HAL
open science

Non-rigid correspondences between surfaces embedded in 3D

Dorian Nogneng

To cite this version:

Dorian Nogneng. Non-rigid correspondences between surfaces embedded in 3D. Image Processing [eess.IV]. Université Paris Saclay (COmUE), 2018. English. NNT : 2018SACLX109 . tel-02074764v2

HAL Id: tel-02074764

https://hal.science/tel-02074764v2

Submitted on 4 Apr 2019

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

université PARIS-SACLAY

NNT : 2018SACLX109

Thèse de doctorat

Non-rigid correspondences between surfaces embedded in 3D

Thèse de doctorat de l'Université Paris-Saclay
préparée à l'Ecole Polytechnique

Ecole doctorale n°580 Sciences et technologies de l'information et de la communication (STIC)
Spécialité de doctorat : mathématiques et informatique

Thèse présentée et soutenue à Palaiseau, le 21/12/2018, par

DORIAN NOGNENG

Composition du Jury :

Giuseppe Patane Professeur, IMATI (CNR)	Rapporteur
Adrien Bousseau Chargé de recherche, Université Nice Sophia-Antipolis (INRIA Sophia-Antipolis)	Rapporteur
Julie Digne Chargée de recherche, Université Claude-Bernard Lyon 1 (LIRIS-Géomod)	Examinateur
Primoz Skraba Chargé de recherche, Josef Stefan Institute	Examinateur
Frédéric Chazal Directeur de recherche, Ecole Polytechnique (INRIA Saclay)	Président
Maks Ovsjanikov Professeur, Ecole Polytechnique (LIX)	Directeur de thèse
Gilles Schaeffer Directeur de recherche, Ecole Polytechnique (LIX)	Co-directeur de thèse

Introduction

Handling and processing the massive amount of 3D data has become a challenge with countless applications, such as computer-aided design, biomedical computing, interactive games, machine perception, robotics, etc. Geometry Processing is an area of research at the interface between algorithmics, applied mathematics and computer science related to the above applications, that exists since approximately 50 years. It is a large topic of research that includes sub-areas. Below we mention a few.

Segmentation

Shape segmentation consists in, given a 3D shape, often represented as a triangle mesh, finding a partition or decomposition of this shape into individual components. We may want, for example, a partition into a small number of connected components that satisfy some geometric constraints such as being flat or having a small diameter. See [84] for a survey on shape segmentation techniques.

Remeshing

Remeshing is a classical problem in Geometry Processing, which essentially involves finding a "better" discrete approximation of the geometry of some given 3D shape. By "better", one may mean that the new approximation should be given by a mesh that should be more regular [93], should preserve sharp edges of the surface, should follow the anisotropic curvature of the surface [6], should use a given type of polygons (such as triangles), or should simply have a user-specified smaller (or larger) density, or connectivity for example [93]. An important example of application is that the mesh may be optimized so that the finite element method becomes more accurate [74].

Geometric optimization

Combinatorial optimization consists in optimizing a function under some constraints. In most practical cases, the constraints satisfy some properties such as symmetries, and the complexity of the problem can be reduced. Geometric optimization is a subset of Combinatorial optimization, where the constraints are derived from a geometric setting. See [3] for a survey.

Surface reconstruction from Point clouds

This problem consists in finding an accurate representation of some 3D shape (typically assumed to be a surface), given a noisy point cloud approximation [37]. For example we can be interested in removing outliers from a noisy point cloud [44, 56], or reconstructing the enclosed volume.

Correspondences

The problem of shape correspondence (also known as "shape matching") consists in, given a pair of shapes, finding a "good" correspondence between them. For example we may want the correspondence to preserve geodesic distances, or local geometric features. This problem has received a growing interest, in part due to its wide applicability, for example in animation, shape morphing or statistical shape modeling. See [100] for a survey on shape correspondence.

The functional map framework, originally introduced in [69] is a recent tool that has shown many useful properties for shape matching. This approach provides a smooth compact representation of correspondences between shapes, and most constraints over functional maps can be expressed as linear constraints, which allows a least squares formulation of the problem.

Organization

In this thesis we focus on the problem of shape correspondence, specifically using functional maps. The overall goal of the thesis is to show how the functional maps pipeline can be improved using functional algebra. The main contribution is to improve both the accuracy of the functional map matrix, computed using the same set of descriptors, and the accuracy of the function transfer, computed using the same functional map matrix. These improvements remain compatible with the classical linear formulation of the functional maps framework.

In Chapter 1 we introduce basic notions and notations that will be used throughout the thesis, related to continuous and discrete surfaces, the Laplace-Beltrami operator, the problem of non-rigid shape matching, and the standard functional map computation pipeline.

In Chapter 2 we notice that functional maps that are induced by point-to-point maps should satisfy point-wise product preservation constraints. We apply this observation to shape descriptors in order to improve the previous classical constraints on functional maps. This leads to an approach that allows to extract more information from existing constraints and results in better correspondences, particularly when the number of independent descriptors is small.

In Chapter 3 we build on the previous remark, but this time in the situation where we already have a functional map that was computed by an existing method. We notice that the point-wise product preservation can also be used to extend the domain over which the given functional map can transfer functions. We show that this allows to improve the accuracy of function transfer.

In Chapter 4 we extend the approach proposed in Chapter 3 by noticing that instead of using point-wise function products, the point-wise composition by any fixed operator should also be preserved. We use a neural network that optimizes the approximation of a given function that we want to transfer, as a point-wise function of some basis functions that we already know how to transfer using a given functional map. We then describe how to apply this trained network to the image of the basis functions to construct the

image of the function that we want to transfer. We show preliminary results that suggest that this method can lead to significant improvement for function transfer.

Main limitations:

- The proposed improvement on the functional map matrix heavily relies on the quality of the input descriptors, and is computationally more expensive than the previous approach.
- The proposed improvement on function transfer heavily relies on the quality of the functional map matrix, and seem to imply natural extensions that are not so easy to perform in practice.

As future work, we plan to use a preservation rule different from the product preservation proposed in Chapter 2 ; extend the ideas proposed in Chapters 3 and 4 using the composition with any smooth function (using neural networks is just one possible option) ; use a composition with a function that does not only apply point-wise as proposed in Chapter 4, but also applies to the neighbors of each vertex for example.

Finally, in Chapter 5 we mention other topics studied during the thesis, that are unrelated to non-rigid shape matching.

Publications This thesis is based on the following publications:

- "Informative descriptor preservation via commutativity for shape matching" [68] (EUROGRAPHICS 2017), for Chapter 2.
- "Improved Functional Mappings via Product Preservation" [67] (EUROGRAPHICS 2018), for Chapter 3.
- "Deep Learning for Non-linear Function Approximation and Mapping" (submitted to EUROGRAPHICS 2019), for Chapter 4.
- [23], [30], [27] for Chapter 5.

Introduction en français

La manipulation et le traitement d'énormes quantités de données en 3D est devenu un défi ayant d'innombrables applications, telles que la conception assistée par ordinateur, le calcul biomédical, les jeux interactifs, la perception des machines, la robotique, etc. Le traitement de données géométrique est un sujet de recherche à l'interface entre l'algorithmique, les mathématiques appliquées et l'informatique en lien avec les applications sus-mentionnées, qui existe depuis une cinquantaine d'années. C'est un domaine de recherche vaste qui inclut des sous-domaines. Ci-dessous nous en mentionnons quelques uns.

Segmentation

La segmentation de maillage consiste à, étant donnée une forme en 3D, trouver une partition ou décomposition de la forme en composantes individuelles. On peut vouloir, par exemple, une partition en un petit nombre de composantes connexes qui satisfont une contrainte géométrique telle qu'être plate ou avoir un petit diamètre. Voir [84] pour un état de l'art des techniques de segmentation de maillage.

Remaillage

Le remaillage est un problème classique en traitement de données géométrique, qui consiste essentiellement à trouver une «meilleure» approximation discrète de la géométrie d'une forme 3D donnée. Par «meilleure», on peut entendre que la nouvelle approximation doit être un maillage plus régulier [93], doit préserver les arêtes saillantes de la surface, doit coller à la courbure anisotrope de la surface [6], doit utiliser un certain type de polygones (par exemple des triangles), ou doit simplement avoir une densité plus faible (ou plus élevée) selon les spécifications de l'utilisateur, ou une connectivité spécifiée par exemple [93]. Un exemple important d'application est que le maillage peut être optimisé pour rendre la méthode des éléments finis plus précise [74].

Optimisation géométrique

L'optimisation combinatoire consiste à optimiser une fonction sous contraintes. Dans la plupart des situations réelles, les contraintes satisfont des propriétés telles que des symétries, et la complexité du problème peut être réduite. L'optimisation géométrique est une branche de l'optimisation combinatoire, qui correspond au cas où les contraintes sont issues d'une situation géométrique. Voir [3] pour un état de l'art.

Reconstruction de surface à partir d'un nuage de points

Ce problème consiste à trouver une représentation exacte d'une forme 3D (typiquement une surface), étant donnée une approximation bruitée par un nuage de points [37]. Par exemple on peut vouloir retirer les intrus d'un nuage de points bruité [44, 56], ou reconstruire le volume englobé.

Correspondances

Le problème de correspondances de forme consiste à, étant donnée une paire de formes, trouver une «bonne» correspondance entre elles. Par exemple on peut vouloir que la correspondance préserve les distances géodésiques, ou des caractéristiques locales. Ce problème a attiré un intérêt croissant, en partie dû à ses nombreuses applications, par exemple en animation, interpolation de formes ou modélisation statistique de formes. Voir [100] pour un état de l'art sur les correspondances de formes.

Le cadre des correspondances fonctionnelles, introduit à l'origine dans [69] est un outil récent qui a dévoilé de nombreuses propriétés utiles pour les correspondances de formes. Cette approche donne une représentation régulière et compacte du problème de correspondances entre formes, et la plupart des contraintes sur les correspondances fonctionnelles peuvent s'exprimer sous forme de contraintes linéaires ce qui permet une formulation du problème par moindres carrés.

Organisation

Dans cette thèse on se concentre sur le problème de correspondance de forme, spécifiquement en utilisant des correspondances fonctionnelles. L'objectif général de la thèse est de montrer comment le processus de calcul des correspondances fonctionnelles peut être amélioré en utilisant l'algèbre de fonctions. La contribution principale est l'amélioration de la précision du calcul de la matrice des correspondances fonctionnelles, calculée en utilisant le même ensemble de descripteurs, et la précision du transfert de fonctions, calculé en utilisant la même matrice de correspondances fonctionnelles. Ces améliorations restent compatibles avec la formulation linéaire classique des correspondances fonctionnelles.

Au Chapitre 1 on introduit les notions et notations de base qui seront utilisées le long de la thèse, liées aux surfaces continues ou discrètes, l'opérateur de Laplace-Beltrami, le problème de correspondance de forme non rigide, et le processus standard du calcul d'une correspondance fonctionnelle.

Au Chapitre 2 on remarque que les correspondances fonctionnelles induites par des correspondances point à point doivent satisfaire des contraintes de préservation de produits point par point. On applique cette observation à des descripteurs de formes pour améliorer la formulation classique des contraintes sur les correspondances fonctionnelles. Cela mène à une approche qui permet d'extraire plus d'information des contraintes existantes et donne de meilleures correspondances, surtout lorsqu'il y a peu de descripteurs indépendants.

Au Chapitre 3 on s'appuie sur la remarque précédente, mais cette fois dans le cas où on a déjà obtenu une correspondance fonctionnelle par une méthode existante. On remarque que la préservation du produit point par point peut aussi être utilisée pour étendre le domaine sur lequel la correspondance fonctionnelle peut transférer des fonctions. On montre que cela permet d'améliorer la précision du transfert de fonction.

Au Chapitre 4 on étend l'approche proposée au Chapitre 3 en remarquant qu'au lieu d'utiliser le produit point par point de fonctions, la composition par n'importe quel opérateur fixé doit aussi être préservée. On utilise un réseau de neurones pour optimiser l'approximation d'une fonction donnée qu'on veut transférer, comme fonction

point par point de fonctions d'une base précalculée, qu'on sait déjà transférer à l'aide de la correspondance fonctionnelle. Puis on décrit comment évaluer ce réseau de neurones entrainé sur l'image des fonctions de la base afin de construire l'image de la fonction que l'on souhaite transférer. On montre des résultats préliminaires qui suggèrent que cette méthode peut apporter des améliorations significatives au transfert de fonctions.

Limites principales :

- L'amélioration proposée sur la matrice des correspondances fonctionnelles dépend beaucoup de la qualité des descripteurs utilisés, et repose sur des calculs plus lourds que l'approche précédente.

- L'amélioration proposée sur le transfert de fonctions dépend beaucoup de la qualité de la matrice de correspondance fonctionnelle, et semble naturellement impliquer des extensions qui ne sont pas si faciles à réaliser en pratique.

Pour poursuivre, noux prévoyons d'utiliser une règle différente de la préservation du produit proposée au Chapitre 2; étendre les idées proposées aux Chapitres 3 et 4 en utilisant la composition avec n'importe quelle fonction régulière (l'utilisation de réseaux de neurones n'est qu'une option possible) ; utiliser la composition avec une fonction qui ne s'applique pas que point par point tel que proposé au Chapitre 4, mais aussi aux voisins de chaque sommet par exemple.

Finalement, au Chapitre 5 on aborde les autres sujets étudiés lors de la thèse, qui n'ont aucun lien avec les correspondances non rigides.

Publications Cette thèse est basée sur les publications suivantes :

- "Informative descriptor preservation via commutativity for shape matching" [68] (EUROGRAPHICS 2017), pour le Chapitre 2.

- "Improved Functional Mappings via Product Preservation" [67] (EUROGRAPHICS 2018), pour le Chapitre 3.

- "Deep Learning for Non-linear Function Approximation and Mapping" (soumis à EUROGRAPHICS 2019), pour le Chapitre 4.

- [23], [30], [27] pour le Chapitre 5.

Acknowledgements

I would like to thank my PhD advisors, and the people with whom I had the opportunity to collaborate. Parts of this work were supported by the chaire Jean Marjoulet from Ecole Polytechnique, FUI project TANDEM 2, a Google Focused Research Award, ERC Starting Grant No. 758800 (EXPROTEA), ERC Consolidator Grant No. 724228 (LEMAN), and Rudolf Diesel fellowship from the TUM Institute for Advanced Study, Marie-Curie CIG-334283, a CNRS chaire d'excellence.

Contents

1 Introduction to non-rigid shape matching **13**
1.1 Surface . . . 13
1.1.1 Continuous setting . . . 13
1.1.2 Discrete setting . . . 13
1.1.3 Discretization of a function and its gradient . . . 14
1.1.4 Area weights . . . 14
1.2 Laplace-Beltrami operator . . . 15
1.2.1 Continuous setting . . . 15
1.2.2 Discrete setting . . . 16
1.3 Goal . . . 18
1.4 Related work . . . 19
1.5 Partial correspondences . . . 20
1.6 Overview of the Functional Maps Framework . . . 20
1.6.1 Setup . . . 20
1.7 Definitions and Notations . . . 23

2 Informative descriptor preservation via commutativity for shape matching **25**
2.1 Introduction . . . 25
2.2 Related work . . . 26
2.3 Overview . . . 28
2.4 Novel Approach for Functional Correspondences . . . 28
2.4.1 Motivation . . . 29
2.4.2 Our constraints . . . 30
2.4.3 Properties . . . 30
2.5 Experiments . . . 32
2.5.1 Using few descriptors . . . 32
2.5.2 Changing the dimension of the reduced space . . . 36
2.6 Conclusion, Limitations & Future work . . . 38

3 Improved functional mappings via product preservation **39**
3.1 Introduction . . . 39
3.2 Related Work . . . 41
3.3 Motivation and Overview . . . 43
3.4 Method Description . . . 44
3.4.1 Function Representation . . . 45
3.4.2 Extended Functional Basis . . . 47
3.4.3 Extended Function Transfer . . . 50
3.4.4 Function Comparison and Pointwise Map Recovery . . . 50
3.5 Results . . . 51

3.5.1 Function approximation and transfer . . . 51
3.5.2 HKS and WKS approximation and transfer . . . 56
3.5.3 Point-to-point map recovery . . . 57
3.5.4 Joint quadrangulation . . . 57
3.6 Conclusion, Limitations and Future Work . . . 58
3.7 Appendix . . . 59
3.7.1 Additional Results . . . 59
3.7.2 Proof of Lemma 2: . . . 60
3.7.3 Proof of Theorem 3.4.1 . . . 62
3.7.4 Future work . . . 68

4 Deep learning for non linear function approximation and mapping 69
4.1 Introduction . . . 69
4.2 Related Work . . . 70
4.3 Background . . . 72
4.4 Main Idea . . . 73
4.5 Description . . . 73
4.6 Parameters . . . 74
4.7 Results . . . 75
4.8 Conclusion . . . 76
4.9 Future work . . . 77

5 Miscellaneous 79
5.1 A new analysis method for evolutionary optimization of dynamic and noisy objective functions . . . 79
5.2 On semiring complexity of Schur polynomials . . . 79
5.2.1 Schur polynomial . . . 80
5.2.2 Main result . . . 80
5.3 Scheduling with gaps: New models and algorithms . . . 80

6 General Conclusion 83
6.1 Summary . . . 83
6.2 Future work . . . 83
6.3 Position of the work in the community . . . 84

Bibliography 85

Chapter 1

Introduction to non-rigid shape matching

Non-rigid shape matching is an important task in Geometry Processing, with a range of applications that includes deformation transfer [91], statistical shape modelling [35] and segmentation transfer among others. It consists in, given two shapes $\mathcal{M}$ and $\mathcal{N}$, finding a good correspondence between them. We assume that $\mathcal{M}$ and $\mathcal{N}$ are represented as surfaces in $\mathbb{R}^3$. Each shape is discretized using a triangle mesh. The shapes may represent the same animal / human / item in a different position, or they may represent two different objects that share some similar underlying structure. In any case, there should be a natural correspondence between the two surfaces that should preserve some shared structure. In the case of the same human in different positions, the "natural" correspondence would match the same parts of the body together, i.e. left hand to left hand, etc. Such a correspondence can be naturally discretized as a function defined from the finite set of vertices of $\mathcal{N}$, $V_{\mathcal{N}}$ to the finite set of vertices of $\mathcal{M}$, $V_{\mathcal{M}}$.

1.1 Surface

1.1.1 Continuous setting

In the continuous setting, a surface $\mathcal{M}$ will be represented as a 2D manifold embedded in 3D. This means that $\mathcal{M}$ is a subset of $\mathbb{R}^3$ such that for each $x \in \mathcal{M}$ there exists an open subset of $\mathcal{M}$, U_x, and a smooth map $m_x : U_x \to \mathbb{R}^2$ such that m_x is a diffeomorphism between U_x and an open set of $\mathbb{R}^2$ [34]. The degree of smoothness that applies to all the maps m_x and m_x^{-1} defines the degree of smoothness of the manifold $\mathcal{M}$. For simplicity, we typically assume *smooth* manifolds, for which all diffeomorphisms involved are infinitely differentiable.

1.1.2 Discrete setting

In the discrete setting a continuous surface may be approximated using a point cloud, where each point is represented using three floating point coordinates in practice. In addition the point cloud is often converted to a triangular mesh, see for example [79] for a survey on algorithms that perform this conversion. In order to obtain a triangular mesh, the point cloud should be equipped with a set of triangles that covers the point cloud. Each endpoint of a triangle has to be a point of the point cloud, and a good triangulation should be such that the triangles follow the original surface. For instance, the normal of these triangles should closely match the normals of the represented surface at points that

are near the triangle. A triangulation will be stored as a finite set of triplets of integers (i, j, k), where i, j and k are the respective indices of the points in the point cloud that correspond to the endpoints of the triangle.

In addition we will often assume that our triangle meshes should be connected, manifold and possibly without boundary. For the triangle mesh to be manifold, all edges should be adjacent to at most two triangles (exactly two if we do not allow boundaries). For each vertex we should be able to order its adjacent triangles in a cycle of triangles that share exactly one edge, and all triangles adjacent to a given vertex should not intersect anywhere else than at the shared edge, as discussed in [16]. In my thesis each surface will be represented as a triangle mesh, I will not work directly on point clouds.

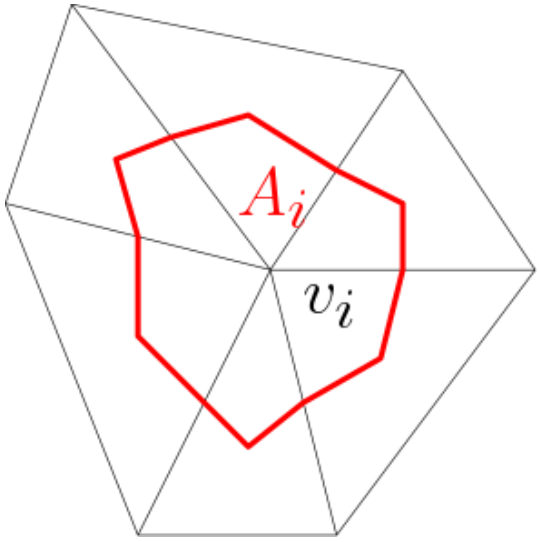


Figure 1.1 – Area associated to the vertex v_i shown in red

1.1.3 Discretization of a function and its gradient

A real-valued function f defined on a surface $\mathcal{M}$ will be discretized by the vector of its values on the vertices of the triangle mesh. For example if $f : \mathcal{M} \to \mathbb{R}$ is a real-valued function, it will be discretized as a vector $f = (f_1, \ldots, f_{n_\mathcal{M}}) \in \mathbb{R}^{n_\mathcal{M}}$, where $n_\mathcal{M}$ is the number of vertices of $\mathcal{M}$.

In order to discretize the gradient ∇f of a real-valued function $f : \mathcal{M} \to \mathbb{R}$, we assume that the function is piece-wise linear on each triangle. This means that on a vertex v situated in the triangle (v_i, v_j, v_k), f takes the value $f(v) = \alpha_i f_i + \alpha_j f_j + \alpha_k f_k$ where $\alpha_i, \alpha_j, \alpha_k \in [0, 1]$ are the barycentric coordinates of v, such that $\alpha_i + \alpha_j + \alpha_k = 1$ and $v = \alpha_i v_i + \alpha_j v_j + \alpha_k v_k$. In the triangle (v_i, v_j, v_k), the gradient of f will take the constant value

$$\nabla f = \frac{1}{2A(v_i, v_j, v_k)} \left(f_i (v_k - v_j)^\perp + f_j (v_i - v_k)^\perp + f_k (v_j - v_i)^\perp \right)$$

where $\perp$ means that we consider the vector rotated by $\frac{\pi}{2}$ in the plane of the triangle (v_i, v_j, v_k), and $A(v_i, v_j, v_k)$ is the area of the triangle (v_i, v_j, v_k). We can then obtain the formula of ∇f by linearity, combining linear functions that take the value 1 on only one vertex among v_i, v_j, v_k, value 0 on the two other vertices. See a complete proof in [16].

1.1.4 Area weights

In the discrete model, we need to associate area weights to each unit, in our case to each vertex. We will denote the area weights associated to vertex v_i by A_{v_i} or simply A_i. To

all the vertices that are within the area range of each point, we will associate the behavior of the point. For example we could discretize the integral of a function by the sum of the discretized function weighted by the area weights: $\int_{x\in\mathcal{M}} f(x)dx \approx \sum_{i=1}^{n_{\mathcal{M}}} f_i A_i$.

For each vertex v_i, this area weight is defined as the area enclosed by a polygon delimited by lines that join each median of edges adjacent to v_i to each barycenter of an adjacent triangle as shown on Figure 1.1. Its value is $\frac{1}{3}$ of the sum of the areas of neighboring triangles: the area of each triangle is equally split between its adjacent vertices.

1.2 Laplace-Beltrami operator

1.2.1 Continuous setting

In the continuous setting, the Laplace-Beltrami operator is a linear operator that takes as input a smooth function f on a surface $\mathcal{M}$, $f : \mathcal{M} \to \mathbb{R}$, and produces as output another function $\Delta f : \mathcal{M} \to \mathbb{R}$ such that $\Delta f = -\mathrm{div}\,(\nabla f)$. We remind that these operators are defined on $\mathbb{R}^2$ by:

- $\nabla : \left(\mathbb{R}^2 \to \mathbb{R}\right) \to \left(\mathbb{R}^2 \to \mathbb{R}^2\right)$: $\forall f : \mathbb{R}^2 \to \mathbb{R}, \nabla f(x,y) = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right)$
- $\mathrm{div} : \left(\mathbb{R}^2 \to \mathbb{R}^2\right) \to \left(\mathbb{R}^2 \to \mathbb{R}\right)$: $\forall \mathbf{F} = (\mathbf{F}_1, \mathbf{F}_2) : \mathbb{R}^2 \to \mathbb{R}^2, \mathrm{div}\mathbf{F}(x,y) = \frac{\partial \mathbf{F}_1}{\partial x} + \frac{\partial \mathbf{F}_2}{\partial y}$.
- $\Delta : \left(\mathbb{R}^2 \to \mathbb{R}\right) \to \left(\mathbb{R}^2 \to \mathbb{R}\right)$: $\forall f : \mathbb{R}^2 \to \mathbb{R}, \Delta f(x,y) = -\frac{\partial^2 f}{\partial x^2} - \frac{\partial^2 f}{\partial y^2}$

We can see that $\Delta = -\mathrm{div} \circ \nabla$, and by computing ∇, div, and Δ in a rotated basis we can show that they do not depend on the choice of the orthonormal basis. This is still true on a general surface $\mathcal{M}$. We refer the reader to [64] for the definition of these operators on a general surface.

The Laplace-Beltrami operator of a constant function is 0 If $f : \mathcal{M} \to \mathbb{R}$ is constant, then $\Delta f = 0$.

The Laplace-Beltrami operator is symmetric If $f^{(1)}, f^{(2)} : \mathcal{M} \to \mathbb{R}$ are smooth functions that vanish on the boundary of $\mathcal{M}$, then

$$\int_{x\in\mathcal{M}} f^{(1)}(x)\Delta f^{(2)}(x)dx = \int_{x\in\mathcal{M}} \Delta f^{(1)}(x) f^{(2)}(x)dx$$

The Laplace-Beltrami operator is locally supported If $x \in \mathcal{M}$, $f : \mathcal{M} \to \mathbb{R}$ is a smooth function, then for any $x' \neq x$, there exists a neighborhood $N(x)$ of x such that $x' \notin N(x)$ and $\Delta f(x)$ is independent from the values of f outside of $N(x)$.

The Laplace-Beltrami operator has a linear precision If $\mathcal{M}$ is included in a Euclidean plane and $f : (x,y,z) \to f_0 + f_1 x + f_2 y + f_3 z$ for some $f_0, f_1, f_2, f_3 \in \mathbb{R}$ is a linear function, then $\Delta f = 0$.

The Laplace-Beltrami operator satisfies the maximum principle If $\Delta f = 0$ in the interior of $\mathcal{M}$, then f does not have any maximum in the interior of $\mathcal{M}$.

The Laplace-Beltrami operator is positive semidefinite For any f, we have $\int_{x\in\mathcal{M}} f(x)\Delta f(x)dx \geq 0$. More precisely:

$$\int_{x\in\mathcal{M}} f(x)\Delta f(x)dx = \int_{x\in\mathcal{M}} \|\nabla f(x)\|^2 dx.$$

It has been shown that all the discrete equivalent of the aforementioned properties cannot be satisfied at once by any discretization [106].

The Sturm-Liouville's decomposition In the continuous setting, the Sturm-Liouville's decomposition states that there is an orthonormal basis $\Phi_1, \Phi_2, \Phi_3, \ldots$ associated to eigenvalues $0 = \lambda_1 \leq \lambda_2 \leq \lambda_3 \leq \ldots \to \infty$ such that any function $f \in L^2(\mathcal{M})$ can be written as a convergent series in $L^2(\mathcal{M})$

$$f = \sum_{i=1}^{\infty} f_i \Phi_i$$

for some coefficients f_i.

The divergence theorem The divergence operator satisfies a property called the divergence theorem (proof in [64]). For any vector valued function $\mathbf{F}$ and a delimited area A included in $\mathcal{M}$, this theorem states that:

$$\int_{u\in A} \mathrm{div}\mathbf{F}(u)du = \int_{\partial A} \mathbf{F}(u)\cdot\mathbf{n}(u)du$$

where $\mathbf{n}(u)$ is the outer normal, and ∂A is the boundary of the enclosed surface A.

For the discrete case we will use this divergence theorem to define our discrete Laplace-Beltrami operator.

1.2.2 Discrete setting

We discretize the Laplace-Beltrami operator using the classical cotangent-weight scheme [63, 76]:

$$\Delta f(v_i) = \frac{1}{2A_i} \sum_{v_j\in N_1(v_i)} (\cot\alpha_{i,j} + \cot\beta_{i,j})(f_i - f_j)$$

where A_i is the area associated to the vertex v_i, $\beta_{i,j}$ is the inner angle of the vertex v_k such that v_i, v_j, v_k is a direct triangle and $\alpha_{i,j}$ is the inner angle of the vertex $v_{k'}$ such that $v_i, v_j, v_{k'}$ is an indirect triangle. $\mathcal{N}_1(v_i)$ is the 1 ring neighborhood around v_i. See notations on Figure 1.2.

First note that this formula defines a discrete Laplacian Δf from a discrete real valued function f. This formula is linear in f and allows us to define a sparse matrix that will operate on vectors f. For completeness of the discussion, we prove the formula below.

For the proof, we use the divergence theorem for a vector valued function $\mathbf{F}$ in the continuous setting, which states that:

$$\int_{A_i} \mathrm{div}\mathbf{F}(u)du = \int_{\partial A_i} \mathbf{F}(u) \cdot \mathbf{n}(u)du$$

where $\mathbf{n}(u)$ is the outer normal, u represents a point on the surface A_i and ∂A_i is the boundary of the enclosed surface A_i.

