Robust representations for supervised and unsupervised 3D shape matching

Nicolas Donati

▶ **To cite this version:**

Nicolas Donati. Robust representations for supervised and unsupervised 3D shape matching. Computational Geometry [cs.CG]. Institut Polytechnique de Paris, 2023. English. NNT : 2023IPPAX004 . tel-04268738

HAL Id: tel-04268738
https://theses.hal.science/tel-04268738

Submitted on 2 Nov 2023

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.

INSTITUT POLYTECHNIQUE DE PARIS

NNT : 2023IPPAX004

Thèse de doctorat

Robust representations for supervised and unsupervised 3D shape matching

Thèse de doctorat de l'Institut Polytechnique de Paris
préparée à École polytechnique

École doctorale n°626 École doctorale de l'Institut Polytechnique de Paris (EDIPP)
Spécialité de doctorat : Informatique

Thèse présentée et soutenue à Palaiseau, le 17 Janvier 2023, par

NICOLAS DONATI

Composition du Jury :

Leif Kobbelt Professeur, RWTH Aachen University	Président
Gerard Pons-Moll Professeur, Tübingen University	Rapporteur
Julie Digne Chargée de recherche, CNRS (LIRIS)	Rapporteur
Jakob Verbeek Research Scientist, Facebook Paris	Examinateur
Maks Ovsjanikov Professeur, École polytechnique (LIX)	Directeur de thèse
Etienne Corman Chargé de recherche, CNRS (LORIA)	Co-directeur

Remerciements

Je remercie chaleureusement toutes les personnes qui m'ont aidé durant mes travaux de recherche, notamment mes directeurs de thèse Maks Ovsjanikov et Etienne Corman, pour leurs idées, leurs conseils et leur soutien durant ces trois années. J'aimerais également remercier mes relecteurs Julie Digne et Gerard Pons-Moll d'avoir eu la patience de relire minutieusement ce qui suit, ainsi que le reste de mon Jury : Leif Kobbelt et Jakob Verbeek. Merci à tous d'avoir accepté de venir m'écouter pour la soutenance. Je tiens à remercier le personnel administratif du LIX / INRIA / CNRS / DIX en particulier Evelyne Rayssac, Jessica Gameiro et Maria Sousa pour les contrats et les missions post-COVID, ainsi que pour la césure Londonienne, et Helena Kutniak sans qui cette soutenance n'aurait pas eu lieu.

Merci aux chercheurs rencontrés au cours de cette thèse et avec qui j'ai eu le plaisir de pouvoir collaborer : Simone Melzi, et Craig Lei Li. Merci aux autres thésards de l'équipe avec qui j'ai eu la chance de discuter, de partager, et même parfois de travailler. Adrien aka N. B. Cage-Villarnault, Marie-Julie (all the way from Fermat), Maud (sans qui Nico n'aurait pas de souvenir précis de ses siestes), Pauline, Thomas et Pierre (le Ping Pong bien sûr), Maxime et les appels pour rester motivés, Mariem, Souhaib et Robin pour la team COVID puis le retour prudent au laboratoire, Ramana and his love of Massy, et Souhail. Merci aux autres, rencontrés sur le chemin de la recherche scientifique, au détour d'un reading group, ou sacrifiés sur l'autel du savoir universel. Ca crée des liens, also called correspondence between 3D shapes.

Et bien sûr merci aux autres, qui m'ont soutenu en étant là pour tout le reste. A Daphné, qui m'a soutenu sans faiblir malgré le poids du bestiau, grâce à son énergie sans faille, pourtant souvent mise a rude épreuve par les joies de l'enseignement, ou du one-woman show. Best partner, best teacher, best climber. A ma famille, mom and dad pour m'avoir donné les outils pour arriver jusqu'ici, ainsi que les passions qui m'animent encore aujourd'hui. A mes frères, Antonin et Matteo et au nom de la grimpe,

des tendinites et des chaussons russes, des korrigans, des cous de taureau et de la potion magique, de sir William Goule et de sir William Peel, des combats de bâton, de Magua, des canards rituels et du reggae shark. Peut-être qu'au final, c'est moi l'gars l'plus chanceux du monde. I hope I can be Sitka, if need be. A Mimi, Mamise, Jean-Roger, Martine, Jean-Damien, et à tous les autres oncles tantes, cousins et cousines. Et aux amis, ce sera impossible de tous vous citer, mais je pense ici à vous tous, et je sais qu'en relisant ces lignes je sourirai : les Matthieu, les Baptiste, les Nico et les Sophie, Pascal, Gauthier, Guillaume, Fred, Sara, Laëtitia, Henri, FH, Marion, Héloïse, Gabrielle, Nina, Sylvain, Vianney, Ryan, Solène, Pierre, Jack, Vincent, Paul, Raphaël, also Ben and Riccardo, Theo, Michał, Andreas and Ade. Merci à tous.

Contents

Abstract

3D data analysis is a fundamental problem in modern science, and recent advances such as deep learning have opened the door to new algorithms and possibilities in this field. Nevertheless, 3D shape analysis presents difficult problems due to its particular structure. Indeed, deep neural networks and more specifically convolutional neural networks were originally tailored to tackle grid-like structures like images. Consequently, the challenge is to adapt deep learning to more complex structures like 3D point cloud or meshes. In this thesis, we focus on the problem of non-rigid 3D shape matching, whose objective is to analyze and compute maps between shapes, typically represented as triangle meshes. Correspondence between a pair of shapes can be used to transfer information such as texture, or segmentation, from one shape onto the other. To tackle this hard non-linear problem, we adopt a functional point of view allowing for a simpler and more efficient representation of maps between shapes. Since deep neural networks have been designed to produce feature functions on point clouds or meshes, for instance for segmentation, we propose to use these network features to match functional spaces using the so-called functional map framework, which we build upon to implement new algorithms for shape matching. In particular, we tackle several key problems which lie at the intersection between shape matching and deep learning. Firstly, we propose a method that helps to address the well-known issue of overfitting, which is a very recurrent problem particularly for 3D data. Secondly, we propose a new way to incorporate orientation information into the functional map pipeline using tangent vector field analysis. We use this novel representation to solve some symmetry issues, difficult to address because of the intrinsic nature of functional maps. Lastly, we propose a global solution that learns features for efficient and robust shape matching, in an unsupervised way. Overall our work proposes efficient methods to explore the space of maps between shapes by exploiting the particular structure of 3D surfaces to build robust regularizers for deep learning correspondence networks.

CHAPTER 1

Introduction

1.1 Background

Recent years have witnessed a massive surge in 3D data consumption, with applications ranging from medical imaging, autonomous driving, scene understanding, to special effects or geometry generation. Consequently the field of 3D data analysis has gained significant momentum as 3D data become more available, and algorithms achieve more efficient results.

Deep Learning, Overfitting and Regularization

One of the contributing factors to this significant rise in accuracy is deep learning. Indeed, deep learning approaches demonstrated their power and versatility on 2D images with tasks like object detection [53] semantic manipulation and editing [69], automatic generation with variational auto-encoders (VAE) [74], then generative adversarial networks (GAN) [54] and more recently diffusion-based techniques [37, 65, 120]. These recent advances made possible with deep learning in image analysis and other fields have been used in a wide range of real world problems, such as *(a)* cancer detection (e.g. [173] using images to detect tumors) *(b)* self-driving vehicles *(c)* language understanding through natural language processing (NLP) [105] and recent methods such as the attention mechanism [162] *(d)* emotion recognition [145, 165] *(e)* fraud detection [133, 160] e.g. with time series analysis and LSTMs [172] *(f)* image restoration with denoising [157] and super-resolution [171], among many others.

It is interesting to note that deep learning also comes with several limitations. As a

data-centered approach, it builds a representation of the world based on a training set and strives to make this representation generalize to a set of unseen data, denoted as test set. A typical and most important pitfall of deep learning approaches is the overfitting issue [137, 150], where the model gives good results on the train set, but fails to generalize, yielding bad results on the test set.

This leads to the second problem of deep learning pipelines: the model weights are optimized to reduce a loss adapted to the task at hand, but this optimization usually happens in a highly non-convex landscape, making it easy for the gradient descent to stop in a local minimum. Moreover, deep learning networks depend on a large number of parameters (typically ranging from 10^6 to 10^9), often more than the number of training instances (which can be addressed with strategies such as data augmentation [147]). As a consequence, the weight optimization process is immensely complex and the network itself is a black box. It is hard to give predictions or theoretical guarantees for convergence, except in simple cases. As a matter of fact, the success of deep learning is rather due to its wide range of practical performance.

Thirdly, despite its incontestable efficiency in many tasks, deep learning has been shown to lack robustness [110]. For instance, given a network trained for image classification, one can "fool" this network by adding a small noise to the input image, essentially invisible to the human eye, yet eventually resulting in the network giving a wrong prediction with strong confidence for this slightly modified image. This strategy is called adversarial attacks, and can be performed with *or without* access to the network weights [60]. This lack of robustness shows that deep learning methods, despite their undeniable success, still need strong *regularizations* to be stable for optimization and ultimately work efficiently.

Generally, a deep learning network should exploit *structure* to build a more accurate and generalizable representation. More precisely, dense neural networks connecting every input dimension to all output neurons for all layers usually have too many parameters to efficiently navigate the landscape, ultimately leading to overfitting. For 2D images, one of the most remarkable instances of such structure being essential to better performance is *convolution filters*, combined with pooling strategies [79, 148, 64]. The idea is to use the translation-invariant property of convolution filters, which is desirable for most image-related tasks, in parallel with pooling layers, whose goal is to reduce the image size, thus letting the network progressively focus on more global information, building features in a hierarchical manner. Indeed, visualizing the convolution filters obtained by such a network can shed light on how meaningful and generalizable the trained representations for images are: typically, some filters in the early layers focus on edge detection, whereas deeper in the network filters focus on more high-level structure such as texture or detection of certain objects. Meanwhile, a convolutional neural network (CNN) has considerably fewer parameters, allowing for easier optimization.

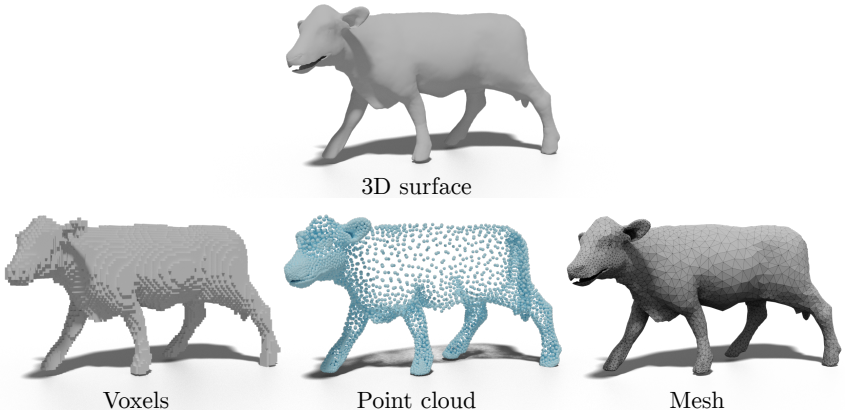


Figure 1.1: For a smooth surface in 3D *(top row)*, there are different possible discretizations *(bottom row)*: *(a)* The voxel grid representation *(b)* The point cloud structure *(c)* The triangle mesh representation, which corresponds to a piecewise-linear encoding of a surface.

