Computing and Processing Correspondences with Functional Maps

SIGGRAPH 2017 COURSE NOTES

Organizers & Lecturers:

Maks Ovsjanikov, Etienne Corman, Michael Bronstein,
Emanuele Rodolà, Mirela Ben-Chen, Leonidas Guibas,
Frederic Chazal, Alex Bronstein

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 greatly facilitate the computation of mappings or correspondences between geometric datasets, such as 3D shapes or 2D images by formulating them as mappings between functions rather than points or triangles.

Methods based on the functional map framework have recently led to state-of-the-art results in problems as diverse as non-rigid shape matching, image co-segmentation and even some aspects of 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

1

Introduction

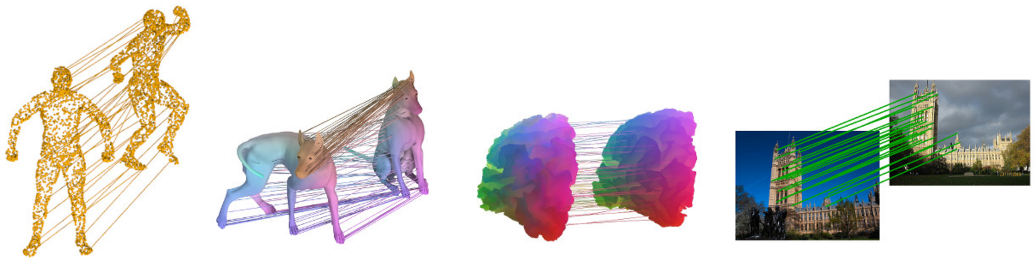


Figure 1.1: Mappings or correspondences between point clouds, triangle meshes, volumetric data and images respectively .

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 [OBCS+12, ROA+13, ABCCO13, HG13, AWO+14, COC14, RMC15] among several others. 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.

Motivation A common task in many fields of applied science is to establish, quantify and predict *mapping* 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: $T : \mathcal{M} \to \mathcal{N}$, where $\mathcal{M}$ and $\mathcal{N}$ are geometric objects and T is a mapping, which takes points on $\mathcal{M}$ to points on $\mathcal{N}$. 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 which can be manipulated, stored, analyzed or optimized in a coherent framework.

1.2 Overview

In this course, we will concentrate on one transformation associated with mappings, which turns out to have particularly nice properties both theoretically and in practice and to facilitate a wide variety of applications from shape matching to tangent vector field design. Namely, we will consider how *mappings act on real-valued functions* defined on the objects. Thus, rather than studying mappings through the lens of couplings or correspondences between points, which has been a standard approach in many domains, we will instead analyze the interaction between mappings and the transportation of real-valued functions across different objects. This approach might seem artificial or unnecessarily complex at first sight. Indeed, why introduce an additional structure (real-valued functions) to a seemingly unrelated problem (finding correspondences between points)? However, as we try to show in this course, considering mappings through their action on functions is both simpler and significantly more flexible than using the classical notions of correspondences between points. To give a hint of the material presented in the following chapters, we note here that by associating functions on different objects, it becomes possible to express correspondences between not only points but also probability densities, which can be useful, for example, when the correspondence is only known approximately. Similarly, regions or patches on the objects can be considered simply as the appropriate indicator functions, and thus, for example, transfered when a functional correspondence is known between objects.

More fundamentally the space of real-valued functions defined on the objects enjoys a special structure that points or regions do not. For example, unlike points or triangles, real-valued functions can be naturally added and multiplied, either pointwise or via an inner product, to define new functions and scalars respectively. This means that in many cases, real-valued functions live in a (suitably defined) *vector space*, which greatly facilitates their manipulation and processing (Chapter 2).

In addition to the nice algebraic structure enjoyed by the space of real-valued functions, another key aspect that we emphasize in this course is that linear maps (operators) between functions can be naturally encoded as matrices in the discrete setting. Moreover the size of these matrices can be controlled and made independent of the number of points on the objects, by using a reduced basis for the space of functions (the details on this are given in Chapter 2). Thus, the task of recovering a mapping can be phrased as an optimization problem over reasonably-sized matrices. This opens the door to many numerical linear algebraic techniques and results in extremely efficient mapping and processing methods. This is perhaps one of the key advantages of the functional maps representation in practice, as it allows to use a new set of numerical tools to study existing problems and define novel analysis techniques (Chapter 3).

A promising area that has only recently started to be explored is the relation between the structural properties of the functional maps and the underlying shapes and their distortions. One remarkable advance in this area is the recognition of the structure of the functional representation of a correspondence in the presence of holes or missing parts. This structure can also be exploited to obtain high quality maps even in such challenging scenarios (Chapter 4). Moreover, the functional representation can be used to study intrinsic distortion induced by a given map and to define the notion of *shape differences* (Chapter 5). The shape differences, which are also defined as linear functional operators, provide a way to compare shapes in a way that is significantly more informative than standard scalar-valued metrics and indeed can be shown to fully encode the distortion and even allow shape recovery and synthesis from the given difference operators (Chapter 5, Section 5.3).

One potential challenge in using functional maps for solving matching problems is the difficulty of converting a functional map to a point-to-point map, which is often preferred in practice. Several methods have been proposed to solve this problem in practice and we take special care to review the progress in this area (Section 2.5.1 in Chapter 2 and entire Chapter 7).

Throughout the course we also try highlight not only the flexibility but also the unity of the proposed approaches in quite diverse application areas. In a way, the functional map framework can be considered as a common language in which many problems in geometry and data processing can be expressed. Two examples of this general property presented in these notes include the idea of functional vector fields (Chapter 6) and the analysis of maps in shape collections (Chapter 5). In both of these scenarios, a very important aspect is the interaction between different operations defined on the objects (vector fields and their flows, vector fields defined on different objects, network of maps defined on shape collections), which can be naturally expressed in the functional framework, by the use of linear operators represented as matrices. For example, map composition becomes matrix multiplication, whereas the operator associated with the flow of a vector field can be computed via matrix exponentiation.

Course Notes Structure and Organization The rest of the notes is organized according to topics each presented in a dedicated chapter. Chapter 2 presents the general background and definitions and a simple way to use the functional map representation for solving the shape matching problem. Chapter 3 discusses more advanced techniques for recovering functional maps by using strong geometric and linear algebraic regularizers. Chapter 4 is dedicated to describing the properties and computing functional maps across partially related shapes, while Chapter 5 gives an overview of techniques for computing better correspodences and for exploring collections using functional maps. Finally, Chapter 6 describes how tangent vector fields can also be naturally expressed as linear functional operators that interact naturally with functional maps, and Chapter 7 describes a set of methods for converting functional maps to point to point maps.

Note that the different chapters are to a large extent mutually independent (apart from the dependence on Chapter 2, and thus can be read in the order that best suits to the readers' interests. Indeed, although we made an effort to make the notation consistent across different chapters, there are nevertheless some discrepancies, and to reduce the reading difficulty to a certain extent we provided a list of key symbols at the end of each chapter.

Finally, we note that our primary goal when preparing these notes was to give an overview of the existing methods, and for this reason, in most cases we only describe the key ideas and contributions of the existing works, and mention the primary applications at the end of each chapter. We therefore invite the interested readers to use these notes to get an intuition and a sense of the scope of the current methods and to consult the original articles, referred to in each chapter, for more detailed technical descriptions and derivations.

2

What are Functional Maps?

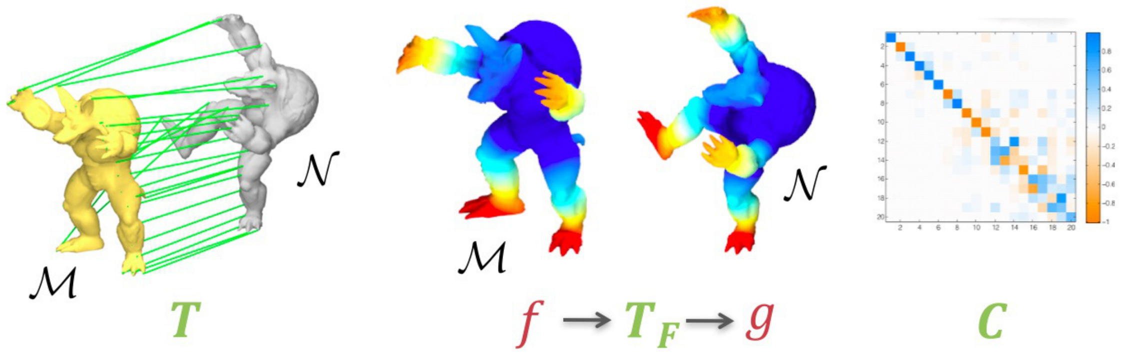


Figure 2.1: A point to point map $T : \mathcal{M} \to \mathcal{N}$ can be represented as a *functional map* T_F: a correspondence between functions $f : \mathcal{M} \to \mathbb{R}$ and functions $g : \mathcal{N} \to \mathbb{R}$. Given a choice of basis for functions on $\mathcal{M}$ and $\mathcal{N}$,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

2.1 Functional Maps in the Continuous Setting

To motivate our definitions, imagine that we start with two geometric objects such as a pair of shapes in 3D, which we will denote by $\mathcal{M}$ and $\mathcal{N}$. Now consider a correspondence or a mapping $T : \mathcal{M} \to \mathcal{N}$ between points on these shapes. In other words if p is a point on $\mathcal{M}$ then $T(p)$ is some corresponding point on $\mathcal{N}$. In practice, we are often interested in both analyzing T (for example when we consider how an object deforms) and conversely computing the optimal T for a given pair of objects, which corresponds to solving the shape matching problem.

Our first observation is that rather than thinking of a mapping as an association between points on the two shapes it is often more productive to think of it as a "transporter" that allows us to move information across the two shapes. In this context it might be useful to consider one of the shapes as a source and the other as a target of this transportation. For example, given a texture on the source, we can use T to transport it onto the target shape.

In this course we will concentrate on one particular transformation enabled by a mapping, which turns out to have extremely beneficial properties both theoretically and in practice. Namely, we will consider a transportation of real-valued functions on the two objects. To be precise, let's suppose that T is a bijection and we are given a scalar function

$f : \mathcal{M} \to \mathbb{R}$, which can represent some geometric quantity such as curvature of different points on $\mathcal{M}$, or something more abstract such as simply some information encoded as a real number $f(p)$ on p.

Using the given mapping T we can transport the function f defined on $\mathcal{M}$ to a function g defined $\mathcal{N}$ simply via composition by $g = f \circ T^{-1}$, which means that $g(p) = f(T^{-1}(p))$ for any point p on $\mathcal{M}$. We will denote by T_F the induced transformation that "transports" real-valued functions on $\mathcal{M}$ to real-valued functions on $\mathcal{N}$. We call T_F the *functional representation* of the mapping T (see Figure 2.1)[1]. We now make the following two simple remarks:

Remark 2.1.1. *The original mapping T can be recovered from T_F.*

Indeed, to recover the image $T(a)$ of any point a on $\mathcal{M}$, construct an indicator function $f : \mathcal{M} \to \mathbb{R}$, s.t. $f(a) = 1$ and $f(x) = 0 \; \forall \; x \neq a \in \mathcal{M}$. 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 $y \in \mathcal{N}$ s.t. $g(y) = 1$.

Remark 2.1.2. *For any fixed bijective map $T : \mathcal{M} \to \mathcal{N}$, 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.

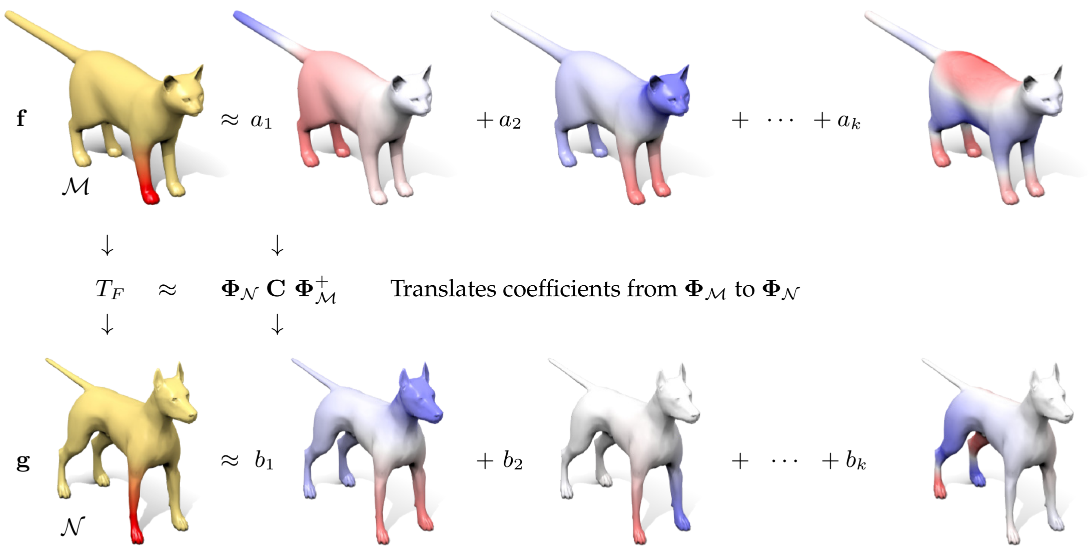
It is hard to overestimate the consequences of these two simple observations, and a large portion of our course will be dedicated to exploring them in full detail. Perhaps the simplest way to appreciate the utility of the functional representation T_F is to note that linear transformations can be represented often be encoded as matrices and there is an intimate relation between different linear algebraic operations (solving linear systems, computing matrix inverses and spectral decompositions, and even matrix exponentiation) turn out all to be very closely related to many operations in manipulating mappings or correspondences between shapes, which have traditionally been considered difficult or cumbersome in practice (matching between non-rigid shapes, inverting and composing different mappings, computing distortion induced by a map or computing flows of tangent vector fields). We will go through each of these constructions (and many more!) and explore their theoretical and computational utility, as well as suggest further possible extensions.

2.2 Functional Maps in a Basis

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

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

[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 $\mathcal{N}$ to $\mathcal{M}$. 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.



$\mathbf{f}$
$\mathcal{M}$
$\approx a_1$
$+ a_2$
$+ \cdots + a_k$
$T_F \approx \Phi_{\mathcal{N}}\, \mathbf{C}\, \Phi^+_{\mathcal{M}}$
Translates coefficients from $\Phi_{\mathcal{M}}$ to $\Phi_{\mathcal{N}}$
$\mathbf{g}$
$\mathcal{N}$
$\approx b_1$
$+ b_2$
$+ \cdots + b_k$

Figure 2.2: When the functional spaces of source and target shapes $\mathcal{M}$ and $\mathcal{N}$ are endowed with bases $\Phi_{\mathcal{M}}$ and thus every function can be written as a linear combination of basis functions, then a linear functional map T_F can be expressed via a matrix $\mathbf{C}$ that intuitively "translates" the coefficients from one basis to another.

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

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

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: $b_j = \sum_i c_{ji}a_i$, 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 $T_F(\phi_i^{\mathcal{M}})$ in the basis $\{\phi_j^{\mathcal{N}}\}$. Note that the matrix $\mathbf{C}$ has a particularly simple representation if the basis functions $\{\phi_i^{\mathcal{N}}\}$ are orthonormal with respect to some inner product $\langle\cdot,\cdot\rangle$, namely $c_{ji} = \langle T_F(\phi_i^{\mathcal{M}}), \phi_j^{\mathcal{N}}\rangle$.

We conclude with the following key observation (See Figure 2.2 for illustration):

Remark 2.2.1. *The map T_F can be represented as a (possibly infinite) matrix* $\mathbf{C}$ *s.t. for any function f represented as a vector of coefficients* $\mathbf{a}$ *we have* $T_F(\mathbf{a}) = \mathbf{C}\mathbf{a}$.

This remark in combination with the previous two remarks shows that the matrix $\mathbf{C}$ fully encodes the original map T.

Functional Representation of Given Map Suppose we are given a map T between two discrete objects $\mathcal{N}$ and $\mathcal{M}$, represented by a collection of n and m vertices respectively. We can represent such a map by a matrix $\mathbf{T}$ of size $m \times n$. In general, when T is not necessarily a bijection, the functional map can only be well defined to transport real-valued functions on $\mathcal{M}$ to their corresponding functions on $\mathcal{N}$ (i.e., in the direction oposite to that of the original map). In the full basis, the functional map associates a real-valued function f on $\mathcal{M}$ to the function $\mathbf{T}^\top f$ on $\mathcal{N}$. If P is a bijection then $\mathbf{T}^{-1} = \mathbf{T}^\top$ and $(\mathbf{T}^\top)^{-1} = \mathbf{T}$, so in this case we can define the functional map between $\mathcal{N}$ and $\mathcal{M}$ (i.e., in the same direction as the map itself) using $f \to \mathbf{T}f$.

When the objects are equipped with a reduced set of basis functions stored as columns of matrices $\mathbf{\Phi}_{\mathcal{N}}$ and $\mathbf{\Phi}_{\mathcal{M}}$ respectively, then the corresponding functional map in the reduced basis can be found by solving the linear system of equations:

$$\mathbf{\Phi}_{\mathcal{N}}\mathbf{C} = \mathbf{T}^{\top}\mathbf{\Phi}_{\mathcal{M}}, \text{ which leads to:}$$

$$\mathbf{C} = \mathbf{\Phi}_{\mathcal{N}}^{\top}\mathbf{T}^{\top}\mathbf{\Phi}_{\mathcal{M}}, \text{ if } \mathbf{\Phi}_{\mathcal{N}}^{\top}\mathbf{\Phi} = \mathbf{I}, \text{ or } \mathbf{C} = \mathbf{\Phi}_{\mathcal{N}}^{\top}\mathbf{A}_{\mathcal{N}}\mathbf{T}^{\top}\mathbf{\Phi}_{\mathcal{M}}, \text{ if } \mathbf{\Phi}_{\mathcal{N}}^{\top}\mathbf{A}_{\mathcal{N}}\mathbf{\Phi} = \mathbf{I},$$

The latter case is especially common in geometry processing, where $\mathbf{A}_{\mathcal{N}}$ is often taken to be a diagonal matrix of area weights of each vertex, and the basis functions are computed to be orthonormal with respect to this set of weights (e.g., [Rus07]).

2.3 General 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 $\mathbf{C}$. We thus define:

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

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

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

Example. As an example, consider a pair of shapes in Figure 2.3 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 $\mathbf{C}_{0..20\times0..20}$, 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].

2.4 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 that maps every function on one shape to the constant zero function on the other. Such a map clearly cannot be

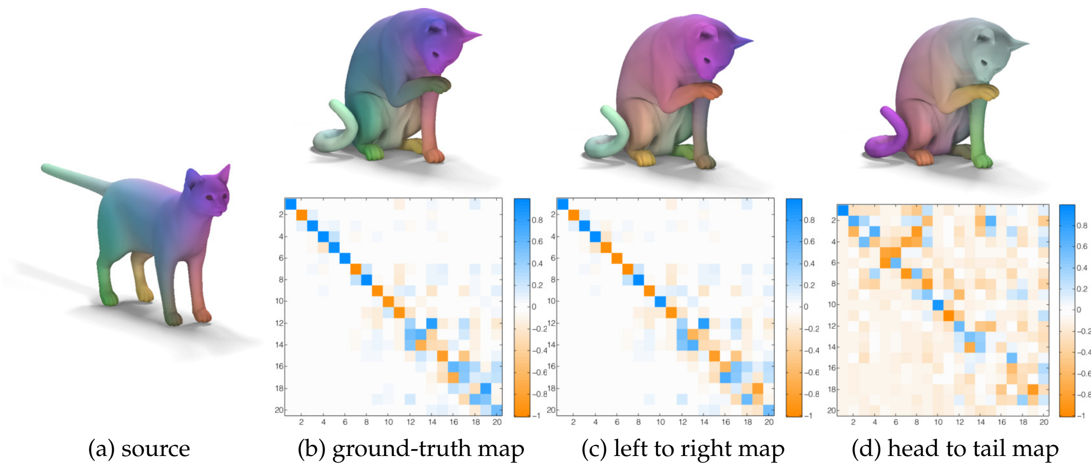


Figure 2.3: Two shapes with three maps between them, each rendered as a point-to-point mapping through color correspondence (top) and its functional representation (bottom) with colors proportional to matrix values. Note that the least isometric map in (d) leads to a more dense matrix.

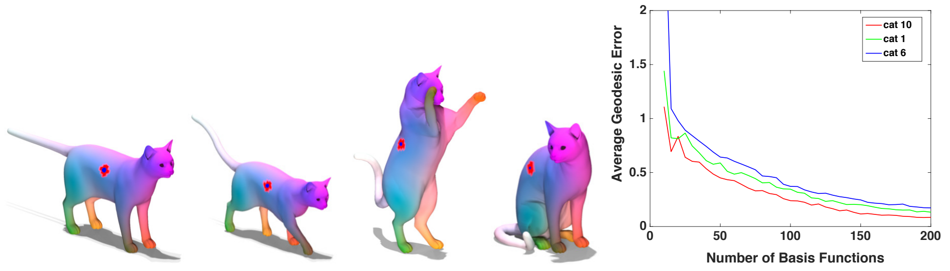
associated with any pointwise correspondences since all such functional maps must, by definition, preserve the set of values of each function. Nevertheless, by going to this richer space of correspondences, we obtain a representation that has several key properties making it more suitable for manipulation 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 2.4.1) and then by showing how many natural constraints on the map can be phrased as linear constraints on the functional map (Section 2.4.2), reducing shape matching to a moderately-sized system of linear equations (Section 2.5).

2.4.1 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 $\mathcal{M}$ and $\mathcal{N}$. Indeed, if we choose the basis functions to be indicator functions at the vertices of the shapes, then $\mathbf{C}$ 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 functions and we need only consider a finite *submatrix* $\mathbf{C}_{0..m\times 0...n}$, for some moder-



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

Figure 2.4: Average error vs. number of basis functions used in the representation. For each shape with a known ground-truth pointwise map, shown as a color correspondence, we computed its functional representation and measured its accuracy in reconstructing the original map. A geodesic disk with radius 1 is shown on each shape for scale.

ate values of m and n, of the infinite matrix $\mathbf{C}$ (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 $\sum_j \sum_i a_i c_{ji} \phi_j^{\mathcal{N}} \approx \sum_{j=0}^{n} \sum_{i=0}^{m} a_i c_{ji} \phi_j^{\mathcal{N}}$, for some fixed small values of m and n.

In the discussion below, 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 2.4 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 2.5.1 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 only $30 - 40$ eigenfunctions are sufficient to represent the ground-truth point-to-point map well.

2.4.2 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 $\mathbf{C}$) 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 $f : \mathcal{M} \to \mathbb{R}$ and $g : \mathcal{N} \to \mathbb{R}$, the correspondence between f and g can be written simply as $\mathbf{Ca} = \mathbf{b}$ where $\mathbf{C}$ 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 $\mathcal{M}$ and $\mathcal{N}$. Note that the function preservation constraint can be phrased entirely in terms of the matrix $\mathbf{C}$ without knowledge of the underlying correspondence T, since $\mathbf{a}$ and $\mathbf{b}$ do not depend on the map $\mathbf{C}$. This is especially useful for shape matching applications where $\mathbf{C}$ is unknown, but could possibly be recovered by phrasing enough constraints of type $\mathbf{Ca} = \mathbf{b}$. 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 $\mathcal{M}$ 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 $x \in \mathcal{M}$ and $y \in \mathcal{N}$ (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.

