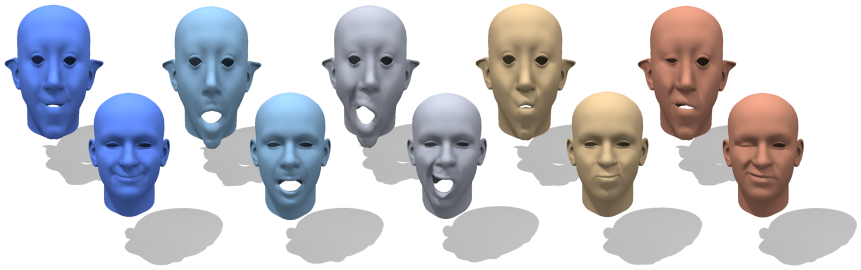


Figure 9.1: Our notion of shape differences provides a way to compare deformations between shape pairs. This allows us to recognize similar expressions of shape A (top row) to those of shape B (bottom row), without correspondences between A and B and without any prior learning process.

can compute the "difference among the differences" of the four shapes. For example, consider the face shapes in Figure 9.1. While the two rows of faces shown differ significantly, the relative changes between the undeformed and deformed version of each are similar, which is captured by our informative descriptors.

One of the key aspects of our shape differences is that they allow both localizing and parameterizing the variability between a single pair or of a collection of shapes. Thus, we can provide not only a canonical descriptor for a difference between a pair of shapes but also use it to analyze and visualize the source of the variability, making the interpretation of results easier and more concrete. In addition, we can now analyze the structure of shape collections based on relating the differences between the shapes and not the shapes themselves. We show several examples of the power of this approach. In particular, unlike almost all existing work, we can look at the variability of related shapes in a collection without necessarily having explicit point-to-point or landmark correspondences between the shapes (though we can use these when we have them).

After discussing related work (Section 9.2), we show how to formally define shape differences (Section 9.3) and compare them in shape collections (Section 9.4). We then discuss the discretization and computational aspects of shape differences (Section 9.5), as well as their key properties (Section 9.6). Finally we proceed to give a number of applications of this notion, including parameterizing the intrinsic variability in a shape collection, exploring shape collections using local variability at different scales, performing shape analogies, and aligning shape collections (Section 9.7).

9.2 Related Work

Shape differences and variability have been of interest in several scientific communities over many decades. One approach, generally termed *Statistical Shape Analysis*, exploits the notion of *Shape Space* introduced by D.G. Kendall [DM98], where a standard set of key points or *landmarks* is selected on each shape and a shape is represented by its vector of landmarks after normalizing for rotation, translation and scale. Multiple shapes are analyzed jointly by first aligning their landmark vectors and then using principal components analysis (PCA) to extract the main modes of shape variation. Such learned shape variability models can also be used in segmenting shapes out of image or volume data (see, e.g. [CT+01]), following the active contour paradigm [KWT88] from computer vision.

The medical research community, and especially brain anatomists, have explored many variations along this general theme, trying to compensate for the fact that exact landmarks may be hard to locate either algorithmically or manually in noisy medical images (2D or 3D). Many other shape features and shape descriptors, local and global, have been tried, including area, volume, spherical harmonics, medial axes or skeletons, etc. (see [GSSL01, GGSK05] among many others) — see also the 2D image analysis survey by F.L. Bookstein [Boo96]. In these works, shape variability is effectively modeled by descriptor variability.

Another important issue is that not all variability carries the same significance. For example, in a population of 3D models of humans, some models may be the same human in different poses. If our goal is to understand the variability of human shapes, we must then factor out the variability due to pose variations among the subjects. Various approaches have been tried towards this end, including PCA in a Riemannian symmetric space [FLPJ04], multivariate tensor-based morphometry using holomorphic forms [WZG+10], tensor ICA [VT07], the use of Laplace coordinates for points [WSX12], and others [NSN+07]. Only recently has an approach been proposed in these communities for comparing shapes intrinsically [LSS+10] using a spectral L^2 distance, but the approach suffers from the usual sign ambiguities (or more generally rotations within an eigenspace) of the eigenfunctions in spectral embeddings.

In the geometry processing area there has been considerable work in

comparing shapes in an indirect way, in the setting of computing good maps between shapes. This is especially true in the context of *non-rigid shape matching* where the goal is to recover the *best* map according to some quality criterion (see e.g. [BBK06, KLF11, SY11] among a myriad of others). Perhaps the most common such criterion for a map between a pair of shapes is preservation of pairwise quantities such as geodesic distances [BBK06, SY11] or spectral quantities such as the heat kernel (e.g. [SH10]). Generally, such measures of quality are both expensive to compute and non-trivial to analyze, making the intuitive understanding of the difference between shapes challenging. Another way of evaluating the distortion of a map, used mostly in shape deformation and parametrization applications (see e.g. [SMW06, BCWG09], among others), is to consider the local affine distortion introduced by the map at every point on the shape, e.g., angular or area distortions. While such local distortion measures are efficient to compute, they can often be too noisy to be used directly for identifying problematic regions. Finally, collections on human shapes were studied in [ACP03, ASK+05, HSS+09]. These papers either explore pose and human shape variability separately, or they use skeleton information to facilitate pose alignment.

Our work is also related to the large volume of research on shape similarity metrics, either map-based or descriptor-based, whose complete survey would be beyond our scope. Shape search using on such metrics has also been intensely studied, but mostly focused on discriminating shapes under large-scale variations (e.g., cars from humans). The current effort is aimed at fine variability, which has received less attention. In a related vein, the problem of how to map shape descriptor variability back onto something semantically meaningful on the original shapes was addressed in [OLGM11]. We also note that the topic of fine classification/categorization has been popular in the computer vision community in the last few years (see, e.g., [FBW11] and the references in the papers therein).

From our point view, all of these approaches suffer from certain drawbacks. First of all, the notion of shape difference is not made explicit — at best only a shape "distance" is defined. With that it is impossible to understand precisely where the variation happens on a shape, as each shape is treated as "atom" — typically, a point in a fixed-dimensional Euclidean space. Furthermore, it is hard to compare differences between shapes —

to express "differences among the shape differences," for the same reason. Second, large amounts of information about the shapes is ignored, and this can affect the results. For example, the connectivity of the landmarks can be just as important as their absolute positions. Third, linear methods such as PCA are most often used — a notable exception being [KMP07] — even when it is not clear that a flat approximation to the shape space, either locally or globally is indicated. Fourth, these works perform *extrinsic* comparisons between the shapes and do not focus on their *intrinsic geometry* which is often what carries the true semantics of the shape. Unfortunately, invariance to isometric deformations is much harder to incorporate than invariance to Euclidean transformations. Finally, unlike earlier works that require vertex-to-vertex or consistent landmark correspondences, our use of the functional framework allows us to compare shapes whose meshes may be entirely different.

9.3 Shape Differences

Similarly to the functional representation of mappings and vector fields described in Chapters 3 and 6, shape differences are linear operators (matrices in the discrete setting) that capture the disparity between shapes M and N under a given map (T or F below) between them. We define two types of shape differences, one based on the area distortion and another based on the conformal distortion, as induced by the map. In this section, we introduce the abstract definition of shape differences, applicable both in the continuous and discrete setting, and then show how they can be computed in practice in Section 9.5.

9.3.1 Background and notation

Our formulation uses the functional maps framework defined in Chapter 3 to represent maps between surfaces. Namely, given two surfaces M and N, a map $T : N \to M$ between them induces a map between functions $F : L^2(M) \to L^2(N)$, where $L^2(\cdot)$ is the set of square integrable real-valued functions on a surface. This functional map T_F takes each function $f : M \to \mathbb{R}$ and maps it to $g : N \to \mathbb{R}$ defined as $g = T_F(f) = f \circ T$. As pointed out in Chapter 3, T_F is a *linear* transformation between function spaces and, therefore, can be represented as a matrix in the discrete setting.

It is crucial to note that functional maps are not limited to point-to-point maps, but provide a general notion of a map that can incorporate mapping ambiguities due to symmetry, slippage, etc. In the formulations below we will directly make use of a linear functional map $T_F : L^2(M) \to L^2(N)$, regardless of whether or not it is associated with a point-to-point map.

