# Functional Maps and Shape/Image Networks

SIGGRAPH 2014 COURSE NOTES

**Organizers & Lecturers:**

Leonidas Guibas, Qi-Xing Huang
Stanford University

Maks Ovsjanikov
Ecole Polytechnique

Mirela Ben-Chen
Technion — Israel Institute of Technology

## Abstract

Notions of *similarity* and *correspondence* between geometric shapes and images are central to many tasks in geometry processing, computer vision, and computer graphics. The goal of this course is to familiarize the audience with a set of recent techniques that treat geometric objects as functional spaces rather than collections of points or triangles and provide a novel perspective on the similarity or correspondence problems—by treating mappings between spaces as objects in their own right. These "functional" representations of mappings have recently led to state-of-the-art results in problems as diverse as non-rigid shape matching, image co-segmentation and tangent vector field design. One challenge in adopting these methods in practice, however, is that their exposition often assumes a significant amount of background in geometry processing, spectral methods and functional analysis, which can make it difficult to gain an intuition about their performance or about their applicability to real-life problems. In this course, we try to provide all the tools necessary to appreciate and use these techniques, while assuming very little background knowledge. We also give a unifying treatment of these techniques, which may be difficult to extract from the individual publications and, at the same time, hint at the generality of this point of view, which can help tackle many problems in the analysis and creation of visual content.

This course is structured as a half day course. We will assume that the participants have knowledge of basic linear algebra and some knowledge of differential geometry, to the extent of being familiar with the concepts of a manifold and a tangent vector space. We will discuss in detail the functional approach to finding correspondences between non-rigid shapes, the design and analysis of tangent vector fields on surfaces, consistent map estimation in networks of shapes and applications to shape and image segmentation, shape variability analysis, and other areas.

# Contents

# Course Schedule

**Course Intro: Leonidas Guibas – 20 mins**
- Course Overview
- Dual representations and the operator point of view
- Functional maps, map networks and notions of map consistency

**Functional Map Estimation: Maks Ovsjanikov – 45 mins**
- Non-rigid shape matching problem for 3D shapes
- Functional map representation and its properties
- Functional map analysis and inference
- Shape differences via functional maps

**Functional Vector Fields on Surfaces: Mirela Ben-Chen – 45 mins**
- Common tangent vector field representations
- Operator approach to tangent vector fields
- The relationship between maps and vector fields
- Design of tangent vector fields

**Short Break – 20 mins**

**Consistent Map Estimation: Qi-Xing Huang – 45 mins**
- Consistency in map networks: low rank, positive semi-definiteness
- Latent spaces and projections
- Map correction

**Applications: Leonidas Guibas – 45 mins**
- Variability analysis is shape collections
- Latent space formulation of the segmentation problem
- Shape segmentation and classification
- Co-segmentation of image collections

**Concluding remarks: Maks Ovsjanikov – 15mins**
- Course wrap-up and conclusion
- Q&A

# 1

# Introduction

> *"When you have to compare images, the question is how you compare them. Amazingly—for a geometer it looks unbelievable—the early work on computer vision was based on matching images with another, taking differences in intensity—which is certainly completely contrary to what your eyes do! [...] Roughly, the eye does this. It does not add images. It moves images. And it has to move them in the right category, which is roughly the category that appears in Riemannian geometry."*
>
> – M. Gromov *Interview with Martin Raussen and Christian Skau*

The ever increasing availability of repositories of visual data such as 3D models and images or videos provides an unprecedented opportunity to facilitate the content creation process in computer graphics—for this data encodes the geometry, appearance, and function of objects in the world. How to extract knowledge from such data, however, and how to use it for computer graphics applications, is not straightforward.

Towards this goal, it is important to obtain a joint understanding of geometric data, looking at each 3D model or image not in isolation but in the context provided by all the other related data. This course presents a set of techniques for making relationships or correspondences between geometric data sets first-class citizens—so that the relationships themselves become explicit, algebraic, storable and searchable objects. In particular the course presents the mathematics and algorithms for computing functional maps between data sets. These dual maps are defined between function spaces over the base objects. Mathematically, they are linear operators represented by ordinary matrices, making familiar tools from linear algebra immediately applicable.

In the setting of a collection of related 3D models or images, the course

explores how entire networks of functional maps can be built and used. Such a network is more powerful than a similarity graph, because its edges are decorated with operators that can be composed to transport information across the network. In particular, the network can be used as a regularizer, providing global context in the computation of individual maps, or as a tool for detecting shared structure across regions of the network. The course covers specific examples of how network analysis can be carried out to derive consistent shape or image segmentations, or to extract variability within a collection of shapes, exploiting in all cases the "wisdom of the collection."