3D Deep Learning

While deep learning has also been applied on 3D data, its use is still not as widespread. Firstly 3D data acquisition is generally more difficult and costly than gathering images. Some popular methods are 3D laser scans [43] which enable precise localization of points on 3D objects facing the scanner, Microsoft Kinnect [63] which can acquire precise depth fields, or Lidar [130] which is used mostly for terrain detection and thus autonomous driving. These techniques usually need post-processing, e.g. point cloud registration to match the acquisition under different viewpoints [81, 12], or denoising to clean the acquired data from outliers [118, 142, 119]. Since the process of getting clean, high-quality 3D data necessitates precise instruments and is generally costly, the amount of 3D data available for large scale experiments is comparatively low. Indeed, any smartphone can generate high-quality 2D images or videos, hence the availability of large-scale image datasets with several billion training samples in some cases.

Secondly, 3D data cannot be treated similarly to 2D images, as there are various types of 3D data, the most commonly used being voxels, point clouds and triangle meshes (see Figure 1.1). All these types of data still differ from their 2D counterpart, owing to either size or non-regular structure: *(a)* voxels extend the regular grid structure to 3D (with often just one binary channel to indicate presence or absence of an object on a given voxel) but usually lack either precision, since even a 256^3-grid takes has as many elements as a 4096^2 image. Additionally, voxels inside the surface bring little to no information, which can make this data structure less competitive compared to sparser structures such as these described below. *(b)* point clouds usually stem from 3D laser

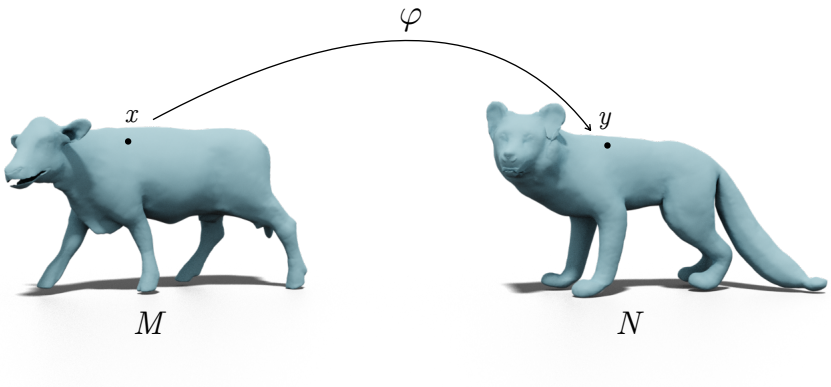


Figure 1.2: A map φ between two smooth shapes M and N. The map takes a point $x \in M$ to $y = \varphi(x) \in N$.

scan acquisition, and are much lighter structures as they only store point positions on the object's *surface*. For reasonable precision, a point cloud (for an organic object such as an animal) only needs to store a few thousand points, but they can be composed of millions of points for high-precision objects or when a whole 3D scene is encoded. However, the point cloud structure is *not regular*, as its local density may vary. Besides, it is difficult to define differential operators and smooth objects on this structure. *(c)* meshes are theoretically more adapted to deal with the issue previously stated, as they are tailored to approximate *smooth surfaces* with piecewise linear elements. The theory of manifolds can be partially adapted, giving access to so-called *intrinsic* differential operators on the mesh structure. These operators will not depend on the *extrinsic* coordinates of the points, making the intrinsic viewpoint independent on translation or rotation of the mesh.

Consequently, for a neural network to exploit these different 3D data structures, one must design specific tools and regularizations, adapted to the problem at hand. This research field is generally called *geometric deep learning* [23] or *3D deep learning*.

Some of the most common tasks in this field are: *(a)* Shape classification and segmentation, with networks such as [115, 116, 156, 114] on point clouds, and [92, 112, 144] on meshes *(b)* Point cloud denoising [118] and meshing [142, 119] *(c)* Learning implicit representations for efficient shape parametrization [155, 100] *(d)* Shape matching [85, 62, 131, 39, 45, 153], potentially with partial correspondence [4] *(e)* Shape interpolation [117, 46].

Shape Matching

In this dissertation, we will mainly focus on one particular shape analysis task, called *shape matching*, whose goal is to analyze and compute maps between shapes. A map between two shapes is represented in Figure 1.2. Given such a map, also called a correspondence between one source and target shape, it is possible to transfer information (e.g. weights, texture, segmentation) from source to target. Essentially, shape matching often relies on differential geometry as shapes are represented with manifolds, as described in Chapter 3. In the discrete case, we typically approximate smooth surfaces as meshes with vertices embedded in 3D space. Note that throughout this manuscript, we focus on the study of surfaces in 3D, but the constructions and considerations developed in subsequent chapters theoretically apply in larger dimensions.

Exploring the space of maps between shapes is challenging, both in continuous and discrete settings. Indeed, the challenge is to look for *natural* correspondences between shapes, which entails enforcing the resulting mappings to have desirable properties, such as bijectivity, continuity, or most importantly *isometry*. While these properties can be expressed as point-wise energies on discretized shapes, both their computation and optimization tend to be time-consuming and non-trivial.

To alleviate this issue, we focus on a particular framework to tackle shape matching, called *functional maps*, introduced in [106]. The key insight proposed by this pipeline holds in the representation of maps between shapes as *small-scale matrices*, on which aforementioned properties like bijectivity and isometry can be expressed as simple *quadratic energies*.

One of the main bottlenecks of the functional map pipeline lies in its dependence on so-called *descriptor functions*, whose purpose is to act as soft constraints to guide the functional map estimation. While some works design descriptor functions in a fully automatic, learning-free way [151, 6, 158], the rise of feature functions from deep learning [115, 116, 156, 144] makes it possible to learn these descriptor functions [85], making shape matching another task on which deep learning can be applied with significant efficiency. Indeed, geometric priors are typically key to generating robust descriptor functions .

However, as mentioned above, deep learning methods are liable to overfit if not combined with the right regularizers or network structures. Besides, this overfitting can happen in surprising places and may therefore remain undetected if not confronted with a precise and thorough experimentation procedure. Consequently, we present in this thesis several works whose overall purpose is to strengthen the general shape matching pipeline. More specifically, we make use of the functional maps framework jointly with 3D deep learning for shape matching. We present new techniques exploiting shape structure or general network optimizations to prevent overfitting, and eventually give insight on how to make this pipeline completely unsupervised.

We believe our work helps make some key shape matching methods more stable, by

opening new possibilities combining both geometry processing and 3D deep learning.

1.2 Contributions

This thesis aims at solving several issues inherent to the functional map pipeline, by incorporating relevant structure into the framework, whether or not it is learning-based. We first investigate how to learn robust features *directly from the raw geometry* of the shapes, rather than using potentially error-prone axiomatically-defined input feature functions. Secondly, we present an extension to the functional map pipeline to address the fundamental issue of intrinsic *orientation-reversing symmetries*. Finally, we present a way to use *unsupervised* deep learning to perform non-rigid shape matching using regularizations derived from the first two parts.

Robust Feature Learning from Raw Geometry

As mentioned above, in shape matching, representing maps can be made considerably easier with the functional maps framework. However, map computation still hinges on finding accurate *descriptor functions* to guide the initialization. Indeed, in the original paper [106] and in follow-up works [107, 31, 32, 103, 104, 113, 124], authors rely on axiomatic descriptors [151, 6] based on heat diffusion. However, manual annotation is costly and time consuming, hence the need for algorithms capable of producing their own descriptor functions without user intervention.

Geometric deep learning algorithms are built to learn their own representation for the input data, through feature function design. While some works aim at adapting convolutional neural networks on surfaces discretized as meshes [92, 21, 112], other methods, pioneered by PointNet [115] and later extended in works such as [116, 5, 156, 114], build feature functions directly on 3D point clouds. Meanwhile, deep learning techniques are applied jointly with functional maps in FMNet [85] to *learn* better descriptor functions. However, FMNet uses as input not the shape itself, but SHOT descriptor functions [158]. These descriptors, based on normal distribution histograms for each vertex, are highly dependent on the shapes discretization. Consequently, FMNet and its follow-up works [62, 132] are liable to overfit to shape discretization rather than exploit relevant geometric details from the input surfaces.

With this in mind, we propose to use a point cloud feature extractor [156] to extract information directly from the shape's geometry, jointly with a differentiable regularizer in the functional map layer to extract robust feature functions tailored to retrieve accurate correspondence. We show through extensive experiments that our method, although supervised, can learn with a relatively small training set, and is stable to changes in triangulation from train to test set since the features are built with a point cloud feature extractor. This work constitutes the first building block of this thesis, where we aim at

designing state-of-the art algorithms for shape matching, while circumventing common pitfalls such as overfitting to the triangulation structure.

Incorporating Orientation into Functional Maps

A second bottleneck of the functional map pipeline is inherent to the fact it is built on *intrinsic quantities*, namely Laplace-Beltrami eigenvalues and eigenbasis (which we will introduce in Chapter 3). Indeed, these objects are built on a continuous surface using only the metric information (which on a discrete mesh corresponds to using only edge length information). Intrinsic pipelines benefit from the fact that they do not use the *extrinsic* 3D coordinates of the input shape, making them independent to rotation or translation of the input.

However, intrinsic pipelines are more affected by the intrinsic self-symmetries of the input shape. As most organic shapes exhibit at least one intrinsic near-isometric symmetry (often referred to in the following as left-right symmetry), this becomes a standard and well-known issue in shape matching. Indeed, for shapes exhibiting these isometric self-maps, the problem of finding the most isometric map becomes ill-defined as the set of solutions includes a non-trivial symmetry group [107]. To tackle this issue one can rely on *some* extrinsic information, for instance by using partially extrinsic descriptor functions as input, such as SHOT descriptors [158]. However, as explained in the previous paragraph, SHOT descriptors are unreliable as they depend more on the discretization than on the underlying surface.

We show that incorporating *orientation information* instead in the functional map pipeline considerably helps in disambiguating symmetries. We build a new spectral representation for push-forwards, which enables to represent conformal orientation-preserving maps as small complex matrices, that we call complex functional maps. We also build and describe the link between functional maps and complex functional maps.

Finally, we show in several experiments that state-of-the-art algorithms of the functional map pipeline all benefit from our modification, and that generally our framework helps with orientation-reversing symmetry aliasing. This work thus paves the way to more robust shape matching and new representations for maps between shapes, that we propose to use for unsupervised efficient shape matching in the last part of this thesis.

Unsupervised Deep Learning for Shape Matching

Chapter 5 and 6 respectively introduce a robust framework for shape matching with deep learning directly from the geometry, and a new representation for orientation-preserving maps between shapes to tackle symmetry issues. In the last work of this thesis, we combine the two previous ideas to build an *unsupervised deep neural network for 3D correspondence*. Indeed, supervised deep learning methods for shape matching require ground-truth maps between all pairs of shapes at training time. While this is typically

feasible for e.g. synthetic datasets, it limits the training scenarios and ultimately makes it harder for the method to generalize.

Consequently, deep unsupervised shape matching has been an active field, initially developed in FMNet follow-up works [62, 132]. In both these papers however, the input signal fed to the network is SHOT descriptors. We show that the use of SHOT descriptors circumvents the problem of symmetry introduced in the previous section through triangulation overfitting. Indeed, in non-rigid shape matching the objects of study are most of the time organic shapes, which exhibit a so-called *left-right symmetry*, which reverses shape orientation. Theoretically, an unsupervised shape matching method must be able to disambiguate between the two near-isometric map solutions that are the ground-truth map and its composition with an intrinsic left-right symmetry. Some works [140, 46] choose to rely directly on shape embedding to solve this issue: the network is fed the 3D coordinate signal of both shapes as input. However, this solution requires all the train and test shapes to be rigidly aligned to the same 3 axes, so that the left and right part of the shapes get separate input signals. These works fall in the category of weak supervision, since they require specific alignments both at training and test times. Finally, for a method to be fully unsupervised in our case, the network loss must be able to rule out undesirable symmetric maps without relying on weak supervision or triangulation overfitting. While Deep Shells [45] proposes such a loss, their structure is still very dependent on having SHOT descriptors as input, which once again proves unstable to triangulation change, or even to some datasets as we describe in Chapter 6.