9.3.2 Formulation

Our goal in defining the shape differences between two shapes M and N, given a functional map T_F is to quantify some measure of distortion induced by T_F between $L^2(M)$ and $L^2(N)$. We compare shapes by comparing corresponding measurements made on the function spaces of the shapes. Following a Riemannian point of view, a measurement over a shape is defined by an *inner product* of functions on the shape. Recall that such an inner product $h(\cdot,\cdot)$ is a bi-linear form taking pairs of functions into real numbers. In our work, for every surface S, we consider the following two inner products on $L^2(S)$:

Definition 2. *We define the* area-based *inner product as:*

$$h_a^S(f,g) = \int_S f(x)g(x)\, d\mu(x).$$

Definition 3. *We define the* conformal *inner product as* $h_c^S(f,g) = \int_S \nabla f(x) \cdot \nabla g(x)\, d\mu(x)$ *on the space of differentiable functions modulo constants.*

These inner products are called area-based and conformal-based because of the following result (see [ROA+13] for a proof):

Theorem 1. *Given a pair of surfaces M, N and a bijection $T : N \to M$ with the functional representation T_F, the following holds:*

1. $h_a^M(f,g) = h_a^N(T_F(f), T_F(g)), \forall f,g$ *if and only if T is locally area preserving.*
2. $h_c^M(f,g) = h_c^N(T_F(f), T_F(g)), \forall f,g$ *if and only if T is conformal.*

If the underlying map is not locally area preserving or conformal, the stated equalities will not hold. It is natural to quantify the distortions induced by the map through the failure of these equalities, perhaps by assigning a single number measuring the discrepancy. Of course the precise notion of discrepancy will depend on the functions f and g chosen. The

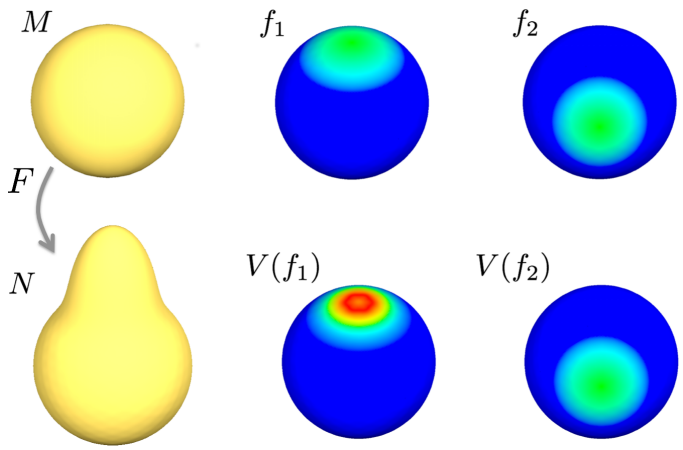


Figure 9.2: Given a pair of shapes M and N (left column) and a functional map F, the shape difference V is a linear operator, which for every function f on M produces another function $V(f)$ on M which intuitively encodes how much f is distorted by F (center and right). Note that f_1 is supported in an area that deforms under the map, and f_2 in an area that does not.

challenge is to encode all these numbers arising out of different f and g into a single richer notion.

Our main observation is that all these discrepancies can be captured by certain matrices (linear operators), where these matrices are not simply tables of numbers, but can be meaningfully manipulated and compared as matrices. These operators effectively compensate for the distortions caused by the map T_F to the measurement in question and allow us to "pull back" the measurement $h_a^N(T_F(f), T_F(g))$ on N to a measurement made on M. Having all measurements on a common space is advantageous, as we want to be able to compare and compute differences between many measurements. Technically, we can accomplish this measurement transportation by using the following consequence of the classical Riesz representation theorem from functional analysis (see the [ROA+13] for proof):

Theorem 2. *Given two shapes M, N, endowed with inner products h^M and h^N respectively, and a general functional map $F : L^2(M) \to L^2(N)$[1], there exists a unique linear operator $D_{h^M,h^N} : L^2(M) \to L^2(M)$ satisfying:*

$$h^M(f, D_{h^M,h^N}(g)) = h^N(F(f), F(g)) \; \forall f, g.$$

We will refer to the operator D_{h^M,h^N} as the difference between h^M and h^N.

[1]We denote the functional map by F rather than T_F to emphasize the fact that it might not come from a pointwise map T

The linear operator D modifies g so as to exactly compensate for the distortions introduced by the map F. It is remarkable that D is a "universal compensator" — a single such operator works simultaneously for all functions f and g. Stated differently, D depends only on the given inner products on M and N, and the functional map F. It is also important to note that D is a linear self-map of the space of functions over M, (see Figure 9.2 for an illustration).

Now, we can apply this theorem to inner products h_a^M and h_a^N (resp. h_c^M and h_c^N) to formally quantify the difference between them. Namely, we define the *area-based shape difference* as:

$$\boxed{V_{M,N,F} = D_{h_a^M, h_a^N}} \tag{9.1}$$

and *conformal-base shape difference as*:

$$\boxed{R_{M,N,F} = D_{h_c^M, h_c^N}} \tag{9.2}$$

Since the map is usually clear from the context, we will often use the abbreviated notation $V_{M,N}$ and $R_{M,N}$.

We stress here that both $V_{M,N}$ and $R_{M,N}$ are not numbers but operators. They yield numbers, once functions specifying the measurement of interest are given. This flexibility enables a rich set of applications as we will show in Section 9.7. Note also that two shape differences V_{M,N_1} and V_{M,N_2}, even if $N_1 \neq N_2$, both represent linear operators with the same domain ($L^2(M)$) and range ($L^2(M)$). This allows us to *compare* shape differences even when they are defined using maps to different shapes (see Section 9.4), as hinted above.

Matrix representation: After a choice of a basis, the linear operators defined above can be made more tangible by expressing them in terms of matrices. Indeed, when dealing with discrete shapes, the underlying function spaces are finite-dimensional vector spaces. Any inner product $h^M(\cdot,\cdot)$ can always be represented via a positive definite matrix H^M such that: $h^M(f,g) = f^\top H^M g$, where f and g are column vectors. Similarly, given the shape N with an inner product h^N, and a functional map F, we can represent $h^N(Ff, Fg) = f^\top F^\top H^N F g$, for some matrix H^N.

When these expressions for discrete inner products are plugged into Theorem 2, we can obtain an explicit expression for the difference operator D between these inner products:

$$D = (H^M)^{-1} F^\top H^N F. \tag{9.3}$$

Explicit formulas for both types of shape differences under various basis choices are provided in Section 9.5.

Discussion: These particular shape difference formulations were chosen for a number of reasons. First, isometric and conformal maps play an important role in shape processing, and so it is important to capture these exact notions. While conformal maps are directly characterized by our framework via the requirement $R_{M,N} = I$, where I is the identity map, note that isometric maps are both area-preserving and conformal and so can be characterized by the equalities $V_{M,N} = I$ and $R_{M,N} = I$. These and several more desirable properties of our shape differences will be discussed in Section 9.6.

Second, inclusion of the map $F : L^2(M) \to L^2(N)$ into our formulation allows the notion of the shape difference to change and depend on the context of a particular application. For example, while a purely geometric notion of shape difference can be obtained by taking the difference induced by some type of geometrically optimal map, such a shape difference may not be optimal for studying, say, differences in human brain shapes. In this latter case, our framework allows use of the maps provided by a specialist to compute a domain-specific shape difference. This is unlike, for example, various notions of Gromov-Hausdorff distances [BBK06], which are often defined with respect to some "optimal" map.

Finally, when working with shape collections, it is crucial to be able to compare the shape difference between a pair of shapes to the shape difference between another pair of shapes. We discuss this in the next section, and show that our shape difference matrices can be rigorously compared to each other in a variety of circumstances without falling into the fallacy of "comparing apples to oranges." Key to our approach is the ability to transport or move a shape difference between two shapes to a third reference shape via connecting maps, so as to make meaningful comparisons possible.

9.4 Differences in Shape Collections

One of the main advantages of the shape differences defined above is that they not only encode detailed knowledge about the distortion under a given map, but also allow distortion comparisons across pairs of shapes, defining

"differences between differences." Here, we outline how such comparisons can be carried out in three different scenarios. The discussion below is valid for both kinds of shape differences; to avoid repetition we will focus on the area-based shape differences.

The first scenario arises when for shapes M, N_1, and N_2, one wants to compare the shape difference V_{M,N_1} to V_{M,N_2}. As mentioned earlier, both of these shape differences are linear operators with the same domain and range $L^2(M)$, and thus can be directly compared and even algebraically combined if needed.

The second scenario arises when one wants to compare the shape difference V_{M_1,N_1} to V_{M_2,N_2}, where $M_1 \neq M_2$, assuming that a linear functional map G between $L^2(M_1)$ and $L^2(M_2)$ is *known*. Note that V_{M_1,N_1} and V_{M_2,N_2}, cannot be directly compared because they are defined over different domains and ranges ($L^2(M_1)$ and $L^2(M_2)$). In order to make the comparison possible, we need to first apply a change of basis transformation to one of the matrices — this is where the cross map $G : L^2(M_1) \to L^2(M_2)$ enters the picture, to allow the transportation of the difference. By applying a matrix conjugation by G to V_{M_2,N_2} we bring it into a common basis with the other matrix, and now the matrices V_{M_1,N_1} and $G^{-1}V_{M_2,N_2}G$ can be compared and algebraically combined as needed. To avoid computing the inverse of matrix G, one can also compare matrix products GV_{M_1,N_1} and $V_{M_2,N_2}G$.

The last, third, scenario arises when one wants to compare the shape difference V_{M_1,N_1} to V_{M_2,N_2}, but a mapping between M_1 and M_2 is *not known*. If we knew the map G, we would have compared after conjugating one of the matrices by G. However, since now we do not know the map, our comparison needs to rely on the quantities that are *invariant* under matrix conjugation. It is well known that the spectrum of a matrix is such an invariant, and therefore, for comparing the shape differences we can compare the spectra of the matrices V_{M_1,N_1} and V_{M_2,N_2}.

In the third scenario, it is crucial that the shape difference matrices are represented in terms of a truncated basis (e.g. low-frequency Laplace-Beltrami eigenfunctions) spanning a subspace of smooth functions. In essence, this adds a regularization on the unknown cross map G, forcing it to be smooth. This is a benefit of the functional map representation, as described in Part I above.