## 1.1 Course Goals

The goal of these course notes is to describe the main mathematical ideas behind the functional mapping framework and to provide implementation details for several applications in geometry processing, computer vision and computer graphics. The text in the course materials is primarily based on previously published work including [16, 19, 3, 8] and [25]. Our aim is to give the reader a more intuitive feel for the methods presented in these papers and to highlight the common "functional" thread that runs through them. We also aim to provide practical implementation details for the methods presented in these works, as well as suggest further readings and extensions of these ideas.

## 1.2 Motivation

A common task in many fields of applied science is to establish, quantify and predict *relations* between various objects. In this course we are primarily interested in geometric and multimedia data, such as 2D images and 3D surfaces. In this context, a common problem is to see whether two images contain the same object or whether two 3D models represent deformations of the same physical entity. If so, then the challenge is to find the points that correspond to each other on the different models (Figure 1.1). Such operations of comparison often revolve around mappings or correspondences between objects, which can be represented as functions of the type: $T : M_1 \rightarrow M_2$, where $M_1$ and $M_2$ are geometric objects and $T$ is a mapping, which takes points on $M_1$ to points on $M_2$. A common unifying theme in the methods that we present in this course is to treat the mappings $T$ as objects in their own right, and in particular, as carriers of information

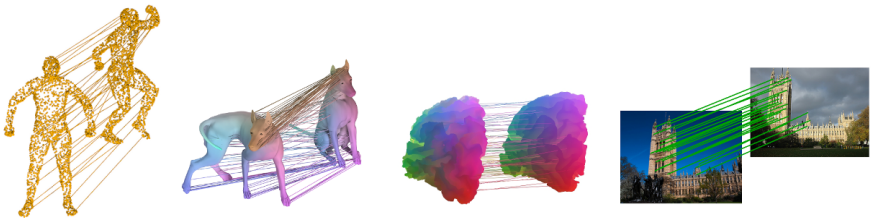


Figure 1.1: Mappings or correspondences between point clouds, triangle meshes, volumetric data and images respectively .

which can be manipulated, stored, analyzed or optimized in a coherent framework.

The notion of mappings is very general and appears in a large number of theoretical and practical settings. For example, as we demonstrate in Chapter 5, tangent vector fields on surfaces can be considered as prescriptions for motion or *infinitesimal mappings* between different points on the surface. Thus designing vector fields can be viewed through the lens of map estimation and analysis.

One difficulty that often arises in practice is that mappings or correspondences are not easy to manipulate due to their inherently high-dimensional (often infinite-dimensional) and non-linear nature. For example, a bijective correspondence between two shapes or images containing $n$ points can be represented as a permutation on $n$ elements, and optimizing a quality metric in the space of permutations most often leads to very difficult combinatorial problems. For this reason many methods in shape or image matching have to deal with non-linear, non-convex optimization problems.

While designing more efficient methods for solving these hard optimization problems has led to significant advances in geometry processing and computer vision, in this course we argue that another useful path to consider is an analysis of the *space of mappings*, which can shed light on both the complexities of the optimization problems as well as on the exact nature of the relations between objects.

Thus, the primary goal of this course is to provide tools for understanding and analyzing spaces of mappings between pairs or sets of geometric objects. A very simple instance of this idea is the continuous "relaxation" techniques employed in integer programming. There, instead of optimizing over the space of discrete variables which can result in NP-hard problems, one is often led to a setting where optimization variables are allowed

to take continuous values (e.g., a real value between $0$ and $1$ instead of an indicator value). Such techniques often allow tackling difficult problems by going to a larger space of solutions, in which optimization can be done efficiently.

The techniques that are presented in this course all share this flavor of considering the space of solutions and analyzing the space of allowable variables rather than concentrating on specific methods for solving complex optimization problems. A particular way is which we propose to do is is to shift our point of view to dual spaces, spaces of functions or attributes defined over 3D meshes or over images. In that framework mappings or correspondences are best viewed as *operators*, or transporters of information between geometric models or images, rather than as static objects. This shift in focus, while more subtle than the previous one, leads to particularly fruitful techniques since it allows us to interpret mappings or correspondences as objects in their own right, whose primary objective is to transfer certain quantities between the original objects. Moreover, the operators associated with mappings are often *linear* even when the underlying spaces are highly complex—for example, we will see that this is the case with finding correspondences between surfaces represented as triangle meshes in 3D. The linear operator representation of mappings allows us to borrow standard tools from linear algebra such as operator inversion, spectral analysis or operator composition which can provide valuable insight into the nature of the connections between the underlying spaces.

In the following chapters we will provide details on how concentrating on efficient representations of mappings and focusing on their operator nature can lead to significant improvement in a large number of common geometry processing, computer vision and computer graphics tasks.