Our work describes a robust and efficient pipeline for unsupervised shape matching, still based on a functional map framework, through a loss mixing functional maps and *complex functional maps* to tackle the symmetry issue. This network is based on the triangulation-agnostic 3D feature extractor DiffusionNet [144], making it robust to changes in discretization. Additionally, our method makes use of *tangent vector field features* which are used as relevant signals for correspondence. Consequently, we believe this work can open new possibilities both for shape matching and shape representation. Indeed, the feature functions and vector fields computed by our network in an unsupervised manner could be used for a variety of other shape analysis tasks.

1.3 List of Publications

This dissertation is based on the following list of publications:

- N. Donati, A. Sharma and M. Ovsjanikov, *Deep Geometric Functional Maps: Robust Feature Learning for Shape Correspondence*, in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020. [39]

- N. DONATI, E.CORMAN, S.MELZI AND M. OVSJANIKOV, *Complex Functional Maps: a Conformal Link between Tangent Bundles*, in Proceedings of the Eurographics Symposium on Geometry Processing, CGF 2022. [40]

- N. DONATI, E. CORMAN AND M. OVSJANIKOV, *Deep Orientation-Aware Functional Maps: Tackling Symmetry Issues in Shape Matching*, in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022. [41]

The following work has also been developed during my PhD and is closely related to the topics mentioned above. However, it is not included in this dissertation:

- L. LI, N. DONATI AND M. OVSJANIKOV, *Learning Multi-resolution Functional Maps with Spectral Attention for Robust Shape Matching*, in NeurIPS 2022. [82]

CHAPITRE 2

Introduction en Français

2.1 Contexte

Ces dernières années ont vu une augmentation massive de la consommation de données 3D, avec des applications allant de la compréhension de scènes 3D et la reconstruction de scans 3D partiels (utilisés par exemple pour la conduite autonome, ou l'imagerie médicale) aux effets spéciaux ou à la génération de géométrie. Par conséquent, le secteur de l'analyse de données 3D a fortement gagné en importance, les données 3D devenant plus accessibles et les algorithmes permettant d'obtenir des résultats plus efficaces.

Apprentissage Profond, Sur-Apprentissage et Regularisation

L'un des facteurs principaux de cette augmentation conséquente en précision est le développement des algorithmes d'apprentissage profond. En effet, cette technique a démontré sa puissance et son adaptabilité sur les images 2D, avec des architectures de modèles performantes et fiables sur des tâches complexes telles que la détection d'objets [53], la manipulation sémantique [69], la génération automatique d'images avec d'une part les auto-encodeurs variationnels (VAE) [74], puis les "generative adversarial networks" (GAN) [54] puis plus récemment, les méthodes basées sur de la diffusion [37, 65, 120]. Ces avancées récentes permises par l'apprentissage profond en matière d'analyse d'images ainsi que dans d'autres domaines ont été largement utilisées pour résoudre des problèmes réels tels que *(a)* la détection automatique de tumeurs cancéreuses [173] *(b)* les voitures autonomes *(c)* l'analyse de langage (NLP) [105] avec des mécanismes tels que l'attention [162] *(d)* la reconnaissance d'émotions [145, 165] *(e)*

la détection de fraudes ou d'incohérences fiscales [133, 160] avec l'analyse de séries temporelles, par exemple avec les LSTMs [172] *(f)* la restauration d'images avec des techniques telles que le débruitage [157] ou la sur-résolution [171], parmi bien d'autres applications.

Il est intéressant de noter que l'apprentissage profond s'accompagne également de plusieurs inconvénients. En tant qu'approche centrée sur les données, elle construit une représentation du monde basée sur un ensemble d'entraînement et tente de généraliser cette représentation à des données inconnues (sur lesquelles elle n'a pas été entraînée) appelées ensemble d'évaluation. Un risque typique et important des approches d'apprentissage profond est le phénomène de sur-apprentissage [137, 150], où le modèle donne de bons résultats sur l'ensemble d'entraînement, mais ne parvient pas à généraliser, donnant ainsi de mauvais résultats sur l'ensemble d'évaluation.

Ceci mène à une autre limite de l'apprentissage profond : les poids du modèle sont optimisés afin réduire une *fonction de coût* adaptée aux besoins du problème, mais cette optimisation est souvent effectuée dans un cadre fortement non-convexe, ce qui bloque la descente de gradient dans des minima locaux. De plus, les réseaux de neurones profonds possèdent un très grand nombre de paramètres (typiquement de 10^6 à 10^6), souvent supérieur au nombre de données d'entraînement (ce qui peut être contrecarré par des procédés comme l'augmentation de données [147]). L'optimisation des poids est ainsi extrêmement complexe ; par essence, le réseau lui-même est une boîte noire et il est difficile de donner des prédictions ou des garanties de convergence, excepté dans les cas les plus simples. Le succès de l'apprentissage profond est plutôt dû à son succès dans une très grande variété d'applications, comme cité plus haut.

Par ailleurs, malgré son efficacité incontestable démontrée dans de nombreuses tâches, il a été démontré que l'apprentissage profond manquait de robustesse [110]. Par exemple, en utilisant un réseau entraîné pour la classification d'images, il est possible de "tromper" le réseau en bruitant artificiellement l'image d'entrée, de façon invisible pour l'œil humain, ce qui résulte en une prédiction erronée du réseau avec un fort degré de certitude, pour cette image très légèrement modifiée. Cette stratégie dite des "adversarial attacks" peut être utilisée en ayant accès ou non aux poids du réseau [60]. Ce manque de robustesse montre que les méthodes d'apprentissage profond, malgré leur taux d'efficacité indéniable, ont encore besoin de fortes régularisations afin d'être suffisamment stables durant l'optimisation, et correctes pendant l'évaluation.

De façon générale, un réseau de neurones profond doit pouvoir s'appuyer sur une notion de structure afin de créer une représentation plus stable et non sur-apprise des données. Plus précisément, les réseaux de neurones denses connectant chaque dimension d'entrée à chaque dimension de sortie contiennent le plus souvent trop de paramètres, rendant l'exploration de l'espace des paramètres trop complexe, ce qui conduit à du sur-apprentissage. Dans le cas des images 2D, un exemple fondamental d'utilisation de structure est celui des filtres de convolution, combinés à des stratégies d'échantillon-

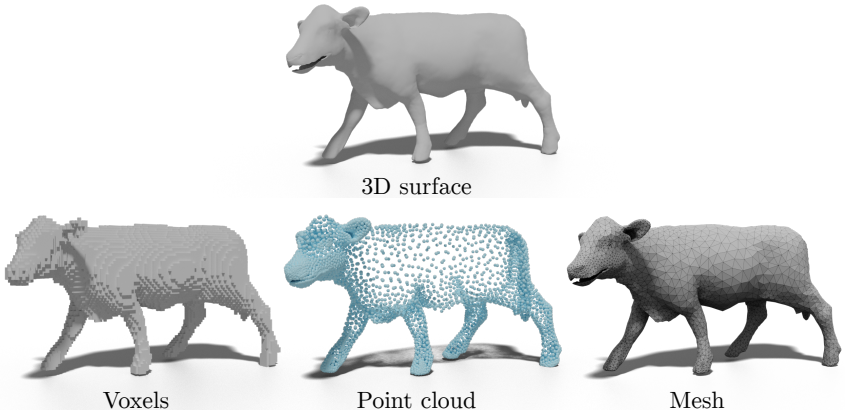


FIGURE 2.1 : Pour une surface 3D donnée *(première ligne)*, il existe plusieurs discretisations possibles *(deuxième ligne)* : *(a)* La grille de voxels *(b)* Le nuage de points *(c)* Le maillage, qui correspond à une représentation linéaire par morceaux de la surface.

nage, ou "pooling" [79, 148, 64]. L'idée principale est d'utiliser la propriété d'invariance par translation des filtres de convolution qui permet de détecter de la même façon un objet sans tenir compte de son emplacement dans l'image, ce qui est généralement utile dans la plupart des tâches liées aux images. En parallèle les couches de "pooling" permettent de réduire l'échelle de l'image afin de laisser le réseau se concentrer progressivement sur des informations de plus en plus globales, construisant ainsi des représentations de manière hiérarchisée. En effet, en visualisant les filtres de convolution obtenus par un réseau entraîné, on observe que l'optimisation a permis de construire des représentations efficaces et généralisables dans le cadres des images : les filtres sur les premières couches du réseau se concentrent sur des notions extrêmement locales telles que la détection de bords, tandis que des couches plus profondes détectent des structures plus globales telles que des textures ou certains objets. Les réseaux de neurones basés sur ces stratégies de filtres de convolution et d'échantillonnage, appelés réseaux de neurones convolutifs (CNN), présentent considérablement moins de paramètres que les réseaux denses, permettant une optimisation plus efficace.

Apprentissage Profond en 3D

Même si l'apprentissage profond a également été utilisé sur des données 3D, ses applications restent à ce jour moins étendues. Premièrement, les méthodes d'acquisition 3D sont généralement plus difficiles et coûteuses à mettre en place que la capture d'images 2D. Les méthodes d'acquisition 3D les plus populaires sont les scans 3D par laser [43] qui permettent la localisation précise de points sur des objets 3D directement en face du

scanner, la Microsoft Kinnect [63] qui permet l'acquisition précise du champ de profondeur, ou encore le Lidar [130] utilisé principalement pour de la détection de reliefs de terrain, puis récemment pour les voitures autonomes. Ces techniques nécessitent habituellement un traitement post-acquisition, tel que l'alignement de nuages de points partiels pris sous des angles différents [81, 12], ou le débruitage pour traiter les données acquises et se débarrasser des "outliers" [118, 142, 119]. Le processus d'obtention de données 3D de qualité propres à l'utilisation nécessitant des instruments précis et coûteux, il est naturel que la quantité de données 3D disponibles pour des expériences à grande échelle soit encore relativement petite comparé au cas de la 2D, où n'importe quel smartphone peut capturer des images ou des vidéos de très bonne qualité en une fraction de seconde.

Deuxièmement, les données 3D peuvent être de différents types (voire Figure 2.1), chacun nécessitant un traitement particulier, puisqu'ils diffèrent des structures régulières de grilles de pixel 2D. Les types de donnée 3D les plus fréquents sont : *(a)* La structure de voxels qui étendent la structure de grille régulière à la troisième dimension (avec le plus souvent un seul canal de donnée binaire indiquant la probabilité de présence d'un objet sur un voxel donné). Le problème majeur de ce type de donnée est qu'il manque souvent de précision : en effet, une grille de format 256^3 pèse autant qu'une image de résolution 4096^2. Additionnellement, les voxels intérieurs à la surface de l'objet d'étude n'apportent qu'une information redondante, ce qui rend parfois les grilles de voxels moins compétitives comparées à des structures plus légères, ou "sparse", comme celles décrites dans les points suivants. *(b)* Les nuages de points, souvent générés lors d'une acquisition scanner, représentent des structures plus légères, puisque sont seulement stockées les positions de points à la *surface* de l'objet d'étude. Pour une précision raisonnable, un nuage de points (dans le cas des objets de type organique) ne demandent que quelques milliers de points, et ce nombre peut monter à quelques millions de points pour des scènes 3D massives représentant par exemple des environnements complexes. Cependant, la structure de nuage de points est *non régulière*, et sa densité locale est souvent variable. Il est alors difficile d'y établir un cadre d'étude pour définir des opérateurs différentiels ou des objets venant de la théorie des surfaces continues. *(c)* Les maillages sont construits pour répondre au problème susmentionné, et sont définis en tant qu'approximation discrète, et linéaire par morceaux, des *surfaces continues*. La théorie des variétés différentielle peut alors partiellement être adaptée, donnant ainsi accès à l'information appelée *intrinsèque* des surfaces, et à certains opérateurs différentiels associés, qui se révèlent très utiles à l'étude des maillages. En effet, ces opérateurs ne dépendent pas des coordonnées 3D, dites *extrinsèques*, des points du maillage, rendant le point de vue intrinsèque indépendant aux translations et aux rotations de la surface considérée.