To summarize, given pair of shapes $\mathcal{M}$ and $\mathcal{N}$ we can often compute a set of pairs of functions f_i, g_i, such that the unknown functional map should satisfy $\mathbf{C}\mathbf{a}_i \approx \mathbf{b}_i$, where $\mathbf{a}_i, \mathbf{b}_i$ are vectors of coefficients representing f_i, g_i in a given basis. By storing these functions as columns of matrices $\mathbf{A}$ and $\mathbf{B}$ respectively (where each column represents the coefficients of a single function expressed in the basis of the corresponding shape), we expect the following energy to have small error:

$$E_1(\mathbf{C}) = \|\mathbf{CA} - \mathbf{B}\|^2.$$

Note that in principle, given enough corresponding functions f_i and g_i, it should be possible to recover $\mathbf{C}$ by solving a least squares system.

2.4.3 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 $\mathcal{M}$ and $\mathcal{N}$. That is, often $\mathcal{M}$ and $\mathcal{N}$ can be endowed with linear functional operators that we may want to preserve. A first example is a symmetry operator $S_F : \mathcal{F}(\mathcal{M}, \mathbb{R}) \to \mathcal{F}(\mathcal{M}, \mathbb{R})$ which associates with every function $f : \mathcal{M} \to \mathbb{R}$ another function $S_F(f) : \mathcal{M} \to \mathbb{R}$ obtained as $S_F(f)(x) = f(S^{-1}(x))$, where $S : \mathcal{M} \to M$ is some symmetry of $\mathcal{M}$. A second example is the Laplace-Beltrami operator and derived operators (e.g. the heat operator), which are preserved under isometries. The operators on $\mathcal{M}$ and $\mathcal{N}$ can be quite general, however, and can represent any association of functions on the manifold. In any case, given functional operators $S_F^{\mathcal{M}}$ and $S_F^{\mathcal{N}}$ on $\mathcal{M}$ and $\mathcal{N}$ respectively, it may be natural to require that the functional map $\mathbf{C}$ commute with these operators. In particular: $S_F^{\mathcal{N}} \circ T_F = T_F \circ S_F^{\mathcal{M}}$, or when all operators are written in a given basis then we obtain in matrix notation, $\|\mathbf{S}_F^{\mathcal{N}}\mathbf{C} - \mathbf{C}\mathbf{S}_F^{\mathcal{M}}\| = 0$. Note that this constraint can be encoded using the following energy:

$$E_2(\mathbf{C}) = \|\mathbf{S}_F^{\mathcal{N}}\mathbf{C} - \mathbf{C}\mathbf{S}_F^{\mathcal{M}}\|^2.$$

Similarly to the E_1 the optimal $\mathbf{C}$ that minimizes $E_2(\mathbf{C})$ can be recovered by solving a linear system of equations. Note that when $\mathbf{C}$ is expressed in the basis given by the first k eigenfunctions of the Laplace-Beltrami operator and $S_F^{\mathcal{M}}$, $S_F^{\mathcal{N}}$ correspond to the LB operators $\Delta^{\mathcal{M}}, \Delta^{\mathcal{N}}$ on $\mathcal{M}$ and $\mathcal{N}$ respectively then $E_2(\mathbf{C})$ has a particularly simple expression

$$E_2(\mathbf{C}) = \sum_{i,j} \mathbf{C}_{ij}^2 (\lambda_i^{\mathcal{N}} - \lambda_j^{\mathcal{M}})^2,$$

where $\lambda^{\mathcal{M}}$ and $\lambda^{\mathcal{N}}$ are the eigenvalues of the corresponding operators. Of course, as mentioned above $E_2(\mathbf{C})$ is quadratic and thus the optimal matrix $\mathbf{C}$ can be found by solving a linear squares system of equations. Note, however, that E_2 alone does not provide enough information to recover $\mathbf{C}$ since the trivial solution $\mathbf{C} = 0$ would result in zero error.

2.4.4 Estimating Functional Maps

In practice, the simplest method for recovering an unknown functional map, (in the basis given by the first k first eigenfunctions of the LB operator) between a pair of shapes is to solve the following optimization problem:

$$C = \arg\min_{\mathbf{X}} E_1(\mathbf{X}) + E_2(\mathbf{X}) = \|\mathbf{X}\mathbf{A} - \mathbf{B}\|^2 + \alpha\|\mathbf{\Lambda}^{\mathcal{N}}\mathbf{X} - \mathbf{X}\mathbf{\Lambda}^{\mathcal{M}}\|^2, \tag{2.2}$$

where $\mathbf{A}$ and $\mathbf{B}$ are the function preservation constraints expressed in the basis of the eigenfunctions of the LB operator, (stored as columns of coefficients in the matrices), $\mathbf{\Lambda}^{\mathcal{M}}$ and $\mathbf{\Lambda}^{\mathcal{N}}$ are diagonal matrices of eigenvalues of Laplacian operators and α is a scalar weight parameter. When the shapes are approximately isometric and the descriptors are well-preserved by the (unknown) map, then this procedure already gives a good approximation of the underlying map. However, as we show below, several regularizations and extensions can help to improve this estimation significantly.

2.4.5 Regularization Constraints

Note that although we mentioned in Section 2.3 that there are no inherent constraints on the matrix $\mathbf{C}$ to be a functional map, this does not mean that *any* matrix $\mathbf{C}$ is associated with a point-to-point map. Indeed, while every bijective map T has a functional representation through the matrix $\mathbf{C}$, 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 2.4.1. *(1) If the underlying map T (discrete or continuous) is volume preserving, i.e. $\mu^{\mathcal{M}}(x) = \mu^{\mathcal{N}}(T(x))$ where $\mu^{\mathcal{M}}$ and $\mu^{\mathcal{N}}$ are volume elements on $\mathcal{M}$ and $\mathcal{N}$ respectively, then the matrix $\mathbf{C}$ associated with the functional representation of T must be* orthonormal. *(2) If the underlying map T is an isometry then the corresponding functional map commutes with the Laplace-Beltrami operator.*

In matrix notation, the first result states that if the underlying point to point map is locally volume preserving then $\mathbf{C}^{\top}\mathbf{C} = \mathbf{I}$, and if it is an isometry then $\mathbf{C}\mathbf{\Lambda}^{\mathcal{M}} = \mathbf{\Lambda}^{\mathcal{N}}\mathbf{C}$. It turns out that when considered in the full basis (or as operators in the continuous case), the converse both of these conditions also holds (e.g. [ROA+13]). Thus, enforcing these constraints provides a very strong regularization on the computed map.

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.

2.4.6 Map Inversion and Composition

A challenging task when considering point-to-point mappings between shapes is map inversion, i.e. given a map $T : \mathcal{M} \rightarrow \mathcal{N}$ that is not necessarily bijective, one is required to find a meaningful version of $T^{-1} : \mathcal{N} \rightarrow \mathcal{M}$. In the functional representation finding an inverse can be done simply by finding an inverse of the mapping matrix $\mathbf{C}$. Moreover, because for near-isometric maps we expect this matrix to be close to diagonal it is reasonable to take the inverse of the approximating submatrix of $\mathbf{C}$. Finally, in light of Theorem 2.4.1 this can be done by simply taking the transpose of $\mathbf{C}$ 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].

2.5 Functional Map Inference

As mentioned in Section 2.4, 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 $\mathbf{C}$. 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

```
Algorithm 1: FUNCTIONAL MAP INFERENCE FOR MATCHING
  1. Compute a set of descriptors for each point on $\mathcal{M}$ and $\mathcal{N}$, 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 $\mathcal{M}$
     and $\mathcal{N}$ (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 $\mathbf{C}$.
  5. Refine the initial solution $\mathbf{C}$ with the iterative method of Section 2.5.2.
  6. If required, compute point correspondences using the method in Section 2.5.1.
```

the best functional map by finding the matrix $\mathbf{C}$ that best satisfies the constraints in the least squares sense.

2.5.1 Efficient Conversion to Point-to-Point

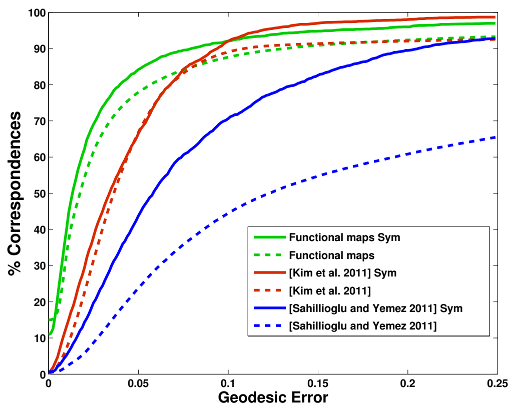
As mentioned in Section 2.2 given a bijection T between two discrete shapes, and the basis vectors of their function spaces, the functional representation $\mathbf{C}$ of the map T can be obtained by solving a linear system.

To reconstruct the original mapping from the functional representation, however, is more challenging. The simplest method alluded to in Remark 2.1.1 to find a corresponding point $y \in N$ to a given point $x \in M$ would require constructing a function $f : \mathcal{M} \to \mathbb{R}$ (either the indicator function, or a highly peaked Gaussian around x) obtaining its image $g = T_F(f)$ using $\mathbf{C}$ and declaring y to be the point at which $g(y)$ obtains the maximum. Such a method, however, would require $O(V_\mathcal{N} V_\mathcal{M})$ operations for a pair of meshes with $V_\mathcal{N}$ and $V_\mathcal{M}$ 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 δ_x, the delta function around a point $x \in M$, has the coefficients: $a_i = \phi_i^\mathcal{M}(x)$. This can be seen for example, since $\delta_x = \lim_{t\to0^+} k_t^\mathcal{M}(x,\cdot) = \lim_{t\to0^+} \sum_{i=0}^{\infty} e^{-t\lambda_i}\phi_i^\mathcal{M}(x)\phi_i^\mathcal{M}(\cdot)$, where $k_t^\mathcal{M}(\cdot,\cdot)$ is the heat kernel at time t on the shape $\mathcal{M}$.

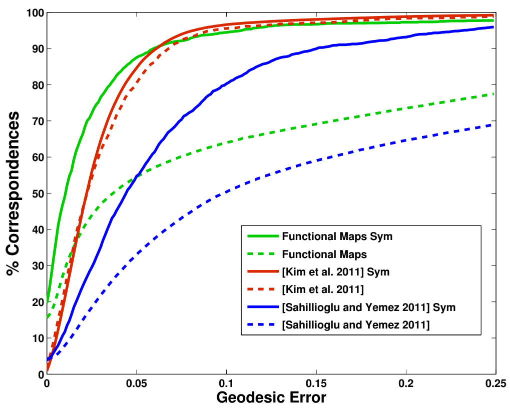
Therefore, given a matrix $\mathbf{\Phi}_\mathcal{M}$ of the Laplace-Beltrami eigenfunctions of $\mathcal{M}$, where each column corresponds to a point and each row to an eigenfunction, one can find the image of *all* of the delta functions centered at points of $\mathcal{M}$ simply as $\mathbf{C\Phi}_\mathcal{M}$.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

2.5.2 Post-Processing Iterative Refinement

The observation made in Section 2.5.1 can also be used to refine a given matrix $\mathbf{C}$ to make it closer to a point-to-point map. Suppose we are given an initial estimate matrix $\mathbf{C}_0$ that



(a) SCAPE



(b) TOSCA

Figure 2.5: 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.

we believe comes from a point-to-point map T. As noted in Section 2.5.1, theoretically $\mathbf{C}_0$ must be such that each column of $\mathbf{C}_0\mathbf{\Phi}_\mathcal{M}$ coincides with some column of $\mathbf{\Phi}_\mathcal{N}$. If we treat $\mathbf{\Phi}_\mathcal{M}$ and $\mathbf{\Phi}_\mathcal{N}$ as two point clouds with dimensionality equal to the number of eigenvalues used in the functional representation $\mathbf{C}_0$ then this means that $\mathbf{C}_0$ must *align* $\mathbf{\Phi}_\mathcal{M}$ and $\mathbf{\Phi}_\mathcal{N}$. Moreover, since by Theorem 2.4.1 we expect the mapping matrix $\mathbf{C}_0$ to be orthonormal, we can phrase the problem of finding the optimal mapping matrix $\mathbf{C}$ as rigid alignment between $\mathbf{\Phi}_\mathcal{M}$ and $\mathbf{\Phi}_\mathcal{N}$. Thus an iterative refinement of $\mathbf{C}_0$ can be obtained via:

1. For each column $\mathbf{x}$ of $\mathbf{C}_0\mathbf{\Phi}_\mathcal{M}$ find the closest $\tilde{\mathbf{x}}$ in $\mathbf{\Phi}_\mathcal{N}$.
2. Find the optimal orthonormal $\mathbf{C}$ minimizing $\sum \|\mathbf{C} - \tilde{\mathbf{x}}\|$.
3. Set $\mathbf{C}_0 = \mathbf{C}$ 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 $\mathbf{C}$ because the embedding $\mathbf{\Phi}_\mathcal{M}$ and $\mathbf{\Phi}_\mathcal{N}$ 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 $\mathbf{C}_0$. Finally, note that the output of this procedure is not only a functional matrix $\mathbf{C}$ but also a point-to-point correspondence given by nearest neighbor assignment between points on $\mathcal{M}$ and $\mathcal{N}$. 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 2.6.

2.6 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 $\mathcal{M}$ and $\mathcal{N}$ in their discrete (e.g. mesh) representation, and the Laplace-Beltrami eigen-decomposition. 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 2.5.2 and compute the point-to-point map using the method in Section 2.5.1.

2.6.1 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 $\mathcal{M}$ and $\mathcal{N}$ 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 $\mathbf{C}$. Finally, we refine the solution using the iterative method described in Section 2.5.2 and find the final dense point-to-point correspondences using the method in 2.5.1.

2.6.2 Results

In this section we present an evaluation of the 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 2.5 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 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.

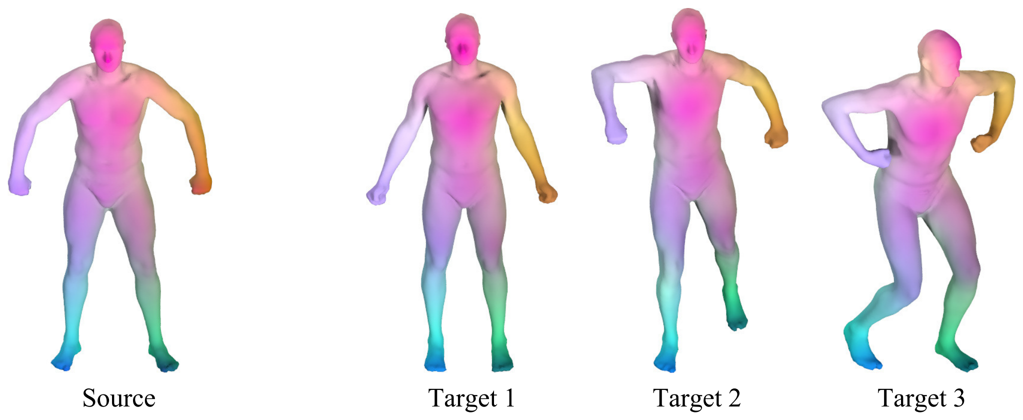


Figure 2.6: Maps between remeshed versions of shapes from the SCAPE collection, mapping the coordinate functions from the source to the three target shapes using an inferred functional map.

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 2.6 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.

2.7 Other Applications

2.7.1 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 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 2.7 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.

Other Extensions and Applications In the original article [OBCS$^+$12], it was also shown that the functional map representation can be used to improve the accuracy of other shape matching methods, and also to improve correspondences in the context of shape

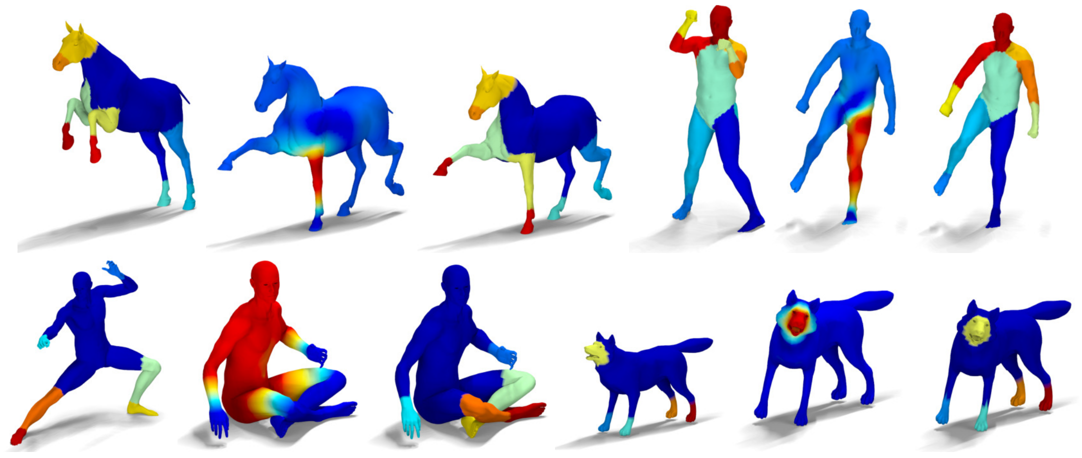


Figure 2.7: Segmentation transfer using the functional map representation. For each pair of shapes we show 3 figures: the user-provided source segmentation of the first shape, the image of one of the indicator functions of a segment using the functional map computed with our method, and the final segmentation transfer onto the target shape. Note that point correspondences were not computed at any point during this procedure.

collections, by using map composition. Furthermore, since then, a large number of extensions have appeared some of which are outlined in the following chapters.

2.8 List of Key Symbols

Symbol	Definition
$\mathcal{M}, \mathcal{N}$	Shapes (in most cases assumed to be either smooth surfaces, or manifold meshes).
$\mathcal{F}(\mathcal{M}, \mathbb{R})$	Space of real-valued functions on shape $\mathcal{M}$
T_F	Functional representation of a given pointwise map T
$\mathbf{C}$	Functional map expressed as a matrix in a given basis.
Δ	Laplace-Beltrami operator on a surface
$\mathbf{\Lambda}$	diagonal matrix of eigenvalues of the mesh Laplacian
$\mathbf{\Phi}$	Functional basis (matrix containing basis functions as columns)
$A_{\mathcal{M}}$	Diagonal matrix of area weights on shape $\mathcal{M}$.
F, G	Function preservation constraints (each column corresponding to a function).
$\mathbf{A}, \mathbf{B}$	Function preservation constraints represented as matrices in a given basis.

3

Computing Functional Maps

In this Chapter, we mainly focus on various formulations of the functional correspondence problem and optimization techniques for solving such problems.

3.1 Joint diagonalization

Let $\mathcal{M}$ and $\mathcal{N}$ be our two shapes with Laplacian eigenbases $\{\phi_i^{\mathcal{M}}\}$ and $\{\phi_i^{\mathcal{N}}\}$ respectively, and let $T : \mathcal{M} \to \mathcal{N}$ be a map between them. In case the map in question is an approximate isometry and in the absence of repeated eigenvalues, the eigenfunctions are defined up to a sign flip, $\phi_i^{\mathcal{M}} = \pm\phi_i^{\mathcal{N}} \circ T^{-1}$. However, in the more general case where the shapes are not isometric (e.g., elephant and horse), the behavior of their eigenspaces can differ dramatically. This poses severe limitations on many applications such as pose transfer or shape retrieval, thus limiting the usefulness of the functional representation by a large margin. For near-isometric shapes, since $\phi_i^{\mathcal{N}} \approx \phi_i^{\mathcal{M}} \circ T^{-1}$, the coefficients $c_{ij} \approx \pm\delta_{ij}$, and thus the matrix $\mathbf{C}$ is nearly diagonal (Figure 3.1, left). However, when trying to express correspondence between non-isometric shapes, the Laplacian eigenfunctions manifest a very different behavior thus breaking this diagonality (Figure 3.1, right).

3.1.1 Coupled bases

A powerful approach to tackle this drawback was formulated in [KBB+12, EKB+15, LRBB17] as a joint diagonalization problem. The main idea is to find a pair of new bases in which the correspondence matrix has a near-diagonal structure (see Figure 3.1, third row). A pair of new orthogonal bases $\{\hat{\phi}_i^{\mathcal{M}}, \hat{\phi}_i^{\mathcal{N}}\}_{i=1}^{k}$ is constructed as a linear combination of $k' \geq k$ standard Laplacian eigenfunctions $\phi_i^{\mathcal{M}}, \phi_i^{\mathcal{N}}$,

$$\hat{\phi}_i^{\mathcal{M}} = \sum_{j=1}^{k'} p_{ji}\phi_j^{\mathcal{M}}, \qquad \hat{\phi}_i^{\mathcal{N}} = \sum_{j=1}^{k'} q_{ji}\phi_j^{\mathcal{N}}, \tag{3.1}$$

where $\mathbf{P}, \mathbf{Q}$ denote the $k' \times k$ matrices of linear combination coefficients. The orthogonality of the new bases $\langle \hat{\phi}_i^{\mathcal{M}}, \hat{\phi}_j^{\mathcal{M}} \rangle_{L^2(\mathcal{M})} = \delta_{ij}$ and $\langle \hat{\phi}_i^{\mathcal{N}}, \hat{\phi}_j^{\mathcal{N}} \rangle_{L^2(\mathcal{N})} = \delta_{ij}$ implies the orthogonality of the matrices $\mathbf{P}^\top\mathbf{P} = \mathbf{I}$ and $\mathbf{Q}^\top\mathbf{Q} = \mathbf{I}$. The orthogonal basis $\{\hat{\phi}_i^{\mathcal{M}}\}$ behaves as eigenfunctions of the Laplacian $\Delta_{\mathcal{M}}$ if it minimizes the *Dirichlet energy* that can be written as

$$\sum_{i=1}^{k} \int_{\mathcal{M}} \|\nabla \hat{\phi}_i^{\mathcal{M}}(x)\|^2 dx = \sum_{i=1}^{k} \int_{\mathcal{M}} \hat{\phi}_i^{\mathcal{M}}(x) \Delta_{\mathcal{M}} \hat{\phi}_i^{\mathcal{M}}(x) dx = \mathrm{trace}(\langle \hat{\phi}_i, \Delta_{\mathcal{M}} \hat{\phi}_j \rangle_{L^2(\mathcal{M})}).$$

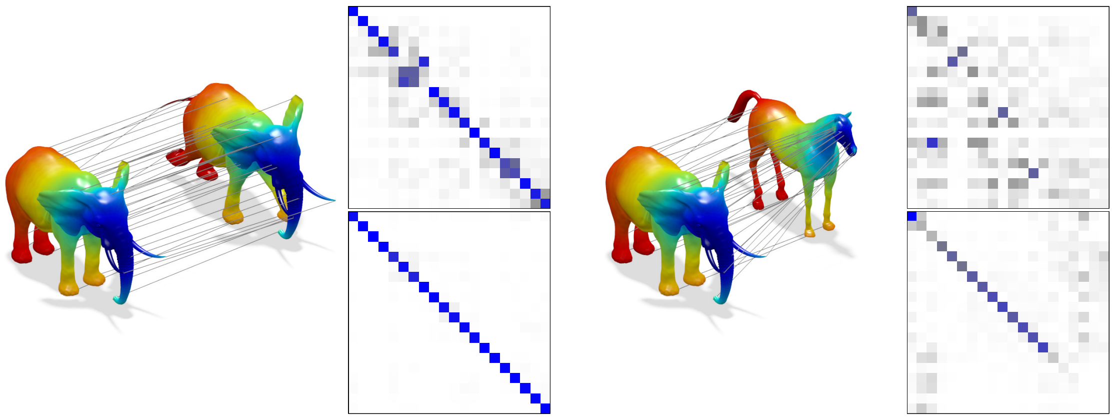


Figure 3.1: Matrix $\mathbf{C}$ representing the functional correspondence between two near-isometric shapes (left) and non-isometric shapes (middle right) expressed in the Laplacian eigenbases (top row) and coupled bases obtained by the joint diagonalization procedure (bottom row).