# 2

# What are Functional Maps?

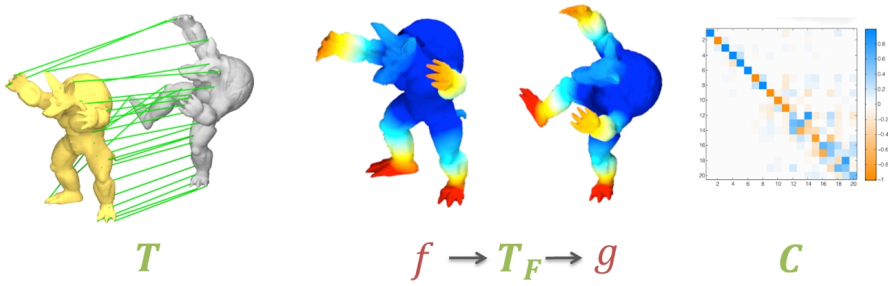



Figure 2.1: A point to point map $T : M \to N$ can be represented as a *functional map* $T_F$: a correspondence between functions $f : M \to \mathbb{R}$ and functions $g : N \to \mathbb{R}$. Given a choice of basis for functions on $M$ and $N$, the functional map can be concisely represented as a matrix $C$.

### 2.0.1 Functional Maps in the Continuous Setting

The functional approach to mappings is based on the idea of "lifting" maps between data sets to maps of function spaces over the data sets. 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 $T$ (see Figure 2.1). We now make the following two simple remarks:

**Remark 2.0.1.** *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 2.0.2.** *For any fixed bijective map $T : M \to N$, $T_F$ is a* linear *map between the corresponding 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 $f = \sum_i a_i \phi_i^M$. Then,

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

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

$$T_F(f) = \sum_i a_i \sum_j c_{ij} \phi_j^N = \sum_j \sum_i a_i c_{ij} \phi_j^N . \tag{2.1}$$

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. 2.1 simply says: $b_j = \sum_i a_i c_{ij}$, where $c_{ij}$ is independent of $f$ and is completely determined by the bases and the map $T$. In particular $c_{ij}$ is the $j^{\text{th}}$ 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 $c_{ij} = \langle T_F(\phi_i^M), \phi_j^N \rangle$.

We conclude with the following key observation:

**Remark 2.0.3.** *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 $T$.

#### Functional Maps

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*

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

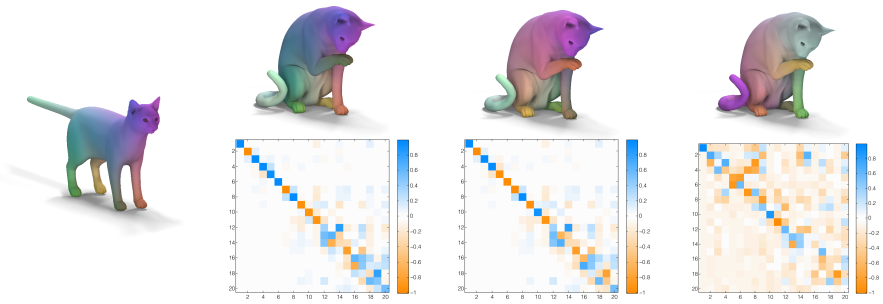
*where $c_{ij}$ is a possibly infinite matrix of real coefficients (subject to conditions that guarantee convergence of the sums above).*

Of course in practice we typically use a hierarchical basis and then truncate it to obtain a finite matrix for $C$. This is discussed in more detail below.

**Example.** As an example, consider a pair of shapes in Figure 2.2 with three bijective maps between them: two approximate isometries (the ground-truth map 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.

## 2.1 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



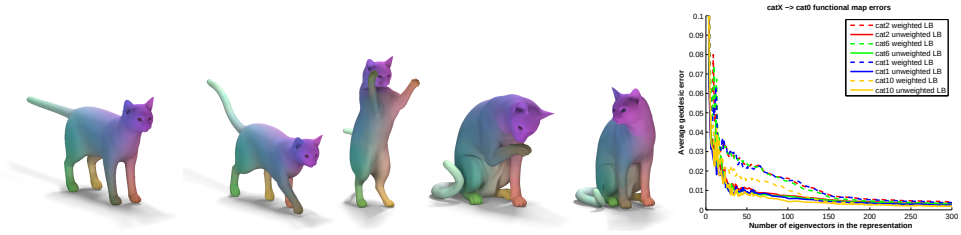
(a) source (b) ground-truth map (c) left to right map (d) head to tail map

Figure 2.2: 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 less sparse functional matrix.

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 0 function on the other—$D$ 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 2.1.1) and then by showing how many natural constraints on the map can be phrased as linear constraints on the functional map (Section 2.1.3), reducing shape matching to a moderately-sized system of linear equations (Section 3).