En conséquence, pour qu'un réseau de neurone puisse utiliser ces types de données non réguliers, des outils spécifiques et des régularisations adaptées doivent être conçus pour le problème d'étude spécifique. Ce domaine de recherche très actif est générale-

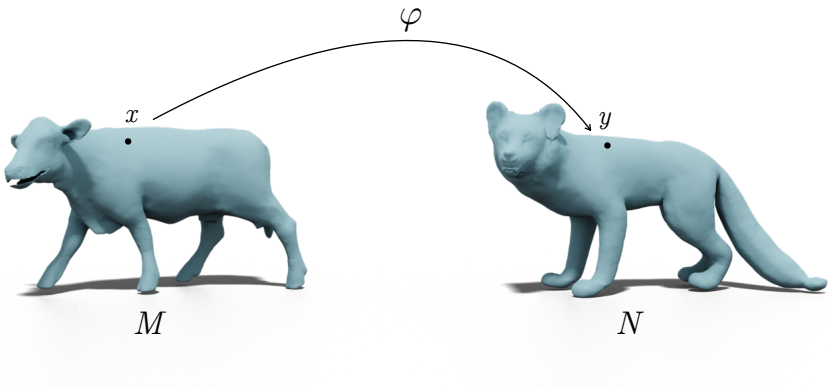


FIGURE 2.2 : Une application φ entre deux formes 3D M and N. L'application associe à un point $x \in M$ le point $y = \varphi(x) \in N$.

ment appelé *apprentissage profond géométrique* [23] ou *apprentissage profond 3D*.

Les tâches et enjeux majeurs de ce domaine sont : *(a)* La classification et la segmentation de formes, avec des réseaux comme [115, 116, 156, 114] pour les nuages de points, et [92, 112, 144] pour les maillages *(b)* Le débruitage de nuages de points [118] ainsi que leur maillage de façon automatique [142, 119] *(c)* L'apprentissage de représentations implicites pour la paramétrisation efficace des formes 3D [155, 100] *(d)* La mise en correspondance de formes 3D [85, 62, 131, 39, 45, 153], avec le cas particulier, plus difficile, où la correspondance est partielle [4] *(e)* L'interpolation entre formes 3D [117, 46].

Mise en correspondance de formes 3D

Dans notre cas, nous nous intéresserons majoritairement dans le cadre de cette dissertation à la *mise en correspondance de formes 3D*, dont l'objectif principal est l'étude et l'analyse des applications entre formes 3D. Une telle application est représentée dans la Figure 2.2. Étant donnée une telle application, aussi appelée correspondance, entre une forme source et une forme cible, il est ensuite possible de transférer de l'information (comme des poids, de la texture ou des informations de segmentation sémantiques) depuis la forme source jusqu'à la forme cible. Essentiellement, ce domaine s'appuie sur la géométrie différentielle et les formes sont représentées par des variétés, comme décrit dans le chapitre 3 en plus de détails. Dans le cas discret, nous approximons les surfaces 3D à l'aide de maillages dont les points sont des éléments de l'espace 3D. À ce stade, nous attirons l'attention du lecteur sur le fait que dans cette dissertation, nous nous concentrons sur l'étude des surfaces 3D, mais les constructions théoriques et les considérations établies au cours de cette thèse restent a priori valables en dimension

supérieure.

L'exploration de l'espace des applications entre formes est extrêmement complexe, à la fois dans le cadre continu et discret. En effet, le problème principal est d'obtenir des correspondances *naturelles* entre formes, ce qui implique d'être capable d'imposer aux applications d'exhiber des propriétés telles que bijectivité, continuité ou de façon plus importante, des propriétés d'isométrie. Ces propriétés peuvent être exprimées et dérivées en énergie sur les maillages 3D, mais à la fois leur estimation et leur optimisation ont tendance à être chronophages et non triviales.

Pour alléger ce problème, nous utilisons le cadre des *applications fonctionnelles*, introduit pour la première fois dans [106]. L'apport principal de cette pipeline tient dans une représentation compactée des applications entre formes, encodée par des *matrices à petite échelle*, sur lesquelles les propriétés susmentionnées (et plus particulièrement la bijectivité et l'isométrie) peuvent s'exprimer comme de simples énergies quadratiques.

En contrepartie, les applications fonctionnelle et leur performance dépendent fortement de certaines données d'entrées, appelées fonctions descriptrices, ou simplement *descripteurs*, qui sont des signaux pre-calculés sur les formes source et cible dont l'objectif est de servir de points de repère pour guider l'estimation d'une application fonctionnelle initiale. Certains travaux proposent des formules pour des descripteurs basées sur des approches purement axiomatiques [151, 6, 158], tandis que des approches plus récentes démontrent l'efficacité de l'apprentissage profond pour *apprendre* ces descripteurs [85] à l'aide de réseaux spécialisés pour les données 3D [115, 116, 156, 144]. Ainsi, la mise en correspondance de formes 3D devient un nouveau problème géométrique sur lequel il est possible d'utiliser de l'apprentissage de façon efficace, puisque les connaissances géométriques obtenues lors de l'entraînement sont pertinentes pour l'estimation de descripteurs généralisables.

Cependant, comme expliqué plus tôt dans cette introduction, les méthodes d'apprentissage profond sont promptes au sur-apprentissage si elles ne sont pas combinées avec les régularisateurs ou les structures de réseau adéquats. Par ailleurs, cette situation peut se présenter inopinément et risque ainsi d'échapper à la détection en l'absence de confrontation avec une procédure d'expérimentation précise et détaillée. Ainsi, nous présentons dans cette thèse plusieurs travaux dont l'objectif global est de renforcer la pipeline générale de correspondance entre formes. Plus précisément, nous utilisons le cadre des applications fonctionnelles associé à l'apprentissage profond en géométrie, avec de nouvelles techniques exploitant la structure des formes 3D ou des optimisations générales du réseau, afin d'éviter le sur-apprentissage, et avec pour objectif final de proposer des solutions visant à rendre la pipeline totalement non-supervisée.

Nous sommes persuadés que notre travail permet de rendre les méthodes de correspondance de formes plus stables, en ouvrant de nouvelles perspectives associant à la fois le calcul en géométrie et l'apprentissage profond de formes 3D.

2.2 Contributions

Cette thèse a pour objectif de résoudre certains des problèmes liés à la pipeline des applications fonctionnelles, en y incorporant des informations pertinentes de structure, à la fois dans la partie de cette pipeline liée à l'apprentissage mais aussi dans ses composantes plus axiomatiques. Nous investiguons dans un premier temps comment apprendre des descripteurs robustes *directement depuis la géométrie "brute"* des formes, plutôt qu'en raffinant des descripteurs potentiellement imprécis ou biaisés. Dans un deuxième temps, nous présentons une extension au cadre des applications fonctionnelles, afin de traiter le problème majeur constitué par les *symétries intrinsèques internes inversant l'orientation* dans la mise en correspondance. Enfin, nous présentons une méthode permettant d'apprendre de manière *non supervisée* des correspondances 3D via de nouvelles régularisations, dérivées des deux parties précédentes.

Correspondances via Apprentissage Supervisé Robuste sur la géométrie "brute"

Dans la correspondance de formes, la représentation d'application est rendue considérablement plus facile avec le cadre des représentations fonctionnelles. Cependant, le calcul de fonctions dépend encore de *descripteurs* précis pour guider l'initialisation. En effet, dans la publication première [106], et dans les travaux qui ont suivi [107, 31, 32, 103, 104, 113, 124]les auteurs s'appuient sur des fonctions repères ainsi que des descripteurs entièrement automatiques [151, 6] s'appuyant sur le processus de diffusion de chaleur. Cependant, l'annotation manuelle est coûteuse et chronophage, d'où le besoin d'algorithmes capables de produire leurs propres fonctions descriptives sans intervention de l'utilisateur.

Les algorithmes d'apprentissage profond géométrique sont construits pour apprendre leur propre représentation des données d'entrée, via l'estimation de descripteurs. Alors que certaines méthodes visent à adapter les réseaux de neurones convolutifs aux surfaces discrétisées en maillage [92, 21, 112], d'autres, présentées pour la première fois par PointNet [115] puis étendues plus tard dans des travaux tels que[116, 5, 156, 114], construisent les descripteurs directement sur des nuages de points 3D. Cependant, les techniques d'apprentissage profond sont combinées aux applications fonctionnelles dans FMNet [85] pour *apprendre* de meilleurs descripteurs. Cependant, FMNet reçoit comme données d'entrées non pas la forme elle-même, mais des descripteurs SHOT [158]. Ces descripteurs, qui s'appuient sur des histogrammes de distribution de normales pour chaque point, sont fortement dépendants de la discrétisation des formes. Ainsi, FMNet et les travaux qui ont suivi [62, 132] sont prompts au sur-apprentissage de la discrétisation de la forme plutôt qu'à l'exploitation de détails géométriques pertinents des surfaces d'entrées.

C'est pourquoi nous proposons d'utiliser un extracteur de fonctions caractéristiques de nuages de points [156] pour extraire de l'information directement depuis la géométrie de la forme, en parallèle d'un régularisateur différenciable dans la couche d'application fonctionnelle afin d'extraire des descripteurs adaptés pour obtenir une correspondance plus précise. Nous montrons donc via des expériences exhaustives que notre méthode, bien que supervisée, peut apprendre avec un ensemble d'entraînement relativement petit, et est résistante au changement de triangulation entre l'entraînement et le test, puisque les descripteurs sont construits avec un extracteur opérant sur un nuage de points. Ce travail est présenté dans le premier axe de cette thèse, dans laquelle nous visons à créer des algorithmes performants et novateurs pour la mise en correspondance de formes, en contournant les écueils classiques tels que le sur-apprentissage de triangulation.

Orientation et Applications Fonctionnelles

Une autre limite de la pipeline de l'application fonctionnelle réside dans le fait qu'elle est construite sur des *quantités intrinsèques*, les valeurs propres et bases propres de Laplace-Beltrami (que nous présenterons dans le chapitre 3). En effet, ces objets sont construits sur une surface continue utilisant uniquement les informations métriques (ce qui dans un maillage discret correspond à utiliser uniquement les informations de longueur des arêtes). Les pipelines intrinsèques bénéficient du fait qu'elles n'utilisent pas les coordonnées 3D *extrinsèques* de la forme d'entrée, les rendant indépendantes de la rotation ou la translation des données d'entrée.

Cependant, les pipelines intrinsèques sont nécessairement davantage affectées par les symétries internes de la forme d'entrée. Comme la plupart des formes organiques présentent au moins une symétrie quasi-isométrique intrinsèque (à laquelle nous ferons souvent référence dans ce qui suit en tant que symétrie gauche-droite), cela devient un problème standard et connu de la reconnaissance de formes. En effet, pour les formes qui présentent ces symétries internes isométriques, le problème de trouver la correspondance la plus isométrique devient mal défini, puisque l'ensemble de solutions inclue un groupe de symétrie non trivial [107]. Pour résoudre ce problème, il est possible de s'appuyer sur de l'information extrinsèque, par exemple en utilisant des descripteurs partiellement extrinsèques en donnée d'entrée, tels que SHOT [158], qui indique une distribution discrète des normales, et est par conséquence aussi indépendante de la rotation ou la translation de l'image d'entrée. Cependant, comme expliqué dans le paragraphe précédent, les descripteurs SHOT ne sont pas fiables puisqu'ils dépendent davantage de la discrétisation que de la surface sous-jacente.