Since the eigenfunctions diagonalize the Laplacian, $\langle \phi_i^{\mathcal{M}}, \Delta_{\mathcal{M}} \phi_j^{\mathcal{M}} \rangle_{L^2(\mathcal{M})} = \lambda_i^{\mathcal{M}} \delta_{ij}$, it is easy to express the Dirichlet energy as

$$\mathrm{trace}(\langle \hat{\phi}_i^{\mathcal{M}}, \Delta_{\mathcal{M}} \hat{\phi}_j^{\mathcal{M}} \rangle_{L^2(\mathcal{M})}) = \mathrm{trace}(\mathbf{P}^\top \mathbf{\Lambda}_{\mathcal{M}} \mathbf{P}),$$

where $\mathbf{\Lambda}_{\mathcal{M}} = \mathrm{diag}(\lambda_1^{\mathcal{M}}, \dots, \lambda_{k'}^{\mathcal{M}})$ is the diagonal matrix of the first k' eigenvalues of $\Delta_{\mathcal{M}}$.

Let us be given q corresponding functions $g_i \approx f_i \circ T^{-1}$, $i = 1, \dots, q$ and let $\mathbf{A} = (\langle \phi_i^{\mathcal{M}}, f_j \rangle_{L^2(\mathcal{M})})$ and $\mathbf{B} = (\langle \phi_i^{\mathcal{N}}, g_j \rangle_{L^2(\mathcal{N})})$ be the $k' \times q$ matrices of coefficients of the given corresponding functions in the standard Laplacian eigenbases. It is easy to see that the coefficients of $\{f_i, g_i\}$ in the new bases can be expressed as $\hat{\mathbf{A}} = \mathbf{P}^\top \mathbf{A}$ and $\hat{\mathbf{B}} = \mathbf{Q}^\top \mathbf{B}$. Our goal is to find $\mathbf{P}, \mathbf{Q}$ resulting in $\{\hat{\phi}_i^{\mathcal{M}}, \hat{\phi}_i^{\mathcal{N}}\}$ that behave approximately as eigenfunctions, while being coupled in the sense $\hat{\mathbf{A}} \approx \hat{\mathbf{B}}$. This is possible by solving the optimization problem

$$\min_{\mathbf{P},\mathbf{Q}} \quad \mathrm{trace}(\mathbf{P}^\top \mathbf{\Lambda}_X \mathbf{P}) + \mathrm{trace}(\mathbf{Q}^\top \mathbf{\Lambda}_Y \mathbf{Q}) + \mu \|\mathbf{P}^\top \mathbf{A} - \mathbf{Q}^\top \mathbf{B}\| \qquad \text{s.t.} \;\; \mathbf{P}^\top \mathbf{P} = \mathbf{I}, \mathbf{Q}^\top \mathbf{Q} = \mathbf{I}. \tag{3.2}$$

Problem (3.2) can be interpreted as a *joint diagonalization* of the Laplacians $\Delta_{\mathcal{M}}$ and $\Delta_{\mathcal{N}}$ [KBB+12, EKB+15]. Its solutions, referred to as *coupled quasi-harmonic bases*, are shown in Figure 3.2. Note that by virtue of parametrization of the basis function the procedure can be applied to any shape representation, for example, meshes and point clouds. This approach was extended to the partial setting in [LRBB17].

3.2 Manifold optimization

Problems with orthogonality constraints like the one arising in joint diagonalization can be efficiently solved by *manifold optimization*, realizing that the feasible set is a Riemannian sub-manifold of the Euclidean space of matrices (in this particular case, optimization in (3.2) is performed over the product of two *Stiefel manifolds* of orthogonal matrices, $\mathcal{X} = \{\mathbf{X} : \mathbf{X}^\top \mathbf{X} = \mathbf{I}\}$). The main idea of manifold optimization is to treat the objective as a function defined on the matrix manifold, and perform descent on the manifold itself rather than in the ambient Euclidean space. A conceptual gradient descent-like manifold

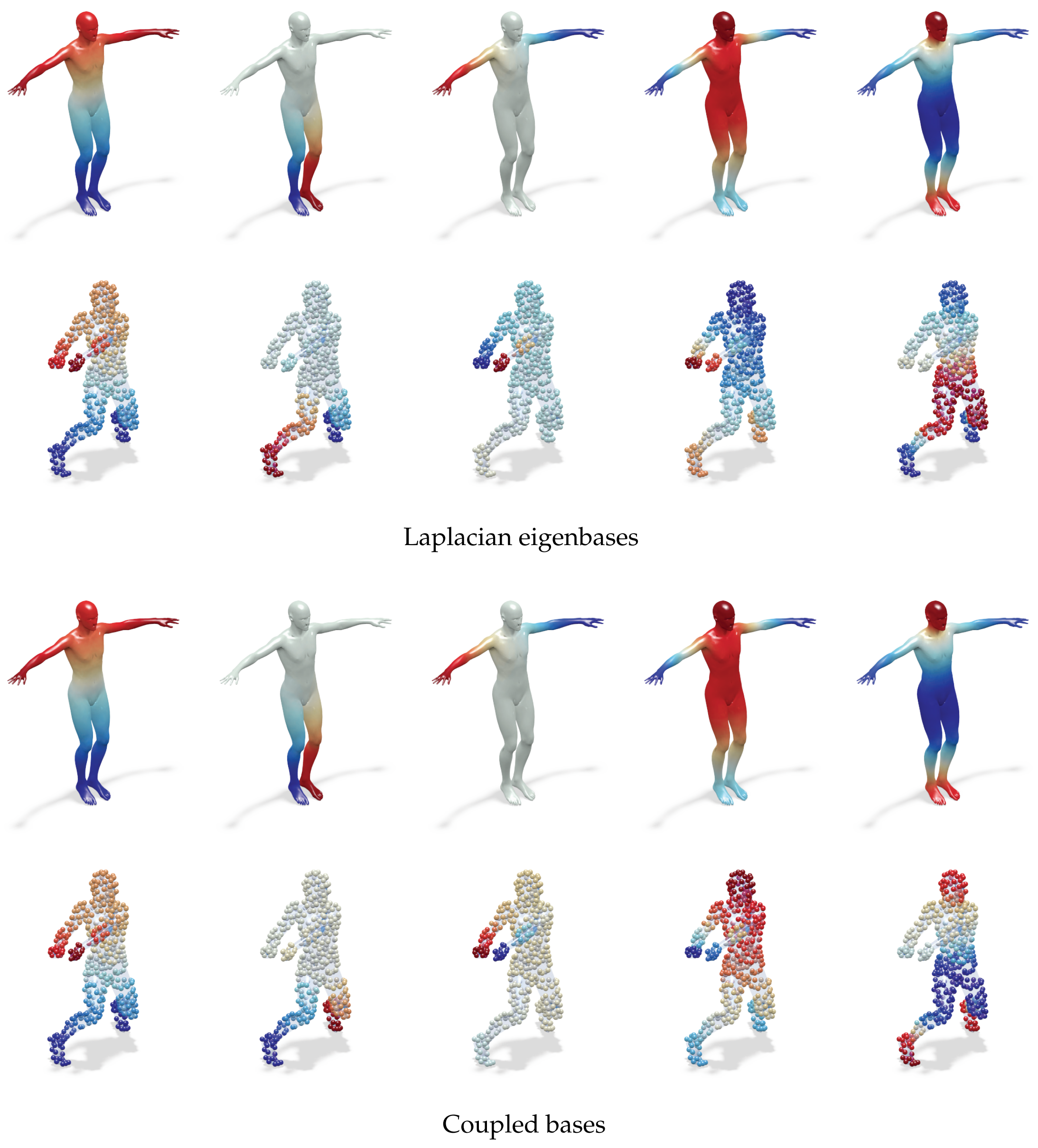


Figure 3.2: Examples of Laplacian eigenbases (top) of two different representations of a human shape (mesh and point clouds) and coupled bases (bottom).

optimization is presented in Algorithm 2. For a comprehensive introduction to manifold optimization, the reader is referred to [AMS09].

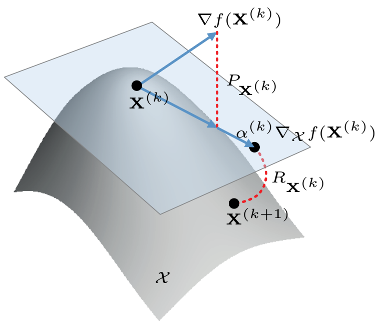
3.2.1 Manifold ADMM

When dealing with non-smooth manifold-constrained optimization problems (for example, when using a robust norm as the data term in the joint diagonalization problem), the *Manifold ADMM* (MADMM) technique introduced in [KGB16] can be used. Given a problem of the form

$$\min_{\mathbf{X} \in \mathcal{X} \subseteq \mathbb{R}^{m \times n}} f(\mathbf{X}) + g(\mathbf{A}\mathbf{X}), \tag{3.3}$$

```
1 repeat
2     Compute the extrinsic gradient $\nabla f(\mathbf{X}^{(k)})$
3     Projection: $\nabla_{\mathcal{X}} f(\mathbf{X}^{(k)}) = P_{\mathbf{X}^{(k)}}(\nabla f(\mathbf{X}^{(k)}))$
4     Compute the step size $\alpha^{(k)}$ along the descent
        direction
5     Retraction: $\mathbf{X}^{(k+1)} = R_{\mathbf{X}^{(k)}}(-\alpha^{(k)} \nabla_{\mathcal{X}} f(\mathbf{X}^{(k)}))$
6 until convergence;
```

Algorithm 2: Smooth optimization on matrix manifold $\mathcal{X}$.



where f and g are smooth and non-smooth real-valued functions, respectively, $\mathbf{A}$ is a fixed $k \times m$ matrix, and $\mathcal{X}$ is a matrix manifold, Algorithm 2 cannot be used directly because of non-smoothness of the objective function.

The key idea of MADMM is that problem (3.3) can be equivalently formulated as

$$\min_{\mathbf{X} \in \mathcal{X}, \mathbf{Z} \in \mathbb{R}^{k \times n}} f(\mathbf{X}) + g(\mathbf{Z}) \quad \text{s.t.} \quad \mathbf{Z} = \mathbf{A}\mathbf{X} \tag{3.4}$$

by introducing an artificial variable $\mathbf{Z}$ and a linear constraint. The method of multipliers applied to only the linear constraints in (3.4), leads to the minimization problem

$$\min_{\mathbf{X} \in \mathcal{X}, \mathbf{Z} \in \mathbb{R}^{k \times n}} f(\mathbf{X}) + g(\mathbf{Z}) + \tfrac{\rho}{2} \|\mathbf{A}\mathbf{X} - \mathbf{Z} + \mathbf{U}\|_{\mathrm{F}}^2 \tag{3.5}$$

where $\rho > 0$ and $\mathbf{U} \in \mathbb{R}^{k \times n}$ have to be chosen and updated appropriately (see below). This formulation now allows splitting the problem into two optimization sub-problems w.r.t. to $\mathbf{X}$ and $\mathbf{Z}$, which are solved in an alternating manner, followed by an updating of $\mathbf{U}$. Observe that in the first sub-problem w.r.t. $\mathbf{X}$ we minimize a *smooth* function with manifold constraints, and in the second sub-problem w.r.t. $\mathbf{Z}$ we minimize a non-smooth function without manifold constraints. Thus, the problem breaks down into two well-known sub-problems. This method, which we call *Manifold Alternating Direction Method of Multipliers* (MADMM), is summarized in Algorithm 3.

```
1 Initialize $k \leftarrow 1$, $\mathbf{Z}^{(1)} = \mathbf{A}\mathbf{X}^{(1)}$, $\mathbf{U}^{(1)} = 0$.
2 repeat
3     X-step: $\mathbf{X}^{(k+1)} = \operatorname{argmin}_{\mathbf{X} \in \mathcal{X}} f(\mathbf{X}) + \tfrac{\rho}{2} \|\mathbf{A}\mathbf{X} - \mathbf{Z}^{(k)} + \mathbf{U}^{(k)}\|_{\mathrm{F}}^2$
4     Z-step: $\mathbf{Z}^{(k+1)} = \operatorname{argmin}_{\mathbf{Z}} g(\mathbf{Z}) + \tfrac{\rho}{2} \|\mathbf{A}\mathbf{X}^{(k+1)} - \mathbf{Z} + \mathbf{U}^{(k)}\|_{\mathrm{F}}^2$
5     $\mathbf{U}^{(k+1)} = \mathbf{U}^{(k)} + \mathbf{A}\mathbf{X}^{(k+1)} - \mathbf{Z}^{(k+1)}$
6     $k \leftarrow k + 1$
7 until convergence;
```

Algorithm 3: Generic MADMM for non-smooth optimization on a manifold $\mathcal{X}$.

The $\mathbf{X}$-step is the setting of Algorithm 2 and can be carried out using any standard smooth manifold optimization method. Similarly to common implementation of ADMM algorithms, there is no need to solve the $\mathbf{X}$-step problem *exactly*; instead, only a few iterations of manifold optimization are done. Furthermore, for some manifolds and some functions f, the $\mathbf{X}$-step has a closed-form solution. The implementation of the $\mathbf{Z}$-step depends on the non-smooth function g, and in many cases has a closed-form expression: for example, when g is the L_1-norm, the $\mathbf{Z}$-step boils down to simple shrinkage.

3.3 Unknown input ordering

Many formulations of the functional map computation assumes the knowledge of a set of *corresponding* functions $\{f_i, g_i\}$ on the two shapes $\mathcal{M}$ and $\mathcal{N}$, respectively. While there exist various methods for extracting repeatable functions stable under wide classes of transformations (see, e.g., [LBB11]), their ordering is usually arbitrary. To overcome this shortcoming, [PBB+13] proposed to solve simultaneously for the functional map and the unknown permutation of the unordered inputs.

We start with the simplified case in which the process generating the inputs is perfectly *repeatable* in the sense that it finds q functions on $\mathcal{M}$ and $\mathcal{N}$, such that for every f_i there exists a $g_j = f_i \circ T^{-1}$ related by the unknown correspondence t. We stress that the ordering of the f_i's and g_j's is *unknown*, i.e., we do not know to which g_j in $\mathcal{N}$ a f_i in $\mathcal{M}$ correspond. This ordering can be expressed by an unknown $q \times q$ permutation matrix $\mathbf{\Pi}$.

Representing the functions in the bases on each shape, we have $a_i = \langle \phi_i^{\mathcal{M}}, f_i \rangle_{L^2(\mathcal{M})}$ and $b_j = \langle \phi_j^{\mathcal{N}}, g_j \rangle_{L^2(\mathcal{N})}$ related (approximately) by $\mathbf{B\Pi} = \mathbf{CA}$, where $\pi_{ji} = 1$ if $\mathbf{a}_i$ corresponds to $\mathbf{b}_j$ and zero otherwise. Note that in the above relation both $\mathbf{\Pi}$ and $\mathbf{C}$ are unknown leading to an ill-posed problem, which can be regularized by adding structure priors on $\mathbf{C}$. While in [PBB+13] only the approximate diagonality prior was considered, the formulation is amenable to more general types of priors. The general correspondence inference problem can be written as

$$\min_{\mathbf{C},\mathbf{\Pi}} \frac{1}{2}\|\mathbf{B\Pi} - \mathbf{CA}\|_{\mathrm{F}}^2 + \rho(\mathbf{C}), \tag{3.6}$$

where the minimum is sought over $k \times k$ matrices $\mathbf{C}$ (representing the correspondence T between the shapes in the functional representation) and $q \times q$ permutations $\mathbf{\Pi}$ (capturing the correspondence between the input functions). The second term promotes solutions respecting the structure of $\mathbf{C}$. In the particular case of ρ being a weighted ℓ_1 norm promoting diagonal structure, the authors of [PBB+13] dubbed problem (3.6) as *permuted sparse coding*.

The solution of (3.6) can be obtained using alternating minimization iterating over $\mathbf{C}$ with fixed $\mathbf{\Pi}$, and $\mathbf{\Pi}$ with fixed $\mathbf{C}$. Note that with fixed $\mathbf{\Pi}$, we can denote $\mathbf{B}' = \mathbf{B\Pi}$ and reduce problem (3.6) to the regularized correspondence inference problem with ordered inputs,

$$\min_{\mathbf{C}} \frac{1}{2}\|\mathbf{B}' - \mathbf{CA}\|_{\mathrm{F}}^2 + \rho(\mathbf{C}). \tag{3.7}$$

On the other hand, when $\mathbf{C}$ is fixed, we set $\mathbf{A}' = \mathbf{CA}$, reducing the optimization objective to

$$\|\mathbf{B\Pi} - \mathbf{A}'\|_{\mathrm{F}}^2 = \mathrm{tr}\left(\mathbf{B}^\top \mathbf{B\Pi\Pi}^\top\right) - 2\mathrm{tr}\left(\mathbf{A}'^\top \mathbf{B\Pi}\right) + \mathrm{tr}\left(\mathbf{A}'^\top \mathbf{A}'\right).$$

Since $\mathbf{\Pi}$ is a permutation matrix, $\mathbf{\Pi\Pi}^\top = \mathbf{I}$, and the only non-constant term remaining in the objective is the second linear term. Problem (3.6) thus becomes the following $q \times q$ linear assignment problem (LAP)

$$\max_{\mathbf{\Pi}} \mathrm{tr}\left(\mathbf{\Pi}^\top \mathbf{E}\right), \tag{3.8}$$

where $\mathbf{E} = \mathbf{A}'^\top \mathbf{B} = \mathbf{A}^\top \mathbf{C}^\top \mathbf{B}$. Due to total unimodularity of LAPs, it can be solved as the following linear problem

$$\max_{\mathbf{\Pi} \geq \mathbf{0}} \mathrm{vec}(\mathbf{E})^\top \mathrm{vec}(\mathbf{\Pi}) \quad \text{s.t.} \quad \begin{cases} \mathbf{\Pi 1} = \mathbf{1} \\ \mathbf{\Pi}^\top \mathbf{1} = \mathbf{1}. \end{cases} \tag{3.9}$$

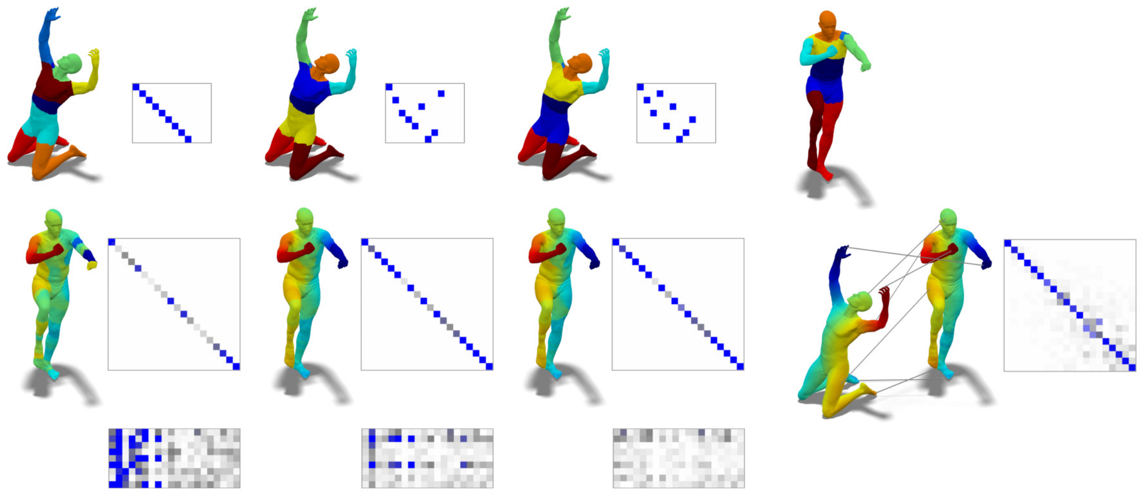


Figure 3.3: Outer iterations of the robust permuted sparse coding alternating the solution of inference problem (3.12) with the linear assignment problem (3.13). Three iterations, shown left-to-right, are required to achieve convergence. Depicted are the permutation matrix $\mathbf{\Pi}$ (first row), the correspondence matrix $\mathbf{C}$ (second row), and the outlier matrix $\mathbf{O}$ (last row). The resulting point-to-point correspondence and the correspondence matrix $\mathbf{C}$ refined using the ICP are shown in the rightmost column.

Problems 3.7 and 3.9 are alternated; in cases where the prior ρ is a convex function, convergence to a local minimum is guaranteed. In practice, excellent convergence is observed after a few outer iterations. Figure 3.3 illustrates convergence in three iterations.

So far, we have assumed the existence of a bijective, albeit unknown, correspondence between the inputs f_i's and the g_j's. In practice, the process detecting these functions is often not perfectly repeatable. In what follows, we discuss the more realistic setting in which q functions f_i are detected on $\mathcal{M}$, and r functions g_j detected on $\mathcal{N}$ (without loss of generality, $q \leq r$), such that some f_i's have no counterpart g_j, and vice versa. This partial correspondence can be described by a $q \times r$ partial permutation matrix $\mathbf{\Pi}$ in which now some columns and rows may vanish. Let us assume that $s \leq q$ f_i's have corresponding g_j's. This means that there is no correspondence between $r - s$ columns of $\mathbf{B}$ and $q - s$ columns of $\mathbf{A}$, and the relation $\mathbf{B\Pi} \approx \mathbf{CA}$ holds only for an unknown subset of its columns. The mismatched columns of $\mathbf{B}$ can be ignored by letting some rows of $\mathbf{\Pi}$ vanish, meaning that the correspondence is no more surjective. This can be achieved by relaxing the equality constraint $\mathbf{\Pi 1} = \mathbf{1}$ in (3.9) replacing it with $\mathbf{\Pi 1} \leq \mathbf{1}$. However, dropping surjectivity as well and relaxing $\mathbf{\Pi}^\top \mathbf{1} = \mathbf{1}$ to $\mathbf{\Pi}^\top \mathbf{1} \leq \mathbf{1}$ would result in the trivial solution $\mathbf{\Pi} = \mathbf{0}$. To overcome this difficulty, we demand every column of $\mathbf{A}$ to have a matching column in $\mathbf{B}$, and absorb the $r - s$ mismatches in a column-sparse $n \times q$ outlier matrix $\mathbf{O}$ that is added to the data term of (3.6). This results in the following problem

$$\min_{\mathbf{C},\mathbf{O},\mathbf{\Pi}} \frac{1}{2}\|\mathbf{B\Pi} - \mathbf{CA} - \mathbf{O}\|_{\mathrm{F}}^2 + \rho(\mathbf{C}) + \mu\|\mathbf{O}\|_{1,2}, \tag{3.10}$$

which can be thought of as a robust version of (3.6). The last term involves the $\ell_{1,2}$ norm

$$\|\mathbf{O}\|_{1,2} = \sum_{i=1}^{n} \|\mathbf{o}_i\|_2, \tag{3.11}$$

which can be thought of as the ℓ_1 norm of the vector of the ℓ_2 norms of the columns $\mathbf{o}_i$ of $\mathbf{O}$. The $\ell_{1,2}$ norm promotes column-wise sparsity, allowing to absorb the errors in the data term corresponding to the columns of $\mathbf{A}$ having no corresponding columns in $\mathbf{B}$; the parameter $\mu \geq 0$ controls the amount of such outliers. The $r \times q$ matrix $\mathbf{\Pi}$ is searched over all surjective correspondences.