9.5 Computation

In this section we present explicit formulas for computing shape differences between two triangle mesh surfaces M and N. As it is clear from formula (Eq. 9.3), to compute the shape differences we need to have access to three matrices: the inner product matrices H^M, H^N and the functional map F. Since these matrices depend on the choice of a basis for the functional spaces $L^2(M)$ and $L^2(N)$, we will consider three options.

Before proceeding, let us fix our discretizations. For a surface mesh S, we discretize the area-based inner product by $h_a^S(f,g) = \sum_{x\in S} f(x)g(x)A^S(x)$ where $A^S(x)$ is the area element (Voronoi area) associated with vertex x. The Laplace-Beltrami operator is discretized as $L = (A^S)^{-1}W^S$, where A^S is the diagonal matrix of area weights and W^S is the stiffness matrix (e.g. the standard cotangent weight matrix) [PP93].

Option 1: Here we use the finite element "hat function" basis for both $L^2(M)$ and $L^2(N)$. In the indicator, hat function basis, the matrix associated with the area-based inner products h_a^M and h_a^N are simply the diagonal matrices of area weights at vertices, A^M and A^N. Using formula (Eq. 9.3) the matrix associated with the area-based shape difference is given by $V_{M,N} = (A^M)^{-1}F^\top A^N F$.

To derive the conformal-based shape difference, we use Stokes' theorem $\int \nabla f(x)\cdot\nabla g(x)\,d\mu(x) = -\int f(x)\Delta g(x)\,d\mu(x)$, where Δ is the Laplace-Beltrami operator. This is valid in the discrete case (even if there is a boundary) due to our choice of discretization. In the discrete case, this means that $h_c^M(f,g) = -f^\top A^M(A^M)^{-1}W^M g$, and thus, the matrix associated with the conformal-based shape difference h_c^M is given simply by $-W$, the stiffness matrix. This implies that the conformal-based shape difference under the functional map F is given by[2]: $R_{M,N} = (W^M)^{-1}F^\top W^N F$.

In a special case when the surfaces M and N have identical tessellations conforming to the map T, the functional map F is simply the identity matrix. Therefore, we obtain the following formulas:

$$V_{M,N} = (A^M)^{-1}A^N \quad \text{and} \quad R_{M,N} = (W^M)^{-1}W^N\,.$$

[2]The conformal shape difference operator is defined on $L^2(M)$ modulo constants, and extended to the entire $L^2(M)$ by setting it to zero for constants. In the discrete setting, the same effect is achieved by using pseudo-inverses; all inverses appearing in formulas for R are pseudo-inverses.

These formulas shed light into the nature of our shape differences. For example, $R_{M,N}$ is seen to capture the change of the conformal cotangent Laplacian (without the area weights) and, as a result, of the angles of the mesh.

Option 2: Here we use the orthonormal Laplace-Beltrami bases for both $L^2(M)$ and $L^2(N)$. First note that the matrix associated with area-based inner product is simply the identity matrix because the Laplace-Beltrami basis is orthonormal. The matrix associated with the conformal-based inner product on M is the diagonal matrix $D_M = \text{diag}(-\{\lambda_i^M\})$, where λ_i^M is the i^{th} eigenvalue of the Laplacian of M; similarly for the conformal inner product on N. Therefore, given a functional map F:

$$V_{M,N} = F^\top F, \quad \text{and} \quad R_{M,N} = (D^M)^{-1} F^\top D^N F. \tag{9.4}$$

Option 3: Here we use the orthonormal Laplace-Beltrami basis for $L^2(M)$, and hat basis for $L^2(N)$. The resulting formulas can be derived by essentially combining the derivations for the two options above to yield:

$$V_{M,N} = F^\top A^N F, \quad \text{and} \quad R_{M,N} = D_M^{-1} F^\top W^N F.$$

Discussion: The first option is presented here only for theoretical reasons, to show that simple and intuitive expressions exist in cases when the shapes are identically tessellated. However, this option is not practically useful: a) the obtained shape difference matrices are sensitive to noise both in the meshes and the maps, b) their sizes scale with the number of mesh vertices, and c) the computation requires the pseudo-inverse of a large matrix.

The second option has the advantages of being generally applicable and allows a smoothed approximation (by using a small (50-100) number of low-frequency eigenfunctions) when dealing with imperfect meshes and/or correspondences. The third option allows a smoothed approximation and speeds up computations in shape collections where all of the meshes are identically tessellated. Indeed, when computing shape differences from one shape to all others in such a collection, the eigenfunction basis is needed on the source mesh only. Due to basis truncation, both of the latter two options result in small shape difference matrices, making joint analysis (e.g. PCA in Section 9.7.1) feasible.

9.6 Properties

In this section we discuss a number of properties of shape differences: functoriality, informativeness, and localization and relation to point-wise measures of distortion.

Functoriality: The shape differences behave functorially under map inversions and compositions. To simplify the exposition let us assume that one uses the Laplacian basis on all shapes, and so the formulas (Eq. 9.4) apply. While we focus on the area based shape differences, similar arguments are valid for the conformal ones.

We start with map inversion: given the maps $F : L^2(M) \to L^2(N)$ and $F^{-1} : L^2(N) \to L^2(M)$, we want the induced shape differences satisfy $V_{N,M} = V_{M,N}^{-1}$. This cannot directly hold because the involved matrices are expressed in terms of different bases. We compute $V_{M,N} = F^\top F$ and $V_{N,M} = (F^{-1})^\top F^{-1}$; next, we need to apply matrix conjugation to the matrix $V_{N,M}$ to transport it into the same basis as $V_{M,N}$; this results in $F^{-1}V_{N,M}F = F^{-1}(F^{-1})^\top F^{-1}F = F^{-1}(F^{-1})^\top$, which indeed is the inverse of $V_{M,N}$. Thus, after transporting to a common basis, $V_{N,M} = V_{M,N}^{-1}$ holds.

As for map compositions, given the maps $F_1 : L^2(M) \to L^2(N)$, $F_2 : L^2(N) \to L^2(K)$ and the composition $F_2F_1 : L^2(M) \to L^2(K)$, we want $V_{M,K} = V_{M,N}V_{N,K}$ to hold. We compute $V_{M,N} = F_1^\top F_1$, $V_{N,K} = F_2^\top F_2$, and $V_{M,K} = (F_2F_1)^\top F_2F_1$. Note that only the matrix $V_{N,K}$ needs to be transported to the same basis as the other matrices; this requires conjugation by F_1, and gives the matrix $F_1^{-1}V_{N,K}F_1 = F_1^{-1}F_2^\top F_2F_1$. Now, the sought equality can be easily seen to hold.

The latter property can be used to speed up computations as follows. Suppose that we have a collection of shapes with functional maps $F_i : L^2(M) \to L^2(N_i)$, and we compute all of the shape differences V_{M,N_i}. Now, the shape differences between any pair of shapes N_i and N_j, after transporting to M, is given by $V_{N_i,N_j} = V_{M,N_i}^{-1}V_{M,N_j}$.

Informativeness: We have seen in Section 9.3.2, that $V_{M,N} = I$ and $R_{M,N} = I$, if and only if the underlying maps are area preserving and conformal respectively. Combining this with functoriality properties we have: if $V_{M,N,F} = V_{M,N,G}$ then the map $F^{-1}G$ is area preserving (resp. conformal for R). This means, in particular, that the shape difference matrices *encode*

the map up to an area preserving, or conformal self-map. In other words, the shape difference matrices that we define *are fully informative* up to the given notion of distortion.

Localization and relation to existing measures: When the functional map $F : L^2(M) \to L^2(N)$ is associated with a point-to-point bijection $T : N \to M$, we can extract local distortion information from the shape difference operators. Let ρ be a compactly supported function on $\Omega \subset M$ (i.e. $\rho(x) = 0$ when $x \notin \Omega$), then $V_{M,N}\rho$ and $R_{M,N}\rho$ only depend on the restricted map $T|_{T^{-1}(\Omega)} : T^{-1}(\Omega) \to \Omega$. In other words, if the map T is modified outside the region Ω, then $V_{M,N}\rho$ and $R_{M,N}\rho$ would not change.

To prove this for, say, the conformal shape differences, note that for any $f : M \to \mathbb{R}$, the operator $R_{M,N}$ satisfies $\int_M (\nabla f)(\nabla R_{M,N}\rho)\, d\mu^M = \int_N \nabla(f \circ T)\nabla(\rho \circ T)\, d\mu^N = \int_{T^{-1}(\Omega)} \nabla(f \circ T)\nabla(\rho \circ T)\, d\mu^N$, where the latter equality follows from ρ being supported in Ω, and by bijectivity of T, $\rho \circ T$ being supported within $T^{-1}(\Omega)$. The last expression involves T only in an integral over the region $T^{-1}(\Omega)$, which proves the claim.

This property means that by selecting a function ρ supported within some ROI, we can use $V_{M,N}\rho$ and $R_{M,N}\rho$ as descriptors of distortion happening along this region. As a result, we can make localized comparisons between different maps such as $T_1 : N_1 \to M$ and $T_2 : N_2 \to M$; see Section 9.7.2 for an application.