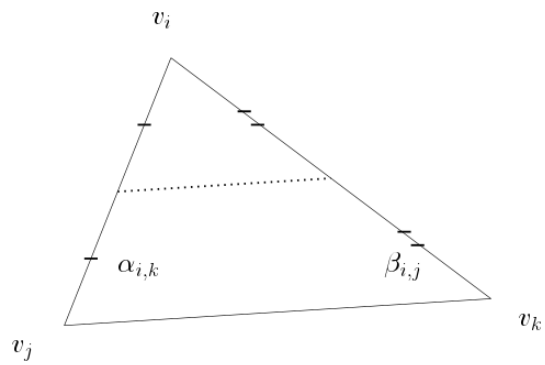


Figure 1.2 – notations for a triangle v_i, v_j, v_k

We apply it to $-\nabla f$, which is discretized as a piecewise constant vector on triangles:

$$A_i \Delta f(v_i) = -\int_{\partial A_i} \nabla f(u) \cdot \mathbf{n}(u)dl$$

Only triangles that are adjacent to v_i have a non zero contribution, and using Figure 1.2, we can see that the contribution of a direct triangle (v_i, v_j, v_k) adjacent to v_i is given by

$$\begin{aligned}
-\frac{1}{2}\left\langle (v_j - v_k)^\perp | \nabla f \right\rangle &= -\frac{1}{2A(v_i, v_j, v_k)} \Big(f_i \left\langle (v_j - v_k)^\perp | (v_k - v_j)^\perp \right\rangle \\
&\quad + f_j \left\langle (v_j - v_k)^\perp | (v_i - v_k)^\perp \right\rangle + f_k \left\langle (v_j - v_k)^\perp | (v_j - v_i)^\perp \right\rangle \Big) \\
&= \frac{1}{2} \left((f_i - f_j)\cot(\beta_{i,j}) + (f_i - f_k)\cot(\alpha_{i,k}) \right)
\end{aligned}$$

where $A(v_i, v_j, v_k)$ denotes the area of the triangle (v_i, v_j, v_k).

Summing over all triangles gives the cotangent formula.

The Laplace-Beltrami operator is symmetric Similarly to the continuous counterpart, the discrete Laplace-Beltrami operator is symmetric. This is a consequence of the symmetry in the coefficients: $\cot \alpha_{i,j} + \cot \beta_{i,j} = \cot \beta_{j,i} + \cot \alpha_{j,i}$

The Laplace-Beltrami operator and positive semidefiniteness By analogy to the continuous case, we can show that for any function $f : V_{\mathcal{M}} \to \mathbb{R}$, $\sum_{x \in V_{\mathcal{M}}} A_x f(x) \Delta f(x) = \sum_{(x,x') \text{ edge of } \mathcal{M}} \left(\cot \alpha_{x,x'} + \cot \beta_{x,x'} \right) (f(x) - f(x'))^2$.

While in most practical cases $\left(\cot \alpha_{x,x'} + \cot \beta_{x,x'} \right) > 0$ and the discrete Laplace-Beltrami operator is semidefinite positive, i.e., $\sum_{x \in V_{\mathcal{M}}} A_x f(x) \Delta f(x) \geq 0$ for all f, this may not always be true and there are counter-examples of specific triangle mesh configurations that lead to a failure of this property. Indeed, as mentioned above, a

seminal result in Geometry Processing is that no discretization of the Laplace-Beltrami operator exists that would always satisfy several basic properties of the continuous operator in all cases [106].

In addition we can see in the equation above that in the situation where the coefficients $\left(\cot \alpha_{x,x'} + \cot \beta_{x,x'}\right) > 0$ are positive, $\sum_{x \in V_{\mathcal{M}}} A_x f(x) \Delta f(x) = 0$ if and only if f takes constant values over all edges, i.e. if f takes constant values on all connected components. Since we only work with connected manifolds, this means for us that the Laplace-Beltrami operator usually has the eigenvalue 0 with multiplicity 1, associated to a constant eigenfunction.

Whenever the discrete Laplace-Beltrami operator is positive semi-definite, and the triangle mesh consists of a single connected component, as is most often the case in practice, this will lead to exactly one eigenvalue equal to 0 and the associated to the constant eigenfunction, and all other eigenvalues being positive.

Discrete basis approximation We use the analogy between the Sturm-Liouville's decomposition of the continuous setting and our discretization to justify that we will compute the first few eigenfunctions of our discretized Laplacian and stack them into a matrix $\mathbf{\Phi}_{\mathcal{M}}$. Then, for a given function $f : V_{\mathcal{M}} \to \mathbb{R}$ we will project it on $\mathbf{\Phi}_{\mathcal{M}}$.

1.3 Goal

As mentionned in the previous chapter, our goal is to study problems related to finding correspondences and relations between items in 3D. An item in 3D is represented by its surface, that may have been scanned using recent technological tools such as Microsoft Kinect or a CT scan. After scanning the surface we get a triangle mesh, for example using surface reconstruction techniques mentionned above. We consider two items that can naturally be mapped to each other. This means that each part of one item has a meaningfully corresponding part on the other. Our main problem consists in finding, for each vertex of one item, the best corresponding vertex on the other. For example if one of the surface represents a sitting cat and another represents a standing cat, then a good solution should be a correspondence that maps the tip of the left paw of the sitting cat to the tip of the left paw of the standing cat, the nose of the sitting cat to the nose of the standing cat, etc. We measure our error using geodesic distances: a good correspondence will preserve geodesic distances as well as possible. Theoretically it is possible to exactly preserve geodesic distances if and only if the shapes are isometric (we mean non-rigid isometry). Ideally our method should be as general as possible. For example we should also be able to find a correspondence between a cat and a dog, since both animals have similar body parts: four legs, a head, a tail, etc. Pushing this idea further, we may want to find (at least partial) correspondences between a cat and a human for instance.

There are multiple reasons why we may be interested in finding such a correspondence. Shape matching has a wide range of applications ranging from shape morphing [42], statistical shape modeling [13, 35], deformation transfer [91].

1.4 Related work

The problem of rigid shape registration consists in aligning a 3D model to another, using a rigid motion defined by a rotation and a translation. It has been well studied, and efficient solutions already exist. Rigid alignment is typically solved using methods like ICP or sampling the shapes and checking the consistency of potential transforms. See [94] for a survey on rigid shape matching.

The problem of non-rigid shape matching is harder to formulate and harder to evaluate, one of the reasons being that it cannot be parameterized by a few parameters (a rotation and a translation).

We look for a correspondence T between a surface $\mathcal{N}$ and a surface $\mathcal{M}$. In the discussion below we will interchangeably refer to the discrete or the continuous versions: $\mathcal{N}$ and $\mathcal{M}$ represent both surfaces and triangulations. $T : \mathcal{N} \to \mathcal{M}$ represents both a map between surfaces and a map $V_{\mathcal{N}} \to V_{\mathcal{M}}$ between the $n_{\mathcal{N}}$ vertices of the discretized $\mathcal{N}$ and the $n_{\mathcal{M}}$ vertices of the discretized $\mathcal{M}$. This is because we want to develop tools that have a well founded equivalence in the continuous setting. In order to represent the correspondence $T : V_{\mathcal{N}} \to V_{\mathcal{M}}$, the most natural way is to use a list of integers. The value at position i, $T(i)$, will be the index of the vertex of $\mathcal{M}$ that best corresponds to the ith vertex of $\mathcal{N}$. Most existing methods formulate the problem using this combinatorial representation of point-to-point correspondences [20, 103]. Using this list T as variable, we often get a huge search space that leads to non-convex problems that are hard to solve. Existing methods use hierarchical matching, probabilistic matching, or a relaxation [18].

Below we discuss the works that are most closely related to ours, especially those based on the functional map framework.

Most early methods designed to find correspondences between shapes undergoing non-rigid transformation have concentrated on establishing mappings that minimize some distortion energy, such as conformality (locally angle preservation) [5, 43, 50], or approximate intrinsic isometries (preserving geodesic distances) (e.g., [18, 71, 96] among many others). Both the theoretical formalism and the computational methods associated with these approaches are mature and can often result in high-quality mappings whenever the deformations follow the prescribed models. However, such methods often lack flexibility, making it hard to introduce additional information, in the form of expected geometric or appearance properties (descriptors) that should be preserved by the map, and are badly-suited in the presence of more general non-rigid deformations. Another, more recent, set of techniques has been proposed to obtain soft, or approximate correspondences rather than point-to-point maps [69, 89]. This includes both maps between probability densities [62, 89, 90] and region-level maps [21, 31], which can be used in a multi-scale way to obtain accurate (sometimes even pointwise) correspondences. These techniques are often more robust in the presence of geometric and structural variability, and in many cases allow to inject domain-specific knowledge, such as expected descriptor preservation into the computational pipeline.

Although in recent years, there have been several approaches proposed to address the point-wise correspondence problem, especially when formulated as a special case of more general quadratic assignment [29, 41, 90, 102], these methods are still typically

computationally very expensive and do not scale well beyond a relatively small number of points on the shapes. As a result, several alternatives have been proposed, including the functional maps framework that we concentrate on throughout our work.

1.5 Partial correspondences

Some works deal with partial correspondences, which leads to additional complexity in the formulation of the objective since the parts that are to be found in correspondence are not specified in advance. We are interested in shape matching with potentially large variations. Previous methods include surface deformation [39,107], matching approximately isometric shapes [17,50], or including high level geometric informations such as labeled segments [99]. We refer the interested reader to the surveys on shape matching for a more in-depth overview of the field [100].

1.6 Overview of the Functional Maps Framework

In this section we describe the general setting of non-rigid shape matching, introduce the main notations that will be used in the rest of the thesis, and give an overview of the functional map framework introduced in [69], including the main computational steps required for estimating functional maps in practice.

1.6.1 Setup

The main goal in the problem of shape matching is to try to find a correspondence or a mapping between a pair of shapes $\mathcal{M}$ and $\mathcal{N}$ that represent similar physical objects, for which one would expect a natural correspondence to exist.

The simplest and most common approach is to represent a solution to the shape matching problem as a correspondence $T : V_{\mathcal{N}} \to V_{\mathcal{M}}$ that maps each vertex in $\mathcal{N}$ to a vertex in $\mathcal{M}$ according to some quality criteria. Such a correspondence can also be written as a matrix Π of size $n_{\mathcal{N}} \times n_{\mathcal{M}}$, that has exactly one 1 on each line, and zeros everywhere else. When the number of points is the same, $n_{\mathcal{N}} = n_{\mathcal{M}}$, and the map T is a bijection then Π is a standard permutation matrix. If we allow convex combinations of vertices (or equivalently probability distributions) as solutions, as in [90], then we can relax the binary $0, 1$ constraint to allow the entries of Π to lie in the interval $[0, 1]$ with the additional constraint that all lines of Π should sum to 1.

Another, more general relaxation that was considered in [69] is via linear mappings between real-valued functions defined on the shapes. It is a dual point of view: instead of using point-to-point correspondences we consider function to function correspondences that allow to distinctly represent each point-to-point correspondence via composition. A map $T : V_{\mathcal{N}} \to V_{\mathcal{M}}$ is represented via the equivalent map $c : (V_{\mathcal{M}} \to \mathbb{R}) \to (V_{\mathcal{N}} \to \mathbb{R})$ defined by composition: $c(f) = f \circ T$. Thus, given a function $f : \mathcal{M} \to \mathbb{R}$ defined on shape $\mathcal{M}$, we can use T to transfer f onto $\mathcal{N}$ through composition to define $g = f \circ T$. Here, $g : \mathcal{N} \to \mathbb{R}$ and $g(y) = f(T(y))$, for any point y on $\mathcal{N}$. For any fixed T, the mapping c between functions $f \mapsto g$ is linear, and thus can be represented as a matrix in the discrete

setting. If functions are represented as discrete vectors, then using the notation above, we can simply write: $g = \Pi f$, if the functions are expressed as vectors with respect to the standard basis. An advantage of this representation is that c is linear. It maps a space of dimension $n_{\mathcal{M}}$ (number of vertices on shape $\mathcal{M}$) to a space of dimension $n_{\mathcal{N}}$ (number of vertices on shape $\mathcal{N}$), therefore it can be represented using a matrix Π of size $n_{\mathcal{N}} \times n_{\mathcal{M}}$. Note that *any* real-valued matrix Π corresponds to a valid linear functional map, even if it does not represent a correspondence between points or probability distributions. A given linear map $c : (V_{\mathcal{M}} \to \mathbb{R}) \to (V_{\mathcal{N}} \to \mathbb{R})$ may not come from any point-to-point map T. This representation is injective: each correspondence T is associated to a unique c, but the space $(V_{\mathcal{M}} \to \mathbb{R}) \to (V_{\mathcal{N}} \to \mathbb{R})$ in which c is defined is larger, therefore the opposite is not true.

By looking at the image of the indicator function of a vertex, one can show that it is possible to recover T from c. Indeed, if e_p is the indicator function at a vertex $p \in \mathcal{M}$, then $c(e_p)$ is the indicator function of $T^{-1}(p) \subset \mathcal{N}$. This allows to infer $T^{-1}(p)$ for each $p \in \mathcal{M}$, therefore it allows to infer T. Note that we may not assume T to be invertible because $n_{\mathcal{M}}$ may be different from $n_{\mathcal{M}}$, and because even if $n_{\mathcal{M}} = n_{\mathcal{N}}$ the topology of the triangle meshes of $\mathcal{M}$ and $\mathcal{N}$ may not be the same.

The key aspect of the functional map representation proposed in [69] is to use a "reduced basis" to encode the functional map, instead of working in the full spaces $\mathbb{R}^{n_{\mathcal{M}}}$ and $\mathbb{R}^{n_{\mathcal{N}}}$. Thus, suppose we are given some set of basis functions on shapes $\mathcal{M}$ and $\mathcal{N}$, encoded as matrices $\Phi_{\mathcal{M}}, \Phi_{\mathcal{N}}$ respectively, having sizes $n_{\mathcal{M}} \times k_{\mathcal{M}}$ and $n_{\mathcal{N}} \times k_{\mathcal{N}}$ for some $k_{\mathcal{M}} \ll n_{\mathcal{M}}$ and $k_{\mathcal{N}} \ll n_{\mathcal{N}}$, where each column corresponds to a basis function on the corresponding shape. Typically $1000 \leq n_{\mathcal{M}}, n_{\mathcal{N}} \leq 100000$, and we choose $10 \leq k_{\mathcal{M}}, k_{\mathcal{N}} \leq 200$. Then, the *functional map* matrix can be written as $\boldsymbol{C} = {\Phi_{\mathcal{N}}}^{\dagger} \Pi \Phi_{\mathcal{M}}$, where $\dagger$ denotes the Moore-Penrose pseudoinverse. For example, if the basis functions are orthonormal with respect to the standard inner product then $\boldsymbol{C} = \Phi_{\mathcal{N}}^T \Pi \Phi_{\mathcal{M}}$, whereas if the basis functions are orthonormal with respect to a weighted inner product, so that $\Phi_{\mathcal{N}}^T \mathbf{A}^{\mathcal{N}} \Phi_{\mathcal{N}} = I_{n_{\mathcal{N}}}$ where $\mathbf{A}^{\mathcal{N}}$ is a matrix of weights, then $\boldsymbol{C} = {\Phi_{\mathcal{N}}}^T \mathbf{A}^{\mathcal{N}} \Pi \Phi_{\mathcal{M}}$.

We may prefer using smooth elements of our vector space in our reduced basis. Intuitively, finding correspondences in the vector spaces spanned by smooth functions would then be closer to finding correspondences between regions of $\mathcal{M}$ and regions of $\mathcal{N}$ than finding correspondences between vertices of $\mathcal{N}$ and vertices of $\mathcal{M}$. Indeed, transferring a function corresponds to transferring a weighted region, where a value 0 means that the point is out of the region that we want to transfer and a high value means that the point is in the region. This point of view is indeed closer to what one should expect from a correspondence.

The expression above allows to compute the matrix $\boldsymbol{C}$ if the initial correspondence matrix Π is known. In practice, however, the shape matching problem consists precisely in trying to recover this correspondence for a given pair of shapes. This means that the matrix $\boldsymbol{C}$ will be an unknown.

In practice, the most commonly-used basis in functional map computations is given by the eigenfunctions, corresponding to the smallest eigenvalues of the Laplace-Beltrami operator, although the ideas presented in the thesis are not tied to this choice. These eigenfunctions have a multi-scale effect, since the eigenfunctions are ordered from low-frequency

(smoothest) to high-frequency according to the eigenvalues. These eigenfunctions are a generalization of Fourier functions to surfaces, and provide a way to approximate well any smooth function using relatively few functions [1]. By far the most common discretization of this Laplace-Beltrami operator is the classical cotangent-weight scheme introduced in section 1.2.2, which allows to represent it as a matrix $L = A^{-1}W$, where A is a diagonal matrix of area weights and W is a sparse matrix of cotangent weights. In this case the eigenfunctions can be found by solving the generalized eigenvalue problem $W\phi = \lambda A\phi$, and the matrix of eigenfunctions Φ satisfies the relation $\Phi^T A\Phi = Id$, and $\Phi^+ = \Phi^T A$.

Functional Map Estimation Many constraints that should be satisfied by the map T can be formulated as linear constraints over the functional map matrix $\boldsymbol{C}$, which can then be optimized using least squares. An example of such a constraint would be that a function $f^{(1)} : V_{\mathcal{M}} \to \mathbb{R}$ defined using local geometry of the shape $\mathcal{M}$ (we call such a function a **descriptor**) should be mapped to its counterpart on $\mathcal{N}$, $g^{(1)} : V_{\mathcal{N}} \to \mathbb{R}$. Examples of descriptors are the Gaussian curvature, the mean curvature, or multi-scale descriptors such as the Heat Kernel signature or the Wave Kernel signature [8, 73, 92] for some range of parameter choices (i.e., each $f^{(p)}, g^{(p)}$ corresponds to a parameter, such as time in the HKS). The key aspects in estimating functional maps therefore consists in formulating pairs of function preservation constraints $f^{(p)}, g^{(p)}$ and solving the linear system of equations to recover the unknown matrix $\boldsymbol{C}$. Alternatively function preservation constraints can also represent knowledge of parts or feature points that are known to match, in which case the functions can be either indicators of given parts, or derived quantities, such as distance function to a feature. Once all of the function preservation constraints are computed, they can be stacked into matrices f and g whose columns are $f^{(1)}, f^{(2)}, \ldots, g^{(1)}, g^{(2)}, \ldots$). We express these matrices in the given (Laplace-Beltrami) basis: $F = \boldsymbol{\Phi}^{\dagger}_{\mathcal{M}} f, G = \boldsymbol{\Phi}^{\dagger}_{\mathcal{N}} g$, where $\dagger$ denotes the Moore-Penrose pseudoinverse. Then, the optimal functional map is found by solving the following system in the least squares sense:

$$\boldsymbol{C}_{\text{opt}} = \arg\min_{\boldsymbol{C}} \|\boldsymbol{C}F - G\|^2 + \alpha\|\Delta_{\mathcal{N}}\boldsymbol{C} - \boldsymbol{C}\Delta_{\mathcal{M}}\|^2. \tag{1.1}$$

Here, $\Delta_{\mathcal{N}}, \Delta_{\mathcal{M}}$ are diagonal matrices of eigenvalues of the Laplace-Beltrami operator and α is a small scalar weight. In other words, the optimal functional map $\boldsymbol{C}$ can be computed so that it preserves the given functions and commutes with the Laplace-Beltrami operator itself. This latter constraint is associated with the standard assumption that the sought map should be approximately intrinsically isometric (see Apendix of [69]). The use of the small weight α makes it a regularization term. This means that the second term is a corrective term over the main optimization.

One of the main advantages of the functional map representation is that the optimization step above for computing $\boldsymbol{C}_{opt}$ in Eq. 1.1 can be solved efficiently using standard numerical linear algebra tools, and in the most basic case reduces to solving a simple linear system of equations. This step has been extended, by both employing manifold constraints, robust optimization methods [47], and by formulating better descriptor preservation, and regularization terms [53]. On the other hand, as has been observed in

several follow-up works (see, e.g. [81]), converting a functional map to a point-to-point map in step 4. of the pipeline can be a challenging and error-prone step in itself. At the same time, as argued in the original article [69] (Section 8.3), the knowledge of a point-to-point map might not be required in certain applications. Indeed, the functional map matrix $\boldsymbol{C}$ can be used, e.g. to transfer real-valued functions across shapes, which can be directly used, e.g. to transport segmentations across shapes [69] or images [104] as well as other information such as tangent vector fields [9] or cross-fields [10]. Thus, given a real-valued function $\mathbf{f}$, its image under the functional map $\boldsymbol{C}$ can be computed as:

$$\mathbf{g} = \Pi\mathbf{f} = \boldsymbol{\Phi}^{\mathcal{N}}\boldsymbol{C}(\boldsymbol{\Phi}^{\mathcal{M}})^{\dagger}\mathbf{f},$$

where † denotes the Moore-Penrose pseudo-inverse. Note that the large $n_{\mathcal{N}} \times n_{\mathcal{M}}$ matrix Π representing the functional map in the 'spatial' domain is never actually constructed explicitly. Note however that this transfer is only limited to the subspace of functions spanned by $\boldsymbol{\Phi}_{\mathcal{M}}$, which can be restrictive. In some cases we may want to expand the space over which we can accurately transfer functions.

This pipeline was further extended in several follow-up works [24, 77, 80].

In this thesis we show different methods for expanding this transfer (Chapters 3 & 4), and a way of defining a better functional map based on a larger space than standard approaches (Chapter 2).

1.7 Definitions and Notations

Along this thesis, we will keep the notations $\mathcal{M}$, $\mathcal{N}$ respectively for the first shape and the second shape, both continuous and discretized versions. We will use $V_{\mathcal{M}}$, $V_{\mathcal{N}}$ for the set of vertices of $\mathcal{M}$ and $\mathcal{N}$, $n_{\mathcal{M}}$ and $n_{\mathcal{N}}$ will denote their number of vertices. $\Phi^{\mathcal{M}}$ and $\Phi^{\mathcal{N}}$ are matrices of size $n_{\mathcal{M}} \times k_{\mathcal{M}}$ and $n_{\mathcal{N}} \times k_{\mathcal{N}}$ that contain the first k eigenfunctions of the Laplace-Beltrami operator, unless specified otherwise.

Chapter 2

Informative descriptor preservation via commutativity for shape matching

We consider the problem of non-rigid shape matching, and specifically the functional maps framework that was recently proposed to find correspondences between shapes. A key step in this framework is to formulate descriptor preservation constraints that help to encode the information (e.g., geometric or appearance) that must be preserved by the unknown map. In this chapter, we show that considering descriptors as *linear operators* acting on functions through multiplication, rather than as simple scalar-valued signals, allows to extract significantly more information from a given descriptor and ultimately results in a more accurate functional map estimation. Namely, we show that descriptor preservation constraints can be formulated via *commutativity* with respect to the unknown map, which can be conveniently encoded by considering relations between matrices in the discrete setting. As a result, when the vector space spanned by the descriptors has a dimension smaller than that of the reduced basis, our optimization may still provide a fully-constrained system leading to accurate point-to-point correspondences, while previous methods might not. We demonstrate on a wide variety of experiments that our approach leads to significant improvement for functional map estimation by helping to reduce the number of necessary descriptor constraints by an order of magnitude, even given an increase in the size of the reduced basis.

2.1 Introduction

In this chapter we study the problem of non-rigid shape matching, which consists in trying to find a good correspondence between two shapes that might undergo a non-rigid transformation, such as articulated motion of humans. This problem has many applications such as deformation transfer [91], shape interpolation [42] and even statistical shape modeling [35] among myriad others. A wide variety of methods has been used to tackle this problem over the years [100], primarily by restricting the search space either using feature-point correspondences [18], or using a reduced model, such as conformal or isometric shape deformations.

In this chapter, we concentrate on the functional map framework introduced in [69], which has been widely adopted since its introduction due to its efficiency for representing and computing correspondences, which in the most basic case reduces to solving a linear system of equations. A key step in this framework, first introduced in the original

article [69] and then used in most follow-up works, including [24, 53, 77] among others, is to formulate function preservation constraints, which typically encode information (e.g., geometric or appearance) that must be preserved by the unknown map. These constraints are typically enforced simply by requesting that the function values must be globally preserved by the functional map. This means, however, that the constraints formulated using this approach often lead to underconstrained, badly defined optimization problems, especially when the number of linearly-independent descriptor functions is smaller than the number of basis functions, used to represent the map itself. This is especially problematic in the presence of non-rigid, possibly noisy deformations, for which obtaining a large set of informative, linearly independent descriptor functions can be very challenging.

Our main contribution is to notice that the standard approach for enforcing function preservation does not extract all of the information from a given descriptor. For example, the *level-sets* of the given function (i.e., the indicator functions of regions of constant value) are not necessarily preserved when using the basic function preservation constraint. This has two consequences: on the one hand, as mentioned above, this requires many descriptor functions to obtain a good approximation of a functional map, and on the other, perhaps more importantly, a solved-for functional map will not necessarily correspond to a point-to-point map, as it might not respect the "structural" properties of function preservation, as described in Section 2.4.1 in more detail. Indeed, one of our main motivations is to introduce constraints that would help guide the optimization process towards functional maps that are closer to point-to-point maps, without introducing additional computational complexity.

We show that much more information can be encoded into function (or descriptor) preservation constraints, while maintaining the overall *linear* system nature of the functional map framework, making it attractive from the computational standpoint. In particular, we show that when function preservation is encoded via *commutativity* with an underlying map, rather than simply via function value preservation, the resulting maps are both more accurate, and moreover can be obtained by using only a handful of descriptors (sometimes as few as 2-3), compared to hundreds required by the standard approach. We demonstrate on a wide range of experiments that our method helps to obtain better correspondences, largely removes the dependency of the descriptor number on the size of the reduced basis, and helps to obtain functional maps that are closer to point-to-point maps in a theoretically well-justified way.

2.2 Related work

Most closely related to this chapter are the methods based on the functional map framework, initially introduced in [69], and later extended significantly in follow-up works (e.g., [53, 77, 80] to name a few). These methods are based on the notion that it is often easier to obtain correspondences between functions, rather than points, by first using a reduced functional basis and second by formulating many *linear* constraints that allow to recover the functional map by solving a least squares system. An approach that tackles

the problem of extracting a good point-to-point correspondence from a functional map can be found in [81]. This framework has a particular advantage of being flexible and allowing to easily incorporate constraints including preservation of geometric quantities (descriptors), while at the same time being able to incorporate deformation models (e.g., isometries) via commutativity with various operators.

Despite this flexibility, one notable difficulty of using the functional map representation is that typically a large number of constraints is necessary to obtain a good solution. This includes using many descriptor preservation constraints [69], even in the case of partial maps [80] (where for example, the authors use 352-dimensional descriptors). Unfortunately, obtaining a large set of high-quality robust and informative descriptor functions can be challenging [24], and moreover noisy descriptor functions can severely affect the resulting quality of the functional map. This is especially problematic since in the original formulation [69], which has been also used in follow-up works, the number of descriptor preservation constraints is tightly linked to the size of the reduced basis, meaning that in order to obtain better correspondences more constraints are necessary, even in the absence of noise. Thus, several previous methods have tried to use regularization to improve the conditioning of the functional map computation, e.g., via sparsity [45].

In this chapter we argue, that the previously proposed approach for function preservation constraints in the functional map framework does not extract all of the available information from a given function. By drawing a link between theoretical guarantees under which functional maps correspond to point-to-point maps, we show that it is possible to formulate the descriptor preservation constraints in a way that is both more informative, and results in higher quality functional maps even as the number of basis functions increase. Remarkably, we show that this is possible without sacrificing the overall linear least squares computational advantage of this framework. Our approach is general and can be used within any other method based on the functional map representation (e.g., [53, 69, 77, 80]), by simply changing the way that constraints are formulated and solved for.

To summarize, our main contributions include:

- A novel approach to formulating function (e.g., descriptor) preservation constraints within the functional maps framework.

- Theoretical analysis demonstrating that our method results in desired point-to-point maps, in the presence of perfect descriptors.

- Both theoretical guarantees and experimental evidence that our constraints allow to extract strictly more information from descriptor functions compared to previous approaches.

We evaluate our method on a wide variety of data, and show that using our simple modification can result in significant improvement in the quality of functional maps and reduce the number of necessary descriptor constraints by an order of magnitude.

2.3 Overview

Section 2.4 describes our proposed modification to the functional map pipeline introduced in section 1.6, and discusses the main properties of the constraints that we introduce. We start by giving a general motivation and theoretical justification for our constraints in Section 2.4.1 and then describe how they can be introduced into the functional map estimation pipeline in Section 2.4.2. We list some of the properties of these constraints in Section 2.4.3, in particular proving that our approach *is strictly more informative* than the standard method for function preservation, and that it provably allows us to extract more information from the same given descriptors. Section 2.5 is dedicated to the experiments, which demonstrate that our constraints result in more accurate functional maps, and allow to obtain high quality maps with significantly fewer descriptors. Finally we conclude with Section 4.8 by mentioning some interesting challenges and directions for future work.

2.4 Novel Approach for Functional Correspondences

Classical functional map pipeline Recall that the standard functional map pipeline consists in the following steps:

- Compute the "reduced bases" $\mathbf{\Phi}_{\mathcal{M}}$ on $\mathcal{M}$, $\mathbf{\Phi}_{\mathcal{N}}$ on $\mathcal{N}$, using the first eigenfunctions of the Laplace-Beltrami operator
- Compute a set of descriptors f for shape $\mathcal{M}$, and the corresponding set of descriptors g for shape $\mathcal{N}$. Each column p of the matrix f, $f^{(p)} : V_{\mathcal{M}} \to \mathbb{R}$ corresponds to a given descriptor.
- Project these descriptors in the reduced basis $F = \mathbf{\Phi}_{\mathcal{M}}^{\dagger} f$, $G = \mathbf{\Phi}_{\mathcal{N}}^{\dagger} g$
- Find the optimal functional map matrix $\boldsymbol{C}_{opt}$ using least squares, as the minimizer of
$$\|CF - G\|^2 + \alpha \|\Delta_{\mathcal{N}} C - C \Delta_{\mathcal{M}}\|^2$$
where $\Delta_{\mathcal{N}}$, $\Delta_{\mathcal{M}}$ are matrices that represent the Laplace-Beltrami operator and α is a small regularizer weight.
- $\boldsymbol{C}_{opt}$ can then be used to transfer a given function $h : V_{\mathcal{M}} \to \mathbb{R}$ by projection on the reduced basis. h is mapped to $\mathbf{\Phi}_{\mathcal{N}} \boldsymbol{C}_{opt} \left(\mathbf{\Phi}_{\mathcal{M}}^{\dagger} h \right)$

See Section 1.6 for more details.

Limitations of the classical functional map pipeline Although simple and efficient, the basic pipeline, described above, and in Section 1.6, has several limitations: first the number of linearly independent function preservation constraints must be sufficiently high to ensure that the least squares system leads to a good approximation of the functional map. Without additional assumptions, such as sparsity, in most cases this

implies that the number of descriptors must be approximately equal to the number of basis functions (which typically ranges between 80-100). Unfortunately, obtaining a large number of descriptor functions that are robust, informative and linearly independent can often be difficult. Moreover, as described below, this basic method for enforcing descriptor preservation does not extract the full information from the given functions, leading to sub-optimal results. Finally, and perhaps most importantly, this approach does not have constraints or regularizers that would lead to the solution to point-to-point maps, which can affect the overall accuracy of the correspondence estimation pipeline.