As before, problem (3.10) is split into two sub-problems, one with the fixed permutation $\mathbf{\Pi}$,

$$\min_{\mathbf{C},\mathbf{O}} \frac{1}{2}\|\mathbf{B}' - \mathbf{C}\mathbf{A} - \mathbf{O}\|_{\mathrm{F}}^2 + \rho(\mathbf{C}) + \mu\|\mathbf{O}\|_{1,2}, \tag{3.12}$$

with $\mathbf{B}' = \mathbf{B}\mathbf{\Pi}$, and the other one with the fixed $\mathbf{C}$,

$$\max_{\mathbf{\Pi}\geq\mathbf{0}} \ \mathrm{vec}(\mathbf{E})^\top \mathrm{vec}(\mathbf{\Pi}) \quad \text{s.t.} \quad \begin{cases} \mathbf{\Pi}\mathbf{1} \leq \mathbf{1} \\ \mathbf{\Pi}^\top\mathbf{1} = \mathbf{1}, \end{cases} \tag{3.13}$$

Note that an surjective correspondence is relaxed into a column-wise stochastic and row-wise sub-stochastic matrix $\mathbf{\Pi}$.

3.4 Coupled functional maps

Instead of finding one functional map from $L^2(\mathcal{M})$ to $L^2(\mathcal{N})$, it is possible to consider simultaneously two *coupled functional maps* $T_1 : L^2(\mathcal{M}) \to L^2(\mathcal{N})$ and $T_2 : L^2(\mathcal{N}) \to L^2(\mathcal{M})$ satisfying $T_1T_2 = \mathrm{id}$ [ERGB16]. In matrix representation, this coupling constraint amounts to $\mathbf{C}_1\mathbf{C}_2 = \mathbf{I}$. Coupled functional maps are computed by solving the optimization problem

$$\begin{aligned} \min_{\mathbf{C}_1,\mathbf{C}_2} \quad & \|\mathbf{C}_1\mathbf{A} - \mathbf{B}\| + \|\mathbf{A} - \mathbf{C}_2\mathbf{B}\| + \mu\|\mathbf{W}\circ\mathbf{C}_1\| + \mu\|\mathbf{W}\circ\mathbf{C}_2\| \\ & \text{s.t. } \ \mathbf{C}_1\mathbf{C}_2 = \mathbf{I}. \end{aligned} \tag{3.14}$$

where $\mathbf{W}$ is a mask as in the permuted sparse coding problem promoting a funnel-shaped structure of the matrices $\mathbf{C}_1, \mathbf{C}_2$. In [GB16], it was shown that the set of pairs of invertible orthogonal matrices $BO = \{(M,N) : M^\top N = \mathbf{I}\}$ is a Riemannian matrix manifold referred to as *biorthogonal manifold*; it is thus possible to employ manifold optimization techniques to solve (3.14).

3.5 Correspondence by matrix completion

In [KBBV15], it was proposed to formulate functional map computation as a *geometric matrix completion* [KBBV14]. In classical matrix completion problem, one is given a sparse set of observations $a_{ij\in\Omega}$ of a matrix $\mathbf{A}$ and tries to find the lowest rank matrix with elements equal to the given ones on the subset of indices Ω. Since rank minimization turns out to be an NP-hard problem, Candès et al. [CR09] proposed using a convex relaxation of the problem

$$\min_{\mathbf{X}} \|\mathbf{X}\|_* \quad \text{s.t.} \quad x_{ij} = a_{ij}, ij \in \Omega,$$

where the *nuclear norm* $\|\mathbf{X}\|_* = \sum_i \sigma_i$ (σ_i denoting here the singular values of $\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$) is used as the convex proxy of the rank. It is common to replace the constraint by a penalty,

$$\min_{\mathbf{X}} \|\mathbf{X}\|_* + \mu\|\mathcal{P}_\Omega\mathbf{X} - \mathbf{a}\|,$$

where $\mathcal{P}_\Omega \mathbf{X} = (x_{ij \in \Omega})$.

Matrix completion problems are widely used in recommender systems such as the classical Netflix problem, in which the matrix of scores given by users to different movies has to be estimated from a sparse set of samples. However, the standard problem setting does not account for possible geometry of the problem. For instance, if the columns correspond to users and rows to movies, and a friendship relation between users is available in the form of a social graph, one would expect friends to give similar scores. A simple geometric matrix completion model (see Figure 3.4) assuming column- and row-wise smoothness, understood as the Dirichlet energy w.r.t. column- and row-graphs,

$$\min_{\mathbf{X}} \|\mathbf{X}\|_* + \mu_1 \mathrm{trace}(\mathbf{X}^\top \mathbf{\Delta}_r \mathbf{X}) + \mu_1 \mathrm{trace}(\mathbf{X} \mathbf{\Delta}_c \mathbf{X}^\top) + \mu_2 \|\mathcal{P}_\Omega \mathbf{X} - \mathbf{a}\|,$$

was proposed in [KBBV14] (here $\mathbf{\Delta}_c$, $\mathbf{\Delta}_r$ denote the column- and row-graph Laplacians, respectively).

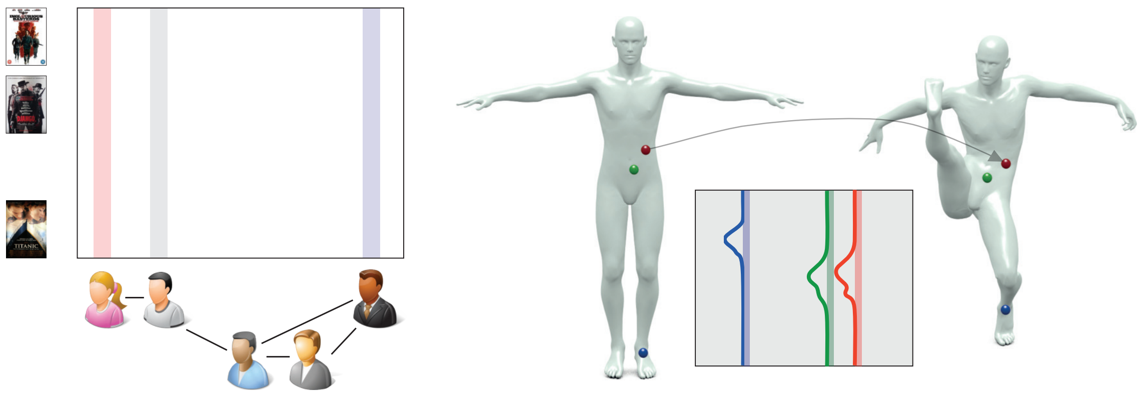


Figure 3.4: Left: geometric matrix completion in the Netflix problem (columns represent users and rows are movies; geometric structure of columns is given in the form of a social graph of users). Right: correspondence as a matrix completion problem.

In [KBBV15], this model was adapted to shape correspondence problems. Considering the functional correspondence operator represented in delta-bases on discrete shapes as an $m \times n$ matrix $\mathbf{T}$ (n and m denote here the number of vertices on the two shapes $\mathcal{M}$ and $\mathcal{N}$, respectively),

$$\min_{\mathbf{T}} \|\mathbf{T}\|_* + \mu_1 \mathrm{trace}(\mathbf{T}^\top \mathbf{\Delta}_{\mathcal{N}} \mathbf{T}) + \mu_1 \mathrm{trace}(\mathbf{T} \mathbf{\Delta}_{\mathcal{M}} \mathbf{T}^\top) + \mu_2 \|\mathbf{TF} - \mathbf{G}\| + \mu_3 \|\mathbf{T}\|_1,$$

where $\mathbf{F} = (\mathbf{f}_1, \dots, \mathbf{f}_q)$ and $\mathbf{G} = (\mathbf{g}_1, \dots, \mathbf{g}_q)$ are $n \times q$ and $m \times q$ matrices representing the corresponding functions, $\mathbf{TF} \approx \mathbf{G}$. The use of the additional L_1-norm penalty promoting sparsity of $\mathbf{T}$, together with the Dirichlet energy promoting its row- and column-wise smoothness, results in correspondence localization [OLCO13]. Parametrizing $\mathbf{T} = \mathbf{UV}^\top$, where $\mathbf{U}, \mathbf{V}$ are $m \times k$ and $n \times k$ matrices, respectively, with arbitrarily large k, the problem can be equivalently posed as [SRJ04]

$$\begin{aligned}\min_{\mathbf{U},\mathbf{V}} \quad & \tfrac{1}{2}(\|\mathbf{U}\|_F^2 + \|\mathbf{V}\|_F^2) + \mu_1 \mathrm{trace}(\mathbf{V}\mathbf{U}^\top \mathbf{\Delta}_{\mathcal{N}} \mathbf{U}\mathbf{V}^\top) + \mu_1 \mathrm{trace}(\mathbf{U}\mathbf{V}^\top \mathbf{\Delta}_{\mathcal{M}} \mathbf{V}\mathbf{U}^\top) \\ & + \mu_2 \|\mathbf{U}\mathbf{V}^\top \mathbf{F} - \mathbf{G}\| + \mu_3 \|\mathbf{U}\mathbf{V}^\top\|_1 \qquad (3.15)\end{aligned}$$

Parametrizing the factors $\mathbf{U} = \mathbf{\Phi}_{\mathcal{N}} \mathbf{P}$, $\mathbf{V} = \mathbf{\Phi}_{\mathcal{M}} \mathbf{Q}$ as linear combinations of k' Laplacian eigenfunctions, problem (3.15) can be rewritten as

$$\min_{\mathbf{P},\mathbf{Q}} \quad \frac{1}{2}(\|\mathbf{\Phi}_{\mathcal{N}}\mathbf{P}\|_{\mathrm{F}}^2 + \|\mathbf{\Phi}_{\mathcal{M}}\mathbf{Q}\|_{\mathrm{F}}^2) + \mu_1 \mathrm{trace}(\mathbf{Q}\mathbf{P}^\top \mathbf{\Lambda}_{\mathcal{N}} \mathbf{P}\mathbf{Q}^\top) + \mu_1 \mathrm{trace}(\mathbf{P}\mathbf{Q}^\top \mathbf{\Lambda}_{\mathcal{M}} \mathbf{Q}\mathbf{P}^\top)$$
$$+\mu_2 \|\mathbf{P}\mathbf{Q}^\top \mathbf{A} - \mathbf{B}\| + \mu_3 \|\mathbf{\Phi}_{\mathcal{N}} \mathbf{P}\mathbf{Q}^\top \mathbf{\Phi}_{\mathcal{M}}^\top\|_1, \tag{3.16}$$

where notation follows our discussion of joint diagonalization. Note that matrices $\mathbf{P}, \mathbf{Q}$ are not orthogonal.

The matrix completion approach turns out to be advantageous when the given correspondence information is very scarce. Since $\mathrm{rank}(\mathbf{T}) \leq k$ and k can be arbitrarily large (in practice, limited only by computational complexity) as opposed to the plain functional maps formulation as a linear system (in which k must be smaller than q in order to make the system determined), the matrix completion approach behaves better in situations where $k \gg q$ (see Figure 3.5).

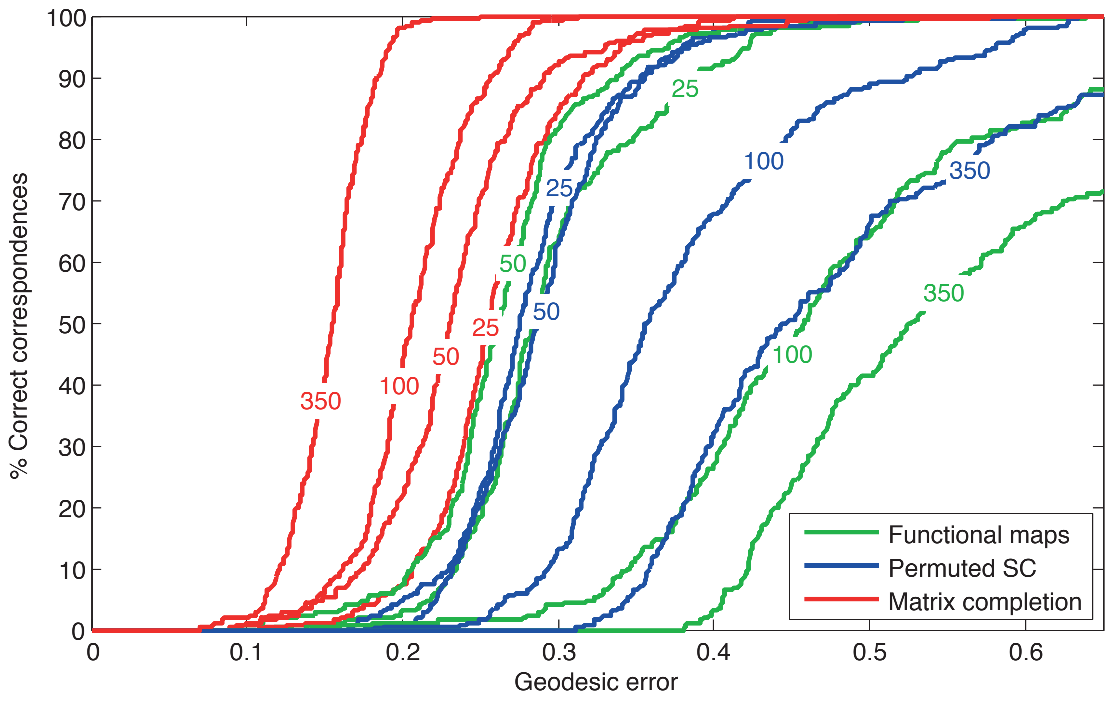


Figure 3.5: Behavior of different functional correspondence model for increasingly large rank of the correspondence matrix k. The matrix completion model manifests better correspondence quality with the increase of k, while other models' quality deteriorates.

3.6 Descriptor Preservation via Commutativity

As mentioned in Section 2.4.2 the simplest way to enforce function (e.g., descriptor) preservation constraints is by formulating constraints on $\mathbf{C}$ of the form: $\mathbf{Ca} = \mathbf{b}$, which can be enforced in the least squares. However, it is easy to see that these constraints typically do not extract all of the information from the descriptor functions if they are phrased in the reduced basis. Indeed, if we are given a single perfect descriptor function, which identifies each point uniquely then it should be sufficient to recover the map $\mathbf{C}$, whereas at least k linearly independent constraints of type $\mathbf{Ca} = \mathbf{b}$ are necessary to recover a matrix $\mathbf{C}$ of size $k \times k$. Previous approaches have tried to address this issue by splitting a given descriptor function into level-sets and deriving additional function preservation constraints (See e.g. Section 4.1 in [OMPG13]). This, however, introduces additional parameters as well as potential sources of instability.

To alleviate this issue, Nogneng and Ovsjanikov [NO17] have proposed a more powerful method for enforcing descriptor preservation constraints, which works as follows: for every pair of descriptor functions f_i, g_i on the source and target shapes respectively, they build *linear functional operators* X_{f_i}, Y_{g_i}, which act on functions via pointwise multiplication. Thus, for a function h defined on the source shape, we have $X_{f_i}(h)(x) = f_i(x)h(x)$ at every point x. Such a linear operator can be represented in a reduced basis as a matrix using: $\mathbf{X}_{f_i} = \mathbf{\Phi}_{\mathcal{M}}^{+} Diag(f_i) \mathbf{\Phi}_{\mathcal{M}}$, where $Diag(f_i)$ is a diagonal matrix containing the values of f_i. Similarly $\mathbf{Y}_{g_i} = \mathbf{\Phi}_{\mathcal{N}}^{+} Diag(g_i) \mathbf{\Phi}_{\mathcal{N}}$. Note that each pair of descriptor functions f_i, g_i thus gives rise to a pair of linear operators $\mathbf{X}_{f_i}$ and $\mathbf{Y}_{g_i}$. Given these operators, we can then add an additional term to the energy in Eq. (2.2) as well as all of the energies described in this section:

$$E_3(\mathbf{C}) = \sum_i \|\mathbf{C}\mathbf{X}_{f_i} - \mathbf{Y}_{g_i}\mathbf{C}\|.$$

Intuitively, this new term promotes preservation of *pointwise function products*, via commutativity with linear operators $\mathbf{X}_{f_i}, \mathbf{Y}_{g_i}$. It can be shown that this constraint is also closely related to promoting a functional map to be associated with a point-to-point one. Indeed it is well-known that a non-trivial functional map T_F corresponds to a point-to-point one if for every pair of functions $T_F(fg) = T_F(f)T_F(g)$. Moreover, although this new energy term remains quadratic in the unknown matrix $\mathbf{C}$, so it can be efficiently optimized for either using a linear or a more general convex solver, this term allows to extract *significantly more information* from the given descriptors and ultimately leads to better maps in practice (see [NO17] for a discussion of these constraints).

3.7 List of Key Symbols

Symbol	Definition
T	Continuous functional map
$\mathbf{T}$	Discrete functional map expressed in the delta basis
$\mathbf{C}$	Functional map expressed as a matrix in the basis of LB eigenfunctions.
$\mathcal{X}$	Matrix manifold
$\mathbf{\Delta}_{\mathcal{M}}$	Discretization of the Laplacian $\Delta_{\mathcal{M}}$
$\mathbf{\Phi}^{\mathcal{M}}$	Matrix of eigenvectors of the Laplacian $\Delta_{\mathcal{M}}$
$\hat{\mathbf{\Phi}}^{\mathcal{M}}$, $\hat{\mathbf{\Phi}}^{\mathcal{N}}$	Coupled bases
$\mathbf{\Lambda}_{\mathcal{M}}$	Diagonal matrix of eigenvalues of the Laplacian $\Delta_{\mathcal{M}}$
$\mathbf{\Pi}$	Permutation establishing input functions ordering
X_{f_i}	Multiplicative operator for a function f_i in the full basis. $X_{f_i} = Diag(f_i)$.
$\mathbf{X}_{f_i}$	Multiplicative for a function f_i in a reduced basis.

4

Partial Functional Maps

This Chapter discusses how the functional map representation can be used to deal with shapes having missing parts, with the presence of clutter, and the two sources of nuisance simultaneously.

4.1 Partial Functional Maps

In case one of the two shapes has holes or missing parts, the functional representation of the correspondence still has a meaningful structure which can be taken advantage of, as recently shown in [RCB+16].

Assume to be given a full shape $\mathcal{M}$ and a *partial* shape $\mathcal{N}$ that is approximately isometric to some (unknown) sub-region $\mathcal{M}' \subset \mathcal{M}$. The authors showed that for each "partial" eigenfunction $\phi_j^{\mathcal{N}}$ of $\mathcal{N}$ there exists a corresponding "full" eigenfunction $\phi_i^{\mathcal{M}}$ of $\mathcal{M}$ for some $i \geq j$, such that $c_{ij} = \langle T_F(\phi_i^{\mathcal{M}}), \phi_j^{\mathcal{N}} \rangle_{L^2(\mathcal{N})} \approx \pm 1$, and zero otherwise. Note that differently from the full-to-full case discussed in the previous chapters, where the approximate equality holds for $i = j$ (see, e.g., Section 3.1), here the inequality $i \geq j$ induces a *slanted*-diagonal structure on matrix $\mathbf{C}$. In particular, it can be shown that the angle of the diagonal can be directly (and quite conveniently) estimated from the area ratio of the two surfaces [RCB+16]. The precomputed angle can then be used as a prior on $\mathbf{C}$ to drive the matching process (we will see how in Section 4.1.2).

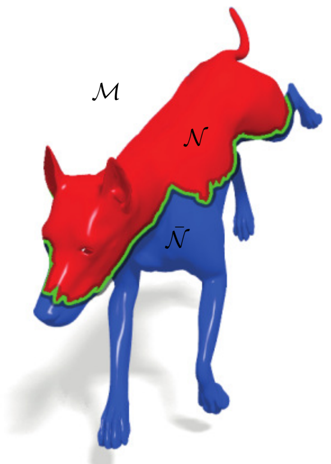
4.1.1 Perturbation analysis

In this Section we sketch an algebraic argument motivating the behavior that we observe for the eigenfunctions of the partial shape. In a nutshell, the idea is to model partiality as a perturbation of the Laplacian matrices $\mathbf{\Delta}_{\mathcal{M}}$, $\mathbf{\Delta}_{\mathcal{N}}$ of the two shapes. Specifically, consider the dog shape $\mathcal{M}$ shown in the inset, and assume a vertex ordering where the points contained in the red region $\mathcal{N}$ appear before those of the blue region $\bar{\mathcal{N}}$. Then, the full Laplacian $\mathbf{\Delta}_{\mathcal{M}}$ will assume the structure

$$\mathbf{\Delta}_{\mathcal{M}} = \begin{pmatrix} \mathbf{\Delta}_{\mathcal{N}} & \mathbf{0} \\ \mathbf{0} & \mathbf{\Delta}_{\bar{\mathcal{N}}} \end{pmatrix} + \begin{pmatrix} \mathbf{P}_{\mathcal{N}} & \mathbf{E} \\ \mathbf{E}^\top & \mathbf{P}_{\bar{\mathcal{N}}} \end{pmatrix}, \tag{4.1}$$

where the second matrix encodes the perturbation due to the boundary interaction between the two regions. Such a matrix is typically very sparse and low-rank, since it contains non-zero elements only in correspondence of the edges connecting the boundary $\partial\mathcal{N}$ to $\partial\bar{\mathcal{N}}$.

If the perturbation matrix is identically zero, then (4.1) is exactly block-diagonal; this describes the case in which $\mathcal{N}$ and $\bar{\mathcal{N}}$ are disjoint parts, and the eigenpairs of $\mathbf{\Delta}_{\mathcal{M}}$ are an interleaved sequence of those of the two blocks. The key result shown in [RCB+16] is that this interleaving property still holds even when considering the full matrix $\mathbf{\Delta}_{\mathcal{M}}$ as given in (4.1): Its eigenpairs consist of those of the blocks $\mathbf{\Delta}_{\mathcal{N}}$, $\mathbf{\Delta}_{\bar{\mathcal{N}}}$, up to some bounded perturbation that depends on the length and position of the boundary $\partial\mathcal{N}$.



4.1.2 Algorithm

In the part-to-full setting, one is interested in determining a functional map built upon a near-isometry $T : \mathcal{N} \to \mathcal{M}'$, where the part $\mathcal{M}' \subset \mathcal{M}$ is an additional *unknown* of the correspondence problem. The idea is to model the part as a soft indicator (or *segmentation*) function $v : \mathcal{M} \to [0,1]$ such that $v(x) = 1$ if $x \in \mathcal{M}'$ and zero otherwise, and to simultaneously solve for the functional map matrix $\mathbf{C}$ and the indicator v. The resulting optimization problem takes the general form:

$$\min_{\mathbf{C},v} \|\mathbf{CA} - \mathbf{B}(v)\|_{2,1} + \rho_{\text{corr}}(\mathbf{C}) + \rho_{\text{part}}(v)\,, \tag{4.2}$$

where $\mathbf{B}(v)$ denotes the spectral coefficients of the descriptor field on $\mathcal{M}$ weighted by v (which thus acts like a mask), ρ_{corr} is a correspondence regularization term, and ρ_{part} is part regularization. The $L_{2,1}$ norm makes the data term more robust to noisy data.