Additionally, the area based shape differences enjoy the following two properties (both proved in [ROA+13]). Given a region $\Omega \subset M$, and a function $\rho : M \to \mathbb{R}$ supported within this region, the support of $V_{M,N}\rho$ lies inside Ω . Finally, by letting χ be the indicator function of Ω (i.e. $\chi(x) = 1$ if $x \in \Omega$ and 0 otherwise), we can extract from $V_{M,N}$ the traditional measure of area distortion using the following formula: $\text{area}(\Omega) \,/\text{area}(T^{-1}(\Omega)) = h_a^M(\chi, \chi) \,/ h_a^M(V_{M,N}\chi, \chi)$.

9.7 Applications

The shape differences provide a general framework with many potential applications in computer graphics, computer vision, medical imaging, structural biology, and a number of other fields that require precise comparisons between shapes. Here we explore a number of prototype applications involv-

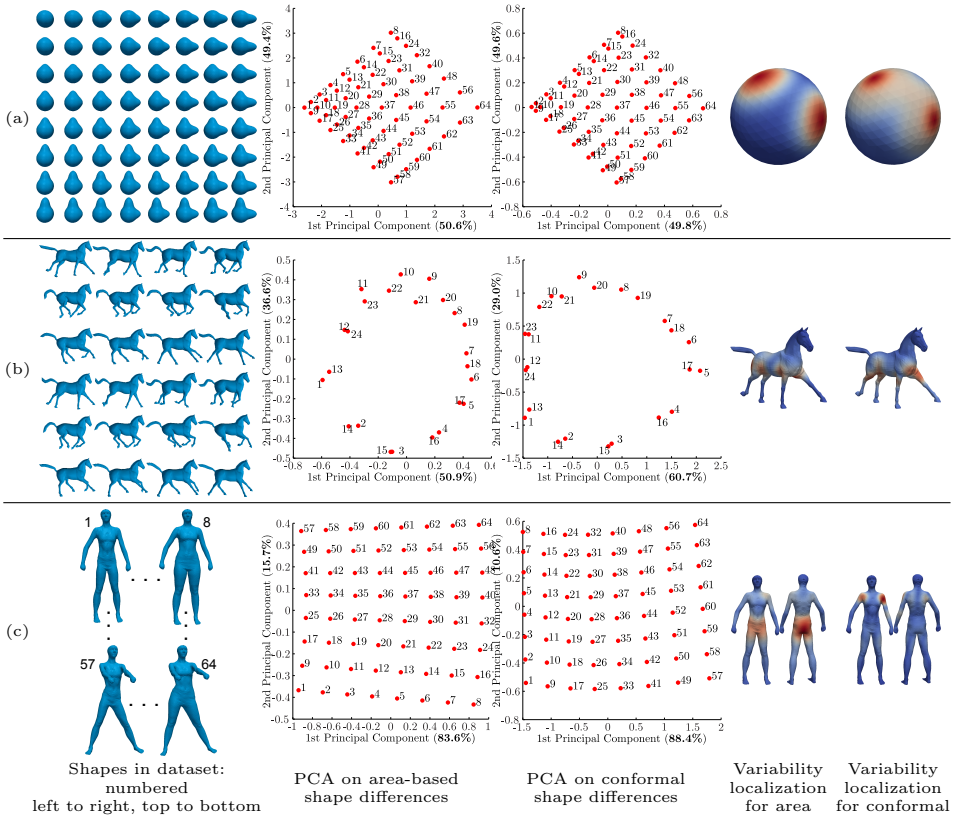


Figure 9.3: The proposed shape differences reveal the major variability in a collection of shapes as well as the locations (sources) of variability, useful for visualization. Each row represents a different shape collection (left), followed by PCA performed on area-based and conformal shape differences (middle) and visualization of the locations of variability color coded from red for high to blue for low variability (right).

ing collections of 3D models.

9.7.1 Intrinsic Shape Space

Shape differences can be used to explore variability in a collection of related shapes. For this purpose it is important to obtain a common representation that captures the landscape where the models live, to determine the "average" shape, and to visualize where variability happens directly on the shapes. We choose to represent a collection of shapes as a collection of shape differences from one of them, what we term the "base" shape. Since shape differences can be transported to different shapes, the choice of the base shape is not at all critical — it is rather like choosing an arbitrary

origin when introducing a coordinate system. We will demonstrate that, by applying Principal Component Analysis (PCA) to these shape differences, we can extract the types of information outlined above.

First, we vectorize the area based and conformal shape difference matrices, apply PCA, and depict each shape's coefficients along the two largest principal component directions. As usual, the vectors determining the PCA directions are normalized to have unit vector norm. Since we are working with matrices, this is equivalent to the unit Frobenius norm of matrices. Due to this normalization, the xy-coordinate ranges in the PCA plot are commensurable both within and across plots.

Second, we obtain the visualization on the base shape of where shape variability localizes. To this end, we convert the principal components into matrices $\{P_i\}$; the amounts of variances explained are $\{\sigma_i^2\}$. For a given function f on the base shape, let $\vec{f}$ be its vector representation in the basis. After normalizing $\|\vec{f}\| = 1$, it is true that $\|P_i\vec{f}\|$ is small for all $i = 1, 2, ...,$ then the distortion that this function undergoes is similar between all the shapes. Note that we are not discussing the average amount of distortion, but rather the deviation of the distortion from the average. Now, we can define an aggregate amount of this deviation over all the principal components as

$$\sum_i \sigma_i^2 \|P_i\vec{f}\|^2 = \sum_i \sigma_i^2 \vec{f}^\top P_i^\top P_i \vec{f} = \vec{f}^\top M \vec{f},$$

where $M = \sum_i \sigma_i^2 P_i^\top P_i$; here the weighting by variances allows giving more importance to more prominent principal directions. To visualize what regions vary most between different shapes (again not the regions of highest distortion, but of highest variability), for every point p on the base shape, we compute the variance function as $var(p) = (\vec{f}^\top M \vec{f})^{1/2}/\|\vec{f}\|$ where $\vec{f}$ represents the delta function centered at p. In practice we replace the delta function by the heat kernel computed at a small value of the time parameter.

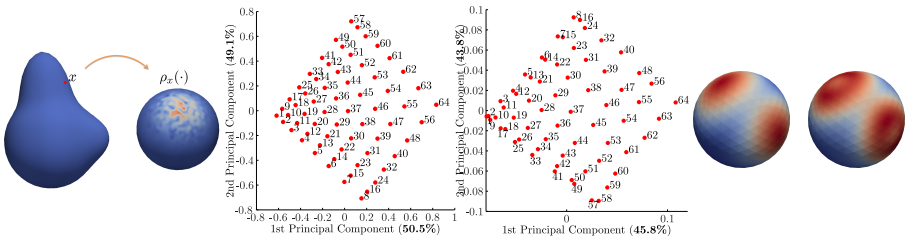
Our first dataset (first column of Figure 9.3(a)) is comprised of 64 deformations of the unit sphere obtained by adding two protrusions to the unit sphere using normal displacement. The sizes of protrusions sample an equally spaced two-dimensional grid of values.

The PCA plots for area based and conformal shape differences are shown respectively in the second and third columns of Figure 9.3(a). These plots uncover the grid structure of the underlying shape space. The fact that the

percent of variance explained by each PCA direction (shown in parenthesis for each axis) nearly add up to 100%, recovers the fact that deformations have two degrees of freedom. The percent of variance explained for both PCA directions are almost the same, meaning that deformations in both of the bumps have similar strength range. Since PCA centers data around the average, we can find the average shape for this collection by looking around the origin, which gives shapes 28, 29, 36, and 37 as the average shapes; these are exactly the four shapes in the center of the image depicting the collection (first column of Figure 9.3(a)). The fourth column of Figure 9.3(a) is the visualization of variability amount (blue is small, red is large) on the base shape for both area and conformal distortions, which are both correctly identified.

Figure 9.3(b) depicts the results on the galloping horse sequence that was used as the frames in a video produced by Sumner and Popović [SP04]. Both of the PCA plots reveal the expected circular "topology" of this dataset, and that this circle is traversed twice. The xy-coordinate range for conformal plot is larger than that of area based PCA plot. This hints that within this collection there is more conformal than area distortion, which is likely due to different parts of body moving relative to each other, inducing higher conformal distortion than area distortion at the joints. Finally, the visualizations of variability in the last column identify the regions of variability correctly.

Figure 9.3(c) depicts the results on a collection of humans synthetically generated using the tools from [HSS+09] as a black-box. This collection



Fuzzy map: each point on one surface is taken to a probability distribution on the other

PCA on area-based shape differences

PCA on conformal shape differences

Variability localization for area

Variability localization for conformal

Figure 9.4: Given fuzzy maps between deformed spheres of Figure 9.3(a), the shape differences correctly identify the major variability as well as the locations of variability.

involves a combination of pose change and body shape variation. Namely, we sample a grid of shapes with two modes of variation: 1) hips get larger and 2) the person raises arms. Both of the PCA plots recover the two dimensional nature of the underlying shape space. From the placement of mesh id numbers in these plots we can see that the main PCA directions for area based and conformal shape differences correspond to different modes of variation: area PCA detects hips getting larger as the main PCA direction, whereas the conformal main PCA direction corresponds to the moving arms. This can also be seen by looking at the visualizations of variability in the last column. Namely, we see that the area variability is maximal around the hips, but the conformal variability is concentrated around the shoulders.