2.4.1 Motivation

One of the primary motivations behind our approach to function preservation within the functional maps framework is a classical result that states that any non-trivial linear functional map C corresponds to a point-to-point map if and only if it preserves pointwise products of functions $c(f \cdot h) = c(f) \cdot c(h)$ for any pair of smooth functions $f, h : M \to \mathbb{R}$ (See for example Corollary 2.1.14 of [87] for a proof). Here $f \cdot h$ represents a function whose value at every point x equals to the product $f(x)h(x)$. Intuitively, this is because a functional map that preserves products of functions must satisfy $c(f^2) = c(f)^2$. If f is an indicator function of a region then $f^2 = f$ and this latter condition implies that $c(f) = c(f^2) = c(f)^2$ which means that $c(f)$ must itself be an indicator function of a region. Thus, the preservation of products of functions is directly related to guiding general functional maps to correspond to point-to-point maps in both the continuous and the discrete setting. Here non-trivial means that $c(\mathbf{1}_M) = \mathbf{1}_N$, where $\mathbf{1}_M$ is the constant function equal to one everywhere on M. This constraints is indeed trivial, because $\mathbf{1} \circ T = \mathbf{1}$ for any T. We also note that without this trivial constraint, the preservation of products of functions is still a very strong condition on a functional map and guarantees a partial correspondence coming from a generalized composition operator (See Example 2.1.10 on p. 21 of [87] for a discussion).

Perhaps the simplest way to introduce this result and intuition into the pipeline described above is by taking multiple pairs of descriptor functions $f^{(p_1)}, g^{(p_1)}$ and $f^{(p_2)}, g^{(p_2)}$ for which we expect $Cf^{(p_1)} \approx g^{(p_1)}$, $Cf^{(p_2)} \approx g^{(p_2)}$, and producing new function preservation constraints $f^{(p_3)}, g^{(p_3)}$ via $f^{(p_3)} = f^{(p_1)} \cdot f^{(p_2)}$ and $g^{(p_3)} = g^{(p_1)} \cdot g^{(p_2)}$. There are however, several issues with such an approach: first, it is not clear how many additional constraints are necessary and what pairs of descriptor functions should be taken. Secondly, any noise in the descriptors will be amplified when pairs of such functions are taken. Thus, we take a slightly different approach as described below.

To motivate our construction further, consider a pair of descriptors $f^{(p)}, g^{(p)}$ that are "fully discriminative," in the sense that for every point $x \in M$ there exists a unique point y on N such that $g^{(p)}(x) = f^{(p)}(y)$. Given such a pair of descriptors, we would expect to recover the underlying point-to-point map using a single function (descriptor) preservation constraint. However, if we simply enforce $Cf^{(p)} = g^{(p)}$, then even in the full basis we will not be able to recover the underlying map, since the function preservation constraint only leads to k_N linear equations instead of the required $k_M k_N$ equations. This is because the simple function preservation constraint does not preserve the individual

level-sets of the function values, which should be expected from a map. A simple method might be to decompose a single pair of descriptor functions $f^{(p)}, g^{(p)}$, into multiple function preservation constraints by introducing new functions by considering level-sets (or Gaussians around certain values, as in [72], Section 4.1), but this again can result in more noise and additional parameters.

2.4.2 Our constraints

In this chapter, we propose a different approach to function preservation constraints. Namely, we start with the observation that in the full basis, given a pair of corresponding probe functions $f^{(p)}, g^{(p)}$, we would expect the mapping matrix Π to be such that $\Pi_{i,j} \cdot (f_j^{(p)} - g_i^{(p)}) = 0$ for all i, j, which is equivalent to saying that indicator functions of regions of constant values of $f^{(p)}$ and $g^{(p)}$ are preserved. This corresponds to the intuition that the level-sets of functions must be preserved along with the values of the functions themselves. This constraint can be rewritten via commutativity as $\Pi Diag(f^{(p)}) = Diag(g^{(p)})\Pi$, where $Diag(v)$ is the matrix that contains the values of the vector v along the diagonal and is zero elsewhere. This form also makes apparent the relation between preservation of level sets and function products. Indeed, if $h : M \to \mathbb{R}$ is any function on M, then its pointwise product with $f^{(p)}$ is obtained, in the discrete formulation, via the matrix vector product $Diag(f^{(p)})h$. Therefore, $\Pi Diag(f^{(p)}) = Diag(g^{(p)})\Pi$ implies $\Pi Diag(f^{(p)})h = Diag(g^{(p)})\Pi h$, which implies that the associated linear mapping C between functions must satisfy $c(f^{(p)} \cdot h) = g^{(p)} \cdot c(h)$ for any $h : M \to \mathbb{R}$, which corresponds exactly to the product rule.

Our constraints in the reduced basis As discussed above, in the functional map framework, the key map estimation step is done in the reduced basis. Thus, we introduce our constraints by following the idea of commutativity with an operator based on the descriptor, as discussed in the previous paragraph. However, in the reduced basis the commuting matrices will not remain diagonal. For a given pair of descriptor functions $f^{(p)}, g^{(p)}$, we therefore create matrices $X^{(p)} = \Phi_M^+ Diag(f^{(p)})\Phi_M$ and $Y^{(p)} = \Phi_N^+ Diag(g^{(p)})\Phi_N$. Finally, we add the corresponding constraints to the system into Eq. 1.1 by requiring the unknown map C to satisfy in the least squares sense:

$$\sum_s \|CX^{(p)} - Y^{(p)}C\|^2, \tag{2.1}$$

where the summation is across the available descriptors, and we use the Frobenius matrix norm.

2.4.3 Properties

As mentioned above, the descriptor preservation constraints alone do not extract all of the information that is present in a given descriptor. In particular even if the descriptors are perfect and identify each vertex uniquely, the classical constraints $CF = G$ may still not be sufficient to identify each vertex. As a toy example, if $n_M = 2$, $n_N = 2$ and

$f = (1, 2)$ and $g = (1, 2)$, then there is a unique point-to-point map that preserves these functions. However, using only the constraint $\Pi \begin{pmatrix} 1 \\ 2 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$ could also lead to the solution $\Pi_1 = \begin{pmatrix} 0 & 0.5 \\ 2 & 0 \end{pmatrix}$. Thus, we can see that although Π_1 preserves the descriptor functions, it fails to preserve the values pointwise: it fails to map vertices that have a value to vertices that have a similar value. Π_1 does not preserve the commutativity constraint, and indeed Π_1 is not a permutation matrix.

Below we show that the above phenomenon cannot happen if Π is enforced to be a doubly stochastic matrix: i.e., having entries that all lie in the interval $[0, 1]$ and whose rows and columns sum to 1:

Theorem 2.4.1 *Let $f \in \mathbb{R}^n$ and $g \in \mathbb{R}^n$ be such that the multiset of values contained in f and in g are the same. Let Π be an $n \times n$ matrix such that $\forall i, j, 0 \leq \Pi_{i,j} \leq 1$ and $\sum_k \Pi_{i,k} = 1$, $\sum_k \Pi_{k,j} = 1$. Then $\Pi f = g$ implies $\Pi_{i,j} = 0$ whenever $f_j \neq g_i$.*

Proof : We proceed by induction on the values of f. Let $L = \max(f) = \max(g)$. By assumption, the sets $I_f = \{k | f_k = L\} \subset \{1, \ldots, n\}$ and $I_g = \{k | g_k = L\} \subset \{1, \ldots, n\}$ must have the same cardinality. Moreover, each g_k for $k \in I_g$ can only be obtained from combinations of f_k for $k \in I_f$. Thus, $\Pi_{k,k'} = 0$ if $k \in I_g$ and $k' \notin I_f$. These constraints also imply $\Pi_{k,k'} = 0$ if $k \notin I_g$ and $k' \in I_f$, because each column of Π should sum to 1. $\square$

Unfortunately, enforcing a matrix to be a stochastic matrix involves inequality constraints that do not translate well in the reduced basis: e.g., the projection of a stochastic matrix in the reduced basis may not remain stochastic. Thus, it is not easy to restrict to such matrices in the reduced basis. Instead, rather than enforcing inequality constraints we propose to introduce the commutativity with respect to the operators derived from the descriptor functions as described above. In our toy example, the commutativity constraint would be

$$\Pi \begin{pmatrix} 1 & 0 \\ 0 & 2 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 2 \end{pmatrix} \Pi$$

We can see that enforcing such a constraint eliminates the wrong solution Π_1.

Recall that any functional map that corresponds to a point-to-point map should satisfy $\Pi \mathbf{1} = \mathbf{1}$, which would thus be a natural constraints to use in our optimization. Interestingly our new commutativity constraint along with the additional regularization, requiring the map to preserve the constant function $\Pi \mathbf{1} = \mathbf{1}$, implies the previously used constraints $\Pi f = g$ even in the reduced basis, as proved in the following theorem:

Theorem 2.4.2 *If $f \in \mathbb{R}^{n_M}$, $g \in \mathbb{R}^{n_N}$ and $\Pi \in M_{n_N, n_M}(\mathbb{R})$, then $\Pi Diag(f) = Diag(g)\Pi$ and $\Pi \mathbf{1} = \mathbf{1}$ implies that $\Pi f = g$. Similarly, if C is in the reduced basis, where the first basis function is a constant function, and $Ce_1 = e_1$, then the commutativity constraint $C\Phi_M^+ Diag(f)\Phi_M = \Phi_N^+ Diag(g)\Phi_N C$ implies that $C\Phi_M^+ f = \Phi_N^+ g$.*

Proof : We consider the first case, in the full basis: $\Pi f = \Pi Diag(f)\mathbf{1} = Diag(g)\Pi\mathbf{1} = Diag(g)\mathbf{1}$. For the second case, by assumption, we have: $C\Phi_M^+ f = C\Phi_M^+ Diag(f)\Phi_M e_1 = \Phi_N^+ Diag(g)\Phi_N C e_1 = \Phi_N^+ Diag(g)\Phi_N e_1 = \Phi_N^+ g$. $\square$

However, in practice, it might still be useful to enforce both the commutativity constraint and the $CF = G$ constraint as the latter might give more control on the importance of preserving the function globally, by e.g., adding a scalar weight. Note that this theorem is only meant to be a theoretical guarantee that our formulation includes at least as much information as the previous one.

2.5 Experiments

As described above, the new commutativity constraints allow us to extract more information from the same set of descriptors, and furthermore allow us to guide the functional map estimation process closer to point-to-point maps, while still maintaining the linear (least squares) complexity of the optimization. Below we demonstrate the utility of these constraints on a wide range of shapes and deformations and show that our approach allows to significantly reduce the number descriptor functions necessary to estimate an accurate functional map, and even improve results with the increase in the size of the basis for a fixed number of descriptors, which is not true for the previously used constraints.

2.5.1 Using few descriptors

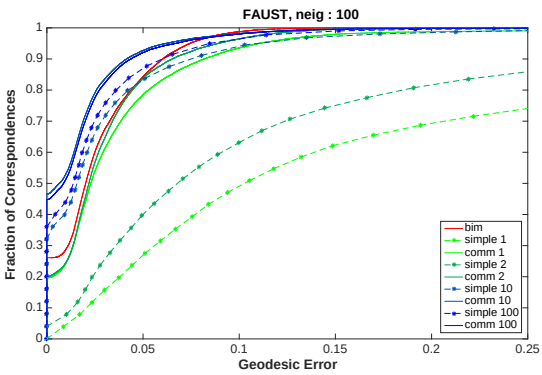
In our first set of experiments, we plot the average correspondence error for several methods on three standard benchmarks: FAUST [13], SCAPE [7], and TOSCA [19].

Our main goal in this experiment is to show that by formulating the descriptor preservation constraints via commutativity, rather than using the original approach based on preservation of values, results in more accurate functional map inference, without requiring any additional information.

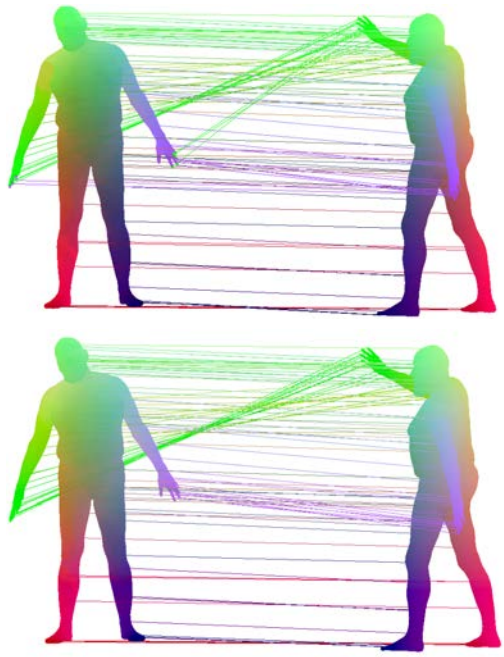
We compare our results to the following methods:

- Blended Intrinsic Maps [43]
- The original method proposed in [69] and used in follow-up works [77, 80], that formulates function preservation constraints, based on values.

We note that our approach can be incorporated into *any* pipeline for estimating functional maps, which uses function (e.g., descriptor) preservation constraints. As such, it can be easily combined with the other techniques that have been introduced for optimizing functional map computations, e.g., based on sparsity [77] or specific prior structure existing in partial correspondences [80]. Therefore, our goal is not to demonstrate that our particular choice of descriptors or parameters results in state-of-the-art correspondences on these benchmarks, but rather to show that our formulation of function preservation via commutativity allows to obtain more accurate results than the one based on function values alone.



(a) Error plots showing the accuracy of our descriptor preservation via commutativity (solid lines) compared to simple value preservation (dotted) and the Blended Intrinsic maps (red) on shape pairs from the FAUST dataset. Our method allows to obtain superior performance using even a very small descriptor set.



(b) Example maps obtained by formulating the descriptor preservation with the simple method (up) and using our commutativity approach (down), using exactly the same descriptor functions.

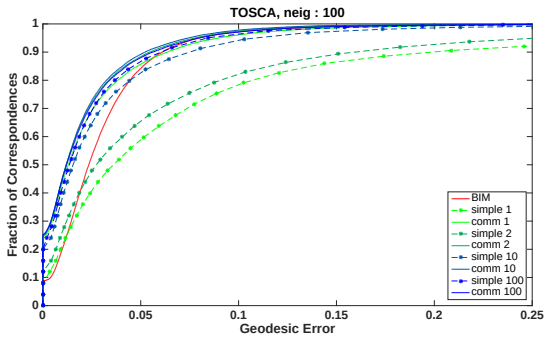
Figure 2.1 – Performance of our technique on the FAUST dataset [13] compared to the standard functional maps approach and Blended Intrinsic Maps [43].

For this, we used the original functional map estimation pipeline introduced in [69], with the same code and parameters. Namely, we used a varying number $(1, 2, 10,$ and $100)$ of descriptor functions, and compared the results of estimating the functional map either by minimizing:

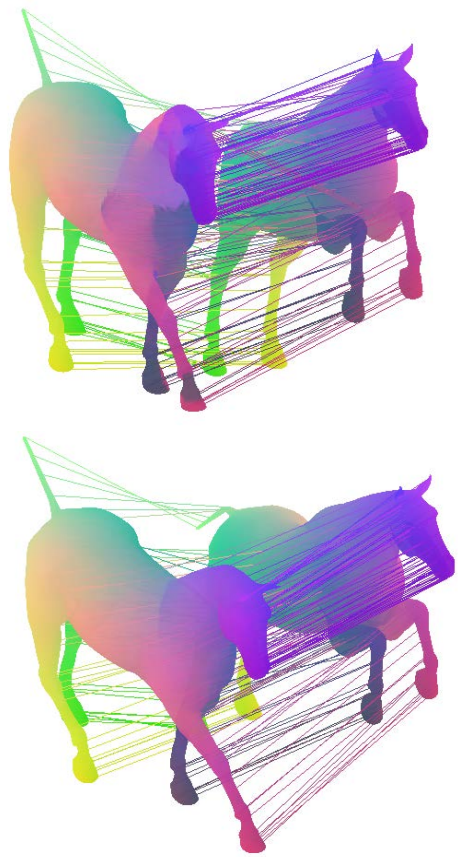
$$C_{\text{opt}} = \arg\min_{C} \|CF - G\|^2 + \alpha\|\Delta_2 C - C\Delta_1\|^2,$$

as described in Section 1.6 or using our constraints, which simply adds an extra term to the energy above, given by $\sum_i \|CX_i - Y_i C\|^2$, as described in Eq. 2.1 above. In both cases we computed the functional map C by solving a linear least squares system, using a vectorization of the functional map C (re-writing it as a vector c), and solving the system $Ac = b$, where A and b are obtained by rewriting the above energy in matrix-vector form. After estimating the functional map C we used the post-processing technique of [69] based on high-dimensional ICP to both refine the functional map and convert it to a point-to-point correspondence.

In all of the experiments in this section we used $n_{\text{eig}} = 100$ eigenfunctions to represent



(a) Error plots showing the accuracy of descriptor preservation via our commutativity approach (solid lines) compared to simple value preservation (dotted) and the Blended Intrinsic maps (red) on the TOSCA dataset.

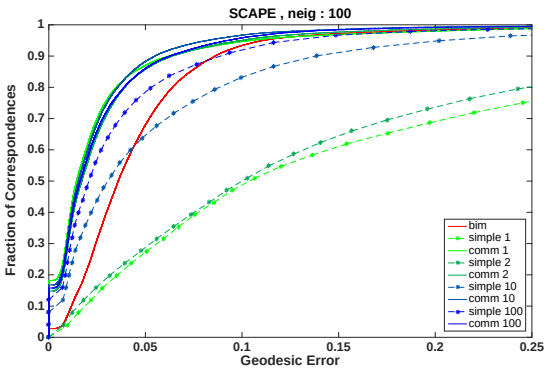


(b) Example maps obtained by formulating the descriptor preservation with the simple method (up) and using our commutativity approach (down), using exactly the same descriptor functions.

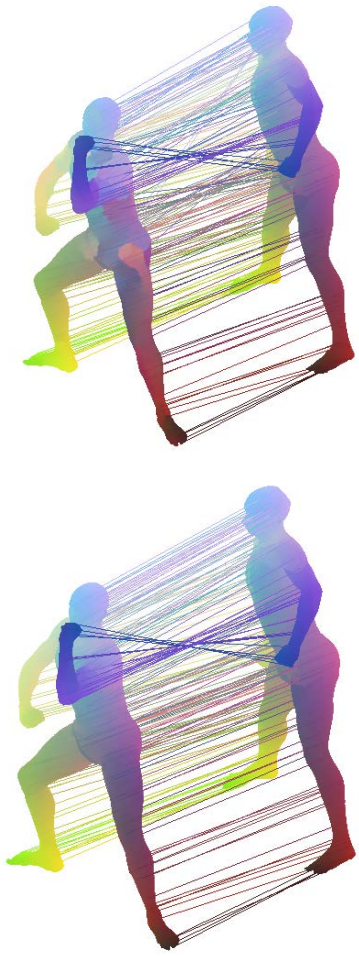
Figure 2.2 – Performance of our technique on the TOSCA dataset [19] compared to the standard functional maps approach and Blended Intrinsic Maps [43].

the functional basis, by discretizing the Laplace-Beltrami operator using the standard cotangent-scheme [63, 76], and a sparse eigensolver, to estimate the basis. We followed the exact pipeline suggested in [69] for map estimation and simply sub-sampled the set of descriptors used in that work (Wave Kernel Signature [8] and Wave Kernel map based on segment correspondences).

Figures 2.1a, 2.2a, 2.3a show the results obtained using our approach compared to the basic function-preservation method on the three benchmarks, using a varying number of descriptor preservation constraints. We evaluated each method on 100 pairs of shapes in the FAUST dataset, 76 shape pairs in TOSCA and 71 pairs in SCAPE, by taking each shape to be a source in exactly one pair. We follow the evaluation protocol introduced



(a) Error plots showing the accuracy of our descriptor preservation via commutativity (solid lines) compared to simple value preservation (dotted) and the Blended Intrinsic maps (red) on the SCAPE dataset. Our method allows to obtain superior performance using even a very small descriptor set.



(b) Example maps obtained by formulating the descriptor preservation with the simple method (up) and using our commutativity approach (down), using exactly the same descriptor functions.

Figure 2.3 – Performance of our technique on the SCAPE dataset [7] compared to the standard functional maps approach and Blended Intrinsic Maps [43].

in [43], by plotting on the x-axis a geodesic threshold, and on the y-axis, the fraction of the correspondences obtained by each method that are at a distance that is less than this threshold from the ground truth map. The geodesic distances (approximated using Dijkstra's algorithm) are divided by the scaling factor $\sqrt{Area}$, where $Area$ is the total area of the shape. In this work, similarly to other non-symmetry aware intrinsic methods, we do not disambiguate left-right symmetries, and thus take the minimum between the distance between the matched vertex and the ground truth, and the distance between

the matched vertex and the symmetric of the ground truth as our distance to target.

In each plot, the red curves represent the performance of [43], the blue/green curves represent approaches based on the functional map framework (green: using only 1 descriptor, blue: using 100 descriptors). The curves with $*$ symbols use the original function-preservation formulation, whereas those with solid lines use our new approach.

We notice the following general trend: the method that uses our commutativity constraints usually leads to better results compared to the classic descriptor preservation constraints, and the improvement is particularly large when only a few descriptors are used. Using more descriptors leads to little improvement for this new method, which, together with the previous fact, highlights that the new formulation helps in extracting more information from the same descriptors.

We also notice that increasing the number of descriptors used to 100 is not generally the best choice for getting better results, which shows that this method is well-suited for performing on few reliable descriptors.

Perhaps most remarkably, our new formulation allows to obtain results with *only two* descriptor functions (e.g., on the SCAPE dataset) that are better than the ones produced by the original method using the full set of 100 descriptors.

In Figures 2.1b, 2.2b and 2.3b we also provide some example maps computed using descriptor preservation with commutativity vs. the simple value-based approach. Note that the resulting maps are typically less noisy and more globally consistent, despite using exactly the same information in the optimization, which also suggests that our formulation helps to obtain more accurate functional map. For visualization, we sampled $100, 300$ and 200 points on the source shape uniformly from the list of all vertices, for the pairs from the FAUST (Fig. 2.1b), TOSCA (Fig. 2.2b), and SCAPE datasets (Fig. 2.3b) respectively.

2.5.2 Changing the dimension of the reduced space

In our second range of experiments, we show the dependence of the results on the number of functions used in the basis for functional maps. Here, rather than changing the number of descriptor functions, we fix the descriptor set and change the dimensionality of the basis and evaluate the quality of the approximation of the point-to-point correspondences using the functional map pipeline.

In Figure 2.4 we show the average correspondence error between a subset of shapes in the FAUST dataset [13], using the same pipeline described in the previous section, for a fixed number (in this case two) of descriptor functions. In particular, we used the Wave Kernel Signature for a single energy value, along with a single descriptor function that is aimed at segment preservation using the Wave Kernel Map with a fixed energy value. This gives us two descriptor functions, which we incorporate into the functional map energy using either the standard descriptor preservation constraints, as done in [69] or using our commutativity-based approach. We then convert the estimated functional map to a point-to-point correspondence and evaluate its accuracy using the distance to the ground truth. We plot the average pointwise map error, computed the same way as in the previous experiment, across the shape pairs for a varying number of basis functions.

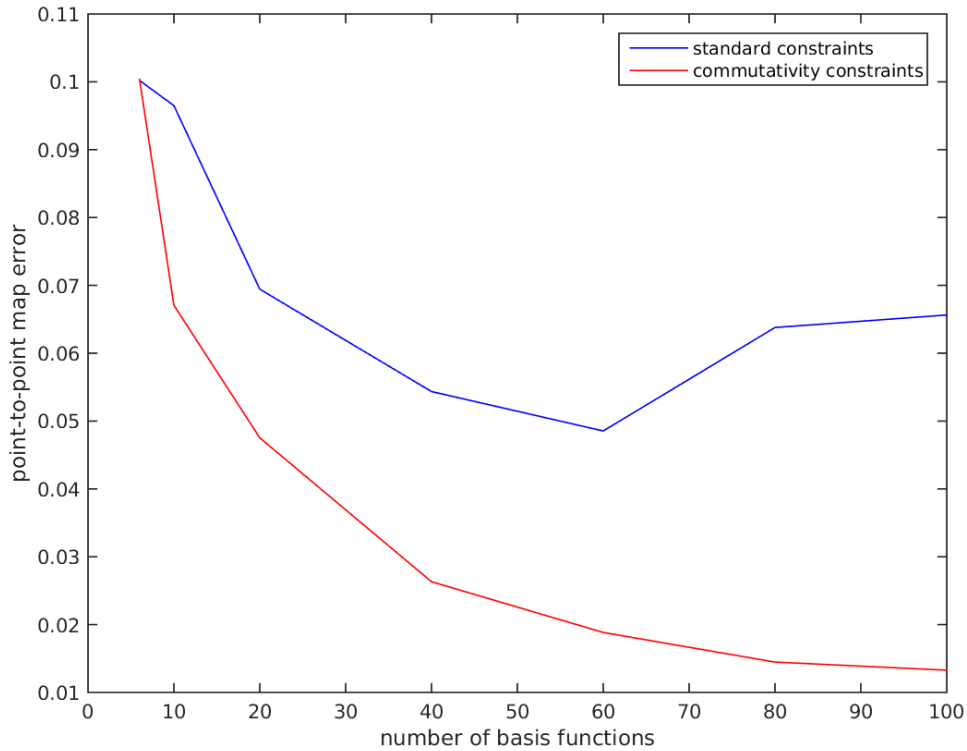


Figure 2.4 – Average error on pairs of shapes in the FAUST dataset, depending on the number of basis functions in the functional map representation, for a fixed number (two in this case) descriptors. Unlike the standard approach of [69], which deteriorates when the size of the basis significantly exceeds the number of descriptors, our method continues to produce high-quality results even for a small number of descriptors and a large number of basis functions. The average error is computed as the average geodesic distance to the ground truth correspondence, symmetries allowed.

As can be seen in Figure 2.4, compared to the basic method for descriptor preservation, our approach allows not only to improve quality of the correspondences significantly, without using any additional information, but also provides more resilience with respect to the choice of the number of basis functions, for a fixed descriptor set. This implies that our approach can potentially enable more accurate correspondence computation based on the functional map pipeline, without requiring any additional information, and supports the idea that using our formulation allows to extract additional information from descriptor functions, which in turns results in better pointwise maps.

2.6 Conclusion, Limitations & Future work

We proposed a new formulation for incorporating descriptor (or more general function) preservation constraints within the functional map framework, which enables finding better solutions to the non-rigid shape matching problem. Our formulation is especially useful when the number of descriptors is lower than the dimension of the reduced space of functions since it allows to extract more information from the same set of given descriptors. Our formulation is applicable in the same settings as the original functional maps framework, with the main limiting factor being the computational time necessary to assemble and solve our optimization problem, which nevertheless remains linear in the unknown map. We also note that we do not enforce the preservation of the constant function in practice, and thus our formulation should, in principle be also applicable even to partial maps. We leave the exploration of this as an interesting direction for future work.

Conceptually, we propose to consider descriptors or functions as *linear functional operators* acting on other functions through multiplication, unlike the standard approach which views them simply as scalar-valued signals. We believe that this idea is particularly exciting for future work, and are planning to investigate other ways in which informative descriptors and constraints can be defined directly as functional operators, opening the door to a much richer way of characterizing shapes and their geometry, which can be useful in shape matching problems.

Chapter 3

Improved functional mappings via product preservation

In this chapter, we consider the problem of information transfer across shapes and propose an extension to the widely used functional map representation. Our main observation is that in addition to the *vector space* structure of the functional spaces, which has been heavily exploited in the functional map framework, the functional *algebra* (i.e., the ability to take pointwise products of functions) can significantly extend the power of this framework. Equipped with this observation, we show how to improve one of the key applications of functional maps, namely transferring real-valued functions without conversion to point-to-point correspondences. We demonstrate through extensive experiments that by decomposing a given function into a linear combination consisting not only of basis functions but also of their pointwise products, both the representation power and the quality of the function transfer can be improved significantly. Our modification, while computationally simple, allows us to achieve higher transfer accuracy while keeping the size of the basis and the functional map fixed. We also analyze the computational complexity of *optimally* representing functions through linear combinations of products in a given basis and prove NP-completeness in some general cases. Finally, we argue that the use of function products can have a wide-reaching effect in extending the power of functional maps in a variety of applications, in particular by enabling the transfer of high-frequency functions without changing the representation size or complexity.

3.1 Introduction

Shape correspondence is one of the key problems in digital geometry processing with applications in fields ranging from manufacturing to shape morphing [42], statistical shape analysis [13,35], texture mapping, animation and deformation transfer [91] among a wide variety of others. In all of these applications, the key requirement of shape matching algorithms is to enable *information transfer* across two or more shapes.

Over the past several decades, a large number of computational techniques has been developed for addressing the shape matching problem. While most early methods have concentrated on rigid shape matching, more recently numerous approaches have also been proposed in the more general context of finding correspondences between non-rigid shapes, such as humans in arbitrary poses [100]. One of the major challenges arising in this setting is that the space of possible correspondences between points on a pair of shapes is exponential, which gives rise to difficult optimization problems when establishing reliable point-to-point maps.

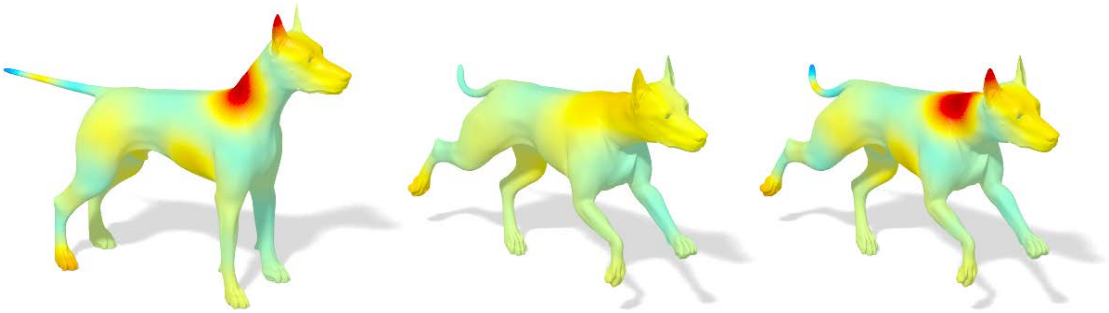


Figure 3.1 – Transferring a real-valued function from a source shape (left) onto the target using a fixed functional map with the standard approach (center) vs. our extended method (right).