Nous montrons qu'incorporer plutôt de l'*information d'orientation* dans la pipeline d'application fonctionnelle améliore considérablement la désambiguïsation des symétries. Nous construisons une nouvelle représentation spectrale de l'application différentielle, qui permet de représenter des applications conformes préservant l'orientation

comme des matrices complexes de petite taille, que nous appelons les applications fonctionnelles complexes. Nous construisons et décrivons également le lien entre les applications fonctionnelles et les applications fonctionnelles complexes.

Enfin, nous montrons dans plusieurs expériences que les algorithmes de l'état de l'art de la pipeline de l'application fonctionnelle bénéficient tous de notre modification, et que généralement notre cadre aide à lutter contre l'aliasing de la symétrie inversant l'orientation. Cette approche ouvre ainsi la voie à des algorithmes de correspondance de formes robustes et à de nouvelles représentations des applications entre formes, que nous proposons d'utiliser pour une correspondance non-supervisée efficace dans la dernière partie de cette thèse.

Correspondances via Apprentissage Non Supervisé et Vecteurs Tangents

Les deux derniers chapitres présentent chacun un cadre robuste pour les correspondances entre formes avec l'apprentissage profond, directement à partir de la géométrie, et une nouvelle représentation des applications entre formes préservant l'orientation afin de résoudre les problèmes de symétrie. Dans cette dernière partie, nous combinons les deux idées précédentes pour construire un *réseau de neurones profonds non supervisé pour les correspondances 3D*. En effet, ce type de méthodes d'apprentissage supervisé nécessite des correspondances entre chaque paire de formes à l'étape de l'entraînement. Même si c'est le cas pour les ensembles de données synthétiques par exemple, cela limite les scénarios d'entraînement et rend finalement plus difficile la généralisation.

Ainsi, l'étude de la correspondance de formes par apprentissage profond non supervisé est devenue un domaine actif, initialement développé par les travaux qui ont suivi FMNet [62, 132]. Dans ces deux publications cependant, le signal d'entrée donné au réseau est un ensemble de descripteurs SHOT qui, comme décrit dans les sections précédentes, est à a fois instable et sujet au sur-apprentissage. En effet, nous prouvons ici que l'utilisation des descripteurs SHOT contourne le problème de symétrie présenté dans la première partie via le sur-apprentissage sur la triangulation. En effet, dans la correspondance de formes non rigides les objets d'étude sont la plupart du temps des formes organiques, qui présentent une symétrie communément appelée *symétrie gauche-droite*, qui inverse l'orientation de la forme. Théoriquement, une méthode de correspondance de formes non supervisée doit pouvoir faire la différence entre les deux solutions les plus proches de l'isométrie que sont la vérité-terrain et sa composition, et une géométrie gauche-droite intrinsèque. Certains travaux [140, 46] choisissent d'utiliser l'espace ambiant de la forme pour résoudre ce problème : on fournit au réseau le signal des coordonnées 3D de chaque forme comme donnée d'entrée. Cependant, cette solution nécessite que toutes les formes d'entraînement et de test soient alignées de façon rigide à ces trois mêmes axes, afin que les parties gauche et droite des formes reçoivent des

signaux de données d'entrées différents. Ces travaux appartiennent à la catégorie de la supervision faible, puisqu'ils requièrent des alignements spécifiques à la fois lors de l'entraînement et des tests. Enfin, pour qu'une méthode soit totalement non-supervisée dans notre cas, la fonction de coût du réseau doit pouvoir écarter les correspondances symétriques indésirables sans avoir recours à la supervision faible ni au sur-apprentissage sur la triangulation. Alors que Deep Shells [45] propose une telle fonction de coût, leur structure reste néanmoins très dépendante des descripteurs SHOT en données d'entrée, et s'avère de nouveau instable en cas de changement de triangulation ou même pour certains ensembles de données comme nous le décrivons dans le chapitre 6.

Notre approche décrit un cadre robuste et efficace pour la correspondance de formes non-supervisée, toujours basé sur une pipeline spectrale, via une fonction de coût mêlant les applications fonctionnelles et *les applications fonctionnelles complexes* pour résoudre le problème de symétrie. Ce réseau s'appuie sur l'extracteur de signaux 3D non sensible à la triangulation DiffusionNet [144], ce qui le rend robuste au changement de discrétisation. De plus, notre méthode utilise *les champs de vecteurs tangents* qui servent de signaux pertinents pour la correspondance. Ainsi, nous sommes convaincus que notre méthode ouvre de nouvelles perspectives à la fois pour la correspondance et pour la représentation de formes. En effet, les fonctions caractéristiques et les champs de vecteurs calculés par notre réseau de façon non supervisée pourraient être transférées à plusieurs autres tâches d'analyse de formes 3D.

2.3 Liste des Publications

Cette thèse s'appuie sur les publications suivantes :

- N. DONATI, A. SHARMA AND M. OVSJANIKOV, *Deep Geometric Functional Maps : Robust Feature Learning for Shape Correspondence*, in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020. [39]

- N. DONATI, E.CORMAN, S.MELZI AND M. OVSJANIKOV, *Complex Functional Maps : a Conformal Link between Tangent Bundles*, in Proceedings of the Eurographics Symposium on Geometry Processing, CGF 2022. [40]

- N. DONATI, E. CORMAN AND M. OVSJANIKOV, *Deep Orientation-Aware Functional Maps : Tackling Symmetry Issues in Shape Matching*, in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022. [41]

La publication suivante a également été développée pendant mon doctorat et est étroitement liée aux sujets mentionnés ci-dessus. Cependant, elle n'est pas incluse dans cette thèse :

- L. LI, N. DONATI AND M. OVSJANIKOV, *Learning Multi-resolution Functional Maps with Spectral Attention for Robust Shape Matching*, in NeurIPS 2022. [82]

CHAPTER 3

Shapes, Shape Matching and Functional Maps

In this thesis, we will focus on a particular shape analysis task: *shape matching*.

The goal of shape matching is to find a meaningful correspondence, also called mapping between a given pair of shapes. Typically, these shapes should exhibit some similarities, e.g. two humans in different poses. This particular example goes in the category of *non-rigid shape matching*, where the two shapes to match differ by a non-rigid transformation (change in species, change in pose, as in Figure 1.2). Non-rigid shape matching extends the problem of rigid alignment where the goal is to find the rigid transformation (translation and rotation, forming the Lie group $SE(3)$ in the 3D case) aligning two *displaced* copies of the same object. However, the output of rigid alignment (in the 3D case) is a 6 parameters global transformation, whereas non-rigid shape matching aims at computing a map in an infinite dimensional space: that of all continuous maps between two shapes. The challenge is to retrieve the correct map (also called *ground-truth* map) or more generally a map with the best properties as possible, given these two shapes for input, and with ideally no user input (e.g. landmarks or manual rigid alignment). To do so, it is fundamental to derive energies and regularization in the space of maps to subsequently naviguate this space in an optimized manner.

In this Chapter, we introduce the concepts and notations that will be useful to the reader in the rest of this manuscript. We first go over differential geometry essentials, then focus on the functional map operator and its associated shape matching pipeline, which is core to our subsequent discussion.

3.1 Differential Geometry Essentials

Given a pair of non-rigid shapes, usually represented as *manifolds* M, N our main goal is to estimate a map between these shapes $\varphi : M \to N$. The output map should have properties such as regularity, near-isometry or bijectivity. Let us first define differential manifolds, which we will use to encode 3D shapes.

3.1.1 Manifolds

Definition 3.1.1. *A manifold M is a topological space in which each point has a neighborhood diffeomorphic to $\mathbb{R}^n$, where n is then called the dimension of the manifold. To perform calculus on a manifold, it is important to keep track of these local maps linking neighborhoods with a coordinate space. For every $x \in M$, we thus define its coordinate chart $\psi_x : U_x \to \mathbb{R}^n$, with U_x the associated neighborhood of x. One can use charts to parametrize the entire manifold with a family (x) such that $\bigcup_x U_x = M$. this corresponding family of charts (U_x, ψ_x) (which is required to be countable for a topological manifold) is then called an atlas.*

For two overlapping charts (U_x, ψ_x) and (U_y, ψ_y), one can consider the change in coordinates through the *transition map*:

$$\psi_{yx} = \psi_y \circ {\psi_x}^{-1} : \psi_x(U_x \cap U_y) \to \psi_y(U_x \cap U_y)$$

The charts allow to transfer objects from $\mathbb{R}^n$ locally onto the manifold. For these objects to be globally coherent, they need to be glued together where two charts intersect. For differentiable objects to be glued together by the transition mapsψ_{yx}, we need $\psi_{yx} \in \mathcal{C}^\infty$. We show an example of such a transition map in Figure 3.1.

Definition 3.1.2. *A manifold with a $\mathcal{C}^\infty$ atlas structure (all transition maps in $\mathcal{C}^\infty$) is called a differentiable manifold.*

We can then introduce regular functions on a differential manifold:

Definition 3.1.3. *Let $f : M \to \mathbb{R}$ be a real-valued function on a differentiable manifold M. f is said to be $\mathcal{C}^\infty$ if for all chart (U, ψ) of a $\mathcal{C}^\infty$ atlas, $f \circ \psi^{-1} : \psi(U) \to \mathbb{R}$ is an element of $\mathcal{C}^\infty(\mathbb{R}^n, \mathbb{R})$.*

More generally, we define regular maps between differentiable manifolds.

Definition 3.1.4. *Let M, N be two differentiable manifolds, and a continuous map $\varphi : M \to N$. This map is said to be $\mathcal{C}^\infty$ if for all charts (U_M, ψ_M) and (U_N, ψ_N) of $\mathcal{C}^\infty$ atlases on M, N, the function $\psi_N \circ \varphi \circ \psi_M^{-1} : \psi_M(U_M \cap \varphi^{-1}(U_N)) \to \mathbb{R}$ is an element of $\mathcal{C}^\infty(\mathbb{R}^n, \mathbb{R}^n)$.*

3.1.2 Tangent Bundles

In a general setting, the *tangent bundle* at $p \in M$, denoted as T_pM is defined intrinsically (without the concept of an ambient space) as the space of all *tangent vectors* at point p.