Since functional maps provide a more general notion than point-to-point maps, our framework is applicable in such more general settings. Here, we show an example of applying shape differences in the setting where only fuzzy correspondences between models in the collection are available. A fuzzy correspondence between two surfaces maps every point on the source surface to a probability distribution on the target. The corresponding functional map goes in the opposite direction, and is obtained by convolving a function on the target surface with the probability distributions. We generated a set of such fuzzy maps between the deformed spheres of Figure 9.3(a) by centering a Gaussian distribution at the corresponding point and adding noise. The first image in Figure 9.4 shows an example probability distribution for a single point. Despite the noise, our shape difference framework can correctly identify both the major variabilities present in the collection and their locations.

9.7.2 Exploring Shape Collections

Another step in understanding and using shape collections is being able to track variation of shapes on a finer level, similarly to [OLGM11, KLM+12]. In this subsection we adopt the shape exploration approach of Kim et al. [KLM+12]: a user paints regions of interest on the shape, and the collection is sorted according to the shape similarity within the user specified ROIs.

Our approach is based on the localization property of our shape differences. We pick one of the shapes in the collection as the base shape M. For a given ROI on any of the shapes, let $\vec{\rho}$ be its smoothed characteristic function expressed in terms of the function basis on the base shape. For each

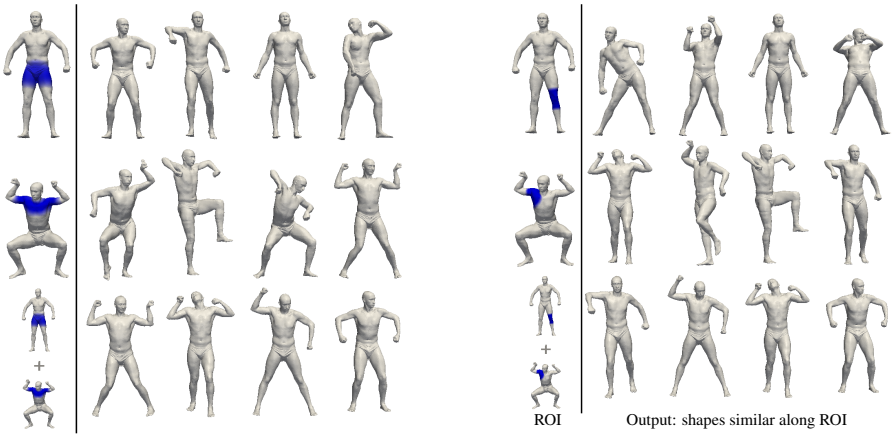


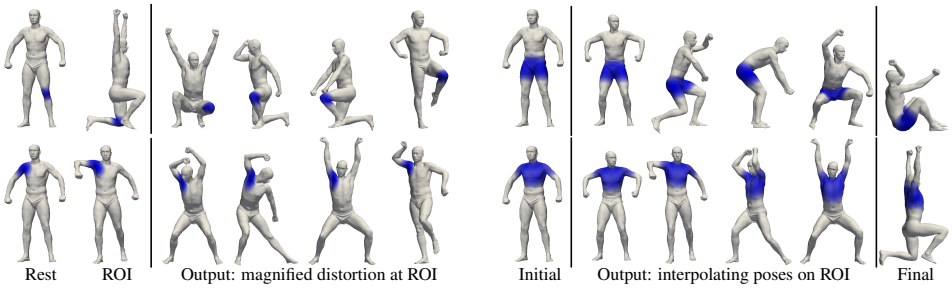
Figure 9.5: Faceted browsing similar to Kim et al. [KLM+12].

shape N_i in the collection, the vectors given by $V_{M,N_i}\vec{\rho}$ and $R_{M,N_i}\vec{\rho}$ carry information about the shape variability within this ROI. We concatenate these vectors into one, and use it to interactively sort the shape collection by similarity/dissimilarity along the ROI, and for operations such as deformation magnification and interpolation. These experiments were run on the SCAPE dataset [ASK+05] which contains 71 poses of the same subject.

Figure 9.5 shows two examples (separated by whitespace) of faceted exploration of human pose. In each of these examples, the first two rows show the selected ROIs and shapes that are most similar to the given shapes along these ROIs. In the third row, similarity along both of the ROIs is sought.

In addition to faceted browsing above, our approach allows the introduction of new exploration capabilities. First, the user may want to see shapes that undergo several times the magnitude of deformation in N_1 along the ROI relative to the base shape M. To this end, we multiply the localized shape differences $V_{M,N_1}\vec{\rho}$ and $R_{M,N_1}\vec{\rho}$ by the user specified amount, and sort the shapes in the collection according to the proximity to these magnified differences. Figure 9.6a shows two examples of this capability. In the first example, the ROI containing the knee is painted on a shape with a bent knee. Magnifying this difference with respect to the base pose, means having the knee bent even more. The shapes retrieved by our method indeed have the most severely bent knees in the dataset.

Another novel exploration capability is shape interpolation along an ROI. Given an initial (N_1) and final (N_2) shape, together with an ROI, we com-



(a) Magnification (b) Interpolation

Figure 9.6: Magnification: given a rest pose (leftmost), and an input shape (second) with an ROI, we find shapes that have 3-4 times the distortion at ROI. Interpolation: given an initial pose (leftmost), and a final pose (rightmost) with an ROI, we find shapes that interpolate between the initial and final poses along the ROI.

pute the vectors $V_{M,N_1}\vec{\rho}$ and $R_{M,N_1}\vec{\rho}$ for the initial shape, and similarly $V_{M,N_2}\vec{\rho}$ and $R_{M,N_2}\vec{\rho}$ for the final shape. Next, we produce equally spaced sample vectors between these initial and final vectors. The shapes having closest vectors to these sample vectors are retrieved. Figure 9.6b shows two examples of this operation.

As a qualitative comparison to the approach of [KLM+12], note that Kim et al. ROI exploration is based on rigidly aligning the ROIs of shapes during the search time, whereas our approach uses intrinsic quantities and is based on directly comparing vectors $V_{M,N_i}\vec{\rho}$ and $R_{M,N_i}\vec{\rho}$. Additionally, simply scaling/linearly interpolating these vectors leads to exaggeration and interpolation capabilities, which are not as straightforward to formulate extrinsically.

9.7.3 Shape Analogies

One of the higher level cognitive operations central to human thought process is constructing analogies. While very difficult to imitate, this operation has received some attention in the image processing context and has led to the concept of image analogies [HJO+01]. In this subsection we show that our shape differences can be used to introduce the notion of *shape analogies*.

Given a pair of 3D models A and B, and another model C, our goal is to retrieve from the collection a shape D such that D relates to C in the same way as B relates to A. We describe an approach to this problem where "in

the same way" is interpreted as having the same or close shape differences. Two sets of experiments will be presented, one when a map between A and C is available, and another when such a map is not available.

We first explain how shape analogies can be obtained when a map between A and C is available. Let be G the corresponding functional map: $G : L^2(A) \to L^2(C)$. We start by computing the area $V_{A,B}$ and conformal $R_{A,B}$ shape differences between A and B. Next, for every shape X in our collection, we compute the corresponding shape differences $V_{C,X}$, $R_{C,X}$ between C and X. Among all X, we select D as

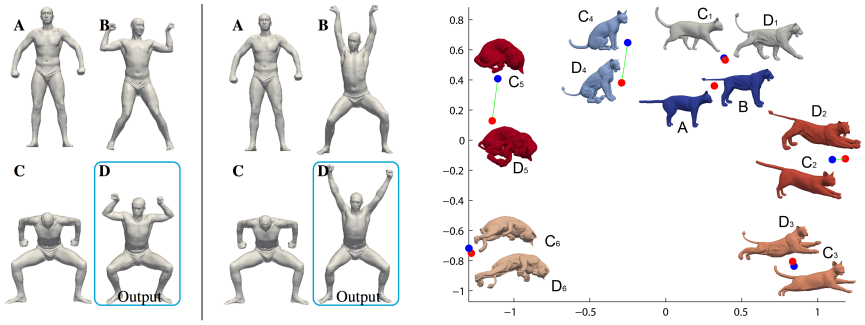
$$D = \arg\min_X \|V_{C,X}G - GV_{A,B}\|_F^2 + \|R_{C,X}G - GR_{A,B}\|_F^2,$$

here we use the Frobenius norm. Note that to compare shape differences we needed to carry out matrix conjugation for transporting the differences to a common comparison ground, which in this case can be achieved without resorting to inverses (see Section 9.4).

Figure 9.7a depicts two examples of analogies constructed using this approach on SCAPE dataset. Here, the map between A and C is known as all of the shapes in the dataset are tessellated in the same way. In the first example (left), as one goes from A to B, the hands get half raised. Therefore, we expect D to differ from C by hand being half raised as well; and indeed our approach retrieves such a pose from SCAPE. The second example involves raising the hands fully, and again our approach succeeds in finding such a pose from SCAPE. Note that the variety of retrieved poses are limited by the dataset being employed.

Figure 9.7b shows a similar experiment but involving multiple analogies based on the Cats and Lions dataset from [SP04]. Here the map between the base cat and base lion is known, the cats are in correspondence and so are the lions. Shape A is the base pose for the cat, and shape B is the base pose for the lion. Then, for multiple shapes C_i (poses of the cat), we find the analogies D_i (poses of the lion). In this case, we embed the V matrices after conjugation in a lower dimensional space using PCA, and compute the distances in this space. Note, that we have recovered all the correct matches between poses.