To address this challenge, recent works have focused either on defining a consistent parameterization for a pair of shapes, often with the use of landmarks (e.g., [4,5,50]), or by choosing a different representation for shape correspondences, which is more amenable to direct optimization and manipulation. These include either soft maps or measure couplings, which can benefit from computational techniques in optimal transport [89,90], and a class of techniques based on the recently introduced functional map representation [69, 70]. This latter is based on the idea of using correspondences between real-valued function rather than points on the shapes. Since certain (e.g., square integrable) function spaces admit a vector space structure and therefore can be endowed with a multi-scale functional basis (e.g., the Laplacian eigenbasis), a common approach is to restrict the search for an optimal correspondence to a subspace spanned by a small number of basis functions. This implies that the optimal functional map can be represented using a moderate-sized matrix, independent of the number of points on a pair of shapes.

This restriction to functional maps between small subspaces significantly reduces the computational complexity of the shape correspondence problem as it leads to simpler optimization problems with fewer unknowns (in the simplest setting, the shape correspondednce boils down to a least-squares problem [70]). At the same time, it also somewhat reduces the utility of the computed functional map especially since converting a functional to a point-to-point map can be a challenging problem in itself [81, 101]. Motivated by this fact, several applications have argued for using functional maps directly, by exploiting their ability to transfer real-valued functions across shapes without recovering the point-to-point map. Examples of applications include segmentation transfer, demonstrated in the original article [69], tangent-vector field design [9], image and shape co-segmentation [38, 104] and even more recently, consistent mesh quadrangulation [10]. In all of these works, a given function on the source shape is represented as a linear combination of the basis functions which are then transferred using the functional map onto the target. Since in most cases, the basis consists of the first few eigenfunctions of the Laplace-Beltrami operator, in practice functional maps can only transfer sufficiently smooth functions, leaving out high-frequency details, which severely limits the applicability of the entire functional maps framework. As a possible remedy, several recent works

considered alternatives to the Laplace-Beltrami basis [46, 60, 66].

In this work, we argue that more accurate function transfer can be achieved within the functional maps framework without changing the basis or increasing its size. In particular, we propose to use the algebraic structure of function spaces [36], which means that in addition to defining a vector space through scalar multiplication and addition, functions can also naturally be multiplied pointwise. Moreover, most natural functional maps (i.e., those arising from point-to-point ones) must preserve this algebraic structure. Putting these two properties together, our key observation is that *the ability to transfer basis functions also gives us the ability to transfer their pointwise products*. This means that if a given function on a source shape can be represented via not just the elements of the basis, but also their pointwise products, a given functional map can be used to transfer it accurately onto the target, as shown in Figure 3.1.

We demonstrate through extensive experiments that this observation can lead to a significant improvement in the function transfer quality without changing the functional map representation. Remarkably, it allows us to map even high frequency information by using only a low-frequency basis without converting a functional map to a point-to-point one. As a result, our approach can have a direct impact on all applications of the functional map framework.

3.2 Related Work

As mentioned above, relating information across different shapes is among the most basic problems in shape analysis. Since our approach is designed to better exploit a *given functional map* and as such is more closely related to *representing* correspondences, rather than *computing* them, in the following we primarily give an overview of the most common representations for maps between shape, and refer an interested reader to recent surveys on shape matching [12, 94, 100] for a more in-depth discussion of correspondence problems.

Most early models of non-rigid correspondences between shapes are based on the standard notion of pointwise mappings, e.g., [18, 43, 50, 71]. A particular instance of this class of methods is based on spectral embeddings defined by the Laplace-Beltrami eigenfunctions, followed by a correspondence procedure in the embedding space, e.g. [28, 57]. Shtern and Kimmel proposed constructing spectral embedding using pointwise products of Laplace-Beltrami eigenfunctions [85] and the triple products of their gradients with the surface normal [86] as a means of capturing additional information. The main drawback of pointwise correspondence models is that such frameworks can often be unstable and lead to difficult non-linear non-convex optimization problems. As a result, several authors have proposed to consider more general notions of mappings, which are more amenable to direct optimization. In the most basic setting, such generalized mappings arise as relaxations of pointwise correspondence problems, as e.g., in [11, 48]. In these scenarios, however, such representations are typically used as intermediate steps, aimed at facilitating pointwise map recovery.

A more principled approach to soft mappings is provided by the notion of measure couplings, that have been used for both representing and finding correspondences [61, 62, 89]

and that are intimately related to the formalism of optimal transport. These representations have a probabilistic interpretation and benefit from computational advances in efficiently solving certain optimal transport problems [26, 88]. Consequently, they have recently gained prominence in addressing shape matching and alignment tasks [54, 90]. Nevertheless, the complexity of these representations is directly related to the sampling density of the shapes, and can quickly become prohibitive, often requiring heuristics and multi-resolution schemes.

A different formalism that this chapter directly builds on is provided by the functional maps framework [69]. This representation is based on representing correspondences through their action (via pull-back) of real-valued functions. Since the pull-back of functions is linear and since functional spaces can be endowed with a multi-scale basis, this leads to a representation of correspondences as moderately-sized matrices, which can be directly manipulated and optimized for. Although initially introduced as a tool for shape matching, functional maps have been used for relating tangent vector fields [9], extending the Generalized Multi-Dimensional Scaling to the spectral domain [2], computing maps between symmetric [72] and partial [52, 53, 80] shapes, coupled bases [46] and even consistent quadrangulation [10] among others. Most recently, functional maps were integrated as differentiable layers into intrinsic deep learning architectures [51].

All of these applications benefit from the properties of the functional representation, including its compactness, which often translates into relatively simple optimization problems, and its capacity for information transfer. Indeed, although the original article [69] proposed an approach for recovering a point-to-point map from a functional one, follow-up works, have observed that, in many scenarios functional maps can be used directly for transferring information such as tangent vector fields [9] or segmentations in shape collections [38] among others. In these works, the information being transferred is represented by using the *vector space* structure of functional spaces. At the same time, as we argue in this chapter, real-valued functions also have a natural *algebra* defined through pointwise products, which can be used directly to improve the quality of function transfer without relying on point-to-point maps. Remarkably, we show how this property can be exploited to transfer high-frequency functions even in the presence of a functional map relating only low-frequency bases.

In the previous chapter we saw that by representing descriptors as linear operators acting on functions through *pointwise multiplication*, it is possible to obtain a significant improvement in the quality of the recovered functional map. Similarly to the previous chapter, this chapter is also motivated by the classical result that a non-trivial functional map acts as a point-to-point map if and only if it preserves pointwise function products [87]. However, rather than trying to improve the quality of an optimized functional map through better use of descriptors, our emphasis is on showing how *a given* functional map can be used more effectively by allowing the transfer of not just basis functions but also their (possibly high-order) point-wise products. More concretely, unlike previous works, including the previous chapter, which have always restricted the functional subspaces to linear combinations of basis functions, we show that a much richer space can be constructed and used without sacrificing the computational and storage efficiency of functional maps, by exploiting pointwise products of basis functions.

3.3 Motivation and Overview

Classical functional map pipeline We recall that the basic functional map pipeline, introduced in [69] consists in the following steps (see also Section 1.6 and the previous chapter for a more detailed discussion):

- Compute the "reduced bases" $\mathbf{\Phi}_{\mathcal{M}}$ on $\mathcal{M}$, $\mathbf{\Phi}_{\mathcal{N}}$ on $\mathcal{N}$, using the first eigenfunctions of the Laplace-Beltrami operator
- Compute a set of descriptors f for shape $\mathcal{M}$, and the corresponding set of descriptors g for shape $\mathcal{N}$. Each column p of the matrix f, $f^{(p)} : V_{\mathcal{M}} \to \mathbb{R}$ corresponds to a given descriptor.
- Project these descriptors in the reduced basis $F = \mathbf{\Phi}_{\mathcal{M}}^{\dagger} f$, $G = \mathbf{\Phi}_{\mathcal{N}}^{\dagger} g$
- Find the optimal functional map matrix $\boldsymbol{C}_{opt}$ using least squares, as the minimizer of
$$\|CF - G\|^2 + \alpha \|\Delta_{\mathcal{N}} C - C\Delta_{\mathcal{M}}\|^2$$
where $\Delta_{\mathcal{N}}$, $\Delta_{\mathcal{M}}$ are matrices that represent the Laplace-Beltrami operator and α is a small regularizer weight.
- $\boldsymbol{C}_{opt}$ can then be used to transfer a given function $h : V_{\mathcal{M}} \to \mathbb{R}$ by projection on the reduced basis. h is mapped to $\mathbf{\Phi}_{\mathcal{N}} \boldsymbol{C}_{opt} \left(\mathbf{\Phi}_{\mathcal{M}}^{\dagger} h \right)$

Limitations of the classical functional map pipeline While very simple and intuitive, this basic functional map procedure has a severe limitation: it only allows to transfer functions (or their projections) that lie within the vector space spanned by $\mathbf{\Phi}^{\mathcal{M}}$. When the basis is given by the first $k_{\mathcal{M}}$ eigenfunctions of the Laplace-Beltrami operator, which is the most popular choice in practice, this implies that only *sufficiently smooth* or low-frequency functions can be transferred using the map $\boldsymbol{C}$.

Functional Algebra In this chapter, we argue that this is an unnecessary restriction, which can be, at least partially lifted in many settings. For this we propose using the algebraic structure of the functional spaces on the shapes, which has so far not been exploited fully. The key property we consider is that in addition to defining a *vector space* via inner products, real-valued (square integrable) functions also have a well-defined point-wise product operation: $f_1 \odot f_2 \to f_3$, where $f_3(x) = f_1(x) f_2(x)$ at every point $x \in \mathcal{M}$.

This point-wise product operation is compatible with functional maps: it is well-known [87] that a non-trivial linear functional map corresponds to a point-to-point one if and only if it preserves the functional algebra. I.e. the distributivity of the functional map over point-wise product:

$$\mathbf{c}(f_1 \odot f_2) = \mathbf{c}(f_1) \odot \mathbf{c}(f_2) \quad \forall\ f_1, f_2. \tag{3.1}$$

Intuitively, if a functional map corresponds to a point-to-point one, then product preservation follows directly from the definition of composition. Conversely, if a functional map preserves pointwise products, then $\mathbf{c}(f^2) = \mathbf{c}(f) \odot \mathbf{c}(f)$, which implies that indicator functions should be mapped to indicator functions.

Remark: It is interesting to measure how Eq. (3.1) changes when the functional map deviates from a pointwise map. A simple computation shows that if $\tilde{\mathbf{c}} = \mathbf{c} + \delta$, where $\mathbf{c}$ satisfies Eq. (3.1), then:

$$\begin{aligned}\tilde{\mathbf{c}}(f_1 \odot f_2) =& \tilde{\mathbf{c}}(f_1) \odot \tilde{\mathbf{c}}(f_2) + \delta(f_1 \odot f_2) - \delta(f_1) \odot \delta(f_2) \\ &- \delta(f_1) \odot \tilde{\mathbf{c}}(f_2) - \tilde{\mathbf{c}}(f_1) \odot \delta(f_2)\end{aligned}$$

Note that the result involves the cross-terms $\tilde{\mathbf{c}}(.) \odot \delta(.)$. Therefore, to bound the deviation, one needs not only a bound on the error δ but also potentially a bound on the functional map itself. Some analysis of such bounds was presented in [40] (Condition 3.1 and Proposition 3.3), and we leave the precise analysis of the failure of product preservation as interesting future work.

Now, suppose that a functional map is expected to be of sufficiently "high-quality" to satisfy the product preservation property. For example, the functional map should approximate some *unknown* point-to-point map. Then, we can use Eq. (3.1) explicitly without computing that point-to-point map. Namely, if we would like to transfer a given real-valued function, we can decompose it into a linear combination of the basis functions *and of their pointwise products*, and then transfer the coefficients by combining the linearity of the map with Eq. (3.1) to compute the image of products of basis functions, and thus of the given function.

Our main motivation for using this construction is that it allows us to extend the space of functions that can be transferred by a given functional map $\boldsymbol{C}$ without changing the basis and without converting it to a point-to-point map. For example, consider the standard Fourier basis functions and their pairwise products shown in Figure 3.2. Note that some products exhibit higher frequency behavior. As a result, they allow a significantly more accurate function reconstruction, shown in Figure 3.3 for different values of $k_{\mathcal{M}}$. Finally, since Eq. (3.1) allows us to transfer coefficients of products of basis functions, given the knowledge of images of basis functions themselves, this enables a more accurate function transfer, which can have a direct effect on all applications that use this aspect of functional maps, e.g., [9, 10, 104].

3.4 Method Description

More concretely, suppose that we are given a pair of discrete shapes $\mathcal{M}$ and $\mathcal{N}$, represented as triangle meshes containing $n_{\mathcal{M}}$ and $n_{\mathcal{N}}$ points respectively. Moreover, suppose both shapes are endowed with a reduced functional basis, stored as columns of matrices $\Phi_{\mathcal{M}}, \Phi_N$ of size $n_{\mathcal{M}} \times k_{\mathcal{M}}, n_{\mathcal{N}} \times k_{\mathcal{N}}$ for some small $k_{\mathcal{M}}, k_{\mathcal{N}}$. In practice we use the eigenfunctions of the Laplace-Beltrami operators to construct the basis, although the ideas presented below are not tied to this choice.

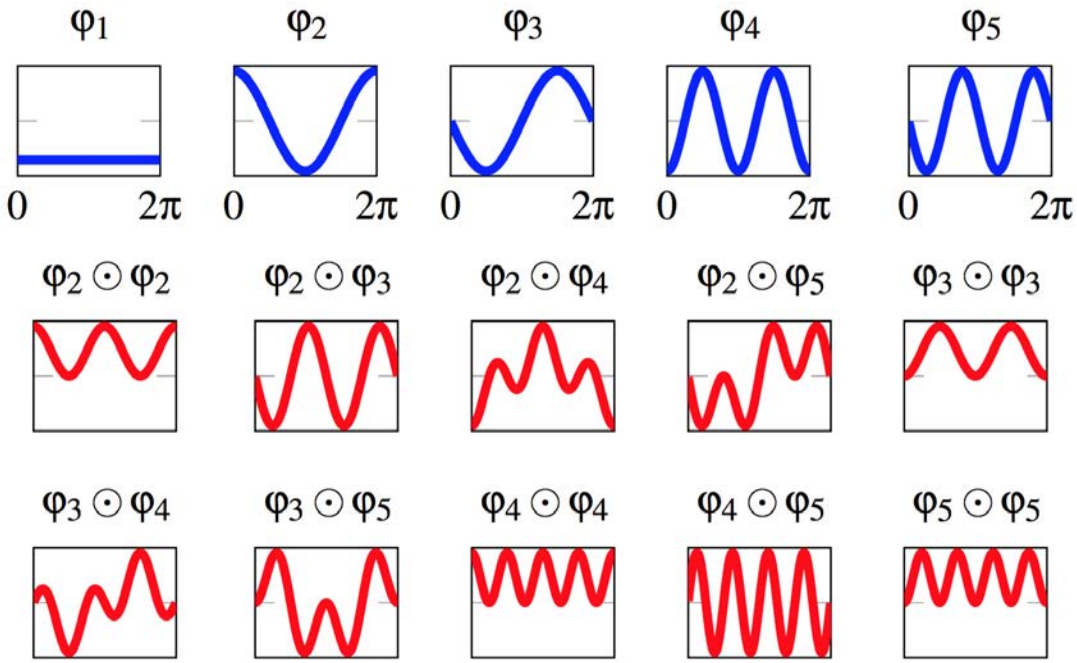


Figure 3.2 – The first 5 standard 1D Fourier basis functions (in blue) and their pairwise products (in red) on a periodic domain. Above each function we report the product from which it is generated. Note how the products exhibit higher frequencies compared to the original functions. Note also some linear dependencies among some of these functions (e.g.: $\varphi_2 \odot \varphi_2 + \varphi_3 \odot \varphi_3 + \lambda\varphi_1 = 0$ for a scalar λ).

We also suppose that we are given a functional map, represented in the reduced basis as a matrix $\boldsymbol{C}$ of size $k_{\mathcal{N}} \times k_{\mathcal{M}}$, which is either induced by a point-to-point map or is close to such a map. For example, if $\boldsymbol{C}$ comes from the optimization pipeline outlined in Section 3.3, then we would expect it to be close to satisfying this property, as this pipeline is intended to recover a point-to-point map. Such a functional map should therefore satisfy the distributive relation of Eq. (3.1). Finally, we are also given a function f, stored simply as a vector $\mathbf{f}$ of size $n_{\mathcal{M}}$, that we would like to transfer from $\mathcal{M}$ to $\mathcal{N}$.

Our goal then is to use the distributive property of multiplication, to extend the definition of our map to a space larger than that spanned by the original basis functions.

3.4.1 Function Representation

To achieve higher accuracy transfer, our approach will be to first represent a given function f as a linear combination of the basis functions $\varphi_1, \ldots, \varphi_k$ *and* of their pointwise

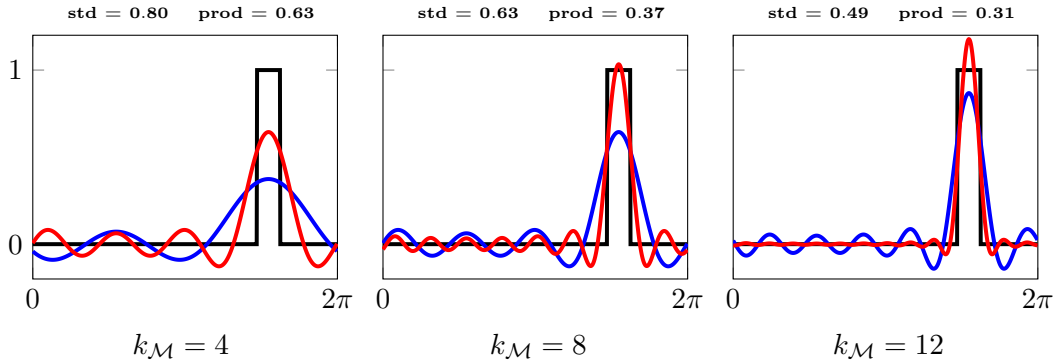


Figure 3.3 – A 1D indicator function defined on the standard periodic domain (in black), its reconstruction using the standard first five Laplacian basis functions (in blue) and using all the products of the first five eigenfunctions (in red). With an increasing value of k from left to right $k_{\mathcal{M}} = 4$, $k_{\mathcal{M}} = 8$ and $k_{\mathcal{M}} = 12$. The bases used are the ones from Figure 3.2. Note the significant improvement in reconstruction error shown above for all the different values of k.

products. I.e., we look for the best coefficients a_i and b_j to approximate f as

$$f \approx \sum_{i=1}^{k} a_i \varphi_i + \sum_{j=1}^{P} b_j \prod_{l=1}^{r_j} \varphi_{i_{jl}}, \tag{3.2}$$

where i_{jl} are indices drawn from $\{1, \ldots, k\}$ (possibly with repetitions) and r_j is the number of terms used in the jth product. Note that a single basis function can appear multiple times in the same product, which allows representing higher order powers of basis functions. Also note that, in principle, both the number of products P and the number of terms r_j in each product is arbitrary.

Once we find such an approximation, we can transfer f by using the given functional map $\boldsymbol{C}$ and exploiting Eq. (3.1).

One difficulty with this approach is that if the functional map is approximate and does not satisfy Eq. (3.1) exactly, then terms involving function products can amplify the noise present in the map. In order to avoid this effect, we should look for a way to make as few products as possible to approximate f. However, the following theorem (proved in the next subsection) shows that this problem is NP-hard:

Theorem 3.4.1 *We consider the following problem (APPROX):*
INPUT: A positive integer $n_{\mathcal{M}}$, K basis functions $\varphi_1, \ldots, \varphi_K : \{1, \ldots, n_{\mathcal{M}}\} \to \mathbb{R}$, a "target" function $f : \{1, \ldots, n_{\mathcal{M}}\} \to \mathbb{R}$, and $\epsilon > 0$
OUTPUT: Minimum cost $c \in \mathbb{N}$, $g_{1,1}, \ldots, g_{1,r_1}, \ldots, g_{P,1}, \ldots, g_{P,r_P} \in \{1, \ldots, k\}$, $\alpha_1, \ldots, \alpha_P \in \mathbb{R}$ such that:

- $\|f - \sum_{i=1}^{P} \alpha_i \cdot G_i\|_\infty < \epsilon$, *where* $G_i = \prod_{j=1}^{r_i} \varphi_{g_{i,j}}$
- $\sum_{i=1}^{P} (r_i - 1) \leq c$

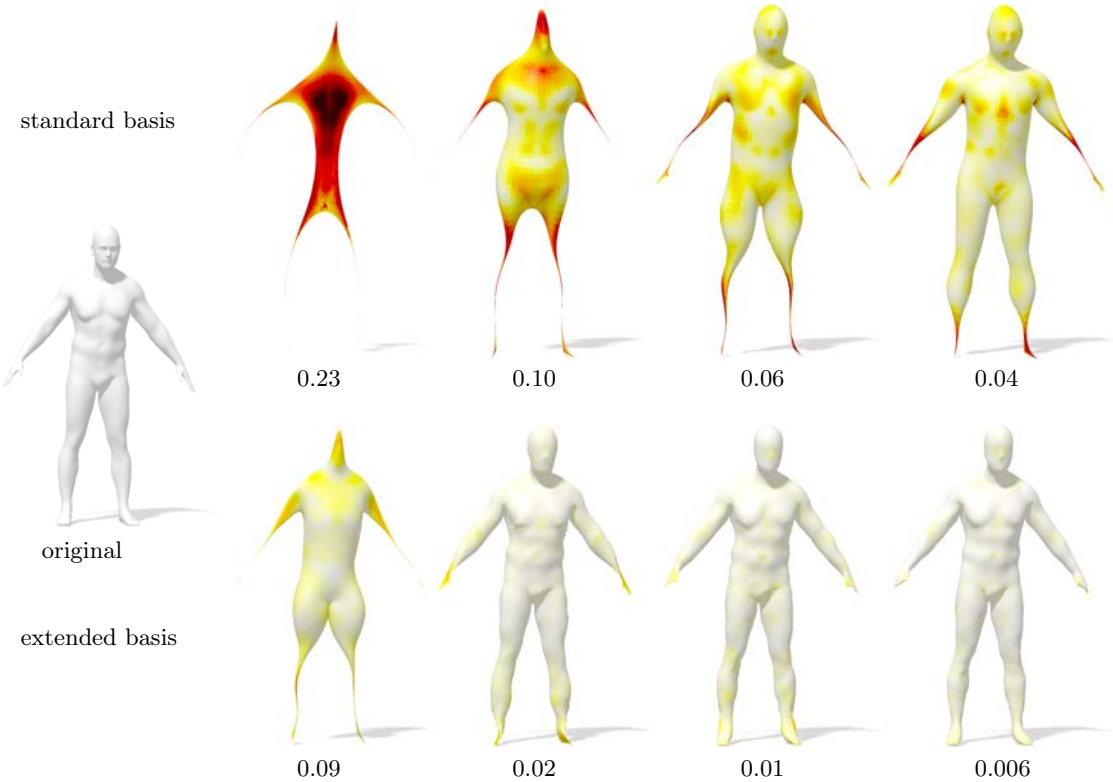


Figure 3.4 – Surface reconstruction via approximation of the 3-coordinates functions. On the left the original shape. On the right, for different values of k $(9, 29, 49, 69)$, some surface reconstruction results. On the top using the first $k+1$ Laplacian eigenfunctions, bottom adding also pairwise products. Under each shape is reported the error. The colormap encodes reconstruction error, decreasing from dark red to white.

(APPROX) is NP-hard.

Proof See Appendix Section 3.7.3 of this chapter. □

Note that c corresponds to the cost that we would like to minimize, since it represents the number of pointwise products used in the approximation. Intuitively, this theorem is related to the optimal sparse (L_0 norm) function approximation, which is also known to be NP-hard [65]. However, due to the special structure of our problem, which involves pointwise products of basis functions, our proof is independent and uses a reduction of 3-SAT directly.

3.4.2 Extended Functional Basis

Due to Theorem 3.4.1 and also because we have observed that using high order products usually brings little improvement in practice, we focus on only using point-wise products

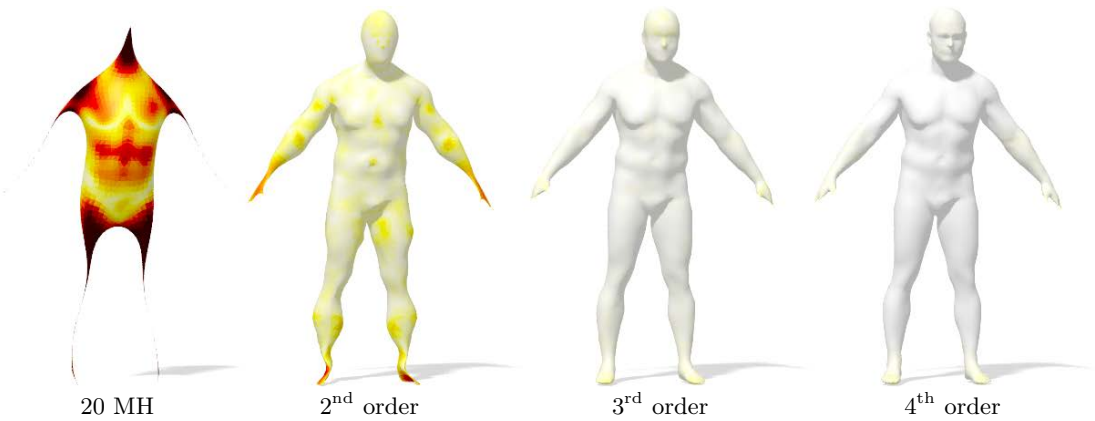


Figure 3.5 – Surface reconstruction via approximation of the 3-coordinates functions using different orders of products. From left to right using 20 eigenfunctions, their second order products, third order and finally fourth order products.

between *pairs* of basis functions $\varphi_i^{\mathcal{M}} \odot \varphi_j^{\mathcal{M}}$, where potentially $i = j$. Therefore, we look for the best coefficients a_i and $b_{i,j}$ to approximate f:

$$f \approx \sum_i a_i \varphi_i^{\mathcal{M}} + \sum_{i,j} b_{ij} \varphi_i^{\mathcal{M}} \odot \varphi_j^{\mathcal{M}}. \tag{3.3}$$

For this, we define an *extended basis* on $\mathcal{M}$ as $\mathbf{B}^{\mathcal{M}} = (\mathbf{\Phi}^{\mathcal{M}}, \widetilde{\mathbf{\Phi}}^{\mathcal{M}})$, where each column of the matrix of $\widetilde{\mathbf{\Phi}}^{\mathcal{M}}$ is of the form $\widetilde{\varphi}_i = \varphi_l^{\mathcal{M}} \odot \varphi_m^{\mathcal{M}}$ for some unique pair $l, m \in \{1, \ldots, k_{\mathcal{M}}\}$. Note that we exclude products with the constant function from $\widetilde{\mathbf{\Phi}}^{\mathcal{M}}$, but include both ordered pairs (l, m) and (m, l). We include both ordered pairs only for simplicity of the computation and the formulas involved, although they carry the same information.

Given the extended basis, our goal then is to approximate a given function f by computing a vector of coefficients $\mathbf{a}$ that would minimize $\|\mathbf{B}^{\mathcal{M}}\mathbf{a} - \mathbf{f}\|$. Note, however, that the matrix $\mathbf{B}^{\mathcal{M}}$ is not full rank, and, therefore the best approximation of f may involve a large amplification of noise.

To handle this issue, we use two solutions:

- **Approach A**: compute the optimal coefficients $\mathbf{a}$ by solving the Lasso-type problem:

$$\arg\min_{\mathbf{a}} \|\mathbf{B}^{\mathcal{M}}\mathbf{a} - \mathbf{f}\|_2 + \epsilon \|\mathbf{a}\|_p, \tag{3.4}$$

  where we use $p = 1$ or $p = 2$, which respectively promote sparsity and penalize large coefficients, and ϵ is a small regularizer. To solve Equation 3.4 with $p = 1$ we use a toolbox for non-smooth convex optimization [75] which implements the FISTA method.

- **Approach B**: compute the singular value decomposition (SVD) of $\sqrt{\mathbf{A}^{\mathcal{M}}}\mathbf{B}^{\mathcal{M}}$ as $\sqrt{\mathbf{A}^{\mathcal{M}}}\mathbf{B}^{\mathcal{M}} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$, where $\mathbf{A}^{\mathcal{M}}$ is the area matrix associated with the triangle

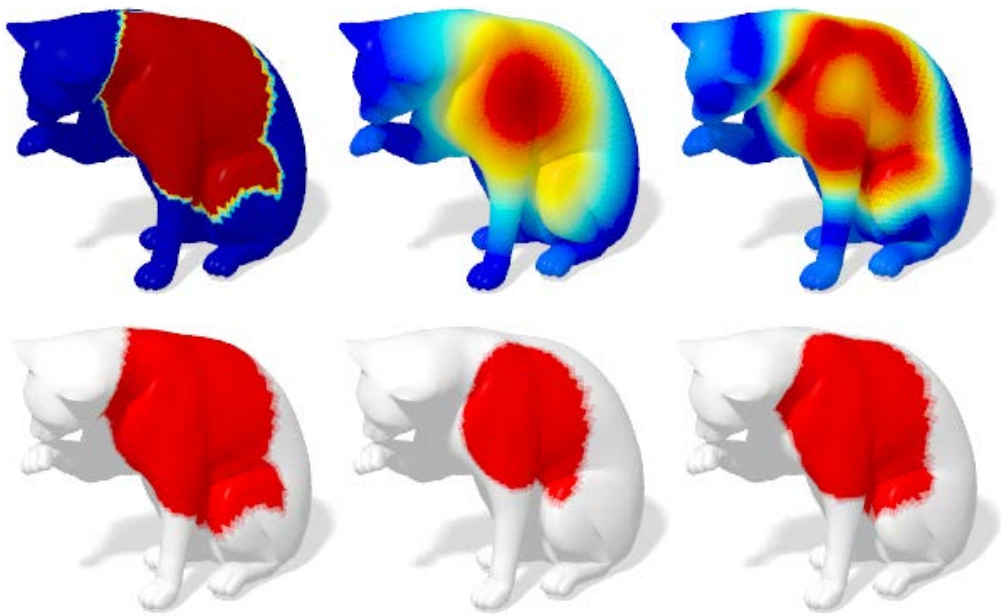


Figure 3.6 – Reconstruction of an indicator function of a region, using the standard and extended bases. On top we plot the original indicator function and its transfer using standard basis (middle) and extended basis (right). The second row shows the regions detected by thresholding the functions above using the same fixed value, the corresponding error shown below each plot.

mesh $\mathcal{M}$. We then set all singular values in $\mathbf{\Sigma}$ that are below 0.1% of the maximal value to zero and compute:

$$\mathbf{a} = \mathbf{V}\mathbf{\Sigma}^{\dagger}\mathbf{U}^{\top}\sqrt{\mathbf{A}^{\mathcal{M}}}\mathbf{f}, \tag{3.5}$$

Note that this approach is nearly identical to using the pseudo-inverse of $\mathbf{B}^{\mathcal{M}}$ to compute $\mathbf{a} = (\mathbf{B}^{\mathcal{M}})^{\dagger}\mathbf{f}$ with the thresholding parameter $0.001\sigma_{\max}$. However we use the area matrix to properly scale the basis with respect to the area elements on the mesh.