Definition 3.1.5. *Tangent vectors at p are defined as linear forms on the space $\mathcal{C}^\infty(M, \mathbb{R})$, which are also differentiations at p. Namely, X is a tangent vector to M at p if:*

(i) $X \cdot (f + \lambda g) = X \cdot f + \lambda X \cdot g$

(ii) $X \cdot (fg) = g(p)(X \cdot f) + f(p)(X \cdot g)$

where $f, g \in \mathcal{C}^\infty(M, \mathbb{R}), \lambda \in \mathbb{R}$

By construction, the tangent vectors at point p form a vector space. The fact that they are also differentiations to functions defined on a space diffeomorphic to $\mathbb{R}^n$ makes the tangent bundle at p a n-dimensional vector space. To explicitate its basis, let $\psi = (x_1, ..., x_n)$ be a chart around p. One can define the tangent vector basis *dual* to the

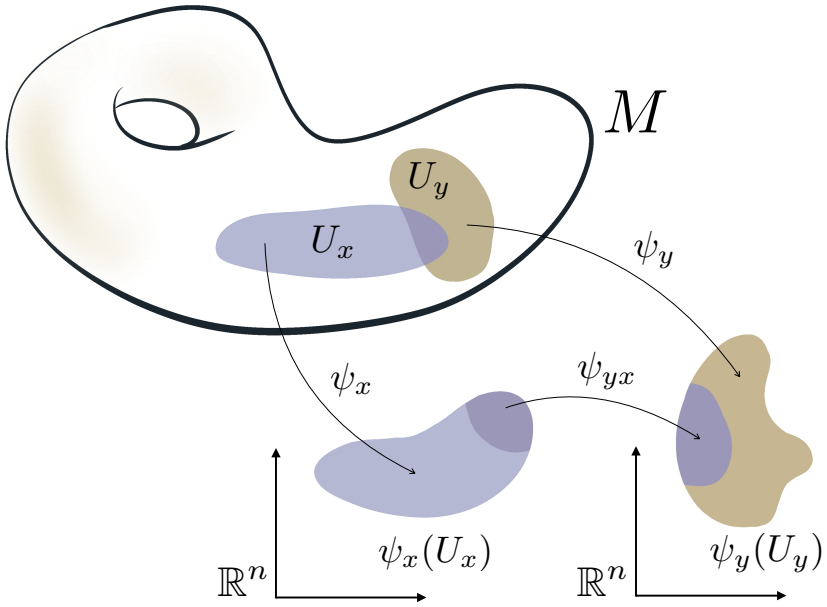


Figure 3.1: A manifold M on which two overlapping charts (U_x, ψ_x) and (U_y, ψ_y). These charts are linked with the transition map ψ_{yx}.

coordinate system (proof in [98], page 32):

$$\left(\frac{\partial}{\partial x_i}\right)_p \cdot f = \left(\frac{\partial(f \circ \psi^{-1})}{\partial x_i} \circ \psi\right)(p)$$

Remark 3.1.1. *For a coordinate change from (x_i) to (y_i) around point p, we have:*

$$\left(\frac{\partial}{\partial x_i}\right)_p = \sum_{j=1}^{n} \frac{\partial y_j}{\partial x_i}(p) \left(\frac{\partial}{\partial y_j}\right)_p$$

We can then build *tangent vector fields* locally on a chart, and thus globally on M:

Definition 3.1.6. *Let $(U, (x_1, ..., x_n))$ be a local chart on M, a tangent vector field X on U is defined as a linear combination of the linear forms $\frac{\partial}{\partial x_i}$:*

$$X = \sum_{i=1}^{n} a_i \frac{\partial}{\partial x_i}$$

where the coefficients $a_i \in \mathcal{C}^\infty(U, \mathbb{R})$ (this regularity does not depend on the choice of coordinate thanks to the previous remark).

The global definition of a tangent vector field is then equivalent to having these $\mathcal{C}^\infty$ coefficients on the whole manifold.

Remark 3.1.2. *The space of tangent vector fields on M is a $\mathcal{C}^\infty(M)$-module.*

It is denoted as TM, the tangent bundle on M.

In $\mathbb{R}^n$, an orientation can be defined on a basis $(e_1, ..., e_n)$ using their determinant. Namely, if $\det(e_1, ..., e_n) > 0$, the basis is said to be of *positive orientation*. If $det(e_1, ..., e_n) < 0$, the basis is said to be of *negative orientation*.
Let (U_x, ψ_x) be a chart on a manifold M. Note that since ψ_x is a diffeomorphism, the orientation of the basis $\left(\frac{\partial}{\partial x_1}, ..., \frac{\partial}{\partial x_n}\right)$ is consistent on U_x.

Moreover, if a consistent orientation can be established on the *whole tangent bundle* of a manifold, it is said to be orientable. More precisely:

Definition 3.1.7. *A manifold M is orientable if there exists an atlas (U_x, ψ_x) for which all transition maps ψ_{xy} have positive Jacobians on all intersections $U_x \cap U_y$, we then call M an oriented manifold.*

Now that the tangent bundle has been defined, we can define *map differentials*. This concept will be used extensively in Chapter 5 and 6.

Definition 3.1.8. *Let $\varphi : M \to N$ be a $\mathcal{C}^\infty(M)$-map between two manifolds M and N.*
For each $p \in M$, we can define the linear map $d_p\varphi : T_pM \to T_{\varphi(p)}N$ as follows:

$$d_p\varphi : X \mapsto Y = (f \mapsto X \cdot (f \circ \varphi))$$

This map is called the differential of φ at p.
For $X \in T_pM$, $f \in \mathcal{C}^\infty(M)$, we can write:

$$X \cdot (f \circ \varphi) = d_p\varphi(X) \cdot f$$

which is sometimes taken to be the definition of the differential, also called push-forward.
The global definition for the map differential $d\varphi : TM \to TN$ is then given by:

$$d\varphi : X \mapsto Y = (p \mapsto Y_p = d_p\varphi(X_p))$$

Remark 3.1.3. *If M, N two oriented manifolds, and $\varphi : M \to N$ a map between them. If $\det(d_p\varphi) > 0$ for all $p \in M$, firstly φ is called an immersion as its differential is injective at each point, secondly φ preserves the orientation of M onto its image $\varphi(M)$, and we can compare the orientation of M and N.*
If $\varphi(M) = N$, then φ is a diffeomorphism between M and N.

3.2 Surfaces in 3D

We will in this manuscript restrict ourselves to the case of surfaces embedded in 3D space. All the notions previously established are of course still valid with $n = 2$.
However, the presence of an embedding space can help define useful objects, e.g. normal fields, or vizualise abstract concepts such as the tangent plane.

3.2.1 First Fundamental Form and Gradient

Remark 3.2.1. *Let M be a differentiable surface of $\mathbb{R}^3$. The tangent plane T_pM at $p \in M$ can be introduced as the plane tangent to M at p.*
Indeed in this case, the basis $\left(\left(\frac{\partial}{\partial x_1}\right)_p, \left(\frac{\partial}{\partial x_2}\right)_p\right)$ can be written in $\mathbb{R}^3$ from the embedded, inverted chart :

$$\nu = \psi^{-1}\big|^{\mathbb{R}^3} : \psi(U) \subset \mathbb{R}^2 \to \mathbb{R}^3$$

The basis can then naturally be defined as $\left(\left(\frac{\partial \nu}{\partial x_1}\right)_p, \left(\frac{\partial \nu}{\partial x_2}\right)_p\right)$, which by construction define a plane tangent to the surface M at p.

We can still however link the abstract concept and this new practical one. Indeed it is possible to see the 3D vector $\left(\frac{\partial \nu}{\partial x_i}\right)_p$ as a tangent vector in the abstract sense (a linear form that is also a differentiation). Indeed the tangent plane T_pM can be equipped with a scalar product inherited from the ambient space.

Definition 3.2.1. *Let the scalar product* $\mathbf{g}_p$ *on* T_pM *be defined as:*

$$\begin{aligned} \mathbf{g}_p \colon (T_pM)^2 &\to \mathbb{R} \\ (X, Y) &\mapsto \langle X, Y \rangle_{\mathbb{R}^3} \end{aligned}$$

Proposition 3.2.1. *For any* X, Y *two tangent vector fields, the function* $p \mapsto \mathbf{g}_p(X_p, Y_p)$ *is* $\mathcal{C}^\infty(M)$*, we say that* $\mathbf{g} : p \mapsto \mathbf{g}_p \in \mathcal{C}^\infty(M, \mathcal{B}(T_pM))$*, or that* $\mathbf{g}$ *defines a smoothly varying scalar product over the tangent bundle.*

Definition 3.2.2. *To have such an application* $\mathbf{g}$ *is the condition to be what is called a* Riemannian manifold*, and* $\mathbf{g}$ *is then called Riemannian metric, or* first fundamental form*.*

In our case we can write $\mathbf{g}_p$ in our local basis of the tangent plane $\left(\left(\frac{\partial \nu}{\partial x_1}\right)_p, \left(\frac{\partial \nu}{\partial x_2}\right)_p\right)$ to get the 2×2 matrix with coefficients:

$$\mathbf{g}_{ij} = \left\langle \left(\frac{\partial \nu}{\partial x_i}\right)_p, \left(\frac{\partial \nu}{\partial x_j}\right)_p \right\rangle \tag{3.1}$$

The abstract definition of a tangent vector is through its action on the functional space. Here with the metric $\mathbf{g}$, we can define this operation more specifically by contextualizing this action. To that end, we can introduce the *gradient* of a function, an element of T_pM acting as a differentiation on that function:

$$X \cdot f = \langle X, \nabla f \rangle_{\mathbb{R}^3}$$

Remark 3.2.2. *Let* $f \in \mathcal{C}^\infty(M)$ *be a function on the surface. Then we can compute the coordinates of the gradient* ∇f *in the local basis* $\frac{\partial \nu}{\partial x_i}$
Indeed:

$$\begin{aligned} X \cdot f &= \langle X, \nabla f \rangle_{\mathbb{R}^3} \\ &= \mathbf{g}(X, \nabla f) \\ &= \sum_{i,j} \mathbf{g}_{ij} X_i (\nabla f)_j \end{aligned}$$

Since $X \cdot f = \sum_i X_i \frac{\partial f}{\partial x_i}$ *by definition, we get:*
$\mathbf{g} \cdot \nabla f = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2} \right)$, *hence:*

$$\nabla f_i = \sum_j (\mathbf{g}^{-1})_{ij} \frac{\partial f}{\partial x_j}$$

3.2.2 Integral and Divergence

To compute an integral on a differential surface M, we simply take into account the deformation of the infinitesimal square $dx_1 \times dx_2$ under ν, which corresponds to the infinitesimal element of T_pM given by $dx_1 \frac{\partial \nu}{\partial x_1} \times dx_2 \frac{\partial \nu}{\partial x_2}$, of area: $\sqrt{\det(\mathbf{g})} dx_1 dx_2$.

We can then write a local integral using the chart (U, ψ):

$$\int_U f = \int_{\psi(U)} (f \sqrt{(\det(\mathbf{g}))} \circ \psi^{-1} dx dy$$

Now we can use a tool fundamental to differential manifolds, called partition of unity:

Definition 3.2.3. *Let* M *be a differentiable manifold. A countable family of* $\mathcal{C}^\infty$ *functions* (h_i) *is said to be a* partition of unity *on* M *if* $h_i \geq 0$, $\text{supp}(h_i)$ *compact, and:*

$$\sum_i h_i = 1$$

An important theorem (in [98], page 29) gives the existence of a partition of unity refining any open covering of M.