As correspondence regularization, the authors in [RCB+16] proposed to use the penalty

$$\rho_{\text{corr}}(\mathbf{C}) = \mu_1\|\mathbf{C}\circ\mathbf{W}\|_F^2 + \mu_2\sum_{i\neq j}(\mathbf{C}^\top\mathbf{C})_{ij}^2 + \mu_3\sum_i((\mathbf{C}^\top\mathbf{C})_{ii} - d_i)^2\,,$$

where $\circ$ is the element-wise product. The μ_1-term models the slanted-diagonal structure of $\mathbf{C}$, where matrix $\mathbf{W}$ is a weight matrix with zeroes along the slanted diagonal and large values outside. A similar term was used in Section 3.3 for full-to-full correspondence. The slope of $\mathbf{W}$ can be estimated in several ways, one simple possibility being the area ratio of the two surfaces [RCB+16]. The μ_2- and μ_3- terms promote orthogonality of the functional map; here, $\mathbf{d}$ is a vector with the first r elements set to 1 and the remaining to 0, where r is the estimated rank of the map (obtained in the same way as the slope estimate).

For part regularization, the following terms (inspired by [BB08]) are considered:

$$\rho_{\text{part}}(v) = \mu_4\left(\text{area}(\mathcal{N}) - \int_{\mathcal{M}} v\,dx\right)^2 + \mu_5\int_{\mathcal{M}}\|\nabla_{\mathcal{M}}v\|dx\,.$$

The μ_4-term asks for the sought part to have area close to the partial shape $\mathcal{N}$. The μ_5-term is an intrinsic equivalent of the Mumford-Shah functional from image processing, measuring the length of the boundary of the part represented by a soft indicator function. Asking for short boundary length prevents the algorithm from getting stuck at local minima with highly fragmented regions.

The optimization problem (4.2) is non-convex, and can be minimized in an iterative fashion by solving for $\mathbf{C}$ and v alternatingly until convergence. Although this approach does not guarantee to reach a global optimum, the obtained solutions are typically very accurate, as also recently demonstrated in the challenging SHREC'16 Partial Correpondence benchmark [CRB+16]. Some qualitative examples are reported in Figure 4.1.

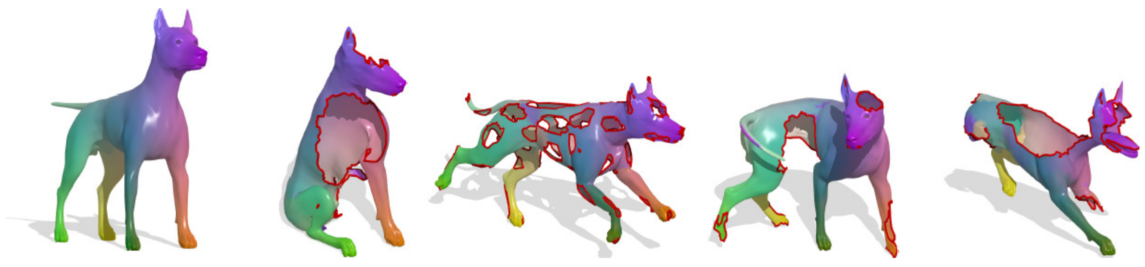


Figure 4.1: Examples of solutions obtained with the partial functional maps algorithm of Section 4.1.2. Each partial shape is matched to the reference full model (leftmost shape).

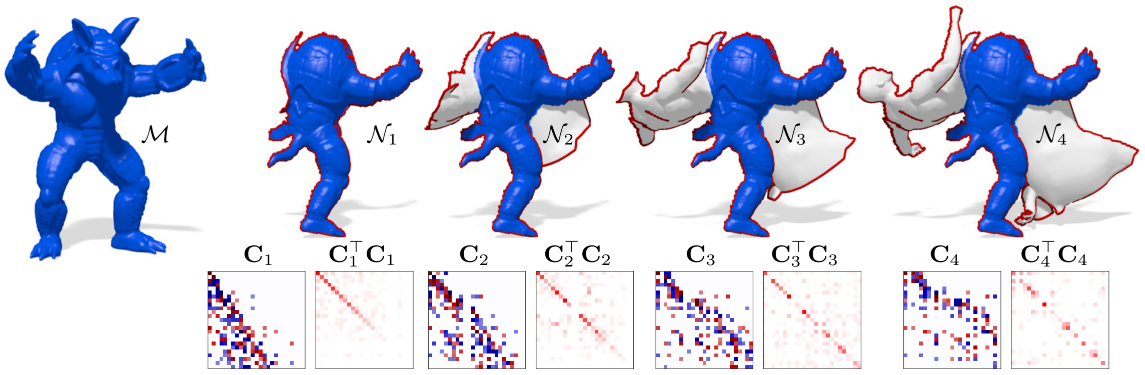


Figure 4.2: Functional maps at increasing amounts of clutter. The model $\mathcal{M}$ is matched to scenes $\mathcal{N}_1$-$\mathcal{N}_4$, giving rise to the matrices of spectral coefficients $\mathbf{C}_1$-$\mathbf{C}_4$. Observe how the dominant slope of $\mathbf{C}_i$ varies with clutter, moving from the lower- to the upper-triangular part of the matrix. The rank of $\mathbf{C}_i$ decreases as more and more clutter is introduced, a fact that is manifested in empty rows and columns in $\mathbf{C}_i$, and in the sparse diagonal structure on $\mathbf{C}_i^\top\mathbf{C}_i$. The latter property can be used as a prior when solving for $\mathbf{C}_i$. The zero-clutter pair $(\mathcal{M}, \mathcal{N}_1)$ is the setting considered in Section 4.1.

4.2 Deformable clutter

The approach described in the previous Section can be applied whenever one is given a partial query $\mathcal{N}$ to be matched to a full model $\mathcal{M}$ (part-to-full matching). If both shapes have missing parts (part-to-part) or if additional geometry (or "clutter") is present in either shape, the approach will fail due to its underlying assumptions. Moreover, it is not clear whether a particular structure is still observable in matrix $\mathbf{C}$ under this challenging setting.

A positive answer to this question was recently given in [CRM+16]: In the presence of clutter, it is still possible to find eigenfunctions $\phi_i^{\mathcal{M}}$ on $\mathcal{M}$ for *some* indices i, having corresponding eigenfunctions $\phi_j^{\mathcal{N}}$ on $\mathcal{N}$ for some indices j. There is a key difference with what we have seen in the previous settings. While in the full-to-full case we had correspondence for $i = j$ and in the part-to-full case for $i \geq j$, here the correspondence among indices cannot be reliably predicted. The diagonal slant of $\mathbf{C}$, which identifies the pairs (i, j) for which $c_{ij} = \langle T_F(\phi_i^{\mathcal{M}}), \phi_j^{\mathcal{N}} \rangle_{L^2(\mathcal{N})} \neq 0$, is now an *unknown* that we need to optimize for. In particular, we expect $c_{ij} \neq 0$ only for a sparse set of indices, i.e., matrix $\mathbf{C}$ will have empty rows and columns. See Figure 4.2 for an illustration at increasing amounts of clutter.

Due to the presence of clutter, a more general formulation for the correspondence

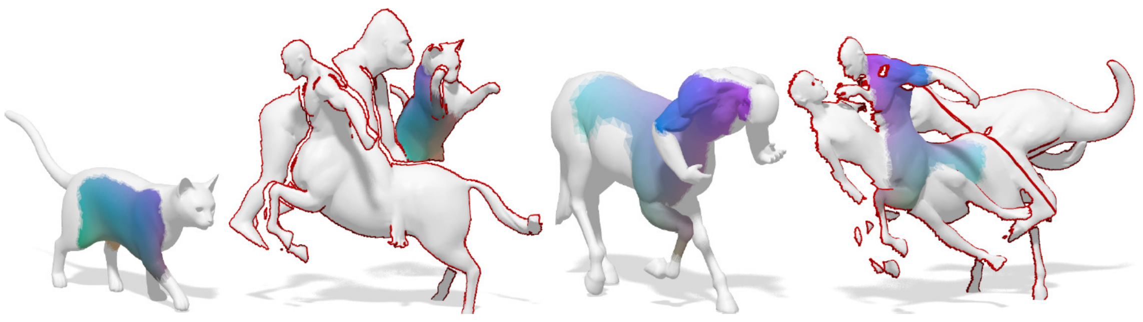


Figure 4.3: Two examples of a deformable object-in-clutter problem, and the corresponding solutions obtained with the method described in Section 4.2.

problem is required since it is now possible that only parts of *both* shapes are matchable. In mathematical terms, this amounts to introducing two segmentation functions $u : \mathcal{M} \to [0,1]$ and $v : \mathcal{N} \to [0,1]$ indicating the shape parts that are put into correspondence. This leads to an optimization problem of the form:

$$\min_{\mathbf{C},\theta,u,v} \|\mathbf{C}\mathbf{A}(u) - \mathbf{B}(v)\|_{2,1} + \rho_{\text{corr}}(\mathbf{C},\theta) + \rho_{\text{part}}(u,v)\,. \tag{4.3}$$

The regularization terms on correspondence and part are defined in a similar manner to Section 4.1.2. Notice, however, that the diagonal slope of $\mathbf{C}$, denoted by θ, is now an optimization variable. This is used to define a parametric weight matrix $\mathbf{W}(\theta)$ that is used to promote a slanted diagonal structure on $\mathbf{C}$ as in the previous case. Similarly to partial functional maps, the optimization process alternates between the two blocks of variables $\{\mathbf{C},\theta\}$ and $\{u,v\}$ until convergence. We refer to [CRM+16] for the technical details.

4.3 Non-rigid puzzles

A problem related to the ones described in the previous Sections was recently tackled in [LRB+16]. Specifically, the authors considered a shape correspondence problem in which *multiple* parts, possibly with additional clutter, are to be matched to a given full model. The query parts may partially overlap, and the model shape might have “missing” regions that do not correspond to any query shape; conversely, there might be “extra” query shapes that have no correspondence to the model shape. Similarly to the previous problems, the final task is to establish a dense part-to-whole correspondence for each partial shape, and simultaneously determine a segmentation of the model. The method can thus be seen as an extension of partial functional maps [RCB+16] for the multiple part setting on one hand, and a non-rigid generalization of the rigid puzzles problem treated in [LBB12] on the other.

This “non-rigid puzzle” problem admits a simple formulation as follows

$$\begin{aligned}
&\min_{\mathbf{C}_i,u_i,v_i} \sum_{i=1}^{p} \|\mathbf{C}_i\mathbf{A}_i(u_i) - \mathbf{B}_i(v_i)\|_{2,1} + \sum_{i=0}^{p} \rho_{\text{corr}}(\mathbf{C}_i) + \sum_{i=1}^{p} \rho_{\text{part}}(u_i,v_i) \\
&\text{s.t.} \quad \sum_{i=0}^{p} u_i = 1\,,
\end{aligned}$$

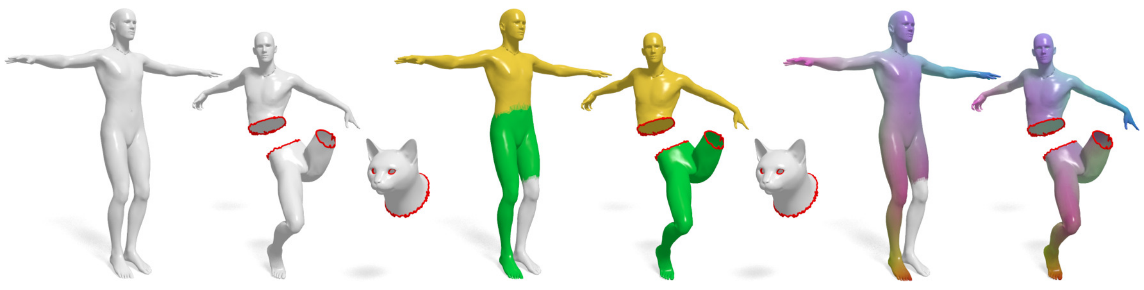


Figure 4.4: Example of a non-rigid puzzle problem: given a model human shape (leftmost, first column) and three query shapes (two deformed parts of the human and one unrelated "extra" shape of a cat head), the goal is to find a segmentation of the model (second column, shown in yellow and green; white encodes parts without correspondence) into parts corresponding to (subsets of) the query shapes. The third column shows the computed correspondence between the parts.

for a problem with p query parts, and with p_0 denoting possible missing parts. Again, the regularization terms on correspondence and parts can be defined in a similar manner as in (4.2). Note that the summation constraint renders the problem a proper segmentation task, enforcing a complete covering of the model. The problem above thus consists in solving p partial functional correspondence problems simultaneously, under covering constraints. Figure 4.4 shows an example of a non-rigid puzzle solved with this method (additional details can be found in [LRB$^+$16]).

4.4 Applications

Partial correspondence problems arise in numerous applications in the computer vision and graphics communities. A few representative examples are given below.

Shape reconstruction from range data is a classical application that involves real data acquisition by 3D sensors, inevitably leading to missing parts due to occlusions or partial view. If the object to be scanned is allowed to undergo non-rigid deformations, the problem is commonly referred to as **dynamic fusion** and is considered one of the most challenging problems of shape analysis and 3D vision.

Object detection and recognition in clutter classically arise in robotics applications, where one has to locate a reference model within a dynamic environment acquired in 3D. Surveillance applications often require the ability to match and compare the (partially) scanned subject against an existing database of shapes, up to body deformation.

Finally, in **computational biology** the identification of similar tertiary structure of proteins provides important information for analyzing their function. Structural similarity is often phrased as a partial correspondence problem between mesh representations of the 3D structures of proteins.

4.5 List of Key Symbols

Symbol	Definition
$\mathcal{M}'$	Sub-region of a full shape $\mathcal{M}$
T_F	Functional representation of a given pointwise map T
$\mathbf{C}$	Functional map expressed as a matrix in the Fourier basis
$\boldsymbol{\Delta}$	Matrix representation of the Laplace operator
$\partial\mathcal{M}$	Boundary of shape $\mathcal{M}$
u, v	Indicator functions with values in $[0, 1]$
$\nabla_{\mathcal{M}}$	Intrinsic gradient operator on shape $\mathcal{M}$

5

Maps in Shape Collections

Considering a collection of shapes related by mappings provide valuable information on deformation model and plausible deviations from this model. The earliest data-driven approaches for devising a deformation model, however, rely heavily on a consistent parametrization of the deformation domain (e.g. on a fixed grid in Euclidean space), and perform statistical analysis on the positions of vertices of the shapes. The functional framework does not need such parametrization, as it is purely intrinsic, making the analysis of a shape collection more tractable and general. In this chapter we introduce two methods to improve the computation of functional maps, first by using a supervised learning technique for feature selection and the second by imposing consistency among mappings with respect to composition. The last section presents the *shape difference operators*, an operator-based representation of intrinsic deformation, initially used for analysis of shape collection.

5.1 Descriptor and subspace learning

The simple algebraic structure of functional maps allows us to handle maps using standard linear algebra tools. However the approximation of a functional maps often relies on intrinsic descriptors computed separately on each shape. Two difficulties can arise from this computation: first the descriptors usually do not span the entire space of functions, second noisy functional correspondences can result in unreliable mappings in some part of the functional space. Using a collection makes the functional map computation more robust to non-isometric deformations and allows us to identify subspace of function on which the map is reliable. In this section we introduce the algorithm suggested in [COC14] closely related to inverse problems.

Functional Map Approximation The basic method described in Chapter 2 approximates the functional map $\mathbf{C}_i$ using a set of linear constraints. The first type of constraints is given by a set of pairs of functions that are expected to be preserved by the deformation. The second is a regularization term coming from the deformation model. This leads to the least squares problem:

$$\mathbf{C}_i^\star = \arg\min_{\mathbf{X}} \|\mathbf{X}\mathbf{A}_0 - \mathbf{A}_i\|_F^2 + \alpha\|\mathbf{X}\mathbf{\Lambda}_0 - \mathbf{\Lambda}_i\mathbf{X}\|_F^2.$$

Here, $\mathbf{A}_0, \mathbf{A}_i$ are the matrices that contain, as columns, pairs of functions that we expect to correspond under the unknown map, written in a given basis (e.g. the basis given

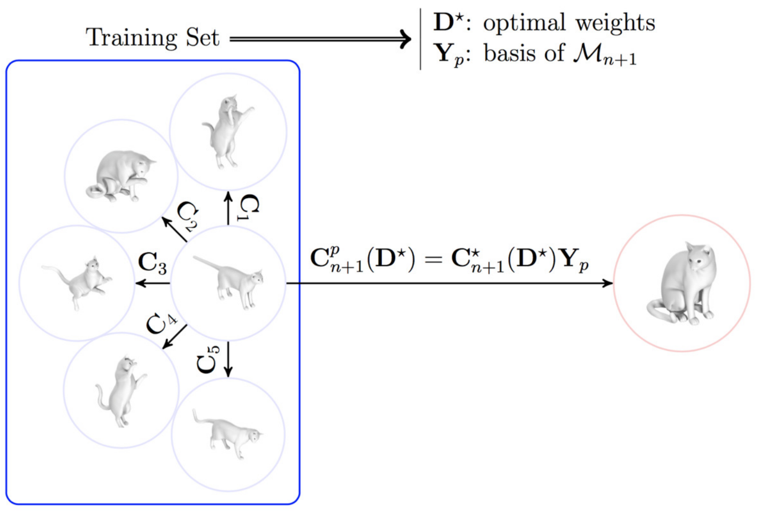

Training Set
$\mathbf{D}^\star$: optimal weights
$\mathbf{Y}_p$: basis of $\mathcal{M}_{n+1}$
$\mathbf{C}_1$
$\mathbf{C}_2$
$\mathbf{C}_3$
$\mathbf{C}_4$
$\mathbf{C}_5$
$\mathbf{C}^p_{n+1}(\mathbf{D}^\star) = \mathbf{C}^\star_{n+1}(\mathbf{D}^\star)\mathbf{Y}_p$

Figure 5.1: Supervised Learning. Given a collection of functional maps, a set of optimal weight $\mathbf{D}^\star$ and a basis of the p-best mapped functions $\mathbf{Y}_p$ are computed. When given a previously unseen shape (circled in red) we obtain an approximated functional map $\mathbf{C}^p_{n+1}(\mathbf{D}^\star)$ restricted to the most reliable function subspace.

by the eigenfunctions of the LB operator), whereas $\mathbf{\Lambda}_0, \mathbf{\Lambda}_i$ are the diagonal matrices of eigenvalues of the LB operator, as discussed in Section 2.4.4 above.

In this basic approach, the functional correspondences (also called *probe functions* below) are assumed to be given. In practice, however, this choice can already be challenging as not all such correspondences result in a useful functional map. The key idea proposed in [COC14] is to introduce weights for probe functions, and to use supervised learning to obtain optimal weights, given a set of example functional maps. Thus, the functional constraint will be replaced by: $\|\mathbf{X}\mathbf{A}_0\mathbf{D} - \mathbf{A}_i\mathbf{D}\|_F^2$ where the weights $\mathbf{D}$ will be optimized so that the weighted descriptors are jointly as informative as possible. This will allow us to improve the quality of the functional maps and moreover to extract the functional subspaces in which the computed maps will be as most reliable.

For this, we can define the function $\mathbf{D} \mapsto \mathbf{C}_i^\star(\mathbf{D})$, which maps a given sets of weight to the corresponding optimal functional map, via the solution of the optimization problem:

$$\mathbf{C}_i^\star = \arg\min_{\mathbf{X}} \|\mathbf{X}\mathbf{A}_0\mathbf{D} - \mathbf{A}_i\mathbf{D}\|_F^2 + \alpha\|\mathbf{X}\mathbf{\Lambda}_0 - \mathbf{\Lambda}_i\mathbf{X}\|_F^2.$$

Finding the best weights We assume that we are given a collection of n deformations of the same object with known functional maps $\mathbf{C}_i$ (Figure 5.1). The optimal weights $\mathbf{D}^\star$ are the ones that produce an approximation $\mathbf{C}_i^\star(\mathbf{D})$ that is closest to the ground truth known $\mathbf{C}_i$. Thus, we want to solve the following optimization problem:

$$\mathbf{D}^\star = \arg\min_{\mathbf{D}} \sum_{i=1}^{n} \|\mathbf{C}_i^\star(\mathbf{D}) - \mathbf{C}_i\|,$$

where the sum is over the set of given training maps $\mathbf{C}_i$. Note that the choice of norm in the above energy is very important and the naive choice of squared Frobenius norm for comparing the optimized with the known functional maps typically leads to poor performance. Note also that although the optimization problem is non-convex, the energy is nevertheless continuous with respect to the weight matrix $\mathbf{D}$, and can be optimized in practice as described in [COC14] in more details.

Basis function extraction Since the probe functions can give redundant information in some shape parts and incomplete information in others, our functional map will map some subspaces of $L^2(\mathcal{M}_0)$ more reliably than others. Using a collection of shapes we would like to extract the most stable subspaces.

For this purpose we propose to use the learned optimal weights $\mathbf{D}^\star$ and the resulting estimated functional maps $\mathbf{C}_i^\star(\mathbf{D}^\star)$ to identify stably mapped functional subspaces by comparing $\mathbf{C}_i^\star(\mathbf{D}^\star)$ to the reference maps $\mathbf{C}_i$. The output will be $\mathbf{Y}$ an orthonormal basis of $L^2(\mathcal{M}_0)$ ordered with decreasing confidence. This can be done efficiently in practice by computing a singular value decomposition of a moderately sized matrix.

Functional map to a test shape using a reduced basis Now if we are given an extra shape $\mathcal{M}_{n+1}$ that does not belong to the training set, we first compute its probe functions and store them in a matrix $\mathbf{A}_{n+1}$. We then compute the functional map $\mathbf{C}_{n+1}^\star(\mathbf{D}^\star)$ by using the optimal weight matrix $\mathbf{D}^\star$. Finally, since we know that $\mathbf{C}_{n+1}^\star(\mathbf{D}^\star)$ contains some badly mapped subspaces (for example the antisymmetric functions), by using $\mathbf{Y}_p$ the p first column of $\mathbf{Y}$, we compute the reduced map $\mathbf{C}_{n+1}^p$

$$\mathbf{C}_{n+1}^p = \mathbf{C}_{n+1}\mathbf{Y}_p : L^2(\mathcal{M}_0) \cap L^2(\mathrm{Span}(\mathbf{Y}_p)) \to L^2(\mathcal{M}_{n+1}).$$

As shown in [COC14], this procedure can lead to significantly better functional map, by first removing the manual process of selecting the right set of probe functions, and instead learning the optimal set from the given input correspondences and moreover by restricting the functional map to the right subspace in which the map is more reliable.

5.2 Networks of Maps

Functional maps enable information transport between between two shapes. When multiple related shapes are available, it is natural to consider multi-hop information transport by composing functional maps along a directed path of pairwise functional maps. Thus suggests organizing a collection of related shapes into a connected network whose nodes are the individual shapes and where certain pairs of shapes are linked by directed edges, each decorated with a functional maps between the corresponding shape pair. An advantage of the network view is that it allows us have access to multiple maps between a given pair of shapes, simply by forming functional map compositions along different paths in the network connecting the two shapes in question. For example, functional maps between relatively dissimilar shapes are likely to be less accurate — but by composing functional maps along a path of interpolating shapes where the relative changes are smaller, we may be able to get a better quality map.