The first approach has the advantage of continuously depending on ϵ, while in the second case computing the optimal coefficients can be done effectively once the matrix $\mathbf{V}\mathbf{\Sigma}^{\dagger}\mathbf{U}^{\top}$ is pre-computed, which allows us to use it to transfer many functions. For these reasons, unless specified, we adopt the **Approach B** below.

We show an example of the reconstruction quality of the coordinate functions using the standard and extended basis in Figure 3.4 and the same for an indicator function of a region in Figure 3.6. Note that the use of the extended basis allows to capture higher frequency details even for the same size of the original basis. Therefore, intuitively, we would expect that it would allow more accurate function transfer across shapes.

3.4.3 Extended Function Transfer

Once the optimal coefficients $\mathbf{a}$ of the function $\mathbf{f}$ in the extended basis $\mathbf{B}^{\mathcal{M}}$ are computed, we use the given functional map $\boldsymbol{C}$ to transfer $\mathbf{f}$ onto shape $\mathcal{N}$. For this, we first construct an extended basis $\mathbf{B}^{\mathcal{N}}$ using the same procedure as described in Section 3.4.2 above. We then construct an extended transfer matrix:

$$\widetilde{\boldsymbol{C}}(\boldsymbol{C}) = \begin{bmatrix} \boldsymbol{C} & \mathcal{R}(\boldsymbol{C}) \\ 0 & \boldsymbol{C}(1:k,1:k) \otimes \boldsymbol{C}(1:k,1:k) \end{bmatrix}$$

with

$$\mathcal{R}(\boldsymbol{C}) = \begin{bmatrix} \phi_0 \boldsymbol{C}(0,1:k) \otimes \boldsymbol{C}(0:k,1:k) \\ +\phi_0 \begin{bmatrix} 0 \dots 0 \\ \boldsymbol{C}(1:k,1:k) \end{bmatrix} \otimes \boldsymbol{C}(0,1:k) \end{bmatrix}$$

where $\otimes$ is the Kronecker product of matrices, $\boldsymbol{C}(0,1:k)$ means that we consider the row of index 0 and the columns of indices going from 1 to k, and $\phi_0 = \Phi_{\mathcal{N}}(0,0)$ is the constant value taken by the constant eigenfunction on any point.

The following result (proved in the Appendix Section 3.7.2) shows that $\widetilde{\boldsymbol{C}}$ allows us to transfer functions expressed in the extended basis

Lemma 3.4.2 *The image of a function* $\mathbf{f} = \mathbf{B}^{\mathcal{M}}\mathbf{a}$ *on shape* $\mathcal{N}$ *is given by* $\mathbf{B}^{\mathcal{N}}\widetilde{\boldsymbol{C}}\mathbf{a}$

This lemma shows that transferring functions in the extended basis can simply be done by matrix-vector multiplication, as is the case in the standard basis. Note that the transfer *is not linear* in the original functional map $\boldsymbol{C}$, since the construction of $\widetilde{\boldsymbol{C}}$ involves Kronecker products. This implies that it is not straightforward to include the extended basis in the pipeline for *computing* functional maps, since this would involve terms with products of the unknown map, which could significantly increase the optimization complexity. Nevertheless, once the map $\boldsymbol{C}$ is computed, using it for function transfer with the extended basis can be done in closed form.

3.4.4 Function Comparison and Pointwise Map Recovery

We note that in some applications (e.g., converting a functional map to a point-to-point one) it is important to compare functions by simply comparing their coefficients in a reduced basis. When considering the extended basis $\mathbf{B}_{\mathcal{N}}$ for example, this can lead to problems because the basis is not orthonormal, and furthermore not necessarily full-rank. This means, in particular, that a single function can be represented using multiple different coefficients in the extended basis. To alleviate this issue, similarly to the procedure described in Section 3.4.2 we compute the SVD of $\sqrt{\mathbf{A}^{\mathcal{N}}}\mathbf{B}^{\mathcal{N}} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^\top$, and for every function $\mathbf{f}$ with coefficients $\mathbf{a}$ in the extended basis, such that $\mathbf{f} = \mathbf{B}^{\mathcal{N}}\mathbf{a}$ we define its *canonical* coefficients using $\tilde{\mathbf{a}} = \boldsymbol{\Sigma}\mathbf{V}^\top\mathbf{a}$. Note that in this case

$$\|\mathbf{f}\|^2_{\mathbf{A}} = \mathbf{f}^\top\mathbf{A}^{\mathcal{N}}\mathbf{f} = \mathbf{a}^\top(\mathbf{B}^{\mathcal{N}})^\top\mathbf{A}^{\mathcal{N}}\mathbf{B}^{\mathcal{N}}\mathbf{a} = \mathbf{a}^\top\mathbf{V}\boldsymbol{\Sigma}^2\mathbf{V}^\top\mathbf{a}^\top = \tilde{\mathbf{a}}^\top\tilde{\mathbf{a}},$$

where the second and third equations hold because $\mathbf{f} = \mathbf{B}^{\mathcal{N}}\mathbf{a}$ by assumption and because the area matrix and the diagonal matrix $\mathbf{\Sigma}$ are symmetric, such that $(\mathbf{B}^{\mathcal{N}})^\top \sqrt{\mathbf{A}^{\mathcal{N}}}^\top \sqrt{\mathbf{A}^{\mathcal{N}}}\mathbf{B}^{\mathcal{N}} = \mathbf{V}\mathbf{\Sigma}^2\mathbf{V}^\top$.

It follows that given two functions $\mathbf{f}_1, \mathbf{f}_2$ with coefficients $\mathbf{a}_1, \mathbf{a}_2$ in the extended basis, we can compute the norm of their difference as simply the L_2-norm of the difference $\|\tilde{\mathbf{a}}_1 - \tilde{\mathbf{a}}_2\|_2$.

Pointwise map recovery With this procedure in place, we can exploit Eq. (3.1) and the extended basis for recovering a point-to-point map from a given functional map $\boldsymbol{C}$. Note that the input functional map is always given in the original basis, and we only use extended basis for more accurate function transfer. For converting $\boldsymbol{C}$ to a pointwise map, we first compute the canonical coefficients of Dirac δ functions at each point y on shape $\mathcal{N}$ in the extended basis. We then compute the image of each δ function of points on shape $\mathcal{M}$ and compute its canonical coefficients. Finally, following the procedure described in [69] we construct a point-to-point map by looking at the nearest neighbors in the coefficient space. This procedure has the advantage of being efficient, since functions are represented in the space of dimension at most $\tilde{k}_{\mathcal{N}} = k_{\mathcal{N}} + \frac{k_{\mathcal{N}}^2 + k_{\mathcal{N}}}{2}$, where the second terms represents the number of distinct pairwise products of basis functions. For small values of $k_{\mathcal{N}}$ the value $\tilde{k}_{\mathcal{N}}$ is still significantly lower than the number of vertices. In practice we use (squared) heat kernel instead of Dirac δ functions for representing points, as described in Section 3.5.3 below.

3.5 Results

In the following experiments we use two standard datasets: FAUST [13] an TOSCA [19]. The former consists of 100 shapes, with 10 subjects in 10 poses, represented as triangle meshes with the same connectivity and with ground truth point-wise correspondences. TOSCA high resolution dataset contains synthetic models in 7 different shape classes with ground truth correspondences given in each class.

3.5.1 Function approximation and transfer

In our first application, we evaluate the utility of our function transfer procedure using both ground truth (arising from known pointwise maps) and computed functional maps. Namely, given a set of pairs of (source, target) shapes and a collection of different functions on each source, we evaluate: i) the approximation of each function on the source shape, and ii) the transfer of a function between the source and the target shape. Figure 3.7 shows a qualitative example of approximation and transfer of a real-valued function, with the original function shown on the left. This function is generated as a combination of an indicator function (on the left leg) a gaussian around a point (top of the right leg) a sine function of the y coordinate (on the tail) and a continuously increasing function (on the ears). We compare the standard approach (std) vs. our extended method (prod), for both function approximation and transfer onto a different pose of the same shape. Note

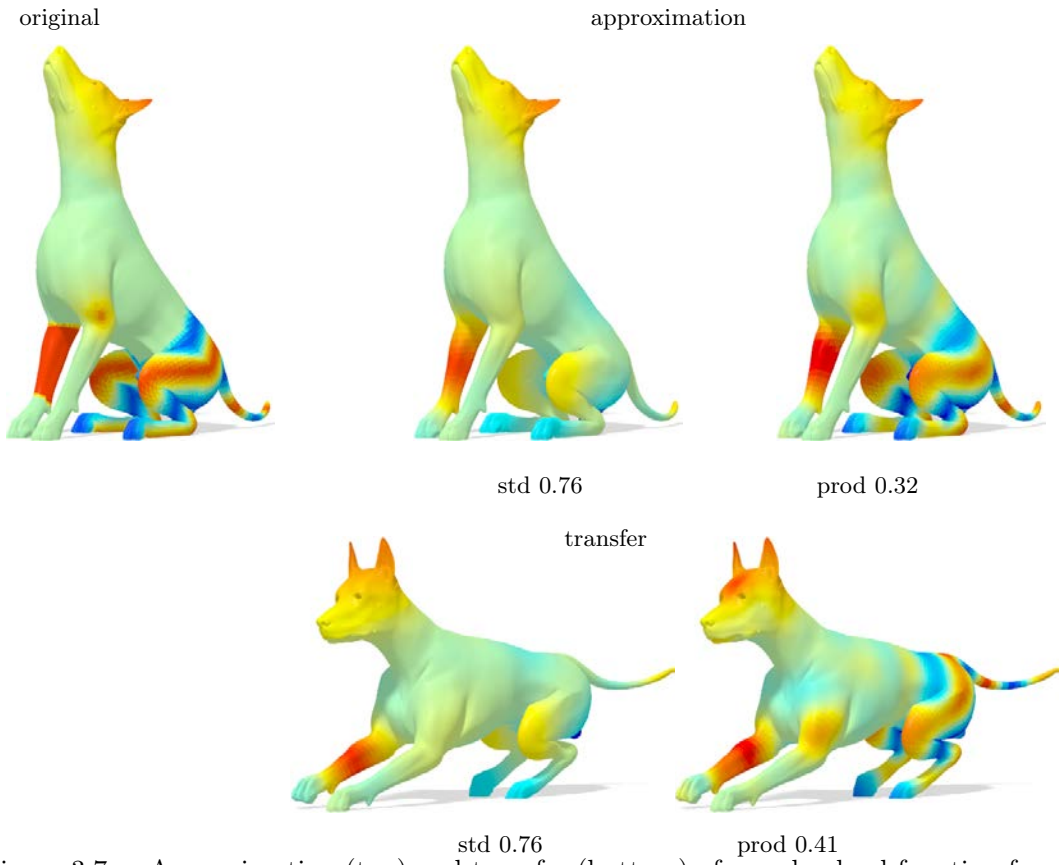


Figure 3.7 – Approximation (top) and transfer (bottom) of a real-valued function from a source shape onto a target shape using a fixed functional map with the standard approach (center) vs. our extended method (right).

the improvement with our approach (Approach B) as captured by the mean squared error reported under each plot.

In our quantitative experiments we consider the following families of functions:

- hk k, hk K: the heat kernel functions $h_t(x, \cdot)$ between a random point x and the rest of the shape approximated using $k_{\mathcal{M}} + 1$ eigenfunctions (for hk k) and using 200 eigenfunctions (for hk K). Note that in the former case, the function is contained in the span of the original basis, while in the latter $200 > k_{\mathcal{M}}$.
- HKS, WKS: the Heat and Wave Kernel Signatures [8, 92] for 10 randomly time and energy values respectively.
- Random: the function obtained as a linear combination of the extended basis using a random set of coefficients.
- XYZ: the X, Y, Z, coordinates of vertices.
- Indicator: the binary indicator function of a random region.

ground truth	approx		transfer			
function	**std**	**our l1**	**std**	**our l1**	**our l2**	**our B**
hk k	**0.00**	0.01	0.06	0.06	0.06	**0.05**
hk K	0.91	**0.86**	0.91	0.87	0.92	**0.86**
HKS	0.54	**0.00**	0.54	**0.15**	**0.15**	**0.15**
WKS	0.21	**0.00**	0.21	**0.04**	**0.04**	**0.04**
Random	0.38	**0.03**	0.38	0.07	**0.06**	**0.06**
XYZ	0.29	**0.21**	0.29	**0.23**	0.27	**0.23**
Indicator	0.36	**0.28**	0.37	**0.28**	0.32	**0.28**
SHOT	0.84	**0.82**	0.84	**0.82**	0.84	**0.82**
AWFT	0.31	**0.26**	0.31	**0.27**	0.30	**0.27**

Table 3.1 – Approximation and transfer quality results using a ground truth functional map with $k_{\mathcal{M}} = k_{\mathcal{N}} = 9$ of various functions on 20 shape pairs from the FAUST dataset. Note that for all functions except hk k, which lies in the span of the original basis, our approach produces a significant improvement.

- SHOT, AWFT: the SHOT [97] and AWFT [59] descriptors for 10 randomly chosen dimensions.

We compare the function approximation and transfer using the standard method (**std**) with the two approaches described in Section 3.4.2. For Approach A we have two different regularizations using the L_1-norm and the L_2-norm (**our l1** and **our l2**). We denote our Approach B with **our B**. For the L_1 regularization we used the *Sparse Optimization Toolbox* for Matlab [75].

Table 3.1 shows the results for function approximation and transfer using ground truth functional maps on a set of shape pairs from the FAUST dataset [13]. Here and below, the errors are computed as the normalized integral of the differrence between the function f and the ground truth g as $err = \sqrt{\int_{\mathcal{M}}(f-g)^2 dx} \Big/ \sqrt{\int_{\mathcal{M}}(g)^2 dx}$, where, dx is the area element induced by the metric. Note that all of our approaches result in a significant improvement for approximation and transfer. Note also that although the L_1 regularization produces good results, it performs similarly to other approaches. At the same time, as it is significantly more time-consuming, and requires solving a convex optimization problem for every function to be transferred, we omit it from further evaluation and rely on the L_2 regularization for Approach A.

In Table 3.2 we show a similar evaluation but on a *computed* functional map, using a recent approach of [68], which estimates a functional map by exploiting a set of descriptors on each shape, where we use the WKS for some energy values. Here again, our approaches show a significant improvement with respect to the baseline standard method for all functions. Throughout our experiments we observed that when increasing the basis size $k_{\mathcal{M}}$ on the source, it is often beneficial to have $k_{\mathcal{N}} > k_{\mathcal{M}}$ since this allows to better represent the transfer of basis functions.

computed	**approx**			**transfer**		
function	**std**	**our A**	**our B**	**std**	**our A**	**our B**
hk k	**0.00**	0.03	0.01	0.13	0.13	0.13
hk K	0.82	0.54	**0.49**	0.82	**0.57**	0.58
HKS	0.55	0.14	**0.00**	0.55	**0.22**	**0.22**
WKS	0.14	0.03	**0.00**	0.14	0.07	**0.06**
Random	0.48	0.04	**0.01**	0.49	**0.15**	**0.15**
XYZ	0.12	0.09	**0.06**	0.13	**0.11**	**0.11**
Indicator	0.28	0.19	**0.17**	0.28	0.20	**0.19**
SHOT	0.81	0.76	**0.74**	0.81	**0.77**	0.78
AWFT	0.24	0.19	**0.17**	0.25	**0.20**	**0.20**

Table 3.2 – Approximation and transfer quality using a *computed* functional map with $k_{\mathcal{M}} = 29, k_{\mathcal{N}} = 39$ using the method from [68] of various functions on 20 shape pairs from the FAUST dataset. Note in particular that our transfer approach produces a significant improvement.

computed	$k_{\mathcal{M}} = 6,\ k_{\mathcal{N}} = 7$		$k_{\mathcal{M}} = 27,\ k_{\mathcal{N}} = 35$	
function	**std**	**our B**	**std**	**our B**
hk k	0.77	**0.60**	0.08	**0.06**
hk K	0.92	**0.87**	0.79	**0.58**
HKS	0.66	**0.61**	0.55	**0.17**
WKS	0.47	**0.25**	0.17	**0.04**
Random	0.75	**0.69**	0.46	**0.09**
Coordinates	0.34	**0.24**	0.12	**0.10**
Indicator	0.58	**0.34**	0.28	**0.19**
SHOT	0.90	**0.88**	0.84	**0.81**
AWFT	0.35	**0.29**	0.25	**0.21**

Table 3.3 – Transfer quality comparison using a ground truth functional map with $k_{\mathcal{M}} = 6$ and $k_{\mathcal{N}} = 7$ and $k_{\mathcal{M}} = 27$ and $k_{\mathcal{N}} = 35$, average on 20 pairs from FAUST dataset. Note in particular that our transfer approach produces a significant improvement as the number of basis functions used increases.

Table 3.3 shows how the transfer using standard and our approach benefit from the increase in the basis size. In the first two columns we use a ground truth functional map represented by a matrix $\boldsymbol{C}$ with $k_{\mathcal{M}} = 6$ and $k_{\mathcal{N}} = 7$. In the last two columns we use $k_{\mathcal{M}} = 27$ and $k_{\mathcal{N}} = 35$ non-constant basis functions. These dimensions are chosen since e.g., starting with k basis functions, the number of different functions obtained by adding the pairwise products is at most: $k + \frac{k(k+1)}{2}$. The transferred functions that depend on the basis dimension (e.g., the HKS) are computed using the larger value

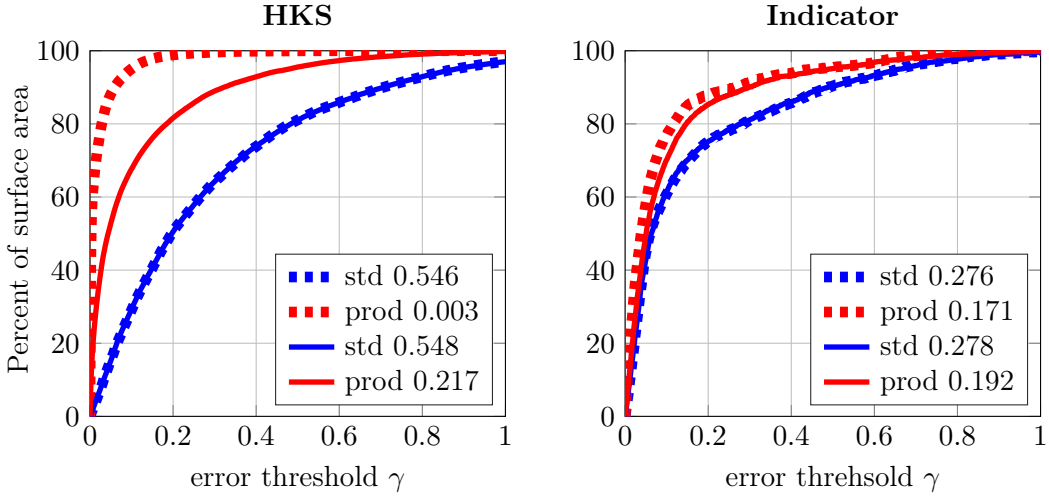


Figure 3.8 – Plots of the percentage of the surface area for which the function approximation and transfer is below some threshold for the HKS (left) and the indicator function of a region (right), using computed functional maps on 20 pairs of shapes from the FAUST dataset. Dashed and solid lines represent function approximation and transfer respectively.

of $k_{\mathcal{M}}$ in all cases. As can be seen in Table 3.3, both the standard and our approach benefit from the increase in the basis size, while our approach is always better for a given basis size. Observe that the results obtained with the standard approach using more basis function are close to those obtained with our approach and fewer basis functions. This is remarkable, given the optimality of the Laplacian eigenfunctions for representing functions with bounded variation, as shown by Aflalo et al. [1]. Furthermore increasing the number of basis functions in the standard approach requires solving an eigen-problem and the estimation of new coefficients in the functional map. Conversely our approach improves the representation and the transfer of functions without any new estimation of coefficients. Please see the Appendix for results using functional maps of various sizes. We also compared our method with first converting the functional map to a point-to-point one and using the latter for function transfer. This approach produces even more error than the standard method, as it introduces an additional source of noise.

In Figure 3.8 we show a different representation of the results from Table 3.2 for the HKS and the indicator functions. Namely, we plot the fraction of the surface area for which the difference between the approximated and the ground truth functions is below a threshold $\gamma \in [0, 1]$. I.e., the x-axis represents the error threshold γ, whereas the y axis shows the surface area of all points x such that $(f(x) - g(x))^2 < \gamma\sqrt{\int g(x)^2 dx}$. Figure 3.9 shows some results on the stability of transfer between different meshes. Starting from an original mesh and a function (left), we compare the transfer using the standard basis and our extended basis. The two meshes on which we compare the transfer have a different connectivity: a mesh with non uniform resampling (fewer vertices on the bottom

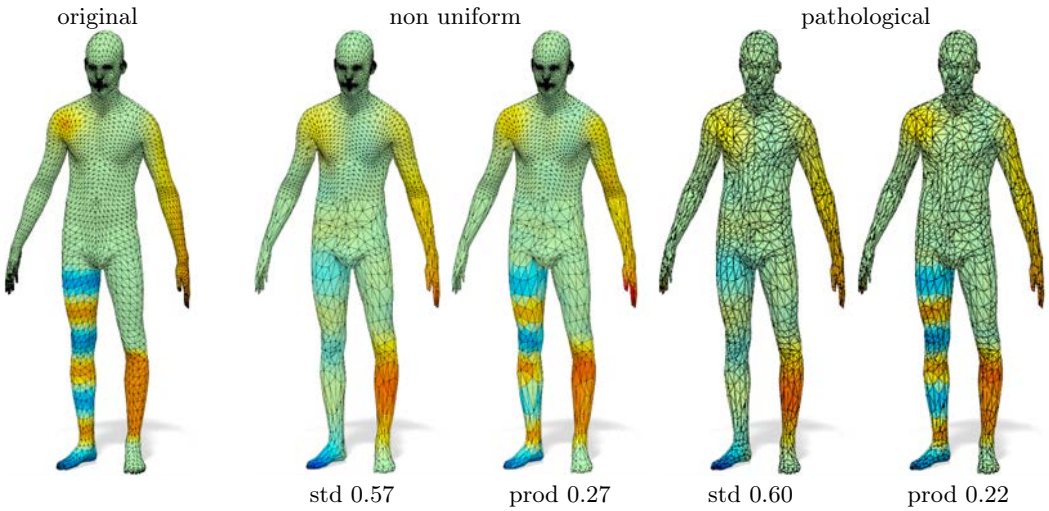


Figure 3.9 – An example of function transfer from a source shape with a regular mesh (left), to two different meshes with different triangulations. From left to right: a non uniform resampling (fewer vertices on the legs), a mesh with pathological triangles (around the 60% are very thin triangles).

of the shape), a mesh with around 60% of pathological triangles (very thin triangles). As can be seen, our approach gives better results and is stable with respect to changes in mesh connectivity.

Approximation in other bases As mentioned above, our extended basis construction is not tied to the Laplacian eigenfunctions. To illustrate this, in Figure 3.10 we show how the pointwise products improve the functional space representation in a different basis. Namely, we perform the same test that is in Figure 3.4, but using different basis manifold harmonics (MH) and the recently proposed localized manifold harmonics (LMH) [60]. As can be seen, for both of these bases pairwise products allow a more accurate representation of the surface.

3.5.2 HKS and WKS approximation and transfer

One interesting observation related to our method is that both the Heat Kernel Signature and the Wave Kernel Signature functions are of the form $h_t(x,x) = \sum_{i=1}^{k_{\mathcal{M}}} \alpha_i \phi_i^2(x)$, for some scalar coefficients α_i. In other words, they are constructed explicitly using squares of eigenfunctions and therefore it must be possible to represent and transfer them *exactly* in our extended basis. Therefore, they provide a good test for the correctness of both our extended function approximation and transfer methods. To demonstrate the difference between the transfer of these functions using the standard and the extended basis we show in Figure 3.11 the transfer error using the ground truth functional map of size $k_{\mathcal{N}} \times k_{\mathcal{M}}$ for increasing $k_{\mathcal{N}}$. Note that for large values of $k_{\mathcal{N}}$ we can approximate the transfer of all $k_{\mathcal{M}}$ basis functions from the source well, which means that we would expect our extended transfer method to produce progressively better results. This can

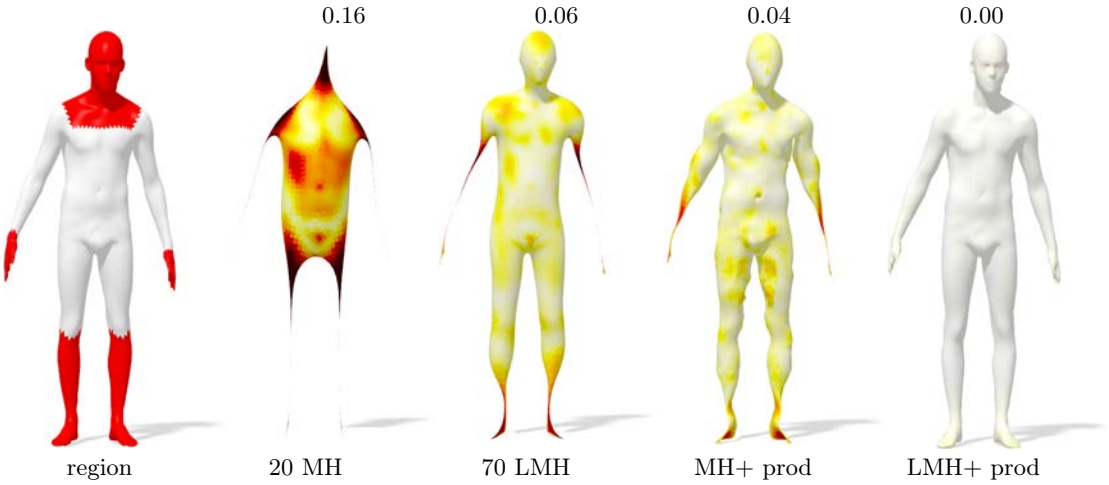


Figure 3.10 – Surface reconstruction via approximation of the 3-coordinates functions using two different bases: the standard manifold harmonics (MH) and localized manifold harmonics (LMH). On the left we show the original shape and the region (in red) in which are localized the LMH. On the right of each shape we report the reconstruction error. The colormap encodes reconstruction error, decreasing from dark red to white.

clearly be seen in Figure 3.11 where our method converges to a very small value while the standard technique does not improve. This is because the functions ϕ_i^2 cannot be well-approximated in the basis of the source shape, which means that regardless of the basis size on the target, we cannot achieve small error.

3.5.3 Point-to-point map recovery

We also demonstrate the utility of our approach for *converting* functional maps to point-to-point ones. As mentioned in Section 3.4.4, our extended basis can also be used to embed points in a coefficient space where function comparison can be done directly, which can be useful for point-to-point map recovery. In particular, we represent each point on the shape as the square of the heat kernel for a small time scale centered at that point. We then convert a functional map to a point-to-point staring with a ground truth map and also a computed one using the method of [68] on a set of pairs from the FAUST [13] TOSCA high resolution datasets [19]. We plot the conversion results in Figures 3.12 and 3.13 using the ground truth and estimated functional maps respectively, both of size 40×30. Note that as explained in Section 3.5.1 it is beneficial to rectangular matrices $\boldsymbol{C}$ in order to take full advantage of the use of the products. As can be seen in Figures 3.12 and 3.13 the extended basis and our improved function transfer also contribute to better point-to-point map recovery.

3.5.4 Joint quadrangulation

We also used our approach in the context of joint quadrangulation of triangle meshes by adapting the recent technique presented in [10]. This method is based on using functional

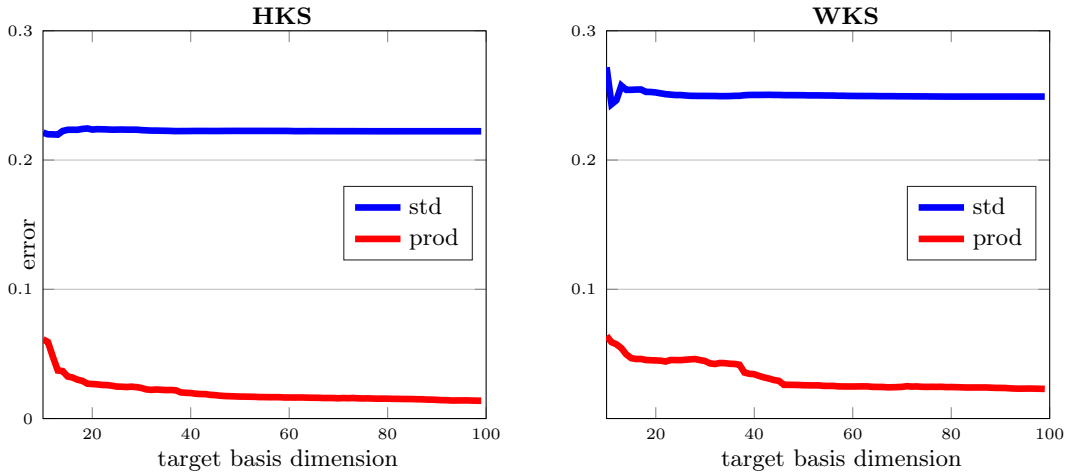


Figure 3.11 – Comparison on the transfer for HKS and WKS, average on 5 pairs from FAUST dataset [13]. Starting with $k+1=10$ basis functions on the source shape we compute HKS and WKS for 1 scale, and their coefficients in the $k+1$ fixed basis and its products extension. Varying the basis dimension on the target shape from 10 to 100, we define the $\boldsymbol{C}$ and the $\widetilde{\boldsymbol{C}}$ and compute the tranfer of HKS and WKS on the target shape. Here we plot the transfer error (y-axis) for HKS (left) and WKS (right) varying the basis dimension on the target shape (x-axis).As can be seen the error with products is smaller and decreases with increasing basis dimension while without products error stops decreasing.

maps for constructing consistent cross fields on a pair of surfaces, which are in turn used for designing approximately consistent quad-meshes. Moreover, the pipeline presented in [10] only relies on the ability of functional maps to transfer real-valued functions and does not require point-wise correspondences. We adapted this method to take advantage of the extended basis by simply enabling the transfer of pointwise products of basis functions, without any other modification, using the code provided by the authors. Figure 3.14 presents a result on a pair of shapes using the standard Laplacian basis with $k=5$ eigenfunctions and using our extended basis. Note that the presence of pointwise products in the extended basis allows us to transfer higher frequency information which leads to a more consistent result overall. In this application we used a functional map arising from a ground truth pointwise correspondence. Although preliminary, this result suggests the utility of our extended function transfer for joint quadrangulation, and we leave a more in-depth exploration of this application as future work.