(a) target (b) Cat10 (c) Cat1 (d) Cat2 (e) Cat6 (f) errors

Figure 2.3: Average mapping error vs. number of eigenvalues used in the functional representation. For each shape with a known ground-truth point-to-point correspondence (shown as a color correspondence), we measure the accuracy of the functional representation of the map. Note that although more eigenvalues lead to an increase in accuracy, maps that correspond to bigger deformations require more basis vectors, capturing the intuition that near-isometric maps are more compactly represented.

### 2.1.1 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 *continuous* mappings between shapes — the desired behavior in the majority of 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 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 $\sum_j \sum_i a_i c_{ij} \phi_j^N \approx \sum_{j=0}^{n} \sum_{i=0}^{m} a_i c_{ij} \phi_j^N$,

for some fixed small values of $m$ and $n$.

In this chapter, 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 $n^{\text{th}}$ and $(n+1)^{\text{st}}$ eigenvalues are well separated, as shown for example in the work of [10].

To illustrate the role of the choice of basis on the functional representation, we compare two widely-used discretizations of the Laplace-Beltrami operator and measure their ability to capture a ground-truth point-to-point correspondence using a fixed number $n$ of basis functions. In particular, we consider, the cotangent weight scheme of Meyer et al. [13] with and without area normalization (in the latter case, each vertex is assigned a uniform weight, while in the former the weight is proportional to the sum of the areas of triangles around the point). Figure 2.3 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 [5] 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: $T(x) = \arg\max_y T_F(f)(y)$ where $f$ is the indicator function at the point $x$ on shape $M$. I.e. $T(x)$ is the point where the function $g = T_F(f)$ is maximal. The error is measured in average geodesic error units (see [11] Figure 7 middle for illustration).

Note that the eigenfunctions of the unweighted discretization of Laplace-Beltrami operator provide a more compact representation of the map for the same quality of reconstruction. This may be because this discretization is less sensitive to volume distortion (and only sensitive to non-conformal distortions). Moreover, note 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. Since a functional representation with 40 eigenvectors implies a matrix of size $40 \times 40$, this means that even without

exploiting its sparsity (described below), this representation has 1600 values, which is a nearly 17 times memory savings over a permutation of size 27.8K.

**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 $\phi_j^M$ and $\phi_i^N$ 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 2.4 shows the sparsity patterns of the matrices $C$ corresponding to two of the maps shown in Figure 2.3. 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 1.1 illustrates the same effect). While this poses fundamental challenges to previous spectral methods [9, 12, 15], the functional representation naturally encodes such changes.

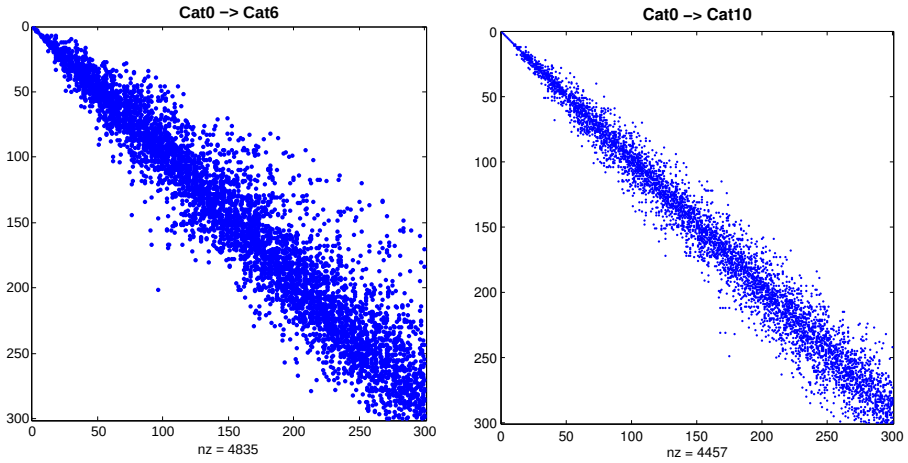
### 2.1.2 Continuity

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. Thus, if we use a truncated functional representation matrix $C_{0..n \times 0...n}$ then the image $C\mathbf{a}$ of any function $f$ will be smooth.

(a) cat0 (b) cat6 (c) cat10




(d) cat0 → cat6 (e) cat0 → cat10

Figure 2.4: Sparsity pattern of the matrices $C$ corresponding to the maps between the models shown. 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.

**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 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 2.5 we show the image of a set of three functions from a fixed source shape (cat0 shown in Figure 2.4) onto a target shape under a mapping matrix, interpolated between two mappings corresponding to the direct and symmetric shape matching.

Note that each mapping is both meaningful and produces intuitive results.

### 2.1.3 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 e.g. 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).