We use a partition of unity refining an atlas (U_i, ψ_i) to define a global integral from local integrals on U_i:

$$\int_M f := \int_M \sum_i h_i f = \sum_i \int_M h_i f = \sum_i \int_{U_i} h_i f$$

This integral automatically defines a scalar product on $\mathcal{C}^\infty(M)$, and we have the space of square integrable functions:

$$\mathcal{L}^2(M) = \left\{ f \in M \;\middle|\; \int_M f^2 < \infty \right\}$$

Note that we can also define a scalar product between two tangent vector fields X, Y:

$$\langle X, Y \rangle_{TM} = \int_M < X, Y >_{\mathbb{R}^3}$$

We can now define a differentiation operator on tangent vector fields, the *divergence* operator. For manifolds without boundaries we can define it through the integration by parts identity. More precisely, if X is a tangent vector field, we simply ask that the divergence be opposite to the adjoint to the gradient on the scalar product $\langle X, \nabla f \rangle_{TM}$, for $f \in \mathcal{C}^\infty(M)$.

$$\int_M \mathrm{div}(X) f = -\int_M \langle X, \nabla f \rangle_{\mathbb{R}^3} = -\int_M \sum_i X_i \frac{\partial f}{\partial x_i}$$

We can follow this integration on the separate charts (U_j, ψ_j) (where we use $dx_1 dx_2$ to signify that integrals are conducted in $\mathbb{R}^2$ rather than on the surface, we then skip the composition with ψ_j^{-1} to lighten the notations), by doing a real integration by parts in $\mathbb{R}^2$ to get:

$$\begin{aligned}\int_{\psi_j(U_j)} \sum_i X_i \frac{\partial f}{\partial x_i} \sqrt{\det(\mathbf{g})} dx_1 dx_2 &= -\int_{\psi_j(U_j)} f \sum_i \frac{\partial}{\partial x_i}(\sqrt{\det(\mathbf{g})} X_i) dx_1 dx_2 \\ &= -\int_{U_j} f \frac{1}{\sqrt{\det(\mathbf{g})}} \sum_i \frac{\partial}{\partial x_i}(\sqrt{\det(\mathbf{g})} X_i)\end{aligned}$$

We can then conclude with an expression for the divergence:

Definition 3.2.4. *The divergence operator is defined on the tangent bundle TM by:*

$$\mathrm{div}(X) = \frac{1}{\sqrt{\det(\mathbf{g})}} \sum_i \frac{\partial}{\partial x_i}(\sqrt{\det(\mathbf{g})} X_i)$$

It is of course possible to derive this expression locally, in a more general setting. This definition of divergence is in fact fully *intrinsic*, which means that it only depends on the first fundamental form $\mathbf{g}$ (also called Riemannian metric). On the other hand, *extrinsic* quantities can only be introduced through the embedding space (e.g. the vector normal to the surface in $\mathbb{R}^3$). Note that here we introduced a metric via the ambient space, but in Riemannian geometry, metrics are introduced *without* the notion of an embedding space. The gradient and divergent operator are then called *intrinsic operators*.

3.2.3 Laplacian

Definition 3.2.5. *We can now introduce the Laplacian operator, also called Laplace-Beltrami operator. This object is also intrinsic, it is defined on $\mathcal{C}^\infty(M)$ and can simply be seen as the composition of divergence and gradient:*

$$\Delta f = \mathrm{div}(\nabla f) = \frac{1}{\sqrt{\det(\mathbf{g})}} \sum_i \frac{\partial}{\partial x_i}(\sqrt{\det(\mathbf{g})} \sum_j (\mathbf{g}^{-1})_{ij} \frac{\partial f}{\partial x_j})$$

The integration by parts discussed in the last section for divergence (which is in fact a consequence of the well-known Stokes' Theorem, see [98], Chapter 3.2) can then be applied in the case of manifolds without boundaries:

$$\forall f_1, f_2 \in \mathcal{C}^\infty(M), \int_M f_1 \Delta f_2 = -\int_M \mathbf{g}(\nabla f_1, \nabla f_2)$$

We remind the reader that $\mathbf{g}(\nabla f_1, \nabla f_2) = \langle \nabla f_1, \nabla f_2 \rangle_{\mathbb{R}^3}$, we kept the metric $\mathbf{g}$ in the previous equation to insist on its *intrinsic* character. Indeed this distinction between intrinsic and extrinsic will become crucial in the rest of this manuscript.

Remark 3.2.3. *This last consideration introduces a new bilinear application on $\mathcal{C}^\infty(M)$, namely $\int_M \mathbf{g}(\nabla f_1, \nabla f_2)$. Note that for this application to be a scalar product, one needs to restrict the space $\mathcal{C}^\infty(M)$ to non-constant functions (without exluding 0) so that the application can be definite. We also need to restrict to the space of functions whose gradients is square integrable, which we call $\mathcal{H}^1(M)$. Calling this scalar product $\langle ., . \rangle_{\mathcal{H}^1}$ then we have:*

$$\forall f_1, f_2 \in \mathcal{H}^1(M), \langle f_1, f_2 \rangle_{\mathcal{H}^1} = -\langle f_1, \Delta f_2 \rangle = -\langle \Delta f_1, f_2 \rangle$$

The Laplacian is therefore symmetric negative.

We can then proceed to introduce the Laplace-Beltrami *eigenbasis*, stemming from the symmetric negative property of the Laplacian (see [22], Section 4.1 for a more detailed analysis).

Proposition 3.2.2. *The eigen-problem $\Delta\Phi = \lambda\Phi$ gives the (countably many) eigenvalues $(\lambda_i)_i$ of Laplace-Beltrami Operator.*
These eigenvalues are negative and form a decreasing sequence:

$$0 = \lambda_0 \geq \lambda_1 \geq ... \geq \lambda_i \geq ...$$

The eigenfunctions $(\Phi_i)_i$ form an orthogonal eigenbasis of $\mathcal{L}^2(M)$, and are ordered from low-to-high frequency. Indeed $\|\nabla\Phi_i\|^2 = \langle \Phi_i, \Delta\Phi_i \rangle = |\lambda_i|$.

Any function in $\mathcal{L}^2(M)$ can be decomposed in the Laplacian eigenbasis as:

$$f = \sum_{i=0}^{\infty} \langle f, \Phi_i \rangle \, \Phi_i$$

If we truncate this sum at a certain number $n \geq 0$, we obtain a smooth approximation of the function f. This remark will be fundamental in Section 3.4. Indeed, the Laplace-Beltrami eigenbasis $(\Phi_i)_i$ plays the same role in geometry processing as the Fourier basis in signal processing.

3.3 Discretization

In this section we detail how to define all the previous concepts on discrete objects. Indeed, limited storage forces us to consider finite representations, which are potentially non-differentiable. We refer the interested reader to [22], Chapter 3, for more information about finite elements.

3.3.1 Point Cloud and Meshes

To discretize 3D objects, one of the most common representation is *3D point clouds*. Indeed, 3D scans are often obtained through Laser-based techniques, which essentially determine point positions in 3D space.

Definition 3.3.1. *A point cloud is defined as a set of* distinct *3D vertices* $\mathcal{X} \in (\mathbb{R}^3)^{(\mathbb{N})}$*, where we note* $(\mathbb{R}^3)^{(\mathbb{N})}$ *the finite sequences over* $\mathbb{R}^3$*. The cardinal* n *of* $\mathcal{X}$ *corresponds to the number of points, and we can then store* $\mathcal{X}$ *as a* $\mathbb{R}^{n\times 3}$*-tensor.*

Despite their efficiency and extensiveness, point clouds do not naturally encode the notion of *surface*. This is the purpose of the mesh data structure:

Definition 3.3.2. *A mesh is defined as a triplet of sets* $(\mathcal{X}, \mathcal{E}, \mathcal{F})$*, where:*
$\mathcal{X}$ *is a 3D point cloud, composing the* vertices (x_i) *of the mesh,*
$\mathcal{E}$ *is the set of undirected edges linking vertices together in a graph structure. We note* (i, j) *the edge linking vertex* x_i *and* x_j*,*
$\mathcal{F}$ *is the set of triangle faces, together forming the piece-wise linear surface. Each face is a set of* 3 *edges forming a cycle* $\{(i, j), (j, k), (k, i)\}$*, thus linking the* 3 *vertices* x_i*,* x_j *and* x_k*.*
For the surface to be well-defined, all undirected edges need to appear in 2 triangles at most*. The mesh is then called* manifold*.*

Remark 3.3.1. *Let* $M = (\mathcal{X}, \mathcal{E}, \mathcal{F})$ *be a mesh. If all undirected edges* $e \in \mathcal{E}$ *appear in* exactly 2 *triangles of* $\mathcal{F}$*, then the manifold is called* closed*.*

Otherwise, the set of edges appearing in only one triangle of $\mathcal{F}$ *is called the* border *of the manifold. Every connected component of this set forms a closed piece-wise linear curve in* $\mathbb{R}^3$*.*

As in the continuous case, a closed manifold is *orientable*, which means that consistent a normal field can be defined on it, thus separating the volume enclosed by that shape and its exterior. One simple way to define normals is on each linear surface element. Namely, for each triangle (x_i, x_j, x_k), we can consider the two 3D vectors $x_j - x_i$ and $x_k - x_i$, then the cross product $(x_j - x_i) \times (x_k - x_i) \in \mathbb{R}^3$ is orthogonal to the triangle. We can then define the normal field:

Definition 3.3.3. *Let $M = (\mathcal{X}, \mathcal{E}, \mathcal{F})$ be a closed mesh. Since M is orientable, there is a way to* orient *all the triangles $(i,j),(j,k),(k,i)$ such that each unoriented edge appears with its two possible orientations in the set of oriented triangles.*
Then the normal field $\vec{n}$ defined on each triangle $T_{ijk} = (x_i, x_j, x_k)$ by:

$$\vec{n}_{T_{ijk}} = \frac{(x_j - x_i) \times (x_k - x_i)}{\|(x_j - x_i) \times (x_k - x_i)\|}$$

is consistent in the sense that it locally separates interior from exterior of M. If $\vec{n}$ points towards the exterior of the shape, the shape is said to be oriented with outer normals. *In the opposite situation, it is said to be oriented with* inner normals.

Having equipped the mesh with a normal field, we can also define the discrete equivalent of its tangent bundle. Indeed, for each normal vector $\vec{n}$, one can consider its orthogonal plane (which can be oriented with the right-hand rule). This set of planes forms the discrete tangent bundle.
If (as in the previous definition) the normals are introduced on triangles, a tangent plane is associated with the corresponding face element, and indeed is the plane generated by the three points constituting this face element.

Remark 3.3.2. *A point cloud $\mathcal{X} \in (\mathbb{R}^3)^n$ can also be equipped with a normal field $\vec{n} \in (\mathbb{R}^3)^n$. We then talk of* oriented point cloud*, where each point has its corresponding normal vector. A tangent plane can then be associated with each vertex of the point cloud.*

3.3.2 Discrete Functional Space

The next thing to introduce on our discrete surfaces are functions. Then, it becomes possible to study these discrete objects through their associated functional space, on which we can subsequently define useful operators.

Finite Elements

Definition 3.3.4. *Let $M = (\mathcal{X}, \mathcal{E}, \mathcal{F})$ be a mesh. To represent functions over M we use the piece-wise linear finite element basis made of hat functions $(h_i)_{i \in [1, |\mathcal{X}|]}$, with :*

$$\forall x_j \in \mathcal{X}, h_i(x_j) = \begin{cases} 1 & \text{if } j = i, \\ 0 & \text{otherwise} \end{cases}$$

And h_i is completed on the faces with the condition that it is linear on the faces.

We illustrate an element of the hat basis in Figure 3.2.
Any function f that is continuous and linear on the faces can be written in this basis as:

$$f = \sum_i \alpha_i h_i$$

We can then represent any function over the mesh $(\mathcal{X}, \mathcal{E}, \mathcal{F})$ as a vector in $\mathbb{R}^n$ (with $n := |\mathcal{X}|$) storing the coefficients of f in the hat basis, which is also the values f_i of f at the vertices $x_i \in \mathcal{X}$.