3.6 Conclusion, Limitations and Future Work

In this chapter, we presented a novel method for function transfer with functional maps, by exploiting the algebraic structure of function spaces. We showed that by extending the functional basis to include *pointwise products* of basis functions, we can significantly

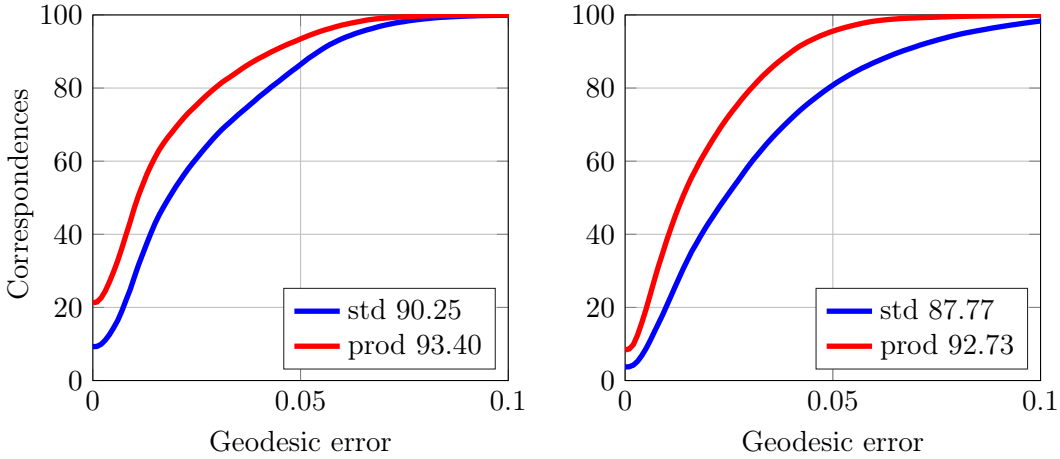


Figure 3.12 – Quality of correspondences obtained from a given ground truth functional map of dimension 40×30, between all the possible pairs of shapes for one subject from FAUST dataset (left), and for all the pairs of shapes for the centaur from TOSCA high resolution dataset (right).

improve both the reconstruction and the transfer quality of functions, while maintaining the computational complexity of the original functional map. Our approach has direct consequences on all applications of functional maps, and in particular shows how high-frequency information can be transferred even in the presence of only low frequency basis functions.

Our main limitation is that in the current formulation we only use products of pairs of basis functions. Although we have observed that higher-order products do not often bring significant improvement, a more in-depth analysis of the scenarios in which they can be useful is necessary. In the future, we also plan to work on more scalable methods for function approximation, by removing the need to explicitly compute the entire extended basis, which can be prohibitive for large bases.

3.7 Appendix

3.7.1 Additional Results

Tables 3.4, 3.5 and 3.6 show approximation and transfer quality of different functions using ground truth and computed functional maps of different sizes.

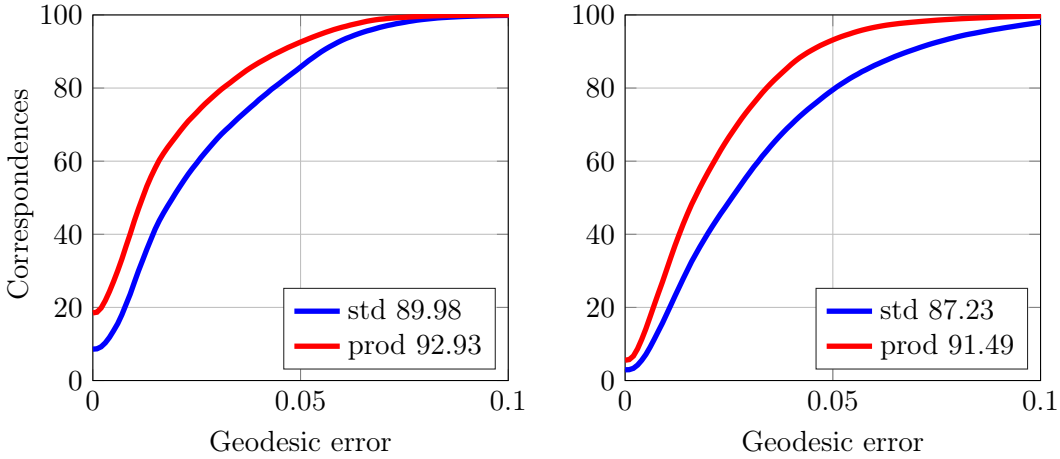


Figure 3.13 – Quality of correspondences obtained from a *computed functional map* of size 40×30, between all the possible pairs of shapes for one subject from FAUST dataset (left), and all pairs of shapes for the centaur from TOSCA high resolution dataset (right).

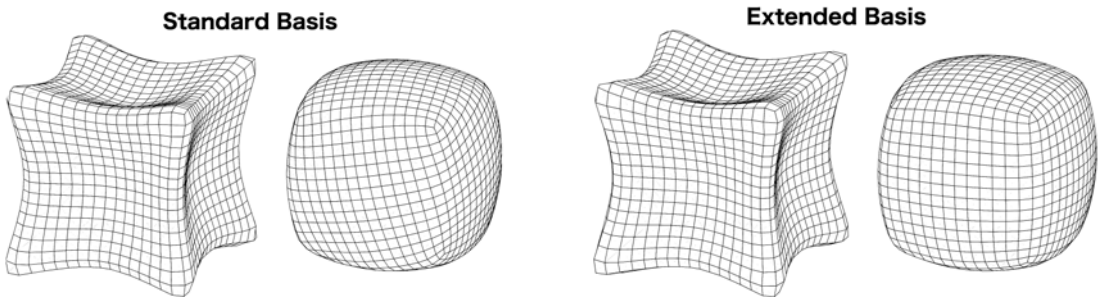


Figure 3.14 – Joint quadrangulation of two triangle meshes using the approach of [10] with the standard Laplacian $k = 5$ eigenfunctions (left) and our extended (right) basis. Note that the presence of products of basis functions allows us to transport higher frequency information, which leads to a more consistent result.

3.7.2 Proof of Lemma 2:

The image of a function $f = B_{\mathcal{M}}\mathbf{f}$ on shape $\mathcal{N}$ is given by $B_{\mathcal{N}}\widetilde{\boldsymbol{C}}\mathbf{f}$. where

$$\widetilde{\boldsymbol{C}} = \begin{bmatrix} \boldsymbol{C} & \begin{matrix} \phi_0 \boldsymbol{C}(0, 1:k) \otimes \boldsymbol{C}(0:k, 1:k) \\ +\phi_0 \begin{bmatrix} 0 \dots 0 \\ \boldsymbol{C}(1:k, 1:k) \end{bmatrix} \otimes \boldsymbol{C}(0, 1:k) \end{matrix} \\ 0 & \boldsymbol{C}(1:k, 1:k) \otimes \boldsymbol{C}(1:k, 1:k) \end{bmatrix} \tag{3.6}$$

Proof Let $K_{\mathcal{M}} = 1 + k_{\mathcal{M}} + {k_{\mathcal{M}}}^2$ be the number of columns in $B_{\mathcal{M}}$. By linearity, it is

computed	approx			transfer		
function	**std**	**our A**	**our B**	**std**	**our A**	**our B**
hk k	**0.00**	0.01	0.01	0.05	0.05	0.05
hk K	0.88	**0.75**	0.78	0.88	0.82	**0.79**
HKS	0.54	**0.00**	**0.00**	0.54	**0.11**	**0.11**
WKS	0.22	**0.00**	0.01	0.22	**0.04**	**0.04**
Random	0.38	**0.00**	**0.00**	0.38	**0.05**	**0.05**
XYZ	0.28	**0.18**	0.20	0.28	0.25	**0.23**
Indicator	0.40	**0.28**	0.29	0.41	0.33	**0.31**
SHOT	0.84	**0.80**	0.81	0.84	0.83	**0.82**
AWFT	0.31	**0.24**	0.25	0.32	0.29	**0.27**

Table 3.4 – $k_{\mathcal{M}} = 9$ and $k_{\mathcal{N}} = 9$, average on 20 pairs from FAUST dataset. Computed functional map

ground truth	approx			transfer		
function	**std**	**our A**	**our B**	**std**	**our A**	**our B**
hk k	**0.00**	0.03	0.01	0.26	**0.20**	**0.20**
hk K	0.79	0.52	**0.45**	0.80	**0.57**	0.75
HKS	0.57	0.14	**0.00**	0.57	**0.43**	0.54
WKS	0.12	0.03	**0.00**	0.12	0.07	**0.05**
Random	0.47	0.04	**0.01**	0.48	**0.17**	0.18
XYZ	0.12	0.09	**0.06**	**0.12**	0.13	0.16
Indicator	0.27	0.18	**0.17**	0.27	**0.20**	0.25
SHOT	0.82	0.76	**0.75**	0.82	**0.78**	0.88
AWFT	0.24	0.19	**0.17**	0.25	**0.22**	0.27

Table 3.5 – $k_{\mathcal{M}} = 29$ and $k_{\mathcal{N}} = 29$, average on 20 pairs from FAUST dataset. Given a ground truth functional map

sufficient to prove the statement for each $e_i^{(K_{\mathcal{M}})} = (0, \dots, 0, 1, 0, \dots, 0) \in \mathbb{R}^{K_{\mathcal{M}}}$ (the 1 is at position i).

- if $0 \leq i \leq k_{\mathcal{M}}$: By definition of $\widetilde{\boldsymbol{C}}$, $B_{\mathcal{N}}\widetilde{\boldsymbol{C}}\mathbf{f} = \Phi_{\mathcal{N}}\boldsymbol{C}e_i^{(k_{\mathcal{M}}+1)}$ which is the image of $B_{\mathcal{M}}e_i^{(k_{\mathcal{M}}+1)} = f$.

- if $i > k_{\mathcal{M}}$: we write $i = k_{\mathcal{M}} + k_{\mathcal{M}}(i_1 - 1) + i_2$, which represents the pair of indices

ground truth	approx			transfer		
function	**std**	**our A**	**our B**	**std**	**our A**	**our B**
hk k	**0.00**	0.03	0.01	0.09	0.09	**0.08**
hk K	0.79	0.54	**0.49**	0.79	**0.57**	0.61
HKS	0.55	0.14	**0.00**	0.56	0.18	**0.14**
WKS	0.13	0.03	**0.00**	0.13	0.05	**0.04**
Random	0.46	0.04	**0.01**	0.46	**0.09**	**0.09**
XYZ	0.12	0.09	**0.06**	0.12	0.10	**0.09**
Indicator	0.28	0.19	**0.18**	0.28	0.20	**0.19**
SHOT	0.82	0.76	**0.75**	0.82	**0.77**	0.78
AWFT	0.24	0.19	**0.17**	0.25	**0.20**	**0.20**

Table 3.6 – $k_{\mathcal{M}} = 29$ and $k_{\mathcal{N}} = 39$, average on 20 pairs from FAUST dataset. Given a ground truth functional map

$1 \leq i_1, i_2 \leq k_{\mathcal{M}}$.

$$\begin{aligned}
B_{\mathcal{N}}\widetilde{\boldsymbol{C}}\mathbf{f} &= \Phi_{\mathcal{N}}(\phi_0(\boldsymbol{C}e_{i_1}^{(k_{\mathcal{M}}+1)})_0(\boldsymbol{C}e_{i_2}^{(k_{\mathcal{M}}+1)})) \\
&\quad + \phi_0(\boldsymbol{C}e_{i_2}^{(k_{\mathcal{M}}+1)})_0(\boldsymbol{C}e_{i_1}^{(k_{\mathcal{M}}+1)})_{1:k_{\mathcal{M}}}) \\
&\quad + (\Phi_{\mathcal{N}} \otimes \Phi_{\mathcal{N}})((\boldsymbol{C}e_{i_1}^{(k_{\mathcal{M}}+1)})_{1:k_{\mathcal{M}}} \otimes (\boldsymbol{C}e_{i_2}^{(k_{\mathcal{M}}+1)})_{1:k_{\mathcal{M}}}) \\
&= (\Phi_{\mathcal{N}}\boldsymbol{C}e_{i_2}^{(k_{\mathcal{M}}+1)}) \odot (\Phi_{\mathcal{N}}(\boldsymbol{C}e_{i_1}^{(k_{\mathcal{M}}+1)})_0) \\
&\quad + (\Phi_{\mathcal{N}}(\boldsymbol{C}e_{i_1}^{(k_{\mathcal{M}}+1)})_{1:k}) \odot (\Phi_{\mathcal{N}}(\boldsymbol{C}e_{i_2}^{(k_{\mathcal{M}}+1)})_0) \\
&\quad + (\Phi_{\mathcal{N}}(\boldsymbol{C}e_{i_1}^{(k_{\mathcal{M}}+1)})_{1:k}) \odot (\Phi_{\mathcal{N}}(\boldsymbol{C}e_{i_2}^{(k_{\mathcal{M}}+1)})_{1:k}) \\
&= (\Phi_{\mathcal{N}}\boldsymbol{C}e_{i_2}^{(k_{\mathcal{M}}+1)}) \odot (\Phi_{\mathcal{N}}(\boldsymbol{C}e_{i_1}^{(k_{\mathcal{M}}+1)})_0) \\
&\quad + (\Phi_{\mathcal{N}}(\boldsymbol{C}e_{i_1}^{(k_{\mathcal{M}}+1)})_{1:k}) \odot \big((\Phi_{\mathcal{N}}(\boldsymbol{C}e_{i_2}^{(k_{\mathcal{M}}+1)})_0) \\
&\qquad\qquad + (\Phi_{\mathcal{N}}(\boldsymbol{C}e_{i_2}^{(k_{\mathcal{M}}+1)})_{1:k})\big) \\
&= (\Phi_{\mathcal{N}}\boldsymbol{C}e_{i_2}^{(k_{\mathcal{M}}+1)}) \odot (\Phi_{\mathcal{N}}(\boldsymbol{C}e_{i_1}^{(k_{\mathcal{M}}+1)})_0) \\
&\quad + (\Phi_{\mathcal{N}}(\boldsymbol{C}e_{i_1}^{(k_{\mathcal{M}}+1)})_{1:k}) \odot (\Phi_{\mathcal{N}}\boldsymbol{C}e_{i_2}^{(k_{\mathcal{M}}+1)}) \\
&= (\Phi_{\mathcal{N}}\boldsymbol{C}e_{i_2}^{(k_{\mathcal{M}}+1)}) \odot (\Phi_{\mathcal{N}}\boldsymbol{C}e_{i_1}^{(k_{\mathcal{M}}+1)}),
\end{aligned}$$

which is the point-wise product of the images of $B_{\mathcal{M}}e_{i_1}^{k_{\mathcal{M}}+1}$ and $B_{\mathcal{M}}e_{i_2}^{(k_{\mathcal{M}}+1)}$, which is, by Eq. (3.1), the image of $B_{\mathcal{M}}e_{i_1}^{(k_{\mathcal{M}}+1)} \odot B_{\mathcal{M}}e_{i_2}^{(k_{\mathcal{M}}+1)} = f$.

□

3.7.3 Proof of Theorem 3.4.1

Note: To avoid heavy notations while proving Theorem 3.4.1, we will use specific notations for this subsection.

We consider the following problem:
(APPROX):
INPUT: $N \in \mathbb{N}$, K "basis" functions $\varphi_1, \ldots, \varphi_K : \{1, \ldots, N\} \to \mathbb{R}$, a "target" function $f : \{1, \ldots, N\} \to \mathbb{R}$, $\epsilon > 0$, a cost $c \in \mathbb{N}$
OUTPUT: $g_{1,1}, \ldots, g_{1,r_1}, \cdots, g_{P,1}, \ldots, g_{P,r_P} \in \{1, \ldots, k\}$, $\alpha_1, \ldots, \alpha_P \in \mathbb{R}$ such that:

- $\|f - \sum_{i=1}^{P} \alpha_i \cdot G_i\|_\infty < \epsilon$, where G_i is defined as $G_i = \prod_{j=1}^{r_i} \varphi_{g_{i,j}}$
- $\sum_{i=1}^{P}(r_i - 1) \leq c$ ($r_i - 1$ represents the number of multiplications that was necessary to perform in order to obtain G_i)

and the boolean **TRUE**, if such a construction exists.
The boolean answer **FALSE** if no such construction exists.

We want to show this problem $NP - hard$. For this, we will make a polynomial reduction from $3 - SAT$:
(3-SAT):
INPUT: $X_1, \ldots, X_Q$ boolean variables, $(T_{1,1} \vee T_{1,2} \vee T_{1,3}) \wedge \ldots \wedge (T_{M,1} \vee T_{M,2} \vee T_{M,3})$ a boolean formula, where each $T_{m,v}$ is some X_i or some $\bar{X}_i$ (negation of X_i).
OUTPUT: The boolean **TRUE**, a function $g : \{1, \ldots, Q\} \to \{FALSE, TRUE\}$ such that assigning each variable X_q to $g(q)$ satisfies the above boolean formula described by the $T_{m,j}$s.
The boolean **FALSE** if no assignment of X_q to boolean values can satisfy the above boolean formula.

Reduction from **(3-SAT)** to **(APPROX)**:
We assume that we are given an initial instance of **(3-SAT)** given with the above notations. We will construct an instance of **(APPROX)** whose solution will be proven convertible into a solution of the initial problem.

Summary of the proof:
STEP 1:
We will define $Q + M$ clusters of variables $n \in \{1, \ldots, N\}$ over which f and a cluster of basis functions f_k for some $k \in \{1, \ldots, K\}$ will take a non-zero value.
STEP 2:
We will prove that each of the $Q + M$ pairs of clusters (variables - functions) defined incurs a cost ≥ 1, therefore the total cost is $\geq Q + M$. In the reduction that we propose, the initial instance of **(3-SAT)** will have a solution if and only if the total cost of the corresponding **(APPROX)** that we built is equal to $Q + M$.
Therefore, we can deduce :
<u>Lemma</u>: any solution **(APPROX)** that we consider can be supposed to involve only 1 product for each pair of clusters.
STEP 3:
We show the first direction: if the initial instance of **(3-SAT)** has a solution then we have a solution to our constructed instance of **(APPROX)**.

$$1 \leq n \leq N = 7Q + 6M + M^2 \quad \longleftrightarrow \quad \begin{array}{ccccccc} (1,1,1) & \dots & (1,1,7) & \dots & (1,Q,1) & \dots & (1,Q,7) \\ (2,1,1) & \dots & (2,1,6) & \dots & (2,M,1) & \dots & (2,M,6) \\ (3,1,1) & \dots & (3,1,M) & \dots & (3,M,1) & \dots & (3,M,M) \end{array}$$

Figure 3.15 – Representation of the variable $1 \leq n \leq N$

$$1 \leq k \leq K = 3Q + M^2 + 3M \quad \longleftrightarrow \quad \begin{array}{ccccccc} (4,1,1) & (4,1,2) & (4,1,3) & \dots & (4,Q,1) & (4,Q,2) & (4,Q,3) \\ (5,1,1) & \dots & (5,1,M) & \dots & (5,M,1) & \dots & (5,M,M) \\ (6,1,1) & (6,1,2) & (6,1,3) & \dots & (6,M,1) & (4,Q,2) & (6,M,3) \end{array}$$

Figure 3.16 – Representation of the variable $1 \leq k \leq K$

The reader should then be able to guess from the structure of this solution how we can prove the other direction.
STEP 4:
Using the Lemma, we prove the other direction.

STEP 1:

We fix ϵ very small, for example $\epsilon = \frac{1}{3M+6}$ will work. For our reduction, we will independently define some basis functions over some $n \in \{1, \dots, N\}$, specifically designed for encoding the initial instance of **(3-SAT)**. Because we want to design these functions over 3 independent set of values taken by n, we will change the notation: we fix $N = 7Q + 6M + M^2$ but for convenience of notation, instead of using an index $1 \leq n \leq N$, we will use indices $(1, q, j)$ with $1 \leq q \leq Q$ and $1 \leq j \leq 7$, indices $(2, m, j)$ with $1 \leq m \leq M$ and $1 \leq j \leq 6$ and $(3, m, m')$ with $1 \leq m, m' \leq M$, as in Figure 3.15. Likewise, we fix $K = 3Q + M^2 + 3M$ but we will use indices $(4, q, j)$ for $1 \leq j \leq 3$, $(5, m, m')$ and $(6, m, j)$ for $1 \leq j \leq 3$, as in Figure 3.16.
Now we want to define the values taken by the functions f and $\varphi_.$ over $n \in \{1, \dots, N\}$ in such a way that:

- f, $\varphi_{(4,q,.)}$ are essentially the only functions that may take a non-zero value at $(1, q, .)$, as shown on Figure 3.17
- f, $\varphi_{(5,m,.)}$, $\varphi_{(6,m,.)}$ are (exactly) the only functions that may take a non-zero value at $(2, m, .)$, as shown on Figure 3.18

In each of the previous cases, we will define the values taken in such a way that, in order to approximate f, the only option will be to make a product of some kind.
Before explaining the behavior that our definitions will induce over a potential solution, let's define the values taken by each function so that the reader can refer to it (or refer to Figures 3.17, 3.18) to follow the analysis:

	(1,q,1)	(1,q,2)	(1,q,3)	(1,q,4)	(1,q,5)	(1,q,6)	(1,q,7)
f	1	0	0	0	0	0	0
$\varphi_{(4,q,1)}$	1	1	0	1	0	1	0
$\varphi_{(4,q,2)}$	1	0	1	-1	1	0	0
$\varphi_{(4,q,3)}$	1	0	1	0	0	-1	1

Figure 3.17 – functions defined over $(1, q, j)$

	(2,m,1)	(2,m,2)	(2,m,3)	(2,m,4)	(2,m,5)	(2,m,6)	(3,m,m')
f	1	0	0	0	0	0	0
$\varphi_{(5,m,m')}$	1	1	0	0	0	0	M
$\varphi_{(6,m,1)}$	1	0	1	1	0	0	0
$\varphi_{(6,m,2)}$	1	0	1	0	1	0	0
$\varphi_{(6,m,3)}$	1	0	1	0	0	1	0

Figure 3.18 – functions defined over $(2, m, j)$

- Target function:
  $f((1, q, j)) = 1$ if $j = 1$, $f((2, m, j)) = 1$ if $j = 1$, $f(.) = 0$ in all other cases.

- Basis functions $\varphi_{(4,q,.)}$: $\varphi_{(4,q,1)}((1, q, j)) = 1$ if $j \in \{1, 2, 4, 6\}$, $\varphi_{(4,q,1)}(.) = 0$ everywhere else.
  $\varphi_{(4,q,2)}((1, q, j)) = 1$ if $j \in \{1, 3, 5\}$, -1 if $j = 4$, 0 for other cases.
  $\varphi_{(4,q,3)}((1, q, j)) = 1$ if $j \in \{1, 3, 7\}$, -1 if $j = 6$, 0 for other cases.

- Basis functions $\varphi_{(5,m,m')}$:
  $\varphi_{(5,m,m')}((2, m, j)) = 1$ if $j \in \{1, 2\}$, $\varphi_{(5,m,m')}((3, m, m')) = M$, $\varphi_{(5,m,m')}((1, q, j)) = \frac{1}{m'}$ if $\exists v : (T_{m,v} = X_q \ \& \ j = 4)$ or $(T_{m,v} = \bar{X}_q \ \& \ j = 6)$, $\varphi_{(5,m,m')}(.) = 0$ for other cases.

- Basis functions $\varphi_{(6,m,.)}$:
  $\varphi_{(6,m,j)}((2, m, j')) = 1$ if $j' \in \{1, 3, 3 + j\}$, $\varphi_{(6,m,j)}((1, q, j')) = 1$ if $(T_{m,j} = X_q \ \& \ j' = 4)$ or $(T_{m,j} = \bar{X}_q \ \& \ j' = 6)$, 0 for other cases.

STEP 2:

On Figure 3.17 we can see the values taken by f, $\varphi_{(4,q,1)}$, $\varphi_{(4,q,2)}$ and $\varphi_{(4,q,3)}$ over $(1,q,1),(1,q,2),\ldots,(1,q,7)$. **Key step:** Because no other function will take a non-zero value at $(1,q,j)$ for $j \neq 4,6$, we will be able to deduce that in order to approximate f there will be no other option than making at least 1 product. Intuitively we can see it because if we try to approximate the value $f((1,q,1)) = 1$, using $\varphi_{(4,q,1)}$ (resp. $\varphi_{(4,q,2)}$, $\varphi_{(4,q,3)}$) would poorly approximate f at $(1,q,2)$ (resp. $(1,q,5)$, $(1,q,7)$)

On Figure 3.18 we can see the values taken by f, $\varphi_{(5,m,m')}$ and $\varphi_{(6,m,j')}$ over $(2,m,.)$. **Key step:** As before, because no other function will take a non-zero value at $(2,m,.)$, we can deduce that in order to approximate f there will be no other option than making at least 1 product.
We formally prove these two key steps below by evaluating some functions at values $1 \leq n \leq N$.

For convenience of notation, we will write $\alpha(G_p)$ instead of α_p, and we may also use this notation for some functions G_p that are not constructed by the solution. In our notation, this will be equivalent to $\alpha(G_p) = 0$.

Let $H(n)$ denote the property: $|f(n) - \sum_{i=1}^{P} \alpha_i \cdot G_i(n)| < \epsilon$ obtained from the evaluation of $\|f - \sum_{i=1}^{P} \alpha_i \cdot G_i\|_\infty < \epsilon$ at n.

Evaluation at $n = (1,q,.)$:

- $H((1,q,2)) \Rightarrow |\alpha(\varphi_{(4,q,1)})| < \epsilon$
- $H((1,q,5)) \Rightarrow |\alpha(\varphi_{(4,q,2)})| < \epsilon$
- $H((1,q,7)) \Rightarrow |\alpha(\varphi_{(4,q,3)})| < \epsilon$
- $H((1,q,1)) \Rightarrow$ Since $\epsilon \leq \frac{1}{5}$, at least one product is made among the functions $\varphi_{(4,q,1)}$, $\varphi_{(4,q,2)}$, $\varphi_{(4,q,3)}$

Evaluation at $n = (2,m,.)$ and $n = (3,m,m')$:

- $H((3,m,m')) \Rightarrow |\alpha(\varphi_{(5,m,m')})| < \frac{\epsilon}{M}$
- $H((2,m,3+j)) \Rightarrow |\alpha(\varphi_{(6,m,j)})| < \epsilon$
- $H((2,m,1)) \Rightarrow$ Since $\epsilon \leq \frac{1}{6}$, at least one product is made among the functions $\varphi_{(5,m,.)}$, $\varphi_{(6,m,.)}$

We can deduce: **Lemma**: any solution **(APPROX)** that we consider can be supposed to involve only 1 product for each pair of clusters.

STEP 3:

We claim that if solving this created **(APPROX)** problem leads to a cost $c = Q + M$ then the corresponding **(3-SAT)** problem has a solution which can be reconstructed from the $g_{.,.}$ s. Otherwise c will be $> Q + M$ and there will be no solution to the corresponding **(3-SAT)**. This will prove that **(APPROX)** is NP-hard.

At this step, we only prove the first direction:
First direction:
We notice that if there is a solution to the given **(3-SAT)** problem, then there is a solution to our constructed **(APPROX)** problem. For this, define for each q some G_p as $\varphi_{(4,q,1)} \times \varphi_{(4,q,2)}$ or $\varphi_{(4,q,1)} \times \varphi_{(4,q,3)}$ depending on whether X_q is $true$ or $false$. Define also, for each m, some G_p as $\varphi_{(5,m,m')} \times \varphi_{(6,m,j)}$ where j is any (let's say the first) value for which $T_{m,j}$ is $true$, and m' is the number of times when we use $\varphi_{(6,m,j)}$ in such products. To approximate f, we add all the G_p constructed ; that is we take $\alpha_p = 1\ \forall p$.

This construction also gives a hint to the reader for guessing the way to recover the solution of **(3-SAT)** from the solution of **(APPROX)**, by looking at which products were computed to construct the G_p s.

STEP 4:

Other direction:

Intuitively we can see that the product made on Figure 3.17 should be $\varphi_{(4,q,1)} \times \varphi_{(4,q,2)}$ or $\varphi_{(4,q,1)} \times \varphi_{(4,q,3)}$ because if we try to approximate the value $f((1,q,1)) = 1$, using $\varphi_{(4,q,2)}\varphi_{(4,q,3)}$ would poorly approximate f at $(1,q,3)$.
We will use this option to encode whether the boolean variable X_q is set to true ($\varphi_{(4,q,1)}\varphi_{(4,q,2)}$ is computed) or to false ($\varphi_{(4,q,1)}\varphi_{(4,q,3)}$ is computed).
Evaluations at $(1,q,.)$:

- $H((1,q,3)) \Rightarrow |\alpha(\varphi_{(4,q,2)} \times \varphi_{(4,q,3)})| < \epsilon$
- $H((1,q,1)) \Rightarrow$ Since $\epsilon \leq \frac{1}{5}$, thanks to the Lemma: $\alpha(\varphi_{(4,q,1)} \times \varphi_{(4,q,2)}) \neq 0$ xor $\alpha(\varphi_{(4,q,1)} \times \varphi_{(4,q,3)}) \neq 0$

Evaluation at $(2,m,.)$ and $(3,m,m')$:

- $H((2,m,2)) \Rightarrow |\alpha(\varphi_{(5,m,m')} \times \varphi_{(5,m,m'')})| < \epsilon$
- $H((2,m,1)) \Rightarrow$ Since $\epsilon \leq \frac{1}{6}$, thanks to the Lemma: $\exists! m', j : \alpha(\varphi_{(5,m,m')} \times \varphi_{(6,m,j)}) \neq 0$.
- $H((1,q,4))$ (respectively $(1,q,6)$) for X_q (resp. $\bar{X}_q$) matching $T_{m,j}$ of the above constraints $\Rightarrow$ the product chosen in the previous analysis should match. That is, $\alpha(\varphi_{(4,q,1)} \times \varphi_{(4,q,2)}) \neq 0$, for this case where $T_{m,j} = X_q$ (resp. $\alpha(\varphi_{(4,q,1)} \times \varphi_{(4,q,3)}) \neq 0$, case where $T_{m,j} = \bar{X}_q$).

Therefore, fixing each X_q to $true$ or $false$ depending on whether $\alpha(\varphi_{(4,q,1)} \times \varphi_{(4,q,2)}) \neq 0$ or $\alpha(\varphi_{(4,q,1)} \times \varphi_{(4,q,3)}) \neq 0$ gives a solution to the initial **(3-SAT)** problem.