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

Figure 2.5: Mapping of the three coordinate functions from the source shape cat0 shown in Figure 2.4 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 $\alpha$.

**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 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. We will discuss the actual choice of the functions used below.

### 2.1.4 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 $S_F$ and $R_F$ on $M$ and $N$ respectively, it may be natural to require that the functional map $C$ commute with these operators. In particular: $R_F C = C S_F$ or $\|R_F C - C S_F\| = 0$. This constraint, despite its second order flavor, leads to linear equations in the elements of $C$.

### 2.1.5 Regularization Constraints

Note that although we mentioned in Section 2.1.2 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 seems to be a difficult open problem, a very useful observation is the following:

**Theorem 2.1.1.** *(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* $\mu^M$ *and* $\mu^N$ *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 commutation with the Laplace-Beltrami operators into the regularization.

### 2.1.6 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 2.4) it is reasonable to take the inverse of the approximating submatrix of $C$. Finally, in light of Theorem 2.1.1 this can be done by simply taking the transpose of $C$ or its approximation.

Similarly, map composition becomes simple matrix multiplication in the functional representation. We exploit these properties when we use our representation for joint map inference on a collection of shapes in Chapter 6.

# 3

# Functional Map Inference

As mentioned in Section 2.1, 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.

### 3.0.7 Linear System Formulation

More concretely, suppose we are given a set of functions on the two shapes that are expected to match. Namely, suppose we have a collection of functions $f_i : M \to \mathbb{R}$ and $g_i : N \to \mathbb{R}$, such that $g_i \approx f_i \circ T$ for some unknown map $T$. In practice, we use two clases of such functions: 1) local descriptors that capture some properties in a neighborhood of each point and 2) functions encoding landmark correspondences between pairs pairs of points or pairs of segments on the shapes (See 3.1 for details). Let $\mathbf{f}_i$ and $\mathbf{g}_i$ be the vectors of coefficients of $f_i$ and $g_i$ in the functional bases of $M$ and $N$ respectively. Then, we expect the functional map $C$ to be such that $C\mathbf{f}_i \approx \mathbf{g}_i$. Moreover, when two shapes are approximately isometric we expect the functional map to commute with the Laplace-Beltrami operators

on $M$ and $N$: $C \circ \Delta_M \approx \Delta_N \circ C$. These two constraints lead to the following linear least squares problem:

$$C_{\text{opt}} = \arg\min_C \sum_i \|C\mathbf{f}_i - \mathbf{g}_i\|_2^2 + \|C\Delta_M - \Delta_N C\|_2^2 \tag{3.1}$$

Note that when the functional bases on $M$ and $N$ are given by the eigenfunctions of the Laplace-Beltrami operator the matrices corresponding to $\Delta_M$ and $\Delta_N$ are diagonal matrices of eigenvalues of $M$ and $N$. In this case we have:

$$C_{\text{opt}} = \arg\min_C \sum_i \|C\mathbf{f}_i - \mathbf{g}_i\|_2^2 + \sum_{i,j}(C_{ij}(\lambda_i^M - \lambda_j^N))^2 \tag{3.2}$$

Note that Eq. 3.2 is a linear least squares system which can be solved efficiently using standard numerical linear algebra techniques. In Section 3.1 we outline the practical implementation details and functional constraints that can be used in the particular case of isometric shape matching.

### 3.0.8 Efficient Conversion to Point-to-Point

As mentioned in Section 3.0.7, 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 2.0.1 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, note that in the Laplace-Beltrami basis $\delta_x$, 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 $\Phi^M$ 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

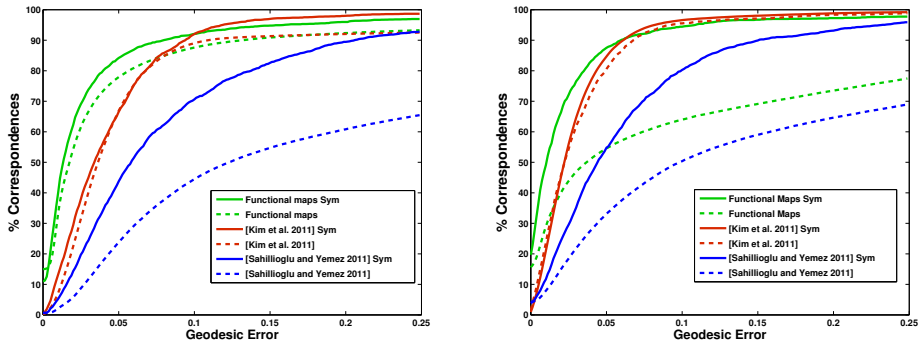
**Algorithm 1** FUNCTIONAL MAP INFERENCE FOR MATCHING

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 $C$ using the iterative method in Section 3.0.9.
6. If point correspondences are required, obtain them using the method in Section 3.0.8.