In fact there are more fundamental ways in which a functional map network can be used to improve the maps decorating its edges, beyond re-writing maps as compositions of other maps. This is because functional maps express notions of function value preservation across two shapes, and value equality is transitive. What this implies is that in an ideal setting map compositions would be path-invariant: compositions along any path connecting the same pair of shapes should yield the same result. Equivalently, this can be stated as cycle closure — composing maps along any edge cycle in the network should yield the identity map. If we have a collection of n shapes and build a big mapping matrix

$\mathbf{C}$ consisting of $n \times n$ blocks $\mathbf{C}_{ij}$, where $\mathbf{C}_{ij}$ is the functional map from shape i to shape j, the cycle closure constraint puts very strong restrictions on the matrix $\mathbf{C}$. As it turns out [HWG14], $\mathbf{C}$ has to be positive semidefinite and of low rank — all the cycle closure conditions introduce many dependencies among the elements of $\mathbf{C}$. This implies that we can use low-rank matrix completion techniques to replace the ordinal functional maps by others that are close to the original but more cycle consistent [HWG14]. In practice we find that this helps improve the original maps.

Cycle consistency. Formulating cycle consistency as an optimization criterion amenable to efficient computation also becomes a challenge in the functional setting, even in the full similarity case, where presumably we want preservation of functions transported around all cycles in the network — equivalently, we want to compositions of operators along any cycle to yield the identity. Recall that the unknowns to be estimated are the elements of the mapping matrices $\mathbf{C}_{ij}$. A network or graph can have an exponential number of cycles. Furthermore, even for short cycles (say 3-cycles) the multiplication of the operator matrices will yield algebraic expressions of degree 3 in the matrix element variables. So it seems that the cycle consistency conditions yield an exponential number of highly nonlinear equations.

Latent spaces. The key notion is to introduce certain *latent functional spaces* that encapsulate the commonalities among the data. In the simplest case of full similarity between the shapes there is just one latent space, the common abstraction or "Platonic ideal" of which all the individual shapes are instances. If we postulate functional maps Y_i that map each individual functional space $\mathcal{F}_i$ over shape i to a common functional space $\mathcal{F}$, then we can factorize each $\mathbf{C}_{ij}$ as $\mathbf{C}_{ij} \approx \mathbf{Y}_j^{-1}\mathbf{Y}_i$. From this expression it is clear the composition of the $\mathbf{C}_{ij}$ around any cycle is a telescoping product, where all the $\mathbf{Y}_i$ cancel out and we get the identity.

This turns out to be equivalent to the existence of row-orthogonal matrices $\mathbf{Y}_i = (\mathbf{y}_{i1}, \cdots, \mathbf{y}_{iL})^T \in \mathbb{R}^{L \times dim(\mathcal{F}_i)}, 1 \leq i \leq N$ such that

$$\mathbf{C}_{ij} = \mathbf{Y}_j^{+}\mathbf{Y}_i, \quad \forall (i,j) \in \mathcal{G}. \tag{5.1}$$

Again, here $\mathbf{Y}^{+}$ denotes the Moore–Penrose pseudo-inverse of $\mathbf{Y}$.

It is clear that matrices $\mathbf{Y}_i$ can be used to specify maps between pairs of shapes that are not neighbors in the original network:

$$\mathbf{C}_{ij} = \mathbf{Y}_j^{+}\mathbf{Y}_i, \quad \forall (i,j) \notin \mathcal{G}. \tag{5.2}$$

If we now let $\mathbf{C}$ be a big matrix that encodes the pair-wise map matrices in blocks, then we can express the relation between $\mathbf{C}$ and matrices $\mathbf{Y}_i$ as

$$\mathbf{C} := \begin{pmatrix} \mathbf{C}_{11} & \cdots & \mathbf{C}_{N1} \\ \vdots & \ddots & \vdots \\ \mathbf{C}_{1N} & \cdots & \mathbf{C}_{NN} \end{pmatrix} = \begin{pmatrix} \mathbf{Y}_1^{+} \\ \vdots \\ \mathbf{Y}_N^{+} \end{pmatrix} \begin{pmatrix} \mathbf{Y}_1 & \cdots & \mathbf{Y}_N \end{pmatrix}. \tag{5.3}$$

Joint map optimization. Adopting the robust principal component analysis framework [CLMW11, WS13], we formulate the following convex program to compute the map matrix:

$$\mathbf{C}^{\star} = \min_{\mathbf{C}} \lambda \|\mathbf{C}\|_{\star} + \sum_{(i,j)\in\mathcal{G}} \|\mathbf{C}_{ij}\mathbf{A}_i - \mathbf{A}_j\|_{2,1}. \tag{5.4}$$

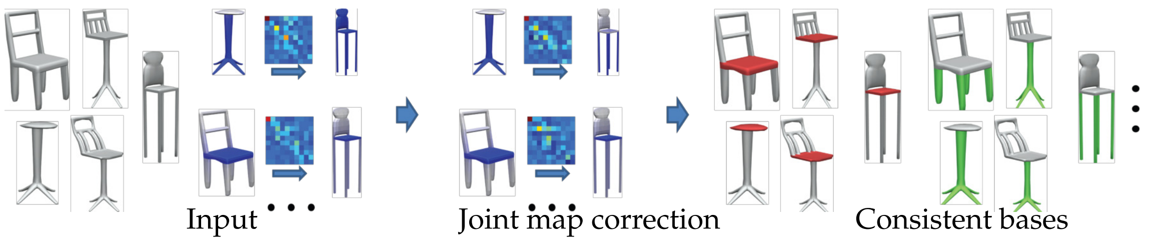


Figure 5.2: **Map Computation Pipeline.** The single-level map construction algorithm consists of two steps. The first step takes a shape collection and (noisy) initial functional maps between pairs of 3D shapes as input and solves a low-rank matrix recovery problem to correct pair-wise maps. The second step then extracts consistent basis functions from optimized pair-wise maps.

The objective function essentially consists of two types of matrix norms. The first component is called the trace-norm defined as $\|\mathbf{C}\|_\star = \sum_i \sigma_i(\mathbf{C})$, where $\sigma_i(\mathbf{C})$ are singular values of $\mathbf{C}$. As discussed in depth in [CLMW11], the trace-norm is a convex proxy for the rank of a matrix. The second component utilizes the $L_{2,1}$ norm, i.e., $\|\mathbf{A}\|_{2,1} = \sum_i \|\mathbf{a}_i\|$, where $\mathbf{a}_i$ are columns of matrix $\mathbf{A}$. This $L_{2,1}$ norm, which is a special group-lasso objective [YL06], has the effect that the optimal value of $\mathbf{C}$ is insensitive to outlier functional correspondences. An alternative is to use the element-wise L_1 norm.

Orthogonal basis synchronization. Assuming that the functional maps are area-preserving and thus represented by orthogonal matrices $\mathbf{C}_{ij} = \mathbf{Y}_j^\top \mathbf{Y}_i$, cycle consistency is automatically guaranteed. In this setting, problem (5.4) can be posed as optimization on the product of Stiefel manifolds [KGB16]

$$\min_{\mathbf{Y}_1,\ldots,\mathbf{Y}_N} \sum_{(i,j)\in\mathcal{G}} \|\mathbf{Y}_i\mathbf{A}_i - \mathbf{Y}_j\mathbf{A}_j\|_{2,1} \;\; \text{s.t.} \;\; \mathbf{Y}_i^\top \mathbf{Y}_i = \mathbf{I}, \tag{5.5}$$

possibly with additional regularization on $\mathbf{Y}_i$ like in the joint diagonalization problem. Due to non-smoothness of the data term, optimization is carried out using Manifold ADMM method [KGB16]. Problem (5.5) can be interpreted as a ‘synchronization’ of orthogonal bases.

5.3 Metrics and Shape Differences

Comparing deformations is a fundamental operation in shape collection. For this purpose we introduce two difference operators acting on functions describing the deformation undergone by the metric. They take into account two orthogonal local distortion: the change in the area and the change in the angle. Interestingly those shape difference operators act on functions on the reference shape and produce functions on the same shape. This simple property allows us to perform deformation comparison and visualization.

Moreover this description is provably comprehensive as the distorted metric can be recovered from the shape operators so the shape difference can also be used for shape synthesis. Given two operators and a reference shape it is possible to retrieve the shape realizing this deformation.

Shape Difference Operators Introduced in [ROA+13], the shape difference operators describe a shape deformation by considering the change of two inner products between functions. Namely, given a pair of shapes $\mathcal{M}, \mathcal{N}$ and a diffeomorphism $T : \mathcal{N} \to \mathcal{M}$, with the associated linear functional map (pullback) defined by $T_F(f) = f \circ T$, the authors introduce the area-based and conformal shape difference operators D_A and D_C respectively, as linear operators acting on (and producing) real-valued functions on $\mathcal{M}$ implicitly via the following equations:

$$\langle f, D_A(g)\rangle_{L^2(\mathcal{M})} := \langle T_F(f), T_F(g)\rangle_{L^2(\mathcal{N})} \qquad \forall f, g \tag{5.6}$$

$$\langle f, D_C(g)\rangle_{H_0^1(\mathcal{M})} := \langle T_F(f), T_F(g)\rangle_{H_0^1(\mathcal{N})} \qquad \forall f, g \tag{5.7}$$

where the inner products are defined as $\langle f, g\rangle_{L^2(\mathcal{M})} := \int_\mathcal{M} fg \mathrm{d}\mu$ and $\langle f, g\rangle_{H_0^1(\mathcal{M})} := \int_\mathcal{M} \langle \nabla f, \nabla g\rangle \mathrm{d}\mu$.

The existence and the linearity of the operators D_A and D_C is guaranteed by the Riesz representation theorem. As shown in [ROA+13], for smooth surfaces, the map T is area-preserving (resp. conformal) if and only if D_A (resp. D_C) is the identity map between functions. From this it follows that T is an isometry if and only if D_A and D_C are *both* identity. Those two operators provide a comprehensive description of intrinsic deformations.

To illustrate the properties of the shape differences we use a simple low-dimensional description of a shape collection. Here we choose a fixed base shape and compute the shape difference matrices with respect to the remaining shapes in a collection. Then, we represent each shape by its shape difference matrix and plot them as points in PCA space. Figure 5.3 represents the conformal deformation of a bunny into a sphere as viewed by the two shape differences. As expected the conformal shape difference is almost identity while the area and isometric shape differences both capture the distortion. In the second experiment, shown in Figure 5.3, we explore another collection obtained by the shearing of a plane patch. As this deformation is area preserving, the area-based shape difference provides no information, unlike the conformal shape difference.

Reconstruction The operator-based representation of metric distortion is not only useful for collection analysis, it can be used for deformation synthesis. At the moment this is done by analyzing the discrete operator for reconstructing triangle meshes.

In a special case when the surfaces $\mathcal{M}$ and $\mathcal{N}$ are triangle meshes with identical connectivity, the functional map $\mathbf{C}_T$ is simply the identity matrix. Therefore, we obtain the following discrete shape differences ([ROA+13] Option 1):

$$\mathbf{D}_A = A_\mathcal{M}^{-1} A_\mathcal{N}, \qquad \mathbf{D}_C = W_\mathcal{M}^{-1} W_\mathcal{N},$$

where A is a diagonal mass matrix and W is a stiffness matrix [Rus07]. According to [ZGLG12], $\mathbf{D}_C$ is the identity matrix if and only if $\mathcal{M}$ and $\mathcal{N}$ have the same edge length up to global scaling. Adding that the area-based shape difference is also identity implies that both meshes share the same discrete metric. Interestingly the continuous statement remains true in the discrete setting.

In [BEKB15] the authors consider the mass and stiffness matrices as functions of the edge lengths ℓ. They are able recover the discrete metric of the deformed mesh given the shape differences $\mathbf{D}_A, \mathbf{D}_C$ by minimizing the energy:

$$\min_\ell \|A_\mathcal{M}^{-1} A(\ell) - \mathbf{D}_A\|_F^2 + \|W_\mathcal{M}^{-1} W(\ell) - \mathbf{D}_C\|_F^2.$$

The embedding is then computed through a *Multidimensional scaling* (MDS) problem (see Figure 5.4).

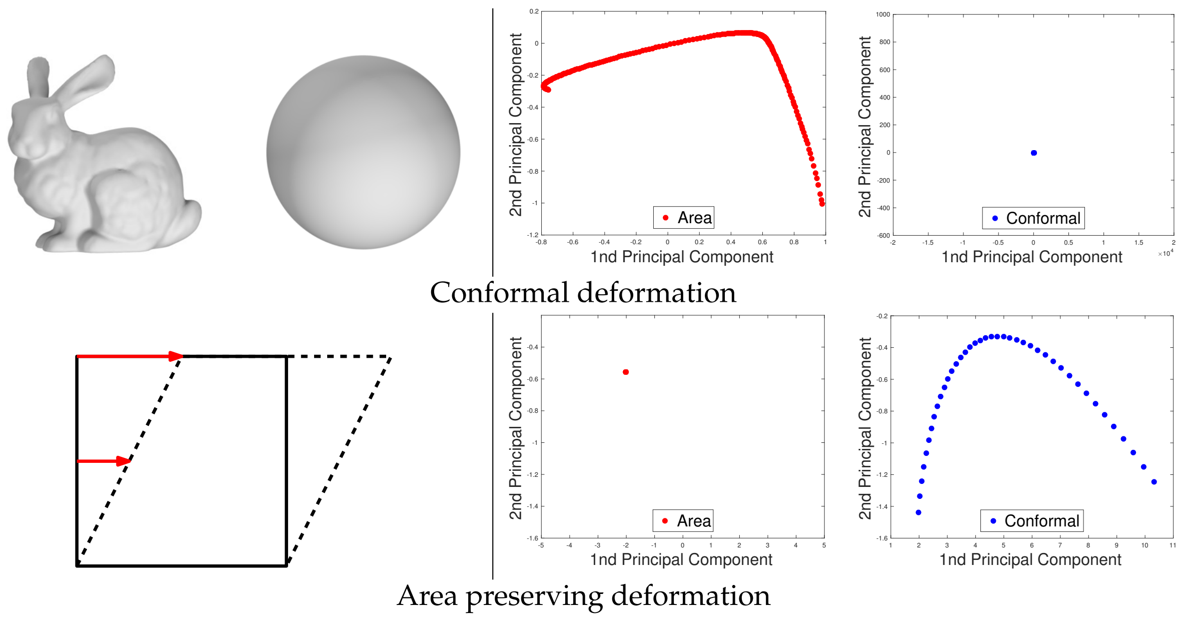


Figure 5.3: Top row: Approximately conformal deformation of a bunny into a sphere. The PCA applied to shape differences confirms the presence of large area distortion in contrast to small conformal distortion. Bottom row: Area preserving deformation of a plane. The area shape difference is almost constant.

However the extrinsic curvature (second fundamental form) is not encoded in this setup leading to possible multiple surfaces with identical metric. To add this information an extension of the shape difference operators is proposed in [CSBC$^+$17] using offset surfaces to capture extrinsic distortion, complementing the purely intrinsic nature of the original shape differences. The authors demonstrate that a set of four operators is complete, capturing intrinsic and extrinsic structure and fully encoding a shape up to rigid motion in both discrete and continuous settings. Moreover, they show that in the presence of full information (without loss due to the basis truncation) the discrete metric (edge lengths) can be obtained by solving two linear systems of equations, and provide a convex optimisation method to approximate the edge lengths given shape difference operators expressed in a reduced basis.

5.4 Applications

Analysis of shape collection arises in many applications in computer graphics.

Shape matching Considering networks of maps has a direct application in improving correspondences between shapes by introducing new constraints such as cycle consistency and identifying outliers in a set of features [COC14, HWG14, KGB16].

Shape collection analysis Shape differences offer the possibility of comparing deformations and therefore finding similar deformation within a shape collection. Moreover it provides a low-dimensional embedding and a notion of distance between 3D models in a shape space [ROA$^+$13, CSBC$^+$17].

Network of maps can be used in for co-segmentation by propagating segments in a collection [HWG14, WHOG14].

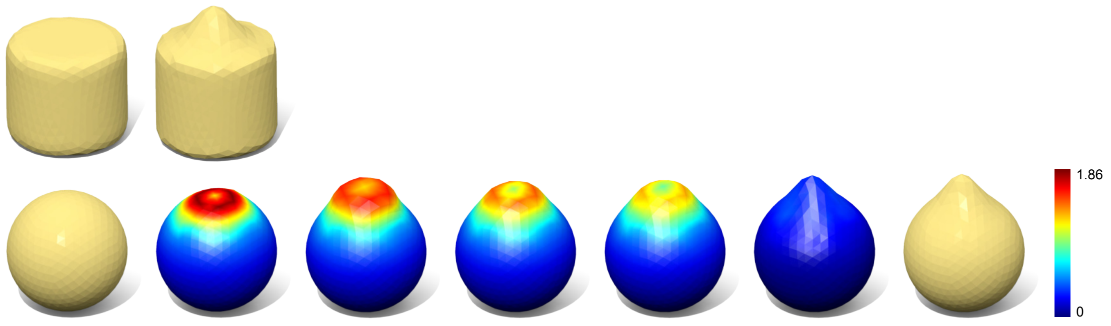


Figure 5.4: Shape analogy synthesis. Given a shape difference operator between the top two shapes (cylinder and cylinder with bump) and another shape (sphere), a new object (sphere with bump) is synthesized. Color code shows the point-wise error during the optimization process.

Shape exploration Shape differences can be easily "edited" by standard linear algebra tools. So, the ability of converting operators to embedded surfaces open the possibility of exploring the space of possible intrinsic deformation by interpolating and extrapolating shapes [CSBC+17].

5.5 List of Key Symbols

Symbol	Definition
$\mathcal{M}, \mathcal{N}$	Shapes (in most cases assumed to be either smooth surfaces, or manifold meshes).
$\mathcal{F}(\mathcal{M}, \mathbb{R})$	Space of real-valued functions on shape $\mathcal{M}$
T_F	Functional representation of a given pointwise map T
$\mathbf{C}$	Functional map expressed as a matrix in a given basis.
Δ	Laplace-Beltrami operator on a surface
$\mathbf{\Lambda}$	diagonal matrix of eigenvalues of the mesh Laplacian
$A_{\mathcal{M}}$	diagonal matrix of area weights on a mesh $\mathcal{M}$
$W_{\mathcal{M}}$	stiffness matrix of the cotangent mesh Laplacian on shape $\mathcal{M}$
$\mathbf{\Phi}$	Functional basis (matrix containing basis functions as columns)
$A_{\mathcal{M}}$	Diagonal matrix of area weights on shape $\mathcal{M}$.
F, G	Function preservation constraints (each column corresponding to a function).
$\mathbf{A}, \mathbf{B}$	Function preservation constraints represented as matrices in a given basis.

6

Functional Vector Fields

The advantages of representing geometric objects as linear operators on scalar functions go beyond maps and correspondences. In this chapter, we will discuss the functional representation of *tangent vector fields*, which are closely related to families of *self-maps*.

Representing tangent vector fields in the discrete setting is a challenging task. The most intuitive representation, namely assigning a single Euclidean vector to each simplex of a polygonal mesh requires careful tracking of the relationships between the tangent spaces at different points, thus complicating tasks such as vector field design and manipulation. For maps we know that the functional map T_F holds the same information as the map T, while being easier to work with since it is a linear operator. Similarly, given a tangent vector field V on $\mathcal{M}$, we can ask whether there exists a linear operator V_F which acts on scalar functions on $\mathcal{M}$, such that V_F completely represents V.

Interestingly, the *directional derivative* operator, which takes a smooth function to its directional derivative in the direction of V, is exactly such an operator. Similarly to the way we defined functional maps, we define a *functional vector field* (FVF) as the linear operator V_F given by: $g = V_F(f) = \langle \nabla f, V \rangle$, where the inner product is defined pointwise in the tangent space of each point on M [ABCCO13]. On the one hand it is easy to see that V_F is linear, and on the other hand it is well known in differential geometry that V_F exactly specifies V. Intuitively, if we know the action of V_F on any function f, then at any point p on M we can find the projection of V on two directions of the tangent plane at p, which is enough to reconstruct V.

As we did for functional maps, given a choice of basis $\phi_{\mathcal{M}}$ for scalar functions on $\mathcal{M}$, we can represent the functional vector field V_F as a matrix D_V (see Figure 6.1). So far, two

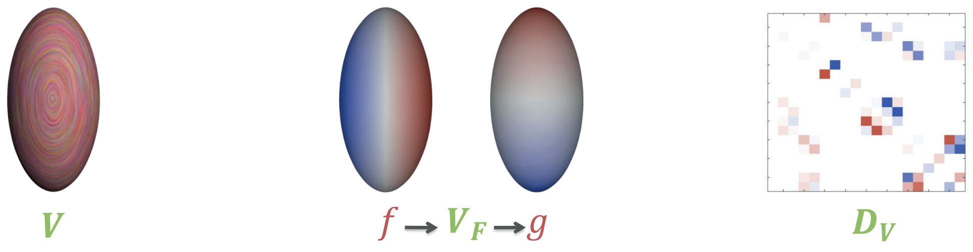


Figure 6.1: A tangent vector field V (visualized using LIC) can be represented as a *functional vector field* V_F: an operator which takes a smooth function $f : \mathcal{M} \to \mathbb{R}$ to the function $g = \langle \nabla f, V \rangle$. Given a choice of basis for functions on $\mathcal{M}$, the functional vector field can be concisely represented as a matrix D_V.

possible choices of basis have been considered: the piecewise linear hat basis functions, commonly used in finite elements, and the eigenvectors of the discrete Laplace-Beltrami operator. In the hat basis D_V is given by a large sparse matrix, whereas using the spectral basis D_V is a small dense matrix.

Equipped with this representation we can tackle classic vector field processing tasks (such as vector field design and fluid simulation) from a new perspective. Instead of working *locally* (or pointwise), e.g. specifying the value of the vector field at a point, we can now work *globally*. For example, for vector field design, we can optimize for a vector field which *commutes* with a given map (e.g. a symmetry map) [ABCCO13]. For fluid simulation, we can use the sparse matrix representation of the FVF to succinctly represent the *flow* of the vector field [AWO+14].

In the following we briefly outline a few useful properties of FVFs. We refer the reader to the original papers for the details and the proofs.

6.1 From Vector Fields to Maps

The Flow Map. Consider a particle at a point p on $\mathcal{M}$ (see Figure 6.2). Given a tangent vector field V, we can ask where will this particle be after time t, if its velocity is given by V. Such a particle will trace a *flow line* of V on $\mathcal{M}$. If we instead consider *all* the the points on $\mathcal{M}$ as the starting points, for every time $t \in \mathbb{R}$ we have new positions which define a self-map on $\mathcal{M}$, known as the *flow-map*. Formally, we define a *one-parameter family* of self maps, $\varphi : \mathbb{R} \times \mathcal{M} \to \mathcal{M}$, which fulfills $\frac{\partial \varphi}{\partial t}(t,p) = V(\varphi(t,p)), \varphi(0,p) = p$. We will often use $\varphi_t(p)$ instead of $\varphi(t,p)$ to denote a single map.

The Functional Flow Map. As with any map, we can think of φ_t as the "transporter" of quantities, and define the corresponding *functional flow map* using composition with the inverse of the flow map: $\varphi_{tF}(f) = f \circ \varphi_{-t}$. The functional flow map naturally describes the transport of a function (any scalar quantity) under the flow of a vector field (see e.g. Figure 6.3), and it is therefore especially meaningful when V describes the velocity of a fluid.

In previous chapters, we computed the functional map either directly from a point-to-point map, or by inferring it using constrained optimization. In this respect, the functional flow map is special, as it can be computed directly from the functional vector field, bypassing the need to compute the point-to-point flow map. This property is quite valuable, since often the flow map is used *solely* for transporting values, and therefore working with the functional flow map is both easier and more natural.