Our final set of experiments considers the case where a cross collection map is not available. In this case, we cannot use the conjugation method in order to bring the shape differences to the same common ground, and therefore we need to use a descriptor of the difference which is invariant



(a) (b)

Figure 9.7: (a) Shape analogies in SCAPE: given shapes A, B, and C, we find a shape D such that the shape difference D to C is close to the shape difference B to A. (b) Multiple shape analogies between the cats and the lions. We find all the analogies — recovering the relationship between the poses of the cat and the lion.

to matrix conjugation. We chose the singular values of the V matrix as such a descriptor. Note, that now we have considerably less information than when a cross map is available, and therefore we need to regularize our experiment by exploiting more of the data. Here we assume that we have to "parallel" shape collections with corresponding shape variants. Instead of simply fixing A, B, C and searching for the best D such that "D is to C like B is to A", we find the best *permutation* of the shapes which simultaneously best aligns all the shapes in one collection with their counterparts in the other collection. Namely, given n shapes, we compute the singular values of all the possible pairwise maps, and use that as a descriptor. We compare this descriptor to the singular values of all the possible pairwise maps in the other collection. To be more precise, for every one of the possible $n!$ permutations we can compute a score measuring the agreement between the descriptors. This provides a ranking of the permutations, which allows us to find all the analogies simultaneously.

Figure 9.8 shows this experiment for a subset of the shapes from the TOSCA dataset [BBK08]. There, poses of the cat and the dog are not marked as corresponding, and in fact there are various changes in the pose between the cat and the dog — for example the tails are geometrically quite different. As in the previous figure, we show the first collection, followed by the color coded best first and second permutations. We can see that

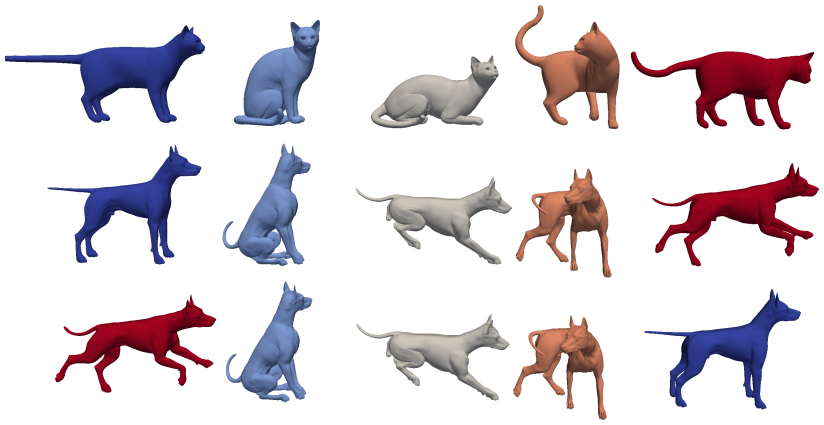


Figure 9.8: Simultaneous analogies between all pairs in two collections (five cats and five dogs from the TOSCA dataset), without a cross map. Note that our best permutation (2nd row) recovers the ground truth, and the second best permutation (3rd row) swaps between two dogs in similar poses.

the best permutation matched correctly between the poses, and the second best permutation confused between similar shapes. This demonstrates that the important information we recover is the *relative* change in pose between the cats within themselves and the dogs within themselves, and there is no requirement for geometric similarity between a cat and a dog.

Finally, note that Figure 9.1 shows another example of such space shape alignment where a cross collection map is not available.

10 Extensions

As mentioned in the previous chapter, one of the appealing properties of shape difference operators is that they provide a convenient way to encode various kinds of distortion induced by a functional map between a pair or a collection of shapes. Moreover, it is possible to show that in the case of smooth surfaces, the shape difference operators fully encode the distortion. I.e., a map is an intrinsic isometry if and only if both the area-based and conformal shape difference operators are identity.

One question that can arrise is to see whether given a shape difference operator it is possible to synthesize or deform a shape so that the induced shape difference is close to a given one. A related question is to see whether the difference operators fully encode the distortion in the discrete setting of triangle meshes.

In [CSBC+17] we proposed an extension of the material presented in this chapter that demonstrates that under mild genericity conditions shape difference operators enjoy the same informativeness properties in the discrete setting as they do in the continuous one, and moreover, in the presence of full information (i.e., without basis reduction), it is possible to intrinsic metric structure (that is, the edge lengths of the triangle mesh) from the shape difference operators by solving two linear systems of equations. We also showed that it is possible to encode extrinsic, or embedding-dependent distortion of the second-fundamental form induced by a map, by considering shape difference operators associated with maps between surface offsets. Together with the statements of completeness mentioned above, this implies that given a base shape of fixed topology, generically any other shape can be encoded by four shape difference operators that describe the intrinsic and extrinsic distortion with respect to this shape. In other words, this implies that shapes themselves can be fully encoded as linear functional operators. Interestingly, these statements hold nearly identically in the con-

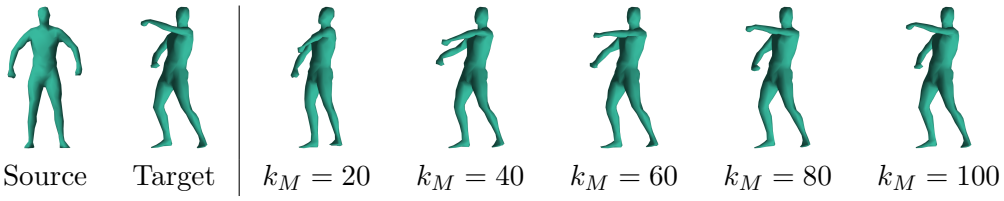


Figure 10.1: Example of mesh recovery from a source mesh with about one thousand vertices and varying-sized shape differences. As the size of the operators increases more details are added to the reconstructed deformation. At $k_M = 100$ and above we achieve a high-quality reconstruction.

tinuous (smooth surfaces) as well as the discrete (triangle mesh) setting. Thus, the linear operator-based representation provides a common language that can be used to describe a coordinate-free representation of surfaces.

In [CSBC+17] we proposed an robust optimization method to reconstruct a shape given a base mesh and the difference operators by recovering the lengths of the edges of the triangle mesh and its offset surface, which works even when the map is represented in a reduced basis. Figure 10.1 shows an example of such reconstruction. At the same time, it would be interesting to see whether performing standard geometry processing operations, such as shape deformation and analysis can be done directly by working with the operator representation, without relying on the potentially resolution and tessellation dependent triangle mesh representation.

11

Conclusions and Future work

In this document we have presented a number of techniques aimed to address various problems in geometry processing, including shape matching, vector field design and manipulation and shape exploration. One of the key observations that is common to all of the methods presented here is that in many cases it is beneficial to represent different geometric concepts as linear operators acting on real-valued functions defined on the shapes. This includes the composition operator (functional maps) in the context of shape matching, which can greatly simplify the problem of finding and manipulating correspondences across pairs of shapes. In addition, we have shown that working with tangent vector fields, represented as derivations (covariant derivatives of functions) naturally encoded as linear functional operators, can be used for vector field design and analysis. Finally, we also demonstrated that it is possible to define linear functional operators, which we call shape differences that allow to capture different kinds of distortion induced by a map between a pair of shapes. These shape differences enable novel exploration interfaces, such as extracting the most distorted areas or finding shape analogies in collections in a robust and efficient way.

There is a number of common themes that run through the applications and techniques that we have described. First, by formulating different structures as linear operators, we can often enable the use of a large number of powerful techniques from numerical linear algebra, such as inference via solution of least squares systems, operator factorisation, PCA, SVD, etc. Interestingly, these operations often have geometric equivalents, that would be somewhat non-trivial to formulate directly on the shapes themselves, especially in a multi-scale and robust way. For example, computing approximate Killing vector fields, whose flows approximately preserve the metric, can be done simply by finding vector fields, whose covariant derivative operators commute with the standard Laplace-Beltrami operator.

In addition to the computational advantages, the operator-based approach is also often useful both for relating different concepts, such as vector fields and maps, and also for providing a common language in which both continuous and discrete properties can be expressed. For example, the definition of shape difference operators is based inner products between functions, and once those are established many properties can be shown in an equivalent way between smooth surfaces and discrete triangle meshes.

Of course many questions are left unexplored in our work so far. First, we are interested to see whether the operator point of view can also be successfully applied to other shape representations, such as point clouds, volumetric data and eventually images and other types of media. We have done some preliminary work in exploring the functional map representation in the context of image co-segmentation [WHOG14] by representing images as graphs. However, much more further analysis is necessary to develop a rigorous framework as was suggested here.

A natural domain, where many of the ideas presented here can be also be explored is graph drawing and analysis. For example, a common problem in visualizing large dynamic graphs or networks, is to find regions where most of the changes occur and to highlight those regions in a robust (and especially multi-scale) way. We are currently working on adapting the techniques presented here to the domain of graph analysis.