$\mathbf{b}_1$ 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 $\Phi^N$. Using an efficient proximity search structure, such as a $k$-$d$ 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.

### 3.0.9 Post-Processing Iterative Refinement

The observation made in Section 3.0.8 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 map $T$. As noted in Section 3.0.8, theoretically $C_0$ must be such that each column of $C_0\Phi^M$ coincides with some column of $\Phi^N$. If we treat $\Phi^M$ and $\Phi^N$ 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* $\Phi^M$ and $\Phi^N$. Moreover, since by Theorem 2.1.1 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 $\Phi^M$ and $\Phi^N$. Thus an iterative refinement of $C_0$ can be obtained via:




(a) SCAPE (b) TOSCA

Figure 3.1: Comparison of the functional method with the state-of-the-art methods of Kim et al. [11] and Sahillioglu and Yemez [20] on two datasets: SCAPE [1] and TOSCA [5] 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.

1. For each column $x$ of $C_0\Phi^M$ find the closest $\tilde{x}$ in $\Phi^N$.
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, [4], 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 $\Phi^M$ and $\Phi^N$ 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 3.1.

**Relation to Existing Methods.** Note that this refinement step is similar to existing spectral matching methods such as [9, 12, 21]. 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 2.1.4) and *represent* the map itself compactly.

## 3.1 Shape Matching

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

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, 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 3.0.9 and compute the point-to-point map using the method in Section 3.0.8.

### 3.1.1 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 [13] for the Laplace-Beltrami operator on meshed surfaces. We also used the Wave Kernel Signature (WKS) and Heat Kernel Signature descriptors of [2] and [23]. 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 [22] 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 [17] 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 3.0.9 and find the final dense point-to-point correspondences using the method in 3.0.8.

# 4

# Shape Differences

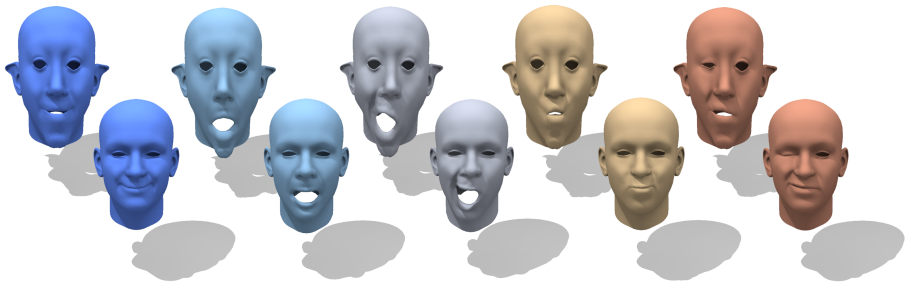


Figure 4.1: The notion of shape difference described in this chapter provides a way to compare deformations between shape pairs. This allows us to recognize similar expressions of shape $A$ (top row) to those of shape $B$ (bottom row), without correspondences between $A$ and $B$ and without any prior learning process.

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, 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.

In this chapter we will see how the notion of functional map between two shapes leads to a natural formulation of a very general and inclusive notion of shape difference.

## 4.1 An Operator View of Shape Differences

### 4.1.1 Area and Conformal Differences

### 4.1.2 Key Properties

## 4.2 Comparing Differences

### 4.2.1 Difference Transport

## 4.3 Discussion

# 5

# Functional Vector Fields

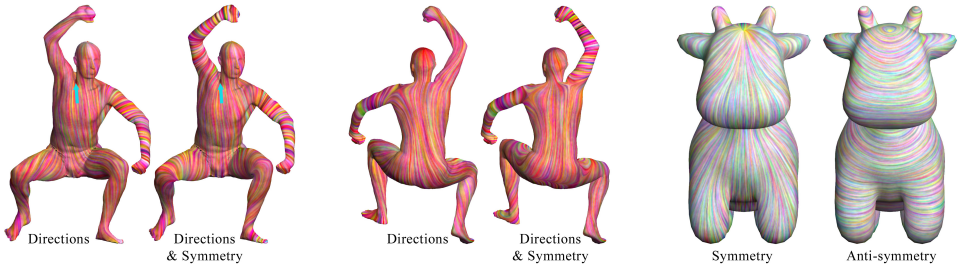



Figure 5.1: Various examples of tangent vector fields designed using functional vector fields.

One of the main advantages of the functional perspective is the ability to solve various problems in the same framework. So far, we have addressed solely maps between shapes, and discussed how these can be represented and computed. However, functional operators can be used to represent other objects useful in geometry processing and graphics. In this chapter, we will focus on the functional representation of *tangent vector fields*.