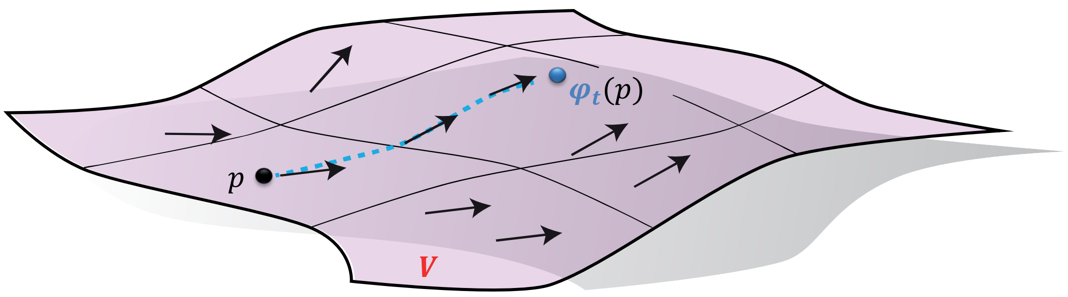


Figure 6.2: A tangent vector field V and its flow φ_t.

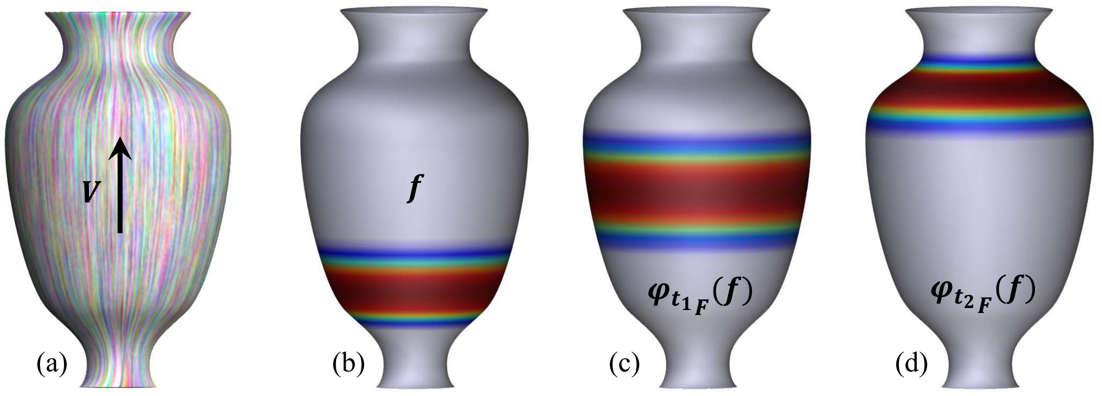


Figure 6.3: Transporting a function (b) on the flow of a vector field (a) (visualized using Line Integral Convolution), for two times t_1, t_2 (c,d).

Computing the Functional Flow. Given an initial function f_0, the functional flow map generates a time-varying function $f : \mathbb{R} \times M \to \mathbb{R}$, given by: $f(t) = \varphi_{tF}(f_0)$. Then f fulfills: $\frac{\partial f}{\partial t} = -\langle \nabla f, V \rangle = -V_F(f), f(0) = f_0$.

We can now consider the space-discrete case, where $\mathcal{M}$ is given as a triangle mesh, and f is represented using a finite set of basis functions. Now the equation becomes $\frac{\partial \mathbf{f}}{\partial t} = -D_V \mathbf{f}_0$, where $D_V, \mathbf{f}, \mathbf{f}_0$ are the representations of V_F, f, f_0 in the chosen basis. Interestingly, this differential equation has a *closed form solution* [HO10], given by: $\mathbf{f}(t) = \exp(-tD_V)\mathbf{f_0}$, where $\exp$ is the matrix exponential. Therefore, the matrix which corresponds to the functional flow is $\exp(-tD_V)$. In practice, there is no need to compute the full exponential of the matrix, it is enough to compute the application of the matrix exponential to a vector, which can be done using efficient algorithms [AMH11].

6.2 Properties.

Functional vector fields and their corresponding functional flow maps have some interesting properties, of which we briefly mention three. Other properties such as a discrete version of the uniqueness property, and discrete integration by parts are discussed in [ABCCO13].

Pushforward. Given a bijective map $T : \mathcal{M} \to \mathcal{N}$, and a vector field V on $\mathcal{M}$, the functional pushforward of V to $\mathcal{N}$ is given by $T_F \circ V_F \circ T^{-1}{}_F$ [ABCCO13]. This property can be used to *jointly* design vector fields on two shapes which correspond under a given map.

Commutation. If D_V commutes with a matrix D, namely $D_V \circ D = D \circ D_V$, then so does its flow for any $t \in \mathbb{R}$. This is straightforward to see, as the matrix exponential is defined as the sum of powers of D_V. This relation is in fact stronger, and holds in both directions: the FVF commutes with D if and only if its flow commutes with D for any t [ABCCO13]. This property can be used to design vector fields whose flow is an isometry (known as Killing vector fields), by constraining D_V to commute with the Laplace-Beltrami operator.

Reconstruction. Given D_V on $\mathcal{M}$ it is possible to reconstruct (an approximation of) V by taking the projection of $(D_V\mathbf{x}, D_V\mathbf{y}, D_V\mathbf{z})$ to the tangent space of M, where $\mathbf{x}$ is the

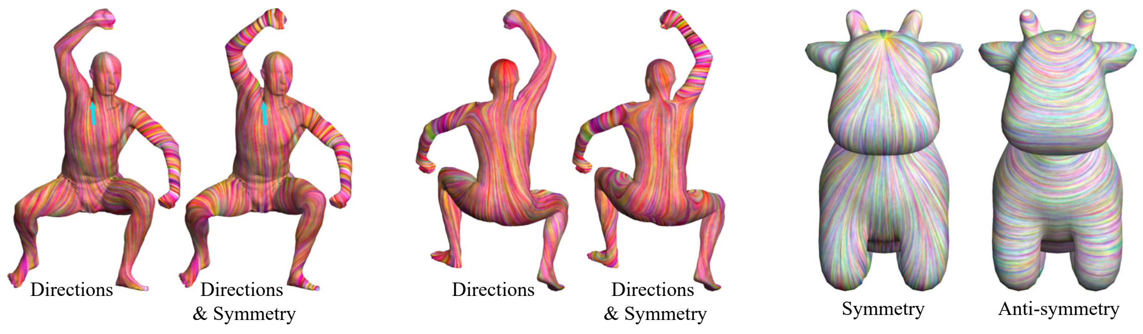


Figure 6.4: Designing symmetric and anti-symmetric vector fields by optimizing for a functional vector field which commutes (or anti-commutes) with an intrinsic symmetry map. Note the symmetric behavior on the hands of the SCAPE model when symmetry constraints are enforced in addition to directional constraints.

representation in the chosen basis of the x coordinate of M, and similarly for $\mathbf{y}, \mathbf{z}$. Depending on the chosen function spaces, this might require interpolation (see [AVBC16]).

6.3 Applications

Vector Field Design. The functional formulation allows to optimize for a vector field which fulfills intricate global constraints, such as commutation with a symmetry map and isometric flow. Similarly to computing maps, we require a regularizer on the structure of the matrix D_V as not any matrix corresponds to a vector field. However, for vector fields the situation is simpler, as they form a linear space. Thus, we can define a basis which spans a subspace of tangent vector fields, and define the optimization problem in terms of this basis. Figure 6.4 shows the result for designing a symmetric and anti-symmetric vector field by requiring commutation and anti-commutation with a pre-computed intrinsic bilateral symmetry map.

Cross Field Design. The functional formulation of vector fields can be generalized to *cross fields*, or more generally to *N-RoSy fields* [ACBCO17], by considering a modified functional operator $D_V(f)$. This operator is no longer linear in its function parameter f, yet it remains linear in the vector field V, and thus can be used for cross field design. Figure 6.5 demonstrates the application of consistent functional cross field design

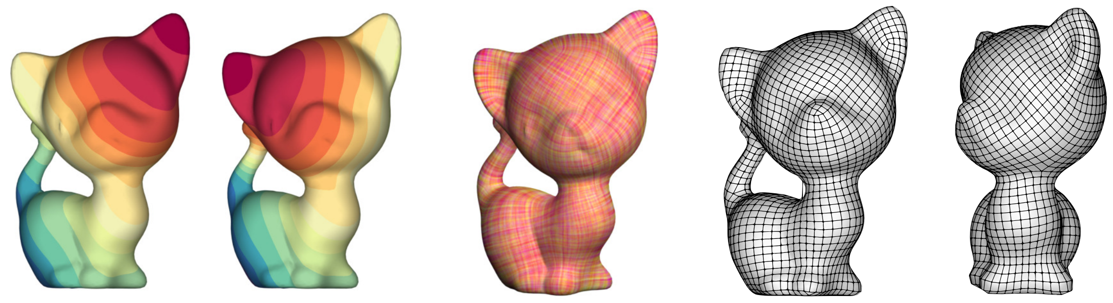


Figure 6.5: Given an input symmetry functional map (left) a consistent cross field is computed by optimizing for operator commutativity (center). An approximately consistent quadrangular mesh (right) is generated from the cross field using standard tools.

for mesh quadrangulation. Given an input symmetry functional map (left), a consistent cross field is computed by requiring that its functional operator commutes with the functional map, when applied to a small subset of smooth functions. In addition, the cross field is required to be smooth and to align with the curvature directions of the surface. This yields a a smooth consistent cross field (center), from which an approximately consistent quadrangular mesh (right) can then be extracted. This is done using the standard pipeline [BLP+13]: first a griddable parameterization which aligns with the cross field directions is computed with MIQ [BZK09] and then a quadrangular mesh is extracted from the parameterization with QEx [EBCK13]. Note that, in this case, the cross field should be designed in the full basis, to allow for the detailed alignment with the curvature directions of the surface.

Fluid Simulation. The ability to compute the functional flow through the matrix exponential is highly valuable in applications where transfer of quantities is required. For example, in fluid simulation, two different fluid models (Euler fluids [AWO+14] and viscous thin films [AVW+15]) can be formulated in terms of the functional flow. Figure 6.6 shows examples of the resulting transported functions which represent (a) vorticity (local spinning of the fluid) and (b) mass density. In (b) we use the density function as a height function, and render the resulting offset surface. In both cases, the operator representation of vector fields allows to utilize complex optimization schemes, which would have been considerably more difficult to implement using the point-to-point flow map.

Self-Map Inference. If we are looking for a smooth self-map, it is possible to constrain the map to be the flow of a vector field (or a composition of flows). Then, the optimization problem is specified in terms of the functional vector fields, instead of directly in terms of the map. In [AVBC16] the authors used this idea to interpolate between functions on the same surface by computing a flow map which transports the source function to the target function. The functional flow map can then be used for interpolation or extrapolation, by transporting the source function for various times t. Figure 6.7 shows how this approach can be used to infer a continuous intrinsic symmetry map from only a function and its image under one instance of the symmetry map. See also chapter 7.6 for an application of self-map inference to map improvement.

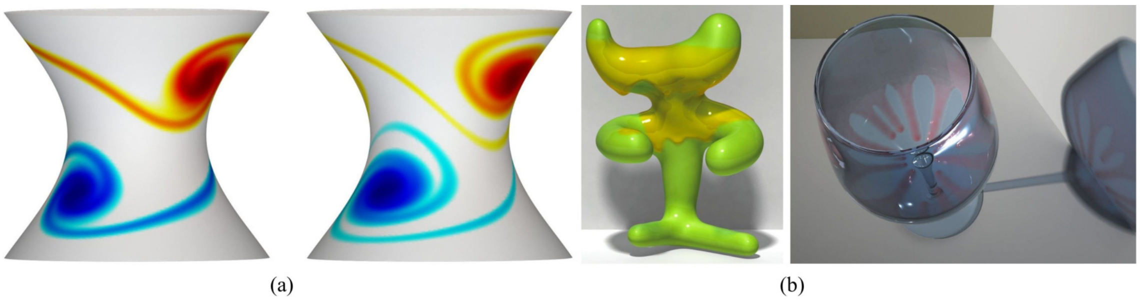


Figure 6.6: Using the functional flow for fluid simulation. (a) Simulating turbulent flow by transporting the vorticity of the fluid. (b) Simulating viscous thin film flow by transporting the mass density (visualized as an offset surface).

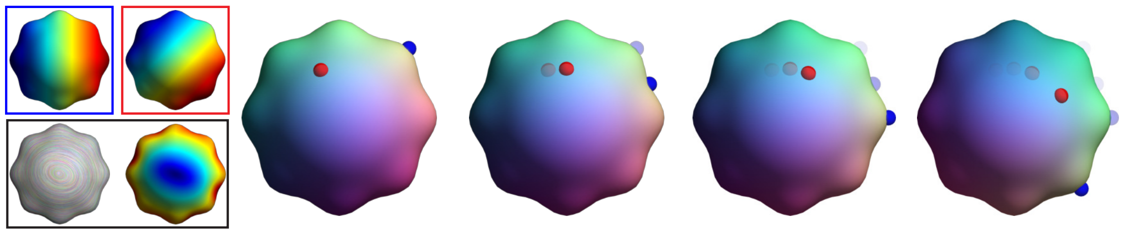


Figure 6.7: Using the functional flow for function matching and inferring a continuous intrinsic symmetry map. (top left) Inputs: source and target functions to match. (bottom left) Outputs: the vector field (direction and norm) whose flow transports the source to the target. (right) Transporting the source function using the computed flow.

6.4 List of Key Symbols

Symbol	Definition
V_F	Functional representation of a given tangent vector field V
D_V	Functional vector field expressed as a matrix in a given basis.
$\varphi_t : \mathcal{M} \to \mathcal{M}$	The flow map of a vector field

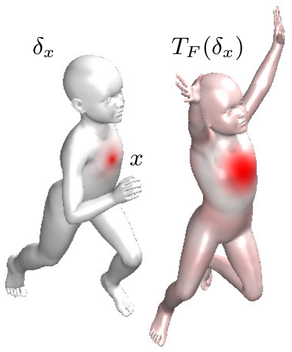
7

Map Conversion

7.1 Converting Functional Maps to Pointwise Maps

The functional map representation greatly simplifies correspondence-based tasks. If the harmonic basis is truncated to k basis functions, the shape matching problem boils down to solving for k^2 unknowns, where k is possibly very small (typically in the range $20 - 300$). At the same time, the truncation has the effect of 'low-pass' filtering, thus producing smooth correspondences. In many applications, however, it is desirable to reconstruct the point-to-point mapping induced by the functional map. Thus, the interest shifts to the *inverse* problem of recovering the map T from its functional representation T_F.

The simplest and most direct way for reconstructing the bijection T from the associated functional map T_F consists in mapping highly peaked Gaussian functions δ_x for each point $x \in \mathcal{M}$ via T_F, obtaining the image $g = T_F(\delta_x)$, and then declaring $T(x) \in \mathcal{N}$ to be the point at which g attains the maximum [OBCS+12]. Such a method, however, suffers from at least two drawbacks. First, it requires constructing and mapping indicator functions for all shape points, which can get easily expensive for large meshes (several thousand vertices). Second, the low-pass filtering due to the basis truncation has a delocalizing effect on the maximum of g, negatively affecting the quality of the final correspondence (see inset).



Assume now that shapes $\mathcal{M}$ and $\mathcal{N}$ have n and m points respectively, and as in the previous sections, let the matrices $\mathbf{\Phi}_\mathcal{M} \in \mathbb{R}^{n \times k}$, $\mathbf{\Phi}_\mathcal{N} \in \mathbb{R}^{m \times k}$ contain the first k eigenfunctions of the respective Laplacians. Further, let matrix $\mathbf{P} \in [0,1]^{n \times m}$ encode the map $T : \mathcal{M} \to \mathcal{N}$. In the general case where $m \neq n$, the map T can be modelled as a *soft assignment*, i.e., by regarding T as a scalar function $T : \mathcal{M} \times \mathcal{N} \to [0,1]$ assigning a value of confidence to each possible match $(x,y) \in \mathcal{M} \times \mathcal{N}$, and setting $\mathbf{P}_{ij} = T(x_i, y_j)$. The expression for $\mathbf{C}$, the spectral representation of the functional map built upon T, can be compactly written as

$$\mathbf{C} = \mathbf{\Phi}_\mathcal{N}{}^\top \mathbf{P}^\top \mathbf{\Phi}_\mathcal{M} \,. \tag{7.1}$$

Note that, despite our specific choice of a basis, the expression above holds for any choice of orthogonal bases $\{\phi_i^\mathcal{M}\}, \{\phi_j^\mathcal{N}\}$; further, matrix $\mathbf{C}$ can be seen as a rank-k approximation of $\mathbf{P}$. For the particular case in which $n = m$ and the underlying map T is a bijection, the pointwise map recovery problem consists in finding a $n \times n$ permutation matrix $\mathbf{\Pi}$ satisfying (7.1). This can be conveniently phrased as a linear assignment problem (LAP), as discussed in the following section.

7.2 Linear Assignment Problem

Consider the simple case in which $n = m$ and $k = n$, i.e., the expression in (7.1) corresponds to an orthogonal change of basis (with no truncation). Since any such change of basis preserves the rank of the transformation, the relation can be directly inverted to obtain $\mathbf{\Pi}^\top = \mathbf{\Phi}_\mathcal{N}\mathbf{C}\mathbf{\Phi}_\mathcal{M}{}^\top$. In the more realistic setting in which $k \ll n$, we can obtain the best possible solution (in the ℓ^2 sense) by looking for a permutation $\mathbf{\Pi}$ that minimizes the linear assignment problem:

$$\min_{\mathbf{\Pi}\in\{0,1\}^{n\times n}} -\langle \mathbf{\Pi}^\top, \mathbf{\Phi}_\mathcal{N}\mathbf{C}\mathbf{\Phi}_\mathcal{M}{}^\top\rangle_F \quad \text{s.t. } \mathbf{\Pi}^\top\mathbf{1} = \mathbf{1}\,, \mathbf{\Pi}\mathbf{1} = \mathbf{1}\,. \tag{7.2}$$

Although this is a linear problem that can be solved in polynomial time [Kuh55], seeking for such a solution can easily become prohibitive in practice for large meshes (n in the order of several thousands). In practice, several more efficient variants are possible that relax the bijectivity constraint on the underlying map, as described in the following sections. We will return to the bijective recovery problem in Section 7.5.

7.3 Nearest Neighbors

A more affordable way to recover the pointwise map $\mathbf{P}$ from its spectral representation can be obtained as a straightforward relaxation to the LAP. We start by deriving an equivalent expression for the objective in (7.2), namely $-2\langle \mathbf{\Pi}^\top, \mathbf{\Phi}_\mathcal{N}\mathbf{C}\mathbf{\Phi}_\mathcal{M}{}^\top\rangle_F = \|\mathbf{C}\mathbf{\Phi}_\mathcal{M}{}^\top - \mathbf{\Phi}_\mathcal{N}{}^\top\mathbf{\Pi}^\top\|_F^2 - \|\mathbf{C}\mathbf{\Phi}_\mathcal{M}{}^\top\|_F^2 - \|\mathbf{\Phi}_\mathcal{N}{}^\top\|_F^2$, holding for permutation matrices $\mathbf{\Pi}$. Minimizing with respect to $\mathbf{\Pi}$, we get an equivalent formulation for the LAP:

$$\min_{\mathbf{\Pi}\in\{0,1\}^{n\times n}} \|\mathbf{C}\mathbf{\Phi}_\mathcal{M}{}^\top - \mathbf{\Phi}_\mathcal{N}{}^\top\mathbf{\Pi}^\top\|_F^2 \quad \text{s.t. } \mathbf{\Pi}^\top\mathbf{1} = \mathbf{1}\,, \mathbf{\Pi}\mathbf{1} = \mathbf{1}\,. \tag{7.3}$$

This problem admits an intuitive interpretation. If we denote by $\mathbf{e}_i$ the indicator vector having the value 1 in the ith position and 0 otherwise, we see that each column of $\mathbf{\Phi}_\mathcal{M}{}^\top$ contains the spectral coefficients $\mathbf{\Phi}_\mathcal{M}{}^\top\mathbf{e}_i$ of delta functions supported at $x_i \in \mathcal{M}$. Hence, $\mathbf{C}\mathbf{\Phi}_\mathcal{M}{}^\top$ contains as its columns the images (via T) of all delta functions on $\mathcal{M}$. Problem (7.3) can then be interpreted as seeking for a permutation $\mathbf{\Pi}$ that minimizes the ℓ^2 distance between columns of $\mathbf{C}\mathbf{\Phi}_\mathcal{M}{}^\top$ and columns of $\mathbf{\Phi}_\mathcal{N}{}^\top\mathbf{\Pi}^\top$.

By allowing $m \neq n$ (i.e. non-bijective maps) and relaxing the permutation constraint to (binary) column-stochasticity, we get to the recovery problem considered in [OBCS+12]:

$$\min_{\mathbf{P}\in\{0,1\}^{n\times m}} \|\mathbf{C}\mathbf{\Phi}_\mathcal{M}{}^\top - \mathbf{\Phi}_\mathcal{N}{}^\top\mathbf{P}^\top\|_F^2 \quad \text{s.t. } \mathbf{P}^\top\mathbf{1} = \mathbf{1}\,. \tag{7.4}$$

This problem can be solved *globally* and efficiently by a simple *nearest-neighbor* search in k-dimensional space: for each index $j = 1, \dots, m$, if the ith column of $\mathbf{C}\mathbf{\Phi}_\mathcal{M}{}^\top$ is the nearest neighbor (in $\mathbb{R}^k$) to the jth column of $\mathbf{\Phi}_\mathcal{N}{}^\top$, then set $\mathbf{P}_{ij} = 1$ (note that going in the other direction, i.e., searching for nearest neighbors for each column of $\mathbf{C}\mathbf{\Phi}_\mathcal{M}{}^\top$, also constitutes a valid global solution to (7.4)).

7.3.1 Orthogonal refinement

By regarding the columns of $\mathbf{\Phi}_\mathcal{M}{}^\top$, $\mathbf{\Phi}_\mathcal{N}{}^\top$ as points in $\mathbb{R}^k$, we may interpret any pointwise map recovery algorithm (including those outlined above) as an attempt to align the k-dimensional *spectral embeddings* of the two shapes [MHK+08], as illustrated in Figure 7.1. In this view, the action of the functional map $\mathbf{C}$ on $\mathbf{\Phi}_\mathcal{M}{}^\top$ can be seen as a pre-alignment

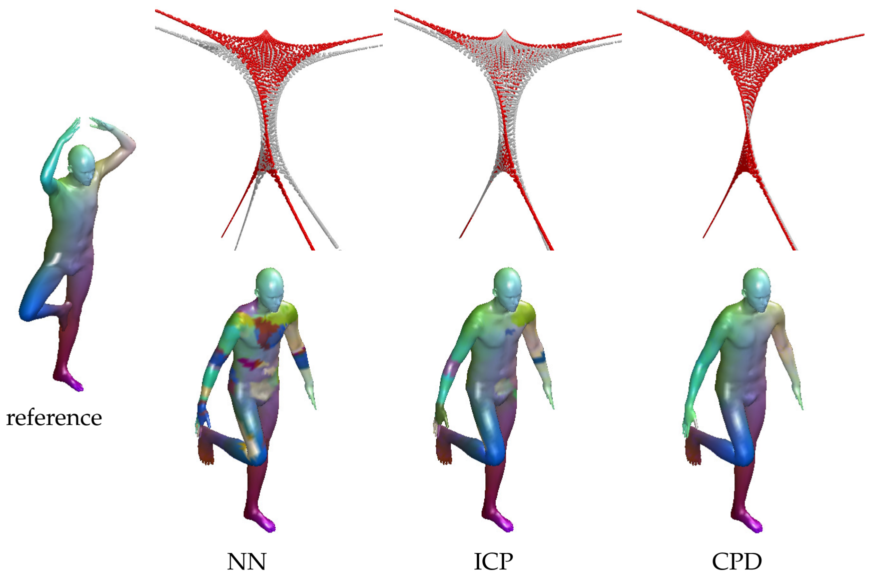


Figure 7.1: The problem of recovering a pointwise map from a functional map can be seen as aligning the spectral embeddings of the two shapes. In the top row we show the first three dimensions of the embeddings for visualization purposes. The given functional map provides the initial alignment, which is further refined by the specific recovery method. Shown here from left to right are nearest neighbors (Sec. 7.3), orthogonal (Sec. 7.3.1) and non-orthogonal (Sec. 7.4) refinement.

of the two embeddings, suggesting the possibility for further refinement by treating the two shapes simply as point clouds in $\mathbb{R}^k$.