3.7.4 Future work

We have shown above that the image of the product of 2 functions can be defined as the product of the image. This can be generalized to higher order products, for example using 3 products. However, using more than 2 products tends to increase noise and does not give better approximations in practice. On the other hand, the product property can be generalized to applying any smooth function: If we already know the images $\Psi_0, \ldots, \Psi_{k-1} \in V_N \to \mathbb{R}$ of some functions $\Phi_0, \ldots, \Phi_{k-1}$ (for example the Laplace basis functions in our case), then we can infer that the image of $h \circ (\Phi_0, \ldots, \Phi_{k-1})$ should be $h \circ (\Psi_0, \ldots, \Psi_{k-1})$ for any smooth function $h : \mathbb{R}^k \to \mathbb{R}$. This follows from $h \circ (\Psi_0, \ldots, \Psi_{k-1}) = h \circ (\Phi_0 \circ T, \ldots, \Phi_{k-1} \circ T) = (h \circ (\Phi_0, \ldots, \Phi_{k-1})) \circ T$. The case of the product of 2 functions corresponds to $h(X_0, \ldots, X_{k-1}) = X_i \times X_j$.

It would be interesting to see how this function h can be best chosen to fit to a function $f \in V_M \to \mathbb{R}$ so that $f \approx h \circ (\Phi_0, \ldots, \Phi_{k-1})$ allows us to transfer f.

Chapter 4

Deep learning for non linear function approximation and mapping

The approximation and transfer of functions between shapes is a widely studied task in geometry processing due to its great importance in several applications. In this chapter, starting from a given functional map between two shapes, we aim at learning how to improve its approximation and transfer power without any additional effort in the correspondence estimation, or increasing the basis in which the map is expressed. We achieve this goal by demonstrating that it is possible to compute a non-linear approximation of a function using a fixed size basis, which can then be transferred directly using the given functional map. For this, we adopt a strategy based on neural networks that gives rise to a non-linear representation, leading to very accurate solutions. We exploit the power of our non-linear functional space representation for improving the quality of functional transfer and point-to-point map recovery with respect to existing methods. We show the improvement provided by our framework in segmentation transfer.

4.1 Introduction

Over the last decades, the use of functional space representation and analysis has grown significantly in the fields of geometry processing and shape analysis. A functional space can be seen as an alternative characterization of the manifold over which it is defined. This characterization is informative and independent from the discretization adopted to describe the manifolds, making it particularly flexible. Functional representation owes most of its strength to the *functional maps* framework, introduced in [69]. Based on the functional shape representation of surfaces, this framework provides a particularly simple and flexible approach for the estimation of correspondences between shapes. Since the original work [69], a large number of applications and methods have utilized this approach for geometry processing (see [70] for a recent overview).

Most of the works that adopt the functional maps framework are based on representing the functional space on a shape by a small set of functions that, combined together, span a sufficient subspace. The extension of the Fourier representation to curved surfaces, based on a small number of eigenfunctions of the Laplace-Beltrami operator (LBO), is by far the most commonly adopted functional space representation for many reasons. First, it has been extensively studied in several areas, including differential geometry, signal processing, and, more recently, computer science and geometry processing. Furthermore

the Laplace-Beltrami operator is completely intrinsic, in other words it is independent from the 3D embedding in which the manifold is represented, which provides robustness to near-isometric shape deformations in practice. Moreover, several standard approaches have been proposed for computing the LBO and its associated eigenfunctions. Finally, using only a relatively small number of such eigenfunctions, it is possible to obtain a very concise representation that still remains informative. Finally, the optimality of the Fourier basis for the representation of smooth functions defined on surface has also been recently proved in [1]. The main drawback of the Laplace-Beltrami eigenbasis is that there is a trade-off between the dimensionality of the basis used and the range of frequencies that can be represented, if the eigenfunctions are combined linearly. In other words it is not possible to obtain at the same time a very compact basis, which can represent a wide range of frequencies of functions. When considering only the first few eigenfunctions, intuitively, this limitation is due to the low pass representation obtained using such a basis. In many recent works, alternative functional space representations have been proposed, [22, 46, 60, 66, 67]. All these methods tried to improve the functional space description allowing the representation of higher frequency with respect to the ones represented by the Laplace-Beltrami basis.

Despite the inherent limitations of representing functional spaces using a reduced basis, it has nevertheless been observed that the functional maps pipeline can allow to efficiently capture the low frequency information present in a map, which can then potentially be extended to other frequencies as well. This fact has been shown in Chapter 3, where functions are represented not only as a linear combination of the standard LBO eigenfunctions but also allowing elementwise products between pairs of basis functions. Since a functional map corresponding to a point-to-point map must also preserve products of functions [87] we showed in Chapter 3 that in many cases higher-frequency information can be transferred without any additional effort in functional map estimation.

In this work we propose to extend this idea and to apply an *arbitrary non-linear transformation of the basis functions* to approximate and transfer real-valued functions using a given functional map. For this, we propose a *deep learning* based solution for non-linear function approximation and mapping. Our formulation exploits the representation power of neural networks to approximate non-linear transformations, and also uses strong regularization that is adapted to the functional map setting, in which the basis can undergo significant changes before and after the transfer. Once we learn an optimal non-linear function approximation, applying it on the target shape is trivial, since functional maps associated with point-to-point correspondences must preserve arbitrary non-linear transformations of functions pointwise.

As we show in our experiments this non-linearity gives rise to substantial improvements.

4.2 Related Work

Shape comparison and matching is among the key problems in shape analysis. This problem usually boils down to computing an accurate pointwise correspondence or map between a pair of shapes. Following this point of view, a large number of methods have

been proposed over the last decades, [18, 28, 43, 48, 54, 61, 88, 90], just to name a few. See also [12, 94, 100] for recent overviews.

In this work, we are interested in a particular class of methods based on the functional maps framework, which was first introduced in [69]. Functional maps model correspondences via linear operators between functional spaces defined on manifolds. Following the original approach, most works in this domain, including e.g. [46, 52, 68, 80] encode functional maps as small-sized matrices that represent linear transformations between spaces spanned by a fixed number of basis functions. Although often easy to compute, this approach is also inherently limited in terms of its power to represent and transfer functions that only lie in the span of the pre-computed basis.

Due to this strong dependence of functional maps on the basis in which they are expressed, the choice of basis plays a crucial role in practice. On the one hand, the size of the basis must be sufficiently small to allow the computation of the functional maps efficiently in practice, but on the other hand it must be sufficiently large to transfer sufficiently "interesting" functions accurately. The standard choice of basis in the functional maps framework consists of the first few eigenfunctions of the Laplace Beltrami operator. In the seminal work [95] Taubin has shown the similarity between the Laplacian basis and the classical Fourier analysis, as well as its utility in geometry processing applications. In [49] and [98] many useful properties of the Laplacian basis were shown, while its optimality for representing smooth functions was demonstrated in [1]. Although by choosing a small number of eigenfunctions, the resulting truncated basis can be made really concise, its main drawback is that it then acts as a low-pass filter by only allowing to represent sufficiently low-frequency functions.

Several alternatives to the Laplacian basis have also been proposed. For example, Kovnatsky et al. [46] proposed a coupled basis obtained for a pair of shapes starting from their Laplacian basis. This basis suffers from the same limitations as the original Laplacian basis but it aligns the functional representations on the two shapes. In [22] and [60], two localized versions of the Laplacian basis have been proposed. In [22] the Laplacian operator is modified in order to distribute the energies of the eigenfunctions with respect to a particular potential defined on the surface. In [60], a local and orthonormal enhancement of a small Laplacian basis is obtained from another modified version of the Laplace-Beltrami operator. Similarly in [66] the authors proposed an approach for promoting sparsity in the computed basis, which also often results in more localized bases. All of these works propose specialized functional bases, aimed at representing a pre-defined (e.g. localized) class of functions more accurately. Instead, we propose to apply a non-linear transformation on a given basis to be able to represent and transfer arbitrary functions.

Since its introduction, the functional map framework has led to several follow-up works and applications. We briefly mention some of them: [2], in which the Generalized Multi-Dimensional Scaling is extended to the spectral domain, partial matching [53, 80], consistent vector field design [9] and joint quadrangulation between shapes [10] among others. All of these works are based on the ability of functional maps to transfer information, and can therefore directly benefit from the ability to extend their expressive and transfer power. To achieve this, we focus on the representation of the functional space.

Our main goal is to obtain an " optimal" representation via a non-linear combination of some given basis, for which we exploit a novel learning-based strategy.

Learning and, in particular, deep learning has been successfully applied in shape correspondence in general [14, 15, 52] and in the functional maps framework in particular. Most notably, in the latter category, in [24], the authors propose to learn a set of weights over given descriptor functions that would result in the most accurate functional map estimation. More closely related to this chapter, is the work of [51] in which the authors use deep learning to non-linearly combine descriptors, with the goal of computing optimal functional maps. Our work is fundamentally different in that rather than computing a non-linear combination of descriptors, we optimize for a combination of basis functions in order to approximate and transfer some function of interest. As we show below, this allows to improve the accuracy of function transfer even when using ground truth functional maps of a fixed size. In this way, the approach of [51] is complementary to this chapter, since it focuses on computing optimal functional maps starting from some fixed descriptors, while we focus on optimally *using* reduced-size functional maps to approximate and transfer certain functions.

This chapter is very related to the approach for extending the representation of the functional spaces, proposed in Chapter 3. In that part, we showed that by computing an "extended basis" consisting of both the Laplacian eigenfunctions and of their pointwise products it is possible to both represent a richer class of functions and also to transfer them using a fixed functional map without any additional computational effort.

We propose to build on that work and show that other non-linear combinations of basis functions can be used to extend the representation and transfer power of functional maps even further, again without requiring to recompute the functional maps or increase the size of the functional basis. This is due to the fact that in addition to products, functional maps arising from point-to-point maps must preserve arbitrary non-linear transformations of functions. To achieve this goal we use a formulation based on a neural network architecture, which is well-adapted to approximate non-linear transformation of input data. As we show below, this leads to significant improvement in terms of the accuracy of function transfer compared both to the basic linear functional maps approach and to the product-based method in Chapter 3.

To summarize, our main goal is to learn a new representation of functions defined on surfaces, achieving better function approximation and transfer between shapes and enhancing the performance of the functional maps framework. With this aim we show how deep learning tools can be used to find an optimal representation.

4.3 Background

In this chapter, we propose an extension of the main idea presented in Chapter 3. We start by observing that a generalization of equation 3.1 is that any pointwise operation applied to a set of functions defined on the source shape $\mathcal{M}$ should transfer as the same pointwise operation applied to the image of this (ordered) set of functions.
This can be formalized as follows:

<u>Lemma:</u>
for any smooth functions $h : \mathbb{R}^k \to \mathbb{R}$, $\varphi_1, \ldots, \varphi_k : \mathcal{M} \to \mathbb{R}$, the image of $h \circ (\varphi_1, \ldots, \varphi_k)$ (pointwise composition) on $\mathcal{N}$ is $h \circ (\psi_1, \ldots, \psi_k)$, where each ψ_i is the image of φ_i.

To see this, notice that

$$h \circ (\psi_1, \ldots, \psi_k) = h \circ (\varphi_1 \circ T, \ldots, \varphi_k \circ T) = (h \circ (\varphi_1, \ldots, \varphi_k)) \circ T$$

Note that equation 3.1 is a particular case of this Lemma, for which we take as function h the function that makes the product between 2 of its variables φ_i and φ_j for some i, j.

4.4 Main Idea

Similarly to the method proposed in Chapter 3, we propose here to improve the accuracy of the transfer of a function $f : \mathcal{M} \to \mathbb{R}$ on $\mathcal{N}$. This time we rely on the Lemma using the basis functions of $\mathbf{\Phi}_{\mathcal{M}}$ as $(\varphi_1, \ldots, \varphi_k)$. Indeed, these are functions whose transfer on $\mathcal{N}$ is given by the functional map, by definition.

Because we want to be as general as possible, we would like to allow any function h. For this, we use a neural network that will optimize for the best possible way of approximating $f(x)$ from $\mathbf{\Phi}_{\mathcal{M}}(x, :)$ (row x of $\mathbf{\Phi}_{\mathcal{M}}$). That is, the neural network h should minimize $\sum_{x \in V_{\mathcal{M}}} A_x \left(f(x) - h\left(\mathbf{\Phi}_{\mathcal{M}}(x, :)\right)\right)^2$, where A_x is the area associated to the vertex x:

$$h_{\text{opt}} = \arg\min_h \sum_{x \in V_{\mathcal{M}}} A_x \left(f(x) - h\left(\mathbf{\Phi}_{\mathcal{M}}(x, :)\right)\right)^2$$

We define the loss function of the neural network as:

$$\frac{\sum_{x \in V_{\mathcal{M}}} A_x \left(f(x) - h\left(\mathbf{\Phi}_{\mathcal{M}}(x, :)\right)\right)^2}{\sum_{x \in V_{\mathcal{M}}} A_x f(x)^2}$$

Then we apply our neural network to $\mathbf{\Phi}_{\mathcal{N}} \boldsymbol{C}$ in order to get the approximate transfer $g : \mathcal{N} \to \mathbb{R}$ of f, output of our method. Thanks to the Lemma discussed in the previous section, the output g is likely to be indeed a good approximation of the exact transfer $f \circ T$.

Similarly to the loss function, we define our test error as

$$\frac{\sum_{y \in V_{\mathcal{N}}} A_y \left((f \circ T)(y) - g(y)\right)^2}{\sum_{y \in V_{\mathcal{N}}} A_y (f \circ T)(y)^2}$$

where $g(y) = h\left((\mathbf{\Phi}_{\mathcal{N}} \boldsymbol{C})(y, :)\right)$ is our approximate transfer.
This is the error that we plot in Figure 4.2 of the section 4.7.

4.5 Description

We specifically design a neural network so that it should approximate any function f at least as well as the standard projection, as shown on Figure 4.1. The input of our

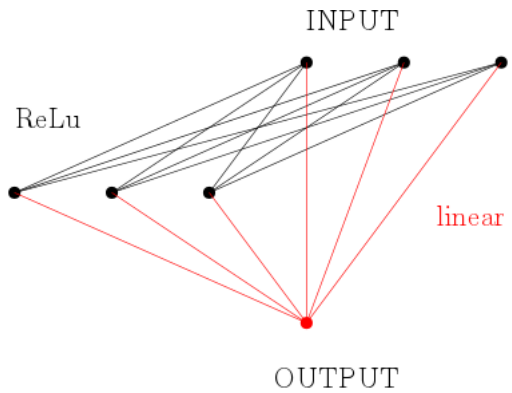


Figure 4.1 – Our neural network, example for $k_{\mathcal{M}} = 3$. It takes as input $k_{\mathcal{M}}$ real numbers that correspond to the spectral embedding of a vertex $x \in V_{\mathcal{M}}$, and gives as output an approximated value $f(x)$ of the function that we want to transfer

network always has size $k_{\mathcal{M}}$ and the output has size 1, because we want our network to act as a function $h : \mathbb{R}^{k_{\mathcal{M}}} \to \mathbb{R}$. Our network is made of 1 intermediate layer of size $k_{\mathcal{M}}$, fully connected and activated with a rectified linear unit activation ("ReLu"). The last layer, of size 1, is a linear combination of the intermediate layer (size $k_{\mathcal{M}}$) and of the input layer (size $k_{\mathcal{M}}$). This design ensures that, if no better approximation is found, our network can at least linearly combine the first $k_{\mathcal{M}}$ inputs as in the basic projection method. Therefore, after some optimization, we should expect our network to improve over the standard projection method, at least for approximating the input function f.

Because our method is based on deriving the transfer of a given function from the transfer of some source functions that we already know how to transfer (in our case the eigenfunctions on the source shape) it it important that our transfer for these source functions should be very accurate. Therefore, instead of using a square matrix $\boldsymbol{C}$, we use a rectangular matrix $\boldsymbol{C}$, with a larger basis on the target shape $\mathcal{N}$ than on the source. These additional frequencies on the target shape allow us to get a more accurate image of the source functions (the basis functions on the source shape).

4.6 Parameters

We implemented the network described in Figure 4.1, in TensorFlow. We trained it using the Adam optimizer with a learning rate of 0.001. We compute our average over our functions, over 10 pairs of shapes from the Faust dataset ([13]).

We use it for both the coordinate functions X, Y, Z, and for 10 indicator functions of different segments, leading to 2 different plots shown on Figure 4.2.

All results are computed for $k_{\mathcal{N}} = 10, 30, 50, 70, 90, 110, 130, 150, 170, 190, 210$ (x axis), the last value at position 230 corresponds to the approximation error

$$\frac{\sum_{x \in V_{\mathcal{M}}} A_x \left(f(x) - h\left(\boldsymbol{\Phi}_{\mathcal{M}}(x,:)\right)\right)^2}{\sum_{x \in V_{\mathcal{M}}} A_x f(x)^2} \tag{4.1}$$

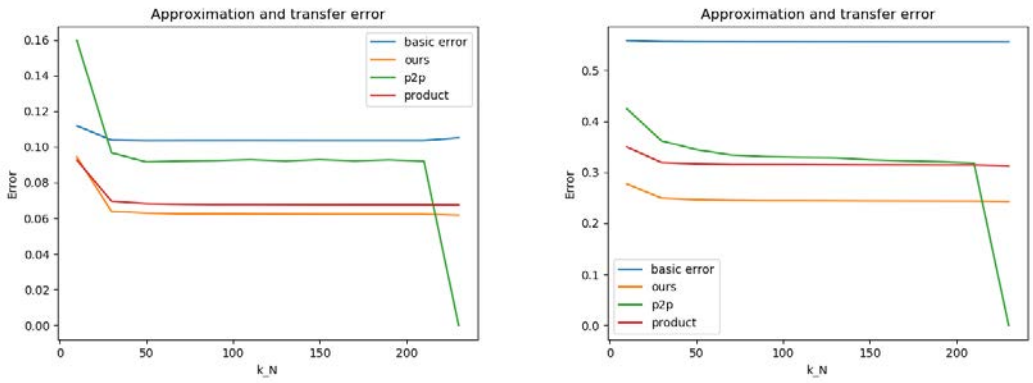


Figure 4.2 – Comparison of the average error made by 4 different methods as a function of $k_{\mathcal{N}}$, for $k_{\mathcal{M}} = 10$ fixed. The last value on each curve represents the approximation error made on the source shape. Left: coordinate functions (X, Y, Z). Right: segment indicator functions, for 10 different segments.

The "p2p" method consists in first converting the map to a point-to-point map, its approximation error is 0 by convention, because that method does not need to approximate f before transferring it. For the same reason, on the column that corresponds to the "p2p" method on the qualitative evaluation (Figures 4.4 and 4.3), the image that corresponds to the approximation (at the top) is simply the exact function on the source.

In the experiments below we use the ground truth functional map $\boldsymbol{C}_{gt}$, as a way of showing that our method can be successful for a sufficiently good map. In practice the interesting case should indeed use a computed map instead, some preliminary results that are not included here show that our improvement can generalize to a computed map.

4.7 Results

On Figure 4.2, we can see a comparison of the approximation and transfer errors made by different methods:

- standard method (std), that consists in simply projecting the function over the space spanned by the basis, then transferring it using the functional map matrix $\boldsymbol{C}$.
- point-to-point conversion (p2p), which first converts the functional map to a point to point map using nearest neighbors, then use the obtained correspondence directly.
- product (prod) is the method of Chapter 3
- Our method (ours) described above

For each method, we plot a curve that shows the average transfer error made for the 3 coordinate functions X, Y, Z (left) or 10 segment indicator functions (right), over 10 pairs of shapes from the FAUST dataset [13]. The error is weighted by the area weights

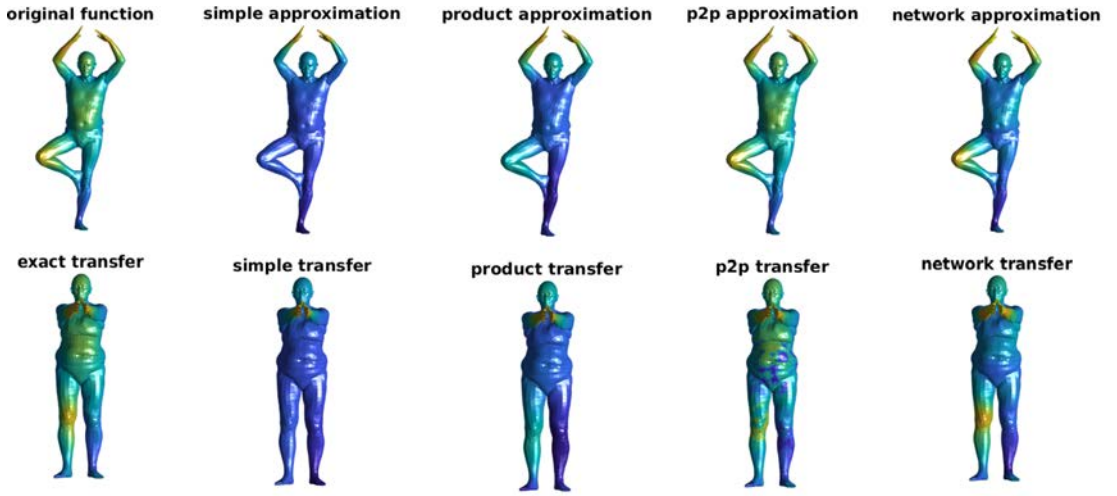


Figure 4.3 – Example comparison of different methods for transfering a coordinate function from the source shape (top row) to the target shape (bottom row). $k_{\mathcal{M}} = 10$, $k_{\mathcal{N}} = 90$. From left to right: exact function and transfer ; standard approximation and transfer ; method of Chapter 3 ; exact function transferred using a point-to-point conversion ; approximation and transfer using our network

associated to each vertex, and rescaled by the squared norm of the ground truth image of the function. For the functional map matrix $\boldsymbol{C}$ we use the ground truth map $\boldsymbol{C}_{gt}$ of size $k_{\mathcal{N}} \times 10$: 10 eigenfunctions on the source shape, $k_{\mathcal{N}}$ on the target shape. $k_{\mathcal{N}}$ goes between 10 and 210, and is on the x-axis of our plot. Finally, the last value on each plot represents the approximation error achieved by each method on the source shape.

As we can see, our network allows a better function transfer than all other methods.

We can see qualitative comparisons for a segment indicator function (Figure 4.4) and a coordinate function (Figure 4.3), where colors represent values of functions. It appears that our method outperforms previous methods.

4.8 Conclusion

The main contribution of this work is a novel, *non-linear* representation of the functional space defined on a shape, that we produce by exploiting a *deep learning* strategy. This non-linear representation allows us to significantly improve the quality of the function approximation and can be used directly in the functional maps framework for function transfer, even when given a fixed-size functional map. Importantly this improvement is achieved without any additional effort in the estimation of the functional map and is complementary to the accuracy achieved in the functional map estimation. In our experiments we show how this non-linearity gives rise to results that outperform the state of the art.

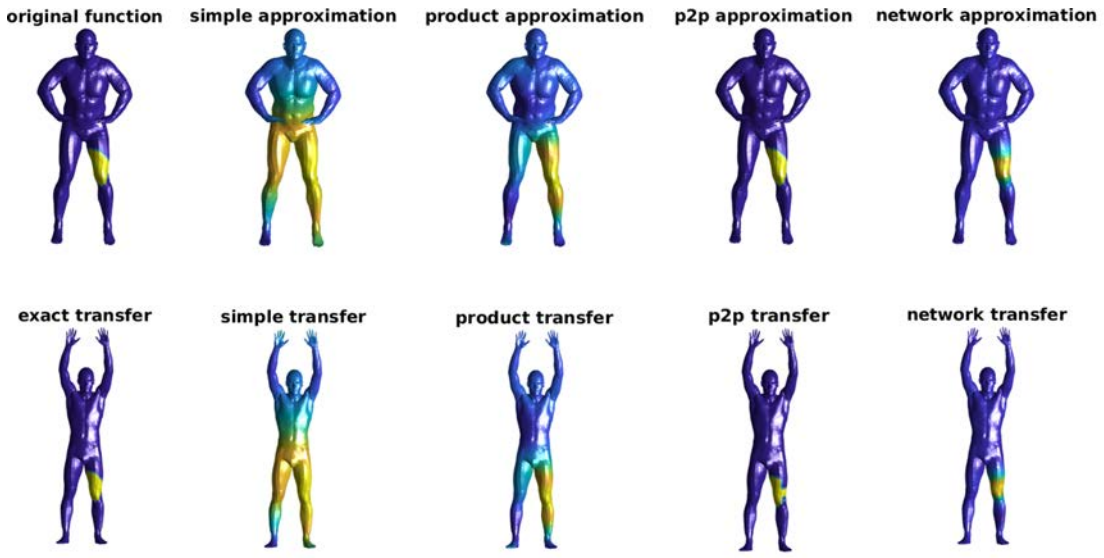


Figure 4.4 – Example comparison of different methods for transfering an indicator function from the source shape (top row) to the target shape (bottom row). $k_{\mathcal{M}} = 10$, $k_{\mathcal{N}} = 90$. From left to right: exact function and transfer ; standard approximation and transfer ; method of Chapter 3 ; exact function transferred using a point-to-point conversion ; approximation and transfer using our network

4.9 Future work

In the future, the most immediate task is to extend the approach described above to *computed*, rather than ground truth functional maps. For this, we would like to improve this method by training over known correspondences from a fixed source shape to various target shapes. This would allow us to define a cost for both function approximation but also for function *transfer*, to each target shape, and therefore should allow resilience to the errors in the functional map matrix. Then we would expect a better network (i.e. a better function h) that we could apply to an unseen shape.

An interesting direction is to use artificially created (synthetic) modified shapes, for which we would know the exact correspondence. Using these shapes for training should give results similar to real shape pairs with ground truth correspondences, but without requiring any input knowledge. This would be a significant achievement, but requires to carefully design an appropriate model for the artificial deformations.

Chapter 5

Miscellaneous

In addition to my work on functional maps, I also worked on other topics that have led to some publications:
_ A new method for analysing the running time of evolutionary algorithms: [27]
_ A method to improve the number of operations required to compute a Schur polynomial, while restricting to using only + and ×: [30]
_ Efficient algorithms for scheduling problems: [23]

5.1 A new analysis method for evolutionary optimization of dynamic and noisy objective functions

Evolutionary algorithms, being problem-independent and randomized heuristics, are generally believed to be robust to dynamic changes and noisy access to the problem instance. We propose a new method to obtain rigorous runtime results for such settings. In contrast to many previous works, our new approach mostly relies on general parameters of the dynamics or the noise models, such as the expected change of the dynamic optimum or the probability to have a dynamic change in one iteration. Consequently, we obtain bounds which are valid for large varieties of such models. Despite this generality, for almost all particular models regarded in the past our bounds are stronger than those given in previous works. As one particular result, we prove that the $(1 + \lambda)$ EA can optimize the OneMax benchmark function efficiently despite a constant rate of 1-bit flip noise. For this, a logarithmic size offspring population suffices (the previous-best result required a super-linear value of λ). Our results suggest that the typical way to find the optimum in such adverse settings is not via a steady approach of the optimum, but rather via an exceptionally fast approach after waiting for a rare phase of low dynamic changes or noise.

5.2 On semiring complexity of Schur polynomials

Semiring complexity is the version of arithmetic circuit complexity that allows only two operations: addition and multiplication. We show that semiring complexity of a Schur polynomial $s_\lambda(x_1, \ldots, x_k)$ labeled by a partition $\lambda = (\lambda_1 \geq \lambda_2 \geq \cdots)$ is bounded by $O(\log(\lambda_1))$ provided the number of variables k is fixed.

5.2.1 Schur polynomial

Let $\lambda = (\lambda_1 \geq \lambda_2 \geq \cdots \geq 0)$ be an integer partition. The *Schur function* (or *Schur polynomial*)) $s_\lambda(x_1, \ldots, x_k)$ is a symmetric polynomial of degree $|\lambda| = \sum_i \lambda_i$ in the variables $x_1, \ldots, x_k$ which can be defined in many different ways. One remarkable feature of Schur polynomials that makes them an exciting object of study in algebraic complexity theory is that the classical formulas defining them fall into two categories. On the one hand, there are determinantal expressions (e.g., the Jacobi-Trudi formula or the bialternant formula) which provide efficient ways to compute Schur functions in an unrestricted setting, i.e., when all arithmetic operations are allowed. On the other hand, Schur functions are generating functions for semistandard Young tableaux. This description represents them as polynomials with manifestly positive coefficients; so they can be computed using addition and multiplication only. We note however that the naïve approach based on these monomial expansions yields algorithms whose (semiring) complexity is very high, and indeed very far from the optimum.

5.2.2 Main result

Our main result is the following. (We use the notation $\lambda' = (\lambda'_1 \geq \lambda'_2 \geq \cdots)$ for the partition conjugate to λ.)

Theorem 5.2.1 *The semiring complexity of a Schur polynomial $s_\lambda(x_1, \ldots, x_k)$ labeled by partition $\lambda = (\lambda_1 \geq \cdots \geq \lambda_\ell)$ is at most $O(\log(\lambda_1) k^5 2^{k\ell} \ell^d)$ where $d = \max_j \lambda'_j (k - \lambda'_j)$.*

Since $\ell \leq k$ (or else $s_\lambda(x_1, \ldots, x_k) = 0$) and $d \leq k^2/4$, we obtain:

Corollary 5.2.2 *The semiring complexity of $s_\lambda(x_1, \ldots, x_k)$ is bounded from above by $k^{k^2(\frac{1}{4}+o(1))} O(\log(\lambda_1))$. If the number of variables k is fixed, then this complexity is $O(\log(\lambda_1))$.*

5.3 Scheduling with gaps: New models and algorithms

We initiate the study of scheduling problems where the number or size of the gaps in the schedule is taken into consideration. We focus on the model with unit jobs. First we examine scheduling problems with release times and deadlines, where we consider variants of minimum-gap scheduling, including maximizing throughput with a budget for gaps or minimizing the number of gaps with a throughput requirement. We then turn to other objective functions. For example, in some scenarios, gaps in a schedule may be actually desirable, leading to the problem of maximizing the number of gaps. The second part of the paper examines the model without deadlines, where we focus on the tradeoff between the number of gaps and flow time.

For all these problems we provide polynomial algorithms. The solutions involve a spectrum of algorithmic techniques, including different dynamic programming formulations, speed-up techniques based on searching Monge arrays, searching $X + Y$ matrices, or implicit binary search. Throughout the paper, we also draw a connection between

our scheduling problems and their continuous analogues, namely hitting set problems for intervals of real numbers.

Chapter 6

General Conclusion

6.1 Summary

We presented different lines of work in this thesis.