## 5.1 Discrete Vector Fields Representations

Given a manifold $M$, a tangent vector field is a smooth assignment of a tangent vector $V(p)$ to each point $p \in M$ (see Figure 5.1). While this definition is straight forward in the continuous case, representing discrete vector fields is a non-trivial problem. One option (e.g. [18, 24]) is to use piece-wise constant vector fields, namely an assignment of a vector to each planar face

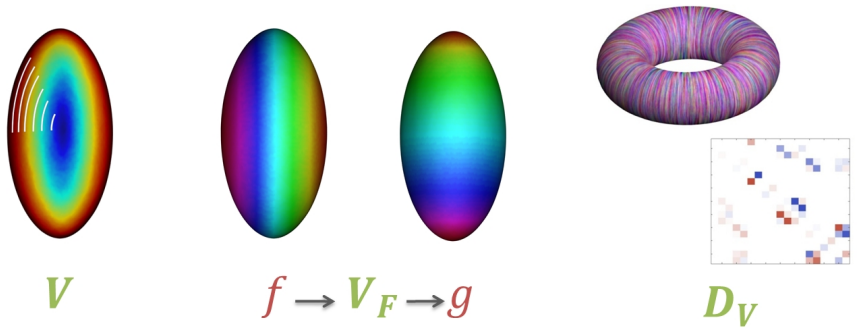



Figure 5.2: The functional point of view allows us to consider a vector field $V$ as a linear operator $V_F$ acting on real-valued functions defined on the shape and represent it simply as a matrix $D_V$ that can be manipulated easily.

of a discrete surface (e.g a triangle mesh). This representation has some drawbacks, however. It is difficult to enforce smoothness on the vector field, and in general is difficult to perform vector field design.

An alternative discretization of vector fields was suggested as part of the formalism of Discrete Exterior Calculus (DEC) [7], where vector fields are identified with discrete 1-forms, represented as a single scalar per edge. This approach is inherently coordinate-free, allowing the formulation of vector field design as a linear system [6]. While this approach is very useful, there are various properties of vector fields, for example their close relationships to maps, which are harder to model using this representation.

## 5.2 The Operator Approach

### 5.2.1 Formulation

As we have seen in previous chapters, we can represent a map $T : M \to N$ as a linear operator $T_F$ which takes scalar functions on $M$ to scalar functions on $N$. We can use a similar approach for representing tangent vector fields. Given a vector field $V : M \to TM$, we define the operator $V_F : \mathcal{F}(M, \mathbb{R}) \to \mathcal{F}(M, \mathbb{R})$ as:

$$V_F(f) = \langle \nabla f, V \rangle \,, \tag{5.1}$$

where $\langle , \rangle$ stands for the standard inner product in the tangent space of a point $p \in M$. We denote $V_F$ as the *functional representation* of the vector field $V$, or a *functional vector field*.

As for maps, this operator has some appealing properties. First, due to the linearity of the gradient operator, it is linear, namely $V_F(\alpha_1 f_1 + \alpha_2 f_2) = \alpha_1 V_F(f_1) + \alpha_2 V_F(f_2)$. Furthermore, it is well known (see [14] p.38) that $V$ can be reconstructed from $V_F$, namely if for any function $f$ we have that $V_F(f) = W_F(f)$, then $V = W$.

These two properties indicate that we can represent $V_F$ (and thus $V$) using a (possibly infinite) matrix $D_V$, given a basis $\{\phi_i^M\}$ of the function space on $M$. Specifically, if the basis is orthogonal with respect to some inner product $\langle\langle , \rangle\rangle$, then the entries of the matrix $D_V$ are given by (See Figure 5.2 for illustration):

$$(D_V)_{ij} = \langle\langle V_F(\phi_i), \phi_j \rangle\rangle \,. \tag{5.2}$$

### 5.2.2 Implementation

**Choice of basis**

**Discrete representation**

**Example application - the Lie Bracket**

**Example application - function symmetrization**

## 5.3 On Tangent Vector Fields and Maps

### 5.3.1 The flow of a vector field

### 5.3.2 The matrix exponent relationship

### 5.3.3 Commutativity properties

**Commutativity with LB**

**Commutativity with symmetric maps**

### 5.3.4 Implementation

**Example application - function advection**

## 5.4 Tangent Vector Field Design

**A basis for vector fields**

**Combining global and local constraints**

**Example application - computing Killing vector fields**

**Example application - joint vector field design**

# 6

# Consistent Map Estimation

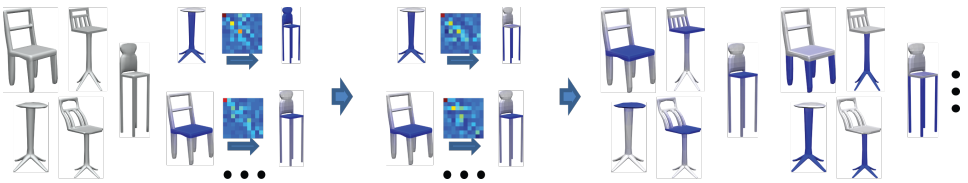



Figure 6.1: The first step takes a shape collection of (noisy) initial functional maps between pairs of shapes as input and solves a low-rank matrix recovery problem to correct the pair-wise maps. The second step then extracts consistent basis functions from optimized pair-wise maps.