With this mindset, in [OBCS+12] it was proposed to consider the orthogonal Procrustes problem:

$$\min_{\mathbf{C}\in\mathbb{R}^{k\times k}} \|\mathbf{C}\mathbf{\Phi}_{\mathcal{M}}{}^\top - \mathbf{\Phi}_{\mathcal{N}}{}^\top\mathbf{P}^\top\|_F^2 \quad \text{s.t. } \mathbf{C}^\top\mathbf{C} = \mathbf{I}\,, \tag{7.5}$$

where the orthogonality constraint on $\mathbf{C}$ implies that the underlying map T is area-preserving (see Chapter 3). A global solution to this problem can be obtained efficiently, and can be interpreted as a k-dimensional rigid alignment of the spectral embedding of $\mathcal{M}$ with the one of $\mathcal{N}$. The $\mathbf{C}$-step (7.5) and the $\mathbf{P}$-step (7.4) are alternated until convergence, in the spirit of the classical Iterative Closest Point algorithm [BM92].

A simple extension of this approach to deal with *partial* functional maps (discussed in Chapter 4) was proposed in [RCB+16], and basically amounts to considering the set of *semi*-orthogonal maps $\mathbf{C}$, i.e., such that $\mathbf{C}^\top\mathbf{C} = \mathbf{I}$ but $\mathbf{C}\mathbf{C}^\top \neq \mathbf{I}$.

7.4 Regularized Map Recovery

The approaches described in the previous section do not incorporate any regularity assumption on the map to be recovered – for example, the natural requirement that the reconstructed pointwise map should be smooth. A first step in this direction was considered in [RMC15]. The authors proposed to consider the density estimation problem:

$$\begin{aligned}\min_{\mathbf{P}\in[0,1]^{n\times m}} \; & D_{\mathrm{KL}}(\mathbf{C}\mathbf{\Phi}_{\mathcal{M}}{}^\top, \mathbf{\Phi}_{\mathcal{N}}{}^\top\mathbf{P}^\top) + \lambda\|\Omega(\mathbf{C}\mathbf{\Phi}_{\mathcal{M}}{}^\top - \mathbf{\Phi}_{\mathcal{N}}{}^\top\mathbf{P}^\top)\|^2 \\ \text{s.t.} \; & \mathbf{P}^\top\mathbf{1} = \mathbf{1}\,,\end{aligned} \tag{7.6}$$

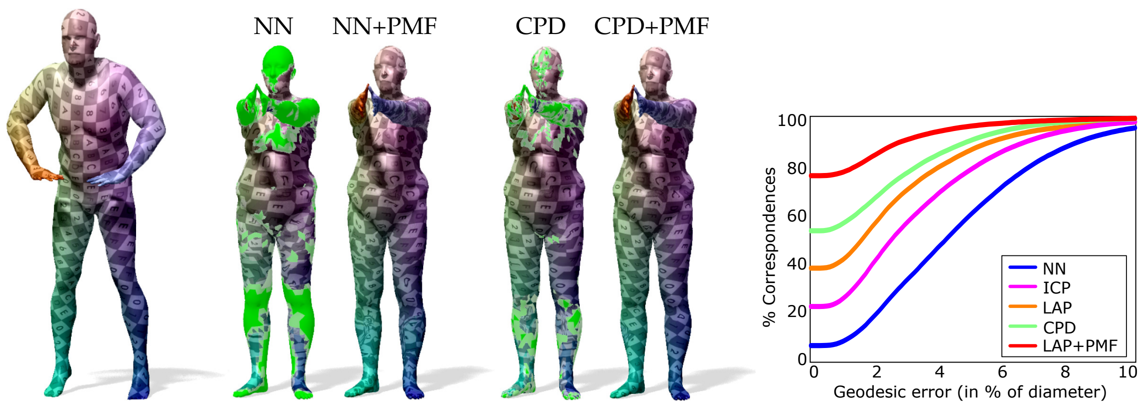


Figure 7.2: Examples of pointwise maps recovered from a functional map matrix of size 20×20. PMF denotes the Product Manifold Filter. The quality of the map is visualized by transferring texture from the reference shape on the left. Green parts correspond to unmatched areas. The plot on the right shows a comparison of the different approaches.

where D_{KL} denotes the Kullback-Leibler divergence between probability distributions, Ω is a low-pass operator promoting smooth vector fields (note that the argument of Ω is a displacement field in $\mathbb{R}^k$), and $\lambda > 0$ controls the regularity of the assignment. The problem can be seen as a Tikhonov regularization of the displacement field relating the two spectral embeddings, where proximity is measured according to the KL divergence between the two.

In other words, this approach is an attempt to cast the map recovery as a probability density estimation problem. Within this model, one can interpret the spectral embedding of $\mathcal{M}$ as modes of a continuous probability distribution defined over $\mathbb{R}^k$, while the embedding of $\mathcal{N}$ constitutes the data, i.e., a discrete sample drawn from the distribution. The task is then to align the modes to the data, such that the point-to-point mapping can be recovered as the maximum posterior probability.

From a geometric perspective, this method can be seen as a *non-rigid* variant of the ICP algorithm described in Section 7.3.1 (see also Figure 7.1 for an example). The non-rigid alignment is performed by the Coherent Point Drift method [MS10], which solves a density estimation problem of the form (7.6). Since the smoothness term in (7.6) tends to match closeby points to closeby points, this induces more natural maps than those obtained by other recovery approaches.

An extension of this approach to partial functional maps was considered in [RMC], by introducing a variable reflecting prior knowledge on the amount of overlap between partial and full shape.

7.5 Product Manifold Filter for Bijective Map Recovery

With the exception of the LAP formulation of Section 7.2, all methods described above can be applied to shapes having different number of points. A novel perspective on the *bijective* map recovery problem was recently proposed in [VLR+17]. Given an initial functional correspondence between two shapes, this can be considered as a noisy realization of a latent bijective correspondence. The key idea behind this method is that such a latent bijection can be recovered by considering an intrinsic equivalent of the standard kernel density estimator (KDE) with Gaussian kernels.

In matrix form, the bijective correspondence estimator of $\mathbf{\Pi}^\top$ given the observation

Algorithm 4: FUNCTIONAL MAP CONVERSION

```
Input  : $\mathbf{C} : L^2(\mathcal{M}) \to L^2(\mathcal{N})$ functional map
         $T^0 : \mathcal{N} \to \mathcal{M}$ initial continuous map
Output: $T_{\mathbf{C}}$: $\mathbf{C}$ converted into a continuous map
1 Convert $T^0$ to a functional map $\mathbf{C}_{T_0}$;
2 Solve: $\mathbf{a}^\star \in \arg\min_{a \in \mathbb{R}^n} \|\mathbf{C}_{T_0} \exp\left(\sum_{i=1}^n \mathbf{a}_i \mathbf{D}_{V_i}\right) - \mathbf{C}\|_\phi$;
3 Set: $V := \sum_{i=1}^n \mathbf{a}_i^\star \mathbf{D}_{V_i}$;
4 Solve: $\frac{\partial}{\partial t}\phi_V^t(p) = V\left(\phi_V^t(p)\right), \ \phi_V^0(p) = p \in \mathcal{N}$;
5 return $T_{\mathbf{C}} := \phi_V^1 \circ T^0$;
```

$\mathbf{P}_0$ can be expressed as the maximizer to the following problem (we refer to [VLR+17] for details):

$$\max_{\mathbf{\Pi} \in \{0,1\}^{n \times n}} \text{trace}(\exp(-\mathbf{D}_{\mathcal{M}}/\sigma^2)\mathbf{P}_0 \exp(-\mathbf{D}_{\mathcal{N}}/\sigma^2)\mathbf{\Pi}) \qquad (7.7)$$
$$\text{s.t.} \quad \mathbf{\Pi}^\top \mathbf{1} = \mathbf{1}\,, \mathbf{\Pi}\mathbf{1} = \mathbf{1}\,,$$

where $\mathbf{D}_{\mathcal{M}}$, $\mathbf{D}_{\mathcal{N}} \in \mathbb{R}^{n \times n}$ are geodesic distance matrices on the respective shapes, exp acts element-wise, and $\mathbf{P}_0$ is an initial (possibly noisy and not necessarily bijective) pointwise correspondence obtained with any of the methods described in the previous sections. Note that problem (7.7) has the structure of a LAP, hence any solution to it is a guaranteed bijection; the authors make use of the auction algorithm [Ber98] in conjunction with a simple multi-scale approach to solve it efficiently.

The recovery process can be reiterated by setting as $\mathbf{P}_0$ the correspondence obtained from the previous step, resulting in an iterative refinement of the initial noisy correspondence. The resulting pointwise map can be seen as a "denoised" version of the input, in analogy with classical KDE-based denoising of images. See Figure 7.2 for qualitative examples, and for a comparison among the different methods.

7.6 Continuous Maps via Vector Field Flows

In [COC15] the authors propose a method for converting a functional map to a point-to-point map, which guarantees continuity and does not rely on any pairwise consistency constraints, making it computationally efficient. The main idea is to represent the target point-to-point map as a composition of an arbitrary continuous map between the two surfaces and a flow associated with an unknown vector field on one of them. By relying on the operator representation of vector fields [ABCCO13], the optimal vector field can be computed efficiently entirely within the functional map framework, and the computation of the final map requires a single discretization of vector field advection.

Algorithm overview The proposed algorithm takes as input a functional map $\mathbf{C} : L^2(\mathcal{M}) \to L^2(\mathcal{N})$ and an arbitrary continuous map $T^0 : \mathcal{N} \to \mathcal{M}$. It then outputs a continuous point-to-point map $T_{\mathbf{C}} : \mathcal{N} \to \mathcal{M}$.

The main idea of the algorithm is to construct the map $T_{\mathbf{C}}$ by composing T^0 with the flow φ_V^t of a well-chosen vector field V. We will choose the vector field V such that $\varphi_V^t \circ T^0$ represented as a functional map is as close as possible to the input $\mathbf{C}$. This can be done efficiently by representing φ_V^t as a functional map, namely $\mathbf{C}_{\varphi_V^t} = \exp(t\mathbf{D}_V)$ (see

[ABCCO13] as well as Chapter 6 of these notes), and then solving a small-scale optimization problem:

$$\min_{\mathbf{a}\in\mathbb{R}^n} \|\mathbf{C}_T \exp\left(\sum_{i=1}^{n} \mathbf{a}_i \mathbf{D}_{V_i}\right) - \mathbf{C}\|_\phi, \tag{7.8}$$

for an appropriate choice of the norm $\|.\|_\phi$. Finally the map T_C is computed by solving a system of ODEs with a simple solver.

The overall algorithm is summarized in Algorithm 4.

7.7 List of Key Symbols

Symbol	Definition
T_F	Functional representation of a given (possibly soft) pointwise map T
$\mathbf{P}$	Pointwise map expressed as a matrix with values in $[0, 1]$
$\mathbf{\Pi}$	Permutation matrix
$\mathbf{C}$	Functional map expressed as a matrix in the Fourier basis
$\mathbf{\Phi}$	Functional basis (matrix containing basis functions as columns)

Bibliography

[ABCCO13] O. Azencot, M. Ben-Chen, F. Chazal Chazal, and M. Ovsjanikov. An operator approach to tangent vector field processing. *Computer Graphics Forum*, 32(5):73–82, 2013.

[ACBCO17] Omri Azencot, Etienne Corman, Mirela Ben-Chen, and Maks Ovsjanikov. Consistent functional cross field design for mesh quadrangulation. *ACM Trans. Graph.*, 36(4):92:1–92:13, 2017.

[AMH11] A. H. Al-Mohy and N. J. Higham. Computing the action of the matrix exponential, with an application to exponential integrators. *SIAM J. Scientific Computing*, 33(2):488–511, 2011.

[AMS09] P.-A. Absil, R. Mahony, and R. Sepulchre. *Optimization algorithms on matrix manifolds*. Princeton University Press, 2009.

[ASC11] M. Aubry, U. Schlickewei, and D. Cremers. The wave kernel signature: A quantum mechanical approach to shape analysis. In *Proc. Workshop on Dynamic Shape Capture and Analysis*, 2011.

[ASK$^+$05] D. Anguelov, P. Srinivasan, D. Koller, S. Thrun, J. Rodgers, and J. Davis. SCAPE: shape completion and animation of people. *ACM Trans. Graphics*, 24(3):408–416, 2005.

[AVBC16] O. Azencot, O. Vantzos, and M. Ben-Chen. Advection-Based Function Matching on Surfaces. *Computer Graphics Forum*, 2016.

[AVW$^+$15] O. Azencot, O. Vantzos, M. Wardetzky, M. Rumpf, and M. Ben-Chen. Functional thin films on surfaces. In *Proc. Symp. Computer Animation*, 2015.

[AWO$^+$14] O. Azencot, S. Weißmann, M. Ovsjanikov, M. Wardetzky, and M. Ben-Chen. Functional fluids on surfaces. *Computer Graphics Forum*, 33(5):237–246, 2014.

[BB08] A. M. Bronstein and M. M. Bronstein. Not only size matters: regularized partial matching of nonrigid shapes. In *Proc. NORDIA*, 2008.

[BBK08] A. M. Bronstein, M. M. Bronstein, and R. Kimmel. *Numerical Geometry of Non-Rigid Shapes*. Springer, 2008.

[BEKB15] D. Boscaini, D. Eynard, D. Kourounis, and M. M. Bronstein. Shape-from-operator: Recovering shapes from intrinsic operators. *Computer Graphics Forum*, 34(2):265–274, 2015.

[Ber98] D. P. Bertsekas. *Network Optimization: Continuous and Discrete Models*. Athena Scientific, 1998.

[BLP$^+$13] David Bommes, Bruno Lévy, Nico Pietroni, Enrico Puppo, Claudio Silva, Marco Tarini, and Denis Zorin. Quad-mesh generation and processing: A survey. In *Computer Graphics Forum*, volume 32, pages 51–76. Wiley Online Library, 2013.

[BM92] P. J. Besl and N. D. McKay. A method for registration of 3-D shapes. *Trans. PAMI*, 14:239–256, 1992.

[BZK09] David Bommes, Henrik Zimmer, and Leif Kobbelt. Mixed-integer quadrangulation. *ACM Transactions On Graphics (TOG)*, 28(3):77, 2009.

[CLMW11] E. J. Candès, X. Li, Y. Ma, and J. Wright. Robust principal component analysis? *J. ACM*, 58(3):1–37, 2011.

[COC14] E. Corman, M. Ovsjanikov, and A. Chambolle. Supervised descriptor learning for non-rigid shape matching. In *Proc. NORDIA*, 2014.

[COC15] E. Corman, M. Ovsjanikov, and A. Chambolle. Continuous matching via vector field flow. *Computer Graphics Forum*, 34(5):129–139, 2015.

[CR09] E. J. Candès and B. Recht. Exact matrix completion via convex optimization. *Foundations of Computational Mathematics*, 9(6):717–772, 2009.

[CRB$^+$16] L. Cosmo, E. Rodolà, M. M. Bronstein, et al. SHREC'16: Partial matching of deformable shapes. In *Proc. 3DOR*, 2016.

[CRM$^+$16] L. Cosmo, E. Rodolà, J. Masci, A. Torsello, and M. Bronstein. Matching deformable objects in clutter. In *Proc. 3DV*, 2016.

[CSBC$^+$17] Etienne Corman, Justin Solomon, Mirela Ben-Chen, Leonidas Guibas, and Maks Ovsjanikov. Functional characterization of intrinsic and extrinsic geometry. *ACM Trans. Graph.*, 36(2):14:1–14:17, 2017.

[EBCK13] Hans-Christian Ebke, David Bommes, Marcel Campen, and Leif Kobbelt. QEx: robust quad mesh extraction. *ACM Transactions on Graphics (TOG)*, 32(6):168, 2013.

[EKB$^+$15] D. Eynard, A. Kovnatsky, M. M. Bronstein, K. Glashoff, and A. M. Bronstein. Multimodal manifold analysis by simultaneous diagonalization of Laplacians. *Trans. PAMI*, 37(12):2505–2517, 2015.

[ERGB16] D. Eynard, E. Rodolà, K. Glashoff, and M. M. Bronstein. Coupled functional maps. In *Proc. 3DV*, 2016.

[GB16] K. Glashoff and M. M. Bronstein. Optimization on the biorthogonal manifold. *arXiv:1609.04161*, 2016.

[HG13] Q. Huang and L. J. Guibas. Consistent shape maps via semidefinite programming. *Computer Graphics Forum*, 32(5):177–186, 2013.

[HO10] M. Hochbruck and A. Ostermann. Exponential integrators. *Acta Numerica*, 19:209–286, 2010.

[HWG14] Q. Huang, F. Wang, and L. J. Guibas. Functional map networks for analyzing and exploring large shape collections. *ACM Trans. Graphics*, 33(4):36:1–36:11, July 2014.

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

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

[KBB$^+$12] A. Kovnatsky, M. Bronstein, A. Bronstein, K. Glashoff, and R. Kimmel. Coupled quasi-harmonic bases. *Computer Graphics Forum*, 32:439–448, 2012.

[KBBV14] V. Kalofolias, X. Bresson, M. M. Bronstein, and P. Vandergheynst. Matrix completion on graphs. *arXiv:1408.1717*, 2014.

[KBBV15] A. Kovnatsky, M. Bronstein, X. Bresson, and P. Vandergheynst. Functional correspondence by matrix completion. In *Proc. CVPR*, 2015.

[KGB16] A. Kovnatsky, K. Glashoff, and M. M. Bronstein. MADMM: A generic algorithm for non-smooth optimization on manifolds. In *Proc. ECCV*, 2016.

[KLF11] V. G. Kim, Y. Lipman, and T. Funkhouser. Blended intrinsic maps. *ACM Trans. Graphics*, 30(4), 2011.

[Kuh55] H. W. Kuhn. The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1–2):83–97, March 1955.

[LBB11] R. Litman, A. M. Bronstein, and M. M. Bronstein. Diffusion-geometric maximally stable component detection in deformable shapes. *Computers & Graphics*, 35(3):549 – 560, 2011.

[LBB12] O. Litany, A. M. Bronstein, and M. M. Bronstein. Putting the pieces together: Regularized multi-part shape matching. In *Proc. NORDIA*, 2012.

[LRB$^+$16] O. Litany, E. Rodolà, A. M. Bronstein, M. M. Bronstein, and D. Cremers. Non-rigid puzzles. *Computer Graphics Forum*, 35(5), 2016.

[LRBB17] O. Litany, E. Rodolà, A. M. Bronstein, and M. M. Bronstein. Fully spectral partial shape matching. *Computer Graphics Forum*, 36(2), 2017.

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

[MS10] A. Myronenko and X. Song. Point set registration: Coherent point drift. *Trans. PAMI*, 32(12):2262–2275, 2010.

[NO17] Dorian Nogneng and Maks Ovsjanikov. Informative Descriptor Preservation via Commutativity for Shape Matching. *Computer Graphics Forum (Proc. Eurographics)*, 2017.

[OBCS$^+$12] M. Ovsjanikov, M. Ben-Chen, J. Solomon, A. Butscher, and L. J. Guibas. Functional maps: a flexible representation of maps between shapes. *ACM Trans. Graphics*, 31(4):30:1–30:11, 2012.

[OLCO13] V. Ozoliņš, R. Lai, R. Caflisch, and S. Osher. Compressed modes for variational problems in mathematics and physics. *PNAS*, 110(46):18368–18373, 2013.

[OMMG10] M. Ovsjanikov, Q. Merigot, F. Memoli, and L. J. Guibas. One point isometric matching with the heat kernel. *Computer Graphics Forum*, 29(5):1555–1564, 2010.

[OMPG13] Maks Ovsjanikov, Quentin Mérigot, Viorica Pătrăucean, and Leonidas Guibas. Shape matching via quotient spaces. In *Computer Graphics Forum*, volume 32, pages 1–11. Wiley Online Library, 2013.

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

[PBB$^+$13] J. Pokrass, A. M. Bronstein, M. M. Bronstein, P. Sprechmann, and G. Sapiro. Sparse modeling of intrinsic correspondences. *Computer Graphics Forum*, 32:459–468, 2013.

[PP93] U. Pinkall and K. Polthier. Computing discrete minimal surfaces and their conjugates. *Experimental Mathematics*, 2(1):15–36, 1993.

[RCB$^+$16] E. Rodolà, L. Cosmo, M. M. Bronstein, A. Torsello, and D. Cremers. Partial functional correspondence. *Computer Graphics Forum*, 2016.

[RMC] E. Rodolà, M. Moeller, and D. Cremers. Regularized point-wise map recovery from functional correspondence. *Computer Graphics Forum*. to appear.

[RMC15] E. Rodolà, M. Moeller, and D. Cremers. Point-wise map recovery and refinement from functional correspondence. In *Proc. VMV*, 2015.

[ROA$^+$13] R. M. Rustamov, M. Ovsjanikov, O. Azencot, M. Ben-Chen, F. Chazal, and L. J. Guibas. Map-based exploration of intrinsic shape differences and variability. *ACM Trans. Graphics*, 32(4):72:1–72:12, July 2013.

[Rus07] R. M. Rustamov. Laplace-Beltrami eigenfunctions for deformation invariant shape representation. In *Proc. SGP*, 2007.

[SOCG10] P. Skraba, M. Ovsjanikov, F. Chazal, and L. J. Guibas. Persistence-based segmentation of deformable shapes. In *Proc. NORDIA*, pages 45–52, June 2010.

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

[SRJ04] N. Srebro, J. Rennie, and T. S. Jaakkola. Maximum-margin matrix factorization. In *Proc. NIPS*, 2004.

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

[VLR$^+$17] M. Vestner, R. Litman, E. Rodolà, A. M. Bronstein, and D. Cremers. Product manifold filter: Non-rigid shape correspondence via kernel density estimation in the product space. In *Proc. CVPR*, 2017.

[WHOG14] F. Wang, Q. Huang, M. Ovsjanikov, and L. J. Guibas. Unsupervised multi-class joint image segmentation. In *Proc. CVPR*, 2014.

[WS13] L. Wang and A. Singer. Exact and stable recovery of rotations for robust synchronization. *Information and Inference*, 2(2):145–193, 2013.

[YL06] M. Yuan and Y. Lin. Model selection and estimation in regression with grouped variables. *J. Royal Statistical Society B*, 68:49–67, 2006.

[ZGLG12] W. Zeng, R. Guo, F. Luo, and X. Gu. Discrete heat kernel determines discrete Riemannian metric. *Graph. Models*, 74(4):121–129, 2012.