In Chapter 2 we presented a method that changes the way descriptor constraints are expressed, and represents descriptors via the associated pointwise multiplication operators that they define instead of simply using descriptor values. We saw that this representation is likely to induce maps that are closer to being point-to-point maps, which is a often a desirable property that one should expect from a good functional map. This method remains in the linear framework of the standard functional maps pipeline, and thus retains its computational efficiency.

In Chapter 3 we presented a method that uses a computed functional map and extends the space of functions that can be transferred by it. Our method assumes that the functional map is approximately induced by a point-to-point map. This property can be assumed to hold on a good functional map given as input. The method uses the pointwise product preservation property of point-to-point maps to define a way to transfer products of basis functions, which enables a way to transfer a functional space richer than the standard approach which is limited to linear combinations of basis functions only.

In Chapter 4 we introduced an extension of the previous method that uses composition with *any* combination of basis functions, instead of only their pointwise products. We use neural networks to generate such a combination, and show its improved accuracy in a variety of settings.

Finally in Chapter 5 we presented different works, unrelated to the main PhD topic.

6.2 Future work

As future work, one interesting direction is to explore preservation rules different from the product preservation used in Chapter 2. For example it could be a rule about the preservation of second-order properties such as geodesic or diffusion distances. Preliminary experiments on these two specific cases do not show any improvement over the product preservation rule, but many other related ideas could be of interest.

We also plan to extend the ideas proposed in Chapters 3 and 4 by using the composition with any smooth function. Trying to simulate a smooth function as general as possible using neural networks is simply one option. It could be more appropriate to use other kinds of composition operators, either more suited for an efficient approximation of a target function or for a better smoothness that would improve the function transfer.

We also plan to use the composition with a function h that can change across different vertices. It could also apply to the neighborhood of the vertex, thus using the local information that could allow, for example, to recognize a part of the body.

6.3 Position of the work in the community

The methods proposed in Chapters 2 and 3 are now considered as state-of-the-art, and are used in the evaluation on functional maps as in [25, 32] (method of Chapter 2) [82] (method of Chapter 3). The method proposed in Chapter 2 specifically allows to better extract and exploit the information contained in a given descriptor, which can then be used for evaluating the quality of newly proposed descriptors as was done, for example in [58]. The commutativity cost defined in Chapter 2 has also been adapted in several follow-up works, as in [32, 55].

Approaches similar to that proposed in Chapter 3 have been recently proposed. An approach for preserving inner products between descriptors rather than descriptor values themselves consists in lifting descriptors to higher dimensional vectors [105]. Another approach uses unsupervised learning in order to compute a non-linear transformation of descriptor functions, that will then lead to improved shape correspondences [82]. A recent method for finding functional maps that are closer to arising from point-to-point maps has also been proposed in [33] and consists in using user-prescribed corresponding curves for finding maps between shapes that are semantically similar but geometrically very different. Another approach relies on a genetic algorithm [83]. Finally, a method that uses persistence diagrams to provide an optimization scheme that helps to promote functional maps associated with continuous bijective point-to-point maps has been proposed in [78].

To summarize, the contributions of this thesis are not limited to the specific improvements that have been shown, but have already inspired follow-up works and have been used as building blocks in other research works in non-rigid shape matching.

Bibliography

[1] Y. Aflalo, H. Brezis, and R. Kimmel, *On the optimality of shape and data representation in the spectral domain*, SIAM Journal on Imaging Sciences, 8 (2015), pp. 1141–1160. (Cited on pages 22, 55, 70 and 71.)

[2] Y. Aflalo and R. Kimmel, *Spectral multidimensional scaling*, PNAS, 110 (2013), pp. 18052–18057. (Cited on pages 42 and 71.)

[3] P. K. Agarwal and M. Sharir, *Efficient algorithms for geometric optimization*, ACM Computing Surveys (CSUR), 30 (1998), pp. 412–458. (Cited on pages 3 and 7.)

[4] N. Aigerman and Y. Lipman, *Hyperbolic orbifold tutte embeddings*, ACM TOG, 35 (2016). (Cited on page 40.)

[5] N. Aigerman, R. Poranne, and Y. Lipman, *Seamless Surface Mappings*, ACM Transactions on Graphics (TOG), 34 (2015), p. 72. (Cited on pages 19 and 40.)

[6] P. Alliez, D. Cohen-Steiner, O. Devillers, B. Lévy, and M. Desbrun, *Anisotropic polygonal remeshing*, in ACM Transactions on Graphics (TOG), vol. 22, ACM, 2003, pp. 485–493. (Cited on pages 3 and 7.)

[7] D. Anguelov, P. Srinivasan, D. Koller, S. Thrun, J. Rodgers, and J. Davis, *SCAPE: Shape Completion and Animation of People*, in ACM Transactions on Graphics (TOG), vol. 24, ACM, 2005, pp. 408–416. (Cited on pages 32 and 35.)

[8] M. Aubry, U. Schlickewei, and D. Cremers, *The Wave Kernel Signature: A Quantum Mechanical Approach to Shape Analysis*, in Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on, IEEE, 2011, pp. 1626–1633. (Cited on pages 22, 34 and 52.)

[9] O. Azencot, M. Ben-Chen, F. Chazal, and M. Ovsjanikov, *An operator approach to tangent vector field processing*, 32 (2013), pp. 73–82. (Cited on pages 23, 40, 42, 44 and 71.)

[10] O. Azencot, E. Corman, M. Ben-Chen, and M. Ovsjanikov, *Consistent functional cross field design for mesh quadrangulation*, ACM Trans. Graph., 36 (2017), pp. 92:1–92:13. (Cited on pages 23, 40, 42, 44, 57, 58, 60 and 71.)

[11] A. C. Berg, T. L. Berg, and J. Malik, *Shape matching and object recognition using low distortion correspondences*, in Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on, vol. 1, 2005, pp. 26–33. (Cited on page 41.)

[12] S. Biasotti, A. Cerri, A. Bronstein, and M. Bronstein, *Recent trends, applications, and perspectives in 3d shape similarity assessment*, 35 (2016), pp. 87–119. (Cited on pages 41 and 71.)

[13] F. Bogo, J. Romero, M. Loper, and M. J. Black, *FAUST: Dataset and Evaluation for 3d Mesh Registration*, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 3794–3801. (Cited on pages 18, 32, 33, 36, 39, 51, 53, 57, 58, 74 and 75.)

[14] D. Boscaini, J. Masci, S. Melzi, M. M. Bronstein, U. Castellani, and P. Vandergheynst, *Learning class-specific descriptors for deformable shapes using localized spectral convolutional networks*, Computer Graphics Forum, 34 (2015), pp. 13–23. (Cited on page 72.)

[15] D. Boscaini, J. Masci, E. Rodolà, and M. M. Bronstein, *Learning shape correspondence with anisotropic convolutional neural networks*, in Proc. NIPS, Barcelona, Spain, 2016. (Cited on page 72.)

[16] M. Botsch, L. Kobbelt, M. Pauly, P. Alliez, and B. Lévy, *Polygon Mesh Processing*, CRC press, 2010. (Cited on page 14.)

[17] A. M. Bronstein, M. M. Bronstein, A. M. Bruckstein, and R. Kimmel, *Partial similarity of objects, or how to compare a centaur to a horse*, International Journal of Computer Vision, 84 (2009), p. 163. (Cited on page 20.)

[18] A. M. Bronstein, M. M. Bronstein, and R. Kimmel, *Generalized Multidimensional Scaling: A Framework for Isometry-Invariant Partial Surface Matching*, Proceedings of the National Academy of Sciences, 103 (2006), pp. 1168–1172. (Cited on pages 19, 25, 41 and 71.)

[19] ——, *Numerical Geometry of Non-Rigid Shapes*, Springer Science & Business Media, 2008. (Cited on pages 32, 34, 51 and 57.)

[20] A. M. Bronstein, M. M. Bronstein, R. Kimmel, M. Mahmoudi, and G. Sapiro, *A gromov-hausdorff framework with diffusion geometry for topologically-robust non-rigid shape matching*, International Journal of Computer Vision, 89 (2010), pp. 266–286. (Cited on page 19.)

[21] Q. Chen and V. Koltun, *Robust Nonrigid Registration by Convex Optimization*, in Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 2039–2047. (Cited on page 19.)

[22] Y. CHOUKROUN, A. SHTERN, A. BRONSTEIN, and R. Kimmel, *Hamiltonian operator for spectral shape analysis*, arXiv:1611.01990, (2016). (Cited on pages 70 and 71.)

[23] M. Chrobak, M. Golin, T.-W. Lam, and D. Nogneng, *Scheduling with gaps: New models and algorithms*, in International Conference on Algorithms and Complexity, Springer, 2015, pp. 114–126. (Cited on pages 5, 9 and 79.)

[24] É. Corman, M. Ovsjanikov, and A. Chambolle, *Supervised Descriptor Learning for Non-Rigid Shape Matching*, in European Conference on Computer Vision, Springer, 2014, pp. 283–298. (Cited on pages 23, 26, 27 and 72.)

[25] L. Cosmo, M. Panine, A. Rampini, M. Ovsjanikov, M. M. Bronstein, and E. Rodolà, *Isospectralization, or how to hear shape, style, and correspondence*, arXiv preprint arXiv:1811.11465, (2018). (Cited on page 84.)

[26] M. Cuturi, *Sinkhorn distances: Lightspeed computation of optimal transport*, in Advances in neural information processing systems, 2013, pp. 2292–2300. (Cited on page 42.)

[27] R. Dang-Nhu, T. Dardinier, B. Doerr, G. Izacard, and D. Nogneng, *A new analysis method for evolutionary optimization of dynamic and noisy objective functions*, in Proceedings of the Genetic and Evolutionary Computation Conference, ACM, 2018, pp. 1467–1474. (Cited on pages 5, 9 and 79.)

[28] A. Dubrovina and R. Kimmel, *Matching shapes by eigendecomposition of the Laplace-Beltrami operator*, Proc. 3DPVT, 2 (2010). (Cited on pages 41 and 71.)

[29] N. Dym, H. Maron, and Y. Lipman, *Ds++: A flexible, scalable and provably tight relaxation for matching problems*, arXiv preprint arXiv:1705.06148, (2017). (Cited on page 19.)

[30] S. Fomin, D. Grigoriev, D. Nogneng, and E. Schost, *On semiring complexity of schur polynomials*, arXiv preprint arXiv:1608.05043, (2016). (Cited on pages 5, 9 and 79.)

[31] V. Ganapathi-Subramanian, B. Thibert, M. Ovsjanikov, and L. Guibas, *Stable Region Correspondences Between Non-Isometric Shapes*, in Computer Graphics Forum, vol. 35, 2016, pp. 121–133. (Cited on page 19.)

[32] A. Gasparetto, L. Cosmo, E. Rodola, M. Bronstein, and A. Torsello, *Spatial maps: From low rank spectral to sparse spatial functional representations*, in 3D Vision (3DV), 2017 International Conference on, IEEE, 2017, pp. 477–485. (Cited on page 84.)

[33] A. Gehre, M. Bronstein, L. Kobbelt, and J. Solomon, *Interactive curve constrained functional maps*, in Computer Graphics Forum, vol. 37, Wiley Online Library, 2018, pp. 1–12. (Cited on page 84.)

[34] V. Guillemin and A. Pollack, *Differential topology*, vol. 370, American Mathematical Soc., 2010. (Cited on page 13.)

[35] N. Hasler, C. Stoll, M. Sunkel, B. Rosenhahn, and H.-P. Seidel, *A Statistical Model of Human Pose and Body Shape*, 28 (2009), pp. 337–346. (Cited on pages 13, 18, 25 and 39.)

[36] M. Hazewinkel, N. Gubareni, and V. Kirichenko, *Algebras, rings and modules*, Kluver Academic Publishers, 2004. (Cited on page 41.)

[37] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle, *Surface reconstruction from unorganized points*, vol. 26, ACM, 1992. (Cited on pages 3 and 7.)

[38] Q. Huang, F. Wang, and L. Guibas, *Functional map networks for analyzing and exploring large shape collections*, ACM Transactions on Graphics (TOG), 33 (2014), p. 36. (Cited on pages 40 and 42.)

[39] Q.-X. Huang, B. Adams, M. Wicke, and L. J. Guibas, *Non-rigid registration under isometric deformations*, in Computer Graphics Forum, vol. 27, Wiley Online Library, 2008, pp. 1449–1457. (Cited on page 20.)

[40] R. Huang, F. Chazal, and M. Ovsjanikov, *On the stability of functional maps and shape difference operators*, Computer Graphics Forum, (2017). (Cited on page 44.)

[41] I. Kezurer, S. Z. Kovalsky, R. Basri, and Y. Lipman, *Tight relaxation of quadratic matching*, in Computer Graphics Forum, vol. 34, Wiley Online Library, 2015, pp. 115–128. (Cited on page 19.)

[42] M. Kilian, N. J. Mitra, and H. Pottmann, *Geometric Modeling in Shape Space*, 26 (2007), p. 64. (Cited on pages 18, 25 and 39.)

[43] V. G. Kim, Y. Lipman, and T. Funkhouser, *Blended Intrinsic Maps*, 30 (2011), p. 79. (Cited on pages 19, 32, 33, 34, 35, 36, 41 and 71.)

[44] R. Kolluri, J. R. Shewchuk, and J. F. O'Brien, *Spectral surface reconstruction from noisy point clouds*, in Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing, ACM, 2004, pp. 11–21. (Cited on pages 3 and 7.)

[45] A. Kovnatsky, M. M. Bronstein, X. Bresson, and P. Vandergheynst, *Functional correspondence by matrix completion*, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 905–914. (Cited on page 27.)

[46] A. Kovnatsky, M. M. Bronstein, A. M. Bronstein, K. Glashoff, and R. Kimmel, *Coupled quasi-harmonic bases*, 32 (2013), pp. 439–448. (Cited on pages 41, 42, 70 and 71.)

[47] A. Kovnatsky, K. Glashoff, and M. M. Bronstein, *MADMM: a generic algorithm for non-smooth optimization on manifolds*, in Proc. ECCV, Springer, 2016, pp. 680–696. (Cited on page 22.)

[48] M. Leordeanu and M. Hebert, *A spectral technique for correspondence problems using pairwise constraints*, in Proc. ICCV, vol. 2, IEEE, 2005, pp. 1482–1489. (Cited on pages 41 and 71.)

[49] B. Lévy, *Laplace-Beltrami eigenfunctions towards an algorithm that understands geometry*, in Proc. SMI, Washington, DC, 2006, IEEE, pp. 13–25. (Cited on page 71.)

[50] Y. Lipman and T. Funkhouser, *Möbius Voting for Surface Correspondence*, 28 (2009), p. 72. (Cited on pages 19, 20, 40 and 41.)

[51] O. Litany, T. Remez, E. Rodolà, A. M. Bronstein, and M. M. Bronstein, *Deep functional maps: Structured prediction for dense shape correspondence*, in Proc. ICCV, 2017. (Cited on pages 42 and 72.)

[52] O. Litany, E. Rodolà, A. M. Bronstein, and M. M. Bronstein, *Fully spectral partial shape matching*, Computer Graphics Forum, 36 (2017), pp. 247–258. (Cited on pages 42, 71 and 72.)

[53] O. Litany, E. Rodolà, A. M. Bronstein, M. M. Bronstein, and D. Cremers, *Non-Rigid Puzzles*, Computer Graphics Forum, 35 (2016), pp. 135–143. (Cited on pages 22, 26, 27, 42 and 71.)

[54] M. Mandad, D. Cohen-Steiner, L. Kobbelt, P. Alliez, and M. Desbrun, *Variance-minimizing transport plans for inter-surface mapping*, ACM Transactions on Graphics, 36 (2017), p. 14. (Cited on pages 42 and 71.)

[55] R. Marin, S. Melzi, E. Rodolà, and U. Castellani, *Farm: Functional automatic registration method for 3d human bodies*, arXiv preprint arXiv:1807.10517, (2018). (Cited on page 84.)

[56] Z. C. Marton, R. B. Rusu, and M. Beetz, *On fast surface reconstruction methods for large and noisy point clouds*, in Robotics and Automation, 2009. ICRA'09. IEEE International Conference on, IEEE, 2009, pp. 3218–3223. (Cited on pages 3 and 7.)

[57] D. Mateus, R. Horaud, D. Knossow, F. Cuzzolin, and E. Boyer, *Articulated shape matching using laplacian eigenfunctions and unsupervised point registration*, in Proc. CVPR, IEEE, 2008, pp. 1–8. (Cited on page 41.)

[58] S. Melzi, M. Ovsjanikov, G. Roffo, M. Cristani, and U. Castellani, *Discrete time evolution process descriptor for shape analysis and matching*, ACM Transactions on Graphics (TOG), 37 (2018), p. 4. (Cited on page 84.)

[59] S. Melzi, E. Rodola, U. Castellani, and M. Bronstein, *Shape analysis with anisotropic windowed fourier transform*, in International Conference on 3D Vision (3DV), 2016. (Cited on page 53.)

[60] S. Melzi, E. Rodolà, U. Castellani, and M. M. Bronstein, *Localized manifold harmonics for spectral shape analysis*, arXiv:1707.02596, (2017). (Cited on pages 41, 56, 70 and 71.)

[61] F. Mémoli, *On the use of Gromov-Hausdorff distances for shape comparison*, Point-Based Graphics, (2007). (Cited on pages 41 and 71.)

[62] ——, *Gromov–Wasserstein Distances and the Metric Approach to Object Matching*, Foundations of computational mathematics, 11 (2011), pp. 417–487. (Cited on pages 19 and 41.)

[63] M. Meyer, M. Desbrun, P. Schröder, and A. H. Barr, *Discrete Differential-Geometry Operators for Triangulated 2-Manifolds*, in Visualization and mathematics III, Springer, 2003, pp. 35–57. (Cited on pages 16 and 34.)

[64] S. Morita, *Geometry of differential forms*, vol. 201, American Mathematical Soc., 2001. (Cited on pages 15 and 16.)

[65] B. K. Natarajan, *Sparse approximate solutions to linear systems*, SIAM Journal on Computing, 24 (1995), pp. 227–234. (Cited on page 47.)

[66] T. Neumann, K. Varanasi, C. Theobalt, M. Magnor, and M. Wacker, *Compressed manifold modes for mesh processing*, 33 (2014), pp. 35–44. (Cited on pages 41, 70 and 71.)

[67] D. Nogneng, S. Melzi, E. Rodolà, U. Castellani, M. Bronstein, and M. Ovsjanikov, *Improved Functional Mappings via Product Preservation*, Computer Graphics Forum, 37 (2018), pp. 179–190. (Cited on pages 5, 9 and 70.)

[68] D. Nogneng and M. Ovsjanikov, *Informative descriptor preservation via commutativity for shape matching*, Computer Graphics Forum, 36 (2017), pp. 259–267. (Cited on pages 5, 9, 53, 54, 57 and 71.)

[69] M. Ovsjanikov, M. Ben-Chen, J. Solomon, A. Butscher, and L. Guibas, *Functional Maps: A Flexible Representation of Maps Between Shapes*, ACM Transactions on Graphics (TOG), 31 (2012), p. 30. (Cited on pages 4, 8, 19, 20, 21, 22, 23, 25, 26, 27, 32, 33, 34, 36, 37, 40, 42, 43, 51, 69 and 71.)

[70] M. Ovsjanikov, E. Corman, M. Bronstein, E. Rodolà, M. Ben-Chen, L. Guibas, F. Chazal, and A. Bronstein, *Computing and processing correspondences with functional maps*, in ACM SIGGRAPH 2017 Courses, 2017, pp. 5:1–5:62. (Cited on pages 40 and 69.)

[71] M. Ovsjanikov, Q. Mérigot, F. Mémoli, and L. Guibas, *One Point Isometric Matching With the Heat Kernel*, 29 (2010), pp. 1555–1564. (Cited on pages 19 and 41.)

[72] M. Ovsjanikov, Q. Mérigot, V. Pătrăucean, and L. Guibas, *Shape matching via quotient spaces*, in Computer Graphics Forum, vol. 32, 2013, pp. 1–11. (Cited on pages 30 and 42.)

[73] G. Patanè, *wFEM heat kernel: Discretization and applications to shape analysis and retrieval*, Computer Aided Geometric Design, 30 (2013), pp. 276 – 295. (Cited on page 22.)

[74] J. Peraire, M. Vahdati, K. Morgan, and O. C. Zienkiewicz, *Adaptive remeshing for compressible flow computations*, Journal of computational physics, 72 (1987), pp. 449–466. (Cited on pages 3 and 7.)

[75] G. Peyré, *Toolbox sparse optimization.* https://www.mathworks.com/matlabcentral/fileexchange/16204. Accessed: 2017-10-10. (Cited on pages 48 and 53.)

[76] U. Pinkall and K. Polthier, *Computing Discrete Minimal Surfaces and their Conjugates*, Experimental mathematics, 2 (1993), pp. 15–36. (Cited on pages 16 and 34.)

[77] J. Pokrass, A. M. Bronstein, M. M. Bronstein, P. Sprechmann, and G. Sapiro, *Sparse Modeling of Intrinsic Correspondences*, in Computer Graphics Forum, vol. 32, 2013, pp. 459–468. (Cited on pages 23, 26, 27 and 32.)

[78] A. Poulenard, P. Skraba, and M. Ovsjanikov, *Topological function optimization for continuous shape matching*, in Computer Graphics Forum, vol. 37, Wiley Online Library, 2018, pp. 13–25. (Cited on page 84.)

[79] F. Remondino, *From point cloud to surface: the modeling and visualization problem*, International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences, 34 (2003). (Cited on page 13.)

[80] E. Rodolà, L. Cosmo, M. M. Bronstein, A. Torsello, and D. Cremers, *Partial Functional Correspondence*, in Computer Graphics Forum, vol. 36, 2017, pp. 222–236. (Cited on pages 23, 26, 27, 32, 42 and 71.)

[81] E. Rodolà, M. Möller, and D. Cremers, *Point-wise map recovery and refinement from functional correspondence*, in Proc. Vision, Modeling and Visualization (VMV), 2015, pp. 25–32. (Cited on pages 23, 27 and 40.)

[82] J.-M. Roufosse and M. Ovsjanikov, *Unsupervised deep learning for structured shape matching*, arXiv preprint arXiv:1812.03794, (2018). (Cited on page 84.)

[83] Y. Sahillioğlu, *A genetic isometric shape correspondence algorithm with adaptive sampling*, ACM Transactions on Graphics (TOG), 37 (2018), p. 175. (Cited on page 84.)

[84] A. Shamir, *A survey on mesh segmentation techniques*, in Computer graphics forum, vol. 27, Wiley Online Library, 2008, pp. 1539–1556. (Cited on pages 3 and 7.)

[85] A. Shtern and R. Kimmel, *Matching the LBO eigenspace of non-rigid shapes via high order statistics*, Axioms, 3 (2014), pp. 300–319. (Cited on page 41.)

[86] ——, *Spectral gradient fields embedding for nonrigid shape matching*, Computer Vision and Image Understanding, 140 (2015), pp. 21–29. (Cited on page 41.)

[87] R. K. Singh and J. S. Manhas, *Composition Operators on Function Spaces*, vol. 179, Elsevier, 1993. (Cited on pages 29, 42, 43 and 70.)

[88] J. Solomon, F. De Goes, G. Peyré, M. Cuturi, A. Butscher, A. Nguyen, T. Du, and L. Guibas, *Convolutional wasserstein distances: Efficient optimal transportation on geometric domains*, ACM Transactions on Graphics (TOG), 34 (2015), p. 66. (Cited on pages 42 and 71.)

[89] J. Solomon, A. Nguyen, A. Butscher, M. Ben-Chen, and L. Guibas, *Soft Maps Between Surfaces*, in Computer Graphics Forum, vol. 31, Wiley Online Library, 2012, pp. 1617–1626. (Cited on pages 19, 40 and 41.)

[90] J. Solomon, G. Peyré, V. G. Kim, and S. Sra, *Entropic metric alignment for correspondence problems*, ACM Transactions on Graphics (TOG), 35 (2016), p. 72. (Cited on pages 19, 20, 40, 42 and 71.)

[91] R. W. Sumner and J. Popović, *Deformation Transfer for Triangle Meshes*, ACM Transactions on Graphics (TOG), 23 (2004), pp. 399–405. (Cited on pages 13, 18, 25 and 39.)

[92] J. Sun, M. Ovsjanikov, and L. Guibas, *A Concise and Provably Informative Multi-Scale Signature Based on Heat Diffusion*, in Computer graphics forum, vol. 28, 2009, pp. 1383–1392. (Cited on pages 22 and 52.)

[93] V. Surazhsky and C. Gotsman, *Explicit surface remeshing*, in Proceedings of the 2003 Eurographics/ACM SIGGRAPH symposium on Geometry processing, Eurographics Association, 2003, pp. 20–30. (Cited on pages 3 and 7.)

[94] G. K. Tam, Z.-Q. Cheng, Y.-K. Lai, F. C. Langbein, Y. Liu, D. Marshall, R. R. Martin, X.-F. Sun, and P. L. Rosin, *Registration of 3D point clouds and meshes: a survey from rigid to nonrigid*, IEEE transactions on visualization and computer graphics, 19 (2013), pp. 1199–1217. (Cited on pages 19, 41 and 71.)

[95] G. Taubin, *A signal processing approach to fair surface design*, in Proc. CGIT, New York, NY, 1995, ACM, pp. 351–358. (Cited on page 71.)

[96] A. Tevs, M. Bokeloh, M. Wand, A. Schilling, and H.-P. Seidel, *Isometric Registration of Ambiguous and Partial Data*, in Computer Vision and Pattern

Recognition, 2009. CVPR 2009. IEEE Conference on, IEEE, 2009, pp. 1185–1192. (Cited on page 19.)

[97] F. Tombari, S. Salti, and L. Di Stefano, *Unique signatures of histograms for local surface description*, in International Conference on Computer Vision (ICCV), 2010, pp. 356–369. (Cited on page 53.)

[98] B. Vallet and B. Lévy, *Spectral geometry processing with manifold harmonics*, Computer Graphics Forum, 27 (2008), pp. 251–260. (Cited on page 71.)

[99] O. Van Kaick, A. Tagliasacchi, O. Sidi, H. Zhang, D. Cohen-Or, L. Wolf, and G. Hamarneh, *Prior knowledge for part correspondence*, in Computer Graphics Forum, vol. 30, Wiley Online Library, 2011, pp. 553–562. (Cited on page 20.)

[100] O. Van Kaick, H. Zhang, G. Hamarneh, and D. Cohen-Or, *A Survey on Shape Correspondence*, in Computer Graphics Forum, vol. 30, Wiley Online Library, 2011, pp. 1681–1707. (Cited on pages 4, 8, 20, 25, 39, 41 and 71.)

[101] M. Vestner, Z. Lähner, A. Boyarski, O. Litany, R. Slossberg, T. Remez, E. Rodola, A. Bronstein, M. Bronstein, R. Kimmel, and D. Cremers, *Efficient deformable shape correspondence via kernel matching*, in Proc. 3DV, 2017. (Cited on page 40.)

[102] 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 CVPR, 2017, pp. 6681–6690. (Cited on page 19.)

[103] C. Wang, M. M. Bronstein, A. M. Bronstein, and N. Paragios, *Discrete minimum distortion correspondence problems for non-rigid shape matching*, in International Conference on Scale Space and Variational Methods in Computer Vision, Springer, 2011, pp. 580–591. (Cited on page 19.)

[104] F. Wang, Q. Huang, M. Ovsjanikov, and L. J. Guibas, *Unsupervised multi-class joint image segmentation*, in Proc. CVPR, 2014, pp. 3142–3149. (Cited on pages 23, 40 and 44.)

[105] L. Wang, A. Gehre, M. M. Bronstein, and J. Solomon, *Kernel functional maps*, in Computer Graphics Forum, vol. 37, Wiley Online Library, 2018, pp. 27–36. (Cited on page 84.)

[106] M. Wardetzky, S. Mathur, F. Kälberer, and E. Grinspun, *Discrete laplace operators: no free lunch*, in Symposium on Geometry processing, Aire-la-Ville, Switzerland, 2007, pp. 33–37. (Cited on pages 16 and 18.)

[107] H. Zhang, A. Sheffer, D. Cohen-Or, Q. Zhou, O. Van Kaick, and A. Tagliasacchi, *Deformation-driven shape correspondence*, in Computer Graphics Forum, vol. 27, Wiley Online Library, 2008, pp. 1431–1439. (Cited on page 20.)

ÉCOLE DOCTORALE

Sciences et technologies de l'information et de la communication (STIC)

Title : Non-rigid correspondences between surfaces embedded in 3D

Keywords : Recognition, Vision, Geometry, Mesh, Optimization

Abstract :
Handling and processing the massive amount of 3D data has become a challenge with countless applications, such as computer-aided design, biomedical computing, interactive games, machine perception, robotics, etc. Geometry Processing is an area of research at the interface between algorithmics, applied mathematics and computer science related to the above applications, that exists since approximately 50 years. It is a large topic of research that includes subareas.

In this thesis we focus on the problem of shape correspondence, specifically using functional maps. The overall goal of the thesis is to show how the functional maps pipeline can be improved using functional algebra. The main contribution is to improve both the accuracy of the functional map matrix, computed using the same set of descriptors, and the accuracy of the function transfer, computed using the same functional map matrix. These improvements remain compatible with the classical linear formulation of the functional maps framework.

Titre : Correspondences non rigides entre surfaces plongées en 3D

Mots clés : Reconnaissance, Vision, Géométrie, Maillage, Optimisation

Résumé :
La manipulation et le traitement d'énormes quantités de données en 3D est devenu un défi ayant d'innombrables applications, telles que la conception assistée par ordinateur, le calcul biomédical, les jeux interactifs, la perception des machines, la robotique, etc. Le traitement de données géométrique est un sujet de recherche à l'interface entre l'algorithmique, les mathématiques appliquées et l'informatique en lien avec les applications sus-mentionnées, qui existe depuis une cinquantaine d'années. C'est un domaine de recherche vaste qui inclut des sous-domaines.

Dans cette thèse on se concentre sur le problème de correspondance de forme, spécifiquement en utilisant des correspondances fonctionnelles. L'objectif général de la thèse est de montrer comment le processus de calcul des correspondances fonctionnelles peut être amélioré en utilisant l'algèbre de fonctions. La contribution principale est l'amélioration de la précision du calcul de la matrice des correspondances fonctionnelles, calculée en utilisant le même ensemble de descripteurs, et la précision du transfert de fonctions, calculé en utilisant la même matrice de correspondances fonctionnelles. Ces améliorations restent compatibles avec la formulation linéaire classique des correspondances fonctionnelles.

Université Paris-Saclay
Espace Technologique / Immeuble Discovery
Route de l'Orme aux Merisiers RD 128 / 91190 Saint-Aubin, France