When constructing functional maps among a collection of objects, we typically build a network to connect these objects and compute maps along edges in this network. In this context, there is a natural constraint that these maps should be consistent with each other. In the full similarity case (i.e., the maps are isomorphisms or homeomorphisms), this constraint is given by the fact map compositions along network cycles yield (at least approximately) the identity map. However, things become more complicated in the partial similarity case. The goal of this chapter is to study how to formulate the consistency constraint in both the full and partial similarity settings and explore how consistency constraints can be exploited in map correction among a network of objects.

## 6.1 Functional Map Networks

### 6.1.1 Map Collection Matrix

### 6.1.2 Consistent Basis and Low-rank Factorization

## 6.2 Map Network Construction

### 6.2.1 Network Based Map Correction

### 6.2.2 Global Map Initialization

### 6.2.3 Local Joint Map Refinement

## 6.3 Results and Discussion

# 7

# Applications

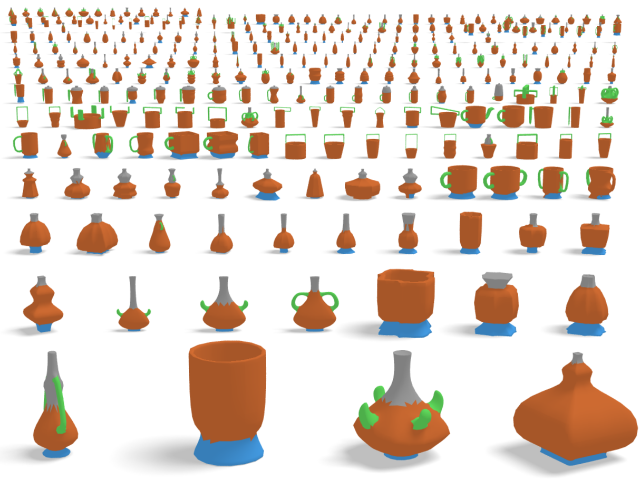


Figure 7.1: Example co-segmentation result on the Vase dataset from [26].

In this chapter we discuss various application of the techniques presented in the previous chapters to the analysis of collections of shapes or images. In particular, we cover tools for exploring and understanding variability in shape collections for the segmentations of shape and images, and for shape classification.

## 7.1 Shape Variability

### 7.1.1 Exploring Shape Collections

### 7.1.2 Shape Analogies

## 7.2 Image and Shape Segmentation via Consistent Map Networks

### 7.2.1 Latent Space Formulation

### 7.2.2 Image Co-Segmentation, Single Entity

### 7.2.3 Image Co-Segmentation, Multiple Entities

### 7.2.4 Shape Segmentation

## 7.3 Shape Classification

# Bibliography


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

[2] M. Aubry, U. Schlickewei, and D. Cremers. The wave kernel signature: A quantum mechanical approach to shape analysis. In *IEEE International Conference on Computer Vision (ICCV) - Workshop on Dynamic Shape Capture and Analysis (4DMOD)*, 2011.

[3] Omri Azencot, Mirela Ben-Chen, Fred Chazal Chazal, and Maks Ovsjanikov. An operator approach to tangent vector field processing. *Computer Graphics Forum*, 32(5):73–82, 2013.

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

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

[6] Matthew Fisher, Peter Schröder, Mathieu Desbrun, and Hugues Hoppe. Design of tangent vector fields. *ACM Transactions on Graphics (TOG)*, 26(3):56, 2007.

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

[8] Qi-Xing Huang and Leonidas Guibas. Consistent shape maps via semidefinite programming. *Computer Graphics Forum*, 32(5):177–186, 2013.

[9] 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.

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

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

[12] 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.

[13] 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.

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

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

[16] 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.

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

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

[19] 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.

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

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

[22] 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.

[23] 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.

[24] 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.

[25] Fan Wang, Qixing Huang, and Leonidas Guibas. Image co-segmentation via consistent functional maps. In *ICCV*, 2013.

[26] Yunhai Wang, Shmulik Asafi, Oliver van Kaick, Hao Zhang, Daniel Cohen-Or, and Baoquan Chen. Active co-analysis of a set of shapes. *ACM Trans. Graph.*, 31(6):165:1–165:10, November 2012.