Finally, most of the techniques discussed here are aimed at analyzing either a single shape or a pair of shapes. However, as hinted in Chapter 9, shape collections can be used to define an entire *space of shapes.* It is very interesting to explore the properties of shape spaces from a functional point of view. For example, we are planning to study ways to defe curvature in shape space, or to formulate deformation of a collection via vector field flow, among many other possible questions.

From a more conceptual point of view, it will be interesting and important to study the limitations of the linear operator approach and consider more general, perhaps non-linear functional operators. Finally, constructing better functional bases (beyond, e.g., the eigenfunctions of the Laplace-Beltrami operator) will be important to extend the ideas presented here to more general scenarios and types of geometric data.

Bibliography

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

[ACP03] Brett Allen, Brian Curless, and Zoran Popović. The space of human body shapes: reconstruction and parameterization from range scans. *ACM Trans. Graph.*, 22(3):587–594, July 2003.

[AFW06] Douglas N Arnold, Richard S Falk, and Ragnar Winther. Finite element exterior calculus, homological techniques, and applications. *Acta numerica*, 15(1):1–155, 2006.

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

[ASC11] M. Aubry, U. Schlickewei, and D. Cremers. The wave kernel signature: A quantum mechanical approach to shape analysis. In *ICCV - Workshop (4DMOD)*, 2011.

[ASK+05] Dragomir Anguelov, Praveen Srinivasan, Daphne Koller, Sebastian Thrun, Jim Rodgers, and James Davis. Scape: shape completion and animation of people. In *Proc. SIGGRAPH*, pages 408–416, 2005.

[BBK06] A. M. Bronstein, M. M. Bronstein, and R. Kimmel. Generalized multidimensional scaling: a framework for isometry-invariant partial surface matching. *PNAS*, 103(5), 2006.

[BBK08] Alexander Bronstein, Michael Bronstein, and Ron Kimmel. *Numerical Geometry of Non-Rigid Shapes*. Springer, 2008.

[BCBSG10] Mirela Ben-Chen, Adrian Butscher, Justin Solomon, and Leonidas Guibas. On discrete killing vector fields and patterns on surfaces. In *CGF*, volume 29, pages 1701–1711, 2010.

[BCWG09] Mirela Ben-Chen, Ofir Weber, and Craig Gotsman. Variational harmonic maps for space deformation. *ACM Trans. Graph.*, 28(3):34:1–34:11, July 2009.

[BKP+10] Mario Botsch, Leif Kobbelt, Mark Pauly, Pierre Alliez, and Bruno Lévy. *Polygon mesh processing*. CRC press, 2010.

[BM92] Paul J. Besl and Neil D. McKay. A method for registration of 3-d shapes. *IEEE TPAMI*, 14:239–256, 1992.

[Boo96] F.L. Bookstein. Shape and the information in medical images: A decade of the morphometric synthesis. In *Proc. IEEE MMBIA*, pages 2–12, 1996.

[Ç98] Eranda Çela. *The Quadratic Assignment Problem: Theory and Agorithms*. Kluwer Academic Publishers, 1998.

[CDS10] Keenan Crane, Mathieu Desbrun, and Peter Schröder. Trivial connections on discrete surfaces. In *CGF*, volume 29, pages 1525–1533, 2010.

[COC14] Étienne Corman, Maks Ovsjanikov, and Antonin Chambolle. Supervised descriptor learning for non-rigid shape matching. In *NORDIA: ECCV 2014 Workshop*. Springer International Publishing, 2014.

[COC15] Étienne Corman, Maks Ovsjanikov, and Antonin Chambolle. Continuous matching via vector field flow. In *Computer Graphics Forum*, volume 34, pages 129–139. Wiley Online Library, 2015.

[CSBC+17] Étienne Corman, Justin Solomon, Mirela Ben-Chen, Leonidas Guibas, and Maks Ovsjanikov. Functional characterization of intrinsic and extrinsic geometry. In *Transactions on Graphics (to appear)*, 2017.

[CT+01] T.F. Cootes, C.J. Taylor, et al. Statistical models of appearance for medical image analysis and computer vision. In *Proc. SPIE Medical Imaging*, volume 4322, pages 236–248, 2001.

[DFN92] BA Dubrovin, A Fomenko, and S Novikov. Modern Geometry-Methods and Applications, Part I. *Graduate texts in mathematics*, 1992.

[DK11] Anastasia Dubrovina and Ron Kimmel. Approximately isometric shape correspondence by matching pointwise spectral features and global geodesic structures. *Advances in Adaptive Data Analysis*, pages 203–228, 2011.

[DM98] I. L. Dryden and K. V. Mardia. *Statistical Shape Analysis*. John Wiley and Sons, 1998.

[FBW11] R. Farrell, S. Branson, and P. Welinder. First Workshop on Fine-Grained Visual Categorization (FGVC) at CVPR 2011. `http://http://www.fgvc.org/`, 2011.

[FLPJ04] P.T. Fletcher, C. Lu, S.M. Pizer, and S. Joshi. Principal geodesic analysis for the study of nonlinear statistics of shape. *Medical Imaging, IEEE Transactions on*, 23(8):995–1005, 2004.

[FSDH07] Matthew Fisher, Peter Schröder, Mathieu Desbrun, and Hugues Hoppe. Design of tangent vector fields. In *ACM Transactions on Graphics (TOG)*, volume 26, page 56. ACM, 2007.

[GF09] Aleksey Golovinskiy and Thomas Funkhouser. Consistent segmentation of 3D models. *Computers and Graphics (Shape Modeling International 09)*, 33(3):262–269, June 2009.

[GGSK05] P. Golland, W.E.L. Grimson, M.E. Shenton, and R. Kikinis. Detection and analysis of statistical differences in anatomical shape. *Medical Image Analysis*, 9(1):69 – 86, 2005.

[GSSL01] G. Gerig, M. Styner, M. Shenton, and J. Lieberman. Shape versus size: Improved understanding of the morphology of brain structures. In *Proc. MICCAI*, pages 24–32, 2001.

[HAWG08] Q-X. Huang, B. Adams, M. Wicke, and L. J. Guibas. Non-rigid registration under isometric deformations. *CGF (Proc. SGP)*, 27(5):1449–1457, 2008.

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

[HJO+01] Aaron Hertzmann, Charles E. Jacobs, Nuria Oliver, Brian Curless, and David H. Salesin. Image analogies. In *Proc. SIGGRAPH*, pages 327–340, 2001.

[HKG11] Qixing Huang, Vladlen Koltun, and Leonidas Guibas. Joint shape segmentation with linear programming. *ACM Trans. Graph.*, 30:125:1–125:12, December 2011.

[HSS+09] N. Hasler, C. Stoll, M. Sunkel, B. Rosenhahn, and H.P. Seidel. A statistical model of human pose and body shape. *Computer Graphics Forum*, 28(2):337–346, 2009.

[JZvK07] V. Jain, H. Zhang, and O. van Kaick. Non-rigid spectral correspondence of triangle meshes. *International Journal on Shape Modeling*, 13(1):101–124, 2007.

[Kat95] T. Kato. *Perturbation Theory for Linear Operators.* Springer-Verlag GmbH, 1995.

[KHS10] Evangelos Kalogerakis, Aaron Hertzmann, and Karan Singh. Learning 3D Mesh Segmentation and Labeling. *ACM Transactions on Graphics*, 29(3), 2010.

[KLF11] Vladimir G. Kim, Yaron Lipman, and Thomas Funkhouser. Blended intrinsic maps. *ACM TOG (Proc. SIGGRAPH)*, 30(4), 2011.

[KLM+12] Vladimir G. Kim, Wilmot Li, Niloy J. Mitra, Stephen DiVerdi, and Thomas Funkhouser. Exploring collections of 3d models using fuzzy correspondences. *ACM Trans. Graph.*, 31(4):54:1–54:11, July 2012.

[KMP07] Martin Kilian, N. J. Mitra, and H. Pottmann. Geometric modeling in shape space. In *Proc. SIGGRAPH*, pages 64:1–64:8, 2007.

[Koo31] Bernard O Koopman. Hamiltonian systems and transformation in hilbert space. *Proceedings of the National Academy of Sciences*, 17(5):315–318, 1931.

[KTCO+10] Oscar Kin-Chung Au, Chiew-Lan Tai, Daniel Cohen-Or, Youyi Zheng, and Hongbo Fu. Electors voting for fast automatic shape correspondence. *Comp. Graph. Forum*, 29(2):645–654, 2010.

[KWT88] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *Int. J. Comput. Vision*, 1(4):321–331, 1988.

[LF09] Y. Lipman and T. Funkhouser. Möbius voting for surface correspondence. In *Proc. of SIGGRAPH*, volume 28:3, pages 72:1–72:12, 2009.

[LSS+10] Rongjie Lai, Yonggang Shi, K. Scheibel, S. Fears, R. Woods, A.W. Toga, and T.F. Chan. Metric-induced optimal embedding for intrinsic 3d shape analysis. In *CVPR*, pages 2871 – 2878, june 2010.

[MDSB02] M. Meyer, M. Desbrun, P. Schröder, and A. H. Barr. Discrete differential geometry operators for triangulated 2-manifolds. In *Proc. VisMath'02*, Berlin, Germany, 2002.

[MHK+08] D. Mateus, R. P. Horaud, D. Knossow, F. Cuzzolin, and E. Boyer. Articulated shape matching using laplacian eigenfunctions and unsupervised point registration. In *Proc. CVPR*, 2008.

[MMP+11] P Mullen, A McKenzie, D Pavlov, L Durant, Y Tong, E Kanso, JE Marsden, and M Desbrun. Discrete lie advection of differential forms. *Found. Comp. Math.*, 11(2):131–149, 2011.

[Mor01] S Morita. *Geometry of differential forms*. American Mathematical Society, Providence, R.I, 2001.

[NBCW+11] A. Nguyen, M. Ben-Chen, K. Welnicka, Y. Ye, and L. Guibas. An optimization approach to improving collections of shape maps. In *Proc. SGP*, pages 1481–1491, 2011.

[NSN+07] D. Nain, M. Styner, M. Niethammer, J.J. Levitt, M.E. Shenton, G. Gerig, A. Bobick, and A. Tannenbaum. Statistical shape analysis of brain structures using spherical wavelets. In *Proc. ISBI*, pages 209–212, 2007.

[OBCS+12] Maks Ovsjanikov, Mirela Ben-Chen, Justin Solomon, Adrian Butscher, and Leonidas Guibas. Functional maps: a flexible representation of maps between shapes. *ACM Trans. Graph.*, 31(4):30:1–30:11, July 2012.

[OHG11] Maks Ovsjanikov, Qi-Xing Huang, and Leonidas J. Guibas. A condition number for non-rigid shape matching. *Comput. Graph. Forum (Proc. SGP)*, 30(5):1503–1512, 2011.

[OLGM11] Maks Ovsjanikov, Wilmot Li, Leonidas Guibas, and Niloy J. Mitra. Exploration of continuous variability in collections of 3d shapes. *ACM Trans. Graph.*, 30(4):33:1–33:10, July 2011.

[OMMG10] Maks Ovsjanikov, Quentin Merigot, Facundo Memoli, and Leonidas Guibas. One point isometric matching with the heat kernel. *CGF*, 29(5):1555–1564, 2010.

[OMPG13] Maks Ovsjanikov, Quentin Mérigot, Viorica Pătrăucean, and Leonidas Guibas. Shape matching via quotient spaces. *Computer Graphics Forum*, 32(5):1–11, 2013.

[OSG08] M. Ovsjanikov, J. Sun, and L. Guibas. Global intrinsic symmetries of shapes. *Comp. Graph. Forum*, 27(5):1341–1348, 2008.

[PBB11] Jonathan Pokrass, Alexander M. Bronstein, and Michael M. Bronstein. A correspondence-less approach to matching of deformable shapes. In *SSVM*, pages 592–603, 2011.

[Pet97] P. Petersen. *Riemannian geometry*. Graduate texts in mathematics. Springer, 1997.

[PLB12] Nick Pears, Yonghuai Liu, and Peter Bunting. *3D imaging, analysis and applications*, volume 3. Springer, 2012.

[PLPZ12] Daniele Panozzo, Yaron Lipman, Enrico Puppo, and Denis Zorin. Fields on symmetric surfaces. *ACM Transactions on Graphics (TOG)*, 31(4):111, 2012.

[PMT+11] D. Pavlov, P. Mullen, Y. Tong, E. Kanso, J.E. Marsden, and M. Desbrun. Structure-preserving discretization of incompressible fluids. *Physica D: Nonlinear Phenomena*, 240(6):443 – 458, 2011.

[PP93] Ulrich Pinkall and Konrad Polthier. Computing discrete minimal surfaces and their conjugates. *Exp. Math.*, 2(1):15–36, 1993.

[PP03] Konrad Polthier and Eike Preuss. Identifying vector field singularities using a discrete hodge decomposition. *Visualization and Mathematics*, 3:113–134, 2003.

[PZ07] Jonathan Palacios and Eugene Zhang. Rotational symmetry field design on surfaces. In *ACM Transactions on Graphics (TOG)*, volume 26, page 55. ACM, 2007.

[PZ11] Jonathan Palacios and Eugene Zhang. Interactive visualization of rotational symmetry fields on surfaces. *Visualization and Computer Graphics, IEEE Transactions on*, 17(7):947–955, 2011.

[ROA+13] Raif M. Rustamov, Maks Ovsjanikov, Omri Azencot, Mirela Ben-Chen, Frédéric Chazal, and Leonidas Guibas. Map-based exploration of intrinsic shape differences and variability. *ACM Trans. Graph.*, 32(4):72:1–72:12, July 2013.

[Rus07] Raif M. Rustamov. Laplace-Beltrami eigenfunctions for deformation invariant shape representation. In *Proc. SGP*, pages 225–233, 2007.

[RVAL09] Nicolas Ray, Bruno Vallet, Laurent Alonso, and Bruno Levy. Geometry-aware direction field processing. *ACM Transactions on Graphics (TOG)*, 29(1):1, 2009.

[SH10] Avinash Sharma and Radu P. Horaud. Shape matching based on diffusion embedding and on mutual isometric consistency. In *Proc. NORDIA Workshop (CVPR)*, June 2010.

[SM93] Raj Kishor Singh and Jasbir Singh Manhas. *Composition operators on function spaces*, volume 179. Elsevier, 1993.

[SMW06] Scott Schaefer, Travis McPhail, and Joe Warren. Image deformation using moving least squares. *ACM TOG*, 25(3):533–540, July 2006.

[SOCG10] P. Skraba, M. Ovsjanikov, F. Chazal, and L. Guibas. Persistence-based segmentation of deformable shapes. In *CVPR Workshop on Non-Rigid Shape Analysis and Deformable Image Alignment*, pages 45–52, June 2010.

[SOG09] J. Sun, M. Ovsjanikov, and L. Guibas. A Concise and Provably Informative Multi-Scale Signature Based on Heat Diffusion. *CGF (Proc. SGP)*, 28(5), 2009.

[SP04] Robert W. Sumner and Jovan Popović. Deformation transfer for triangle meshes. *ACM Trans. Graph.*, 23(3):399–405, August 2004.

[Spi99] Michael Spivak. *A comprehensive introduction to differential geometry. Vol. I.* Publish or Perish Inc., third edition, 1999.

[SY11] Y. Sahillioğlu and Y. Yemez. Coarse-to-fine combinatorial matching for dense isometric shape correspondence. *Computer Graphics Forum*, 30(5):1461–1470, 2011.

[TBW+11] Art Tevs, Alexander Berner, Michael Wand, Ivo Ihrke, and H-P Seidel. Intrinsic shape matching by planned landmark sampling. In *Computer Graphics Forum*, volume 30, pages 543–552. Wiley Online Library, 2011.

[TLHD03] Yiying Tong, Santiago Lombeyda, Anil N Hirani, and Mathieu Desbrun. Discrete multiscale vector field decomposition. In *ACM Transactions on Graphics (TOG)*, volume 22, pages 445–452, 2003.

[vKTS+11] Oliver van Kaick, Andrea Tagliasacchi, Oana Sidi, Hao Zhang, Daniel Cohen-Or, Lior Wolf, , and Ghassan Hamarneh. Prior knowledge for part correspondence. *Computer Graphics Forum (Proc. Eurographics)*, 30(2):553–562, 2011.

[vKZHCO11] Oliver van Kaick, Hao Zhang, Ghassan Hamarneh, and Daniel Cohen-Or. A survey on shape correspondence. *Computer Graphics Forum*, 30(6):1681–1707, 2011.

[VT07] M. Vasilescu and D. Terzopoulos. Multilinear (tensor) ICA and dimensionality reduction. *Independent Component Analysis and Signal Separation*, pages 818–826, 2007.

[Wey46] Hermann Weyl. *The Classical Groups: Their Invariants and Representations*. Princeton University Press, 1946.

[WHOG14] Fan Wang, Qixing Huang, Maks Ovsjanikov, and Leonidas J Guibas. Unsupervised multi-class joint image segmentation. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3142–3149. IEEE, 2014.

[WSX12] Stefanie Wuhrer, Chang Shu, and Pengcheng Xi. Posture-invariant statistical shape analysis using laplace operator. *Computers & Graphics*, 36(5):410–416, 2012.

[WZG+10] Y. Wang, J. Zhang, B. Gutman, T.F. Chan, J.T. Becker, H.J. Aizenstein, O.L. Lopez, R.J. Tamburo, A.W. Toga, and P.M. Thompson. Multivariate tensor-based morphometry on surfaces: Application to mapping ventricular abnormalities in hiv/aids. *Neuroimage*, 49(3):2141–2157, 2010.

[XLZ+10] Kai Xu, Honghua Li, Hao Zhang, Daniel Cohen-Or, Yueshan Xiong, and Zhiquan Cheng. Style-content separation by anisotropic part scales. *ACM Transactions on Graphics, (Proceedings SIGGRAPH Asia 2010)*, 29(5):184:1–184:10, 2010.

[YLSL10] I-Cheng Yeh, Chao-Hung Lin, Olga Sorkine, and Tong-Yee Lee. Template-based 3d model fitting using dual-domain relaxation. *IEEE Transactions on Visualization and Computer Graphics*, 99, 2010.

[ZSCO+08] H. Zhang, A. Sheffer, Cohen-Or, Q. Zhou, O. van Kaick, and A. Tagliasacchi. Deformation-driven shape correspondence. In *Proc. SGP*, pages 1431–1439, 2008.