Remark 3.3.3. *The hat basis can also be defined on point clouds in a very canonical way, by taking the definition 3.3.4 without imposing additional conditions. Any function f on a point cloud $\mathcal{X}$ can then be represented with a vector in $\mathbb{R}^n$.*

Scalar Product

At this point we can transfer the definition of integrals from continuous manifold to our discretization. Indeed, using the linearity of the integral, we simply need to compute integrals of the hat basis elements:

$$a_i := \int_M h_i = \int_{\cup_{i \in T} T} h_i = \sum_{i \in T} \int_T h_i = \frac{1}{3} \sum_{i \in T} \mathcal{A}(T)$$

where $\mathcal{A}(T)$ is the area of triangle T, so that a_i is the area of the barycentric cell around vertex x_i. This a_i can be seen as the area element $\sqrt{\det(g)}$ of the continuous case, as it stems from the metric, which can also be introduced in the discrete setting as we will see in the following.
As a mean to check the validity of this area element, we can assert if the integral of the constant 1 function gives the total area of the surface:

$$\int_M 1 = \sum_i a_i = \sum_i \frac{1}{3} \sum_{i \in T} \mathcal{A}(T) = \sum_T \mathcal{A}(T)$$

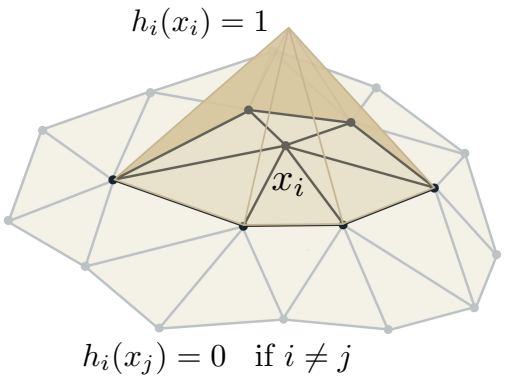


Figure 3.2: The hat basis function h_i on a discrete mesh. Any discrete function f on the mesh can be decomposed in the basis $(h_i)_{i \in [1, |\mathcal{X}|]}$.

since every triangle of the surface appears 3 times in the second sum. The third sum is by construction the total area of the mesh $\mathcal{A}(M)$, which completes the check.

Definition 3.3.5. *On a discrete mesh M, one* potential *discretization of the area element for standard integration is given by $a_i = \frac{1}{3}\sum_{i\in T}\mathcal{A}(T)$.*
The integral of a function f can then be written $\int_M f = \sum_i a_i f_i$.

We can also introduce the scalar product between discrete function. To this end we need to compute the integrals:

$$\langle h_i, h_j\rangle = \int_M h_i h_j = \begin{cases} A_{ij} & \text{if } (i,j)\in\mathcal{E} \text{ or } i=j, \\ 0 & \text{otherwise} \end{cases}$$

Computing the integrals we get:

$$A_{ii} = \frac{1}{6}\sum_{i\in T}\mathcal{A}(T)$$
$$A_{ij} = \frac{1}{12}(\mathcal{A}(T_{ijk}) + \mathcal{A}(T_{ilj})) \text{ if } (i,j)\in\mathcal{E},$$

where (i,j,k) and (i,l,j) are the two oriented triangles containing the unoriented edge (i,j) (if this edge is a border, A_{ij} contains only one term).

The Gram matrix corresponding to this scalar product is then given by the symmetric *sparse* matrix $[A_{ij}]$. This mass matrix is often replaced by its *lumped* approximation, given by the diagonal matrix with i-th coefficient $\sum_j A_{ij} = \frac{1}{3}\sum_{i\in T}\mathcal{A}(T) = a_i$, where the first equality stems from the definition of A_{ij}. Consequently, the lumped mass matrix associated to $[A_{ij}]$ is $\text{diag}(a_i)$.

Remark 3.3.4. *The lumped approximation corresponds to discretizing $\int_M f_1 f_2$ as a simple integral:*

$$\int_M (f_1 f_2) = \sum_i a_i (f_1 f_2)_i = \sum_i a_i (f_1)_i (f_2)_i$$

Definition 3.3.6. *The standard scalar product on the discrete functional space can be defined as:*

$$\langle f_1, f_2\rangle_{\mathcal{L}^2} = \sum_i a_i (f_1)_i (f_2)_i$$

where we often denote the mass matrix as $A = diag(a_i)$.

3.3.3 Operators as Sparse Matrices

The first operator that is useful to discretize is the intrinsic metric g. Indeed it defines a local metric on each tangent plane (here defined on triangles), and the continuous definition of Eq. (3.1) gets the discrete approximation:

$$\mathbf{g}(T_{ijk}) = \begin{pmatrix} x_j - x_i \\ x_k - x_j \end{pmatrix}^T \begin{pmatrix} x_j - x_i \\ x_k - x_j \end{pmatrix}$$

Simplifying the anti-diagonal terms in the previous expression, we get the following discrete metric:

$$\mathbf{g}(T_{ijk}) = \frac{1}{2} \begin{pmatrix} 2l_{ij}^2 & l_{ki}^2 - l_{ij}^2 - l_{jk}^2 \\ l_{ki}^2 - l_{ij}^2 - l_{jk}^2 & 2l_{jk}^2 \end{pmatrix},$$

which *only depends on the edge lengths* $l_{ij}, (i,j) \in \mathcal{E}$. A tensor depending only on these edge length quantities is thus called *intrinsic*.

Gradient This is the case of the gradient operator, defined naturally on the hat basis, as detailed in [22], Chapter 3. Computations lead to the following expression for the gradient of h_i on a triangle T_{ijk} adjacent to x_i:

$$(\nabla h_i)_{T_{ijk}} = \frac{1}{\mathcal{A}(T_{ijk})}(x_k - x_j)^\perp$$

Putting things together, we get the following definition:

Definition 3.3.7. *On a discrete mesh M, the discrete gradient is an intrinsic operator. Its value on the triangle T_{ijk}, which corresponds to the tangent plane, is given by:*

$$(\nabla f)_{T_{ijk}} = \frac{1}{2\mathcal{A}(T_{ijk})} \begin{pmatrix} (x_i - x_k)^\perp \\ (x_j - x_i)^\perp \end{pmatrix} \begin{pmatrix} f_j - f_i \\ f_k - f_j \end{pmatrix}$$

This operator can be assembled in a sparse matrix $G \in \mathbb{R}^{2|\mathcal{F}| \times |\mathcal{X}|}$, taking functions on M as $|\mathcal{X}|$-dimensional vectors and outputting tangent vectors on the tangent planes as $2|\mathcal{F}|$-dimensional vectors.

Divergence This operator is also intrinsic, and following the continuous definition where gradient and divergence are adjoint operators, we can introduce it as:

$$div(X) = A^{-1} G^T \cdot X$$

where X is encoded as a $2|\mathcal{F}|$-dimensional vector, G is the previously defined sparse matrix for the gradient operator, and A the lumped mass matrix.

Remark 3.3.5. *Note that there are other discretizations for the divergence operator, stemming from other continuous properties that we might want to preserve in the discrete world. More generally, there are many potential discretizations for the functional space, tangent bundles and corresponding operators, as we will see in Chapter 5.*

Laplacian Lastly, we can introduce the discretization of Laplace-Beltrami Operator, which is, as previously stated central to spectral methods that constitute the backbone of the following manuscript:

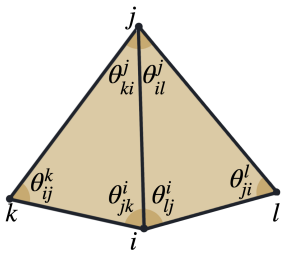
Definition 3.3.8. *The Laplace-Beltrami operator* $\Delta \in \mathbb{R}^{|V|\times|V|}$ *for a piece-wise linear function* f *on a discrete mesh* M *is obtained by the so-called* cotangent weights formula *[22]:*

$$(\Delta f)_i := \frac{1}{2\mathcal{A}(T_{ijk})} \sum_{(ij)\in\mathcal{E}} \left(\cot\theta_{ij}^k + \cot\theta_{ji}^l\right)(f_j - f_i)\,,$$

where the index notation is defined in the inset figure.

We can re-write this operator as $\Delta = A^{-1}W$, where W is the symmetric, sparse matrix:

$$W_{ij} = \begin{cases} \cot\theta_{ij}^k + \cot\theta_{ji}^l & \text{if } (i,j)\in\mathcal{E} \\ -\sum_{(ij)\in\mathcal{E}}\left(\cot\theta_{ij}^k + \cot\theta_{ji}^l\right) & \text{if } i=j \\ 0 & \text{otherwise} \end{cases}$$



As in the continuous case, the non-normalized matrix W (sometimes called stiffness matrix) is a semi-definite negative matrix and is consequently fully diagonalizable. To get the eigenbasis $(\Phi_i)_{i\in\mathbb{N}}$ that we then use to project functions upon, we solve the generalized eigen problem $Wf = \lambda Af$. This basis is naturally ordered in a low-to-high-frequency manner, as displayed in Figure 3.4. Namely, the lowest the module of the eigenvalue λ_i, the lowest the frequency of the eigenfunction φ_i.

3.4 The Functional Map Operator

Given a pair of non-rigid 3D shapes M, N, the functional map approach presents a linear, compact and effective way to encode any map $\varphi : M \to N$.
The method essentially relies on two key remarks:

(a) The pullback of φ is a linear operator between the shapes' functional spaces (Section 3.4.1)

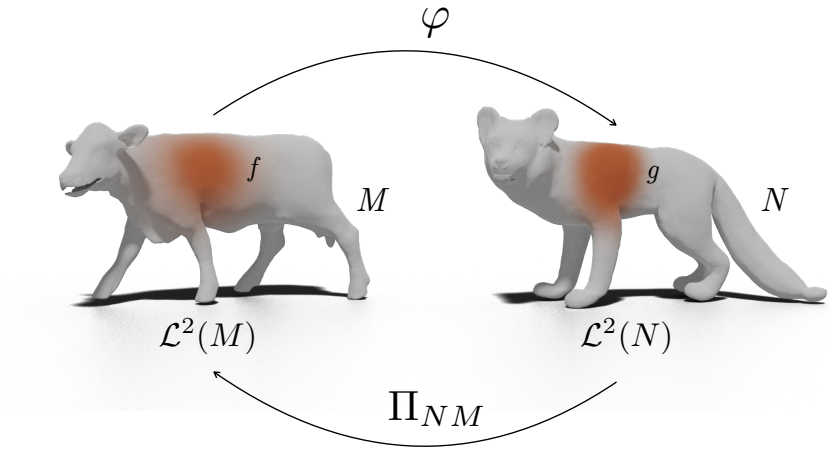


Figure 3.3: A map φ between shapes M and N and its corresponding pullback operator Π_{NM}, sending functions $g \in \mathcal{L}^2(N)$ to $f = \Pi_{NM}(g) \in \mathcal{L}^2(M)$. Orange color corresponds to a maximum of the function, and grey to a minimum.

(b) Using a low-to-high frequency basis on each shape (e.g. the Laplace-Beltrami eigenbasis defined in Section 3.2.3) allows to express the pullback of φ as a small matrix (Section 3.4.2). Indeed, truncating the basis only cuts out little information, making this encoding compact and efficient.

3.4.1 Pullback Operator

Definition 3.4.1. *Let $\varphi : M \to N$ be a mapping between the two manifolds M, N, then for any target function $g \in \mathcal{L}^2(N)$ we can define its* pullback *on the source shape M via the operator:*

$$\Pi_{NM} : g \mapsto f = g \circ \varphi \in \mathcal{L}^2(M)$$

This new functional operator $\Pi_{NM} : \mathcal{L}^2(N) \to \mathcal{L}^2(M)$ is a linear operator. Figure 3.3 shows a map φ and its corresponding pullback defined on functional spaces. Additionally, the original point-to-point mapping φ can be recovered from Π_{NM}, e.g. by using Dirac function approximations to represent a particular point. Consequently, Π_{NM} contains all the information about φ. It should be noted at this point that not all linear maps between $\mathcal{L}^2(N)$ and $\mathcal{L}^2(M)$ are pullbacks. Additional constraints need to be derived in that particular setting, as we will see in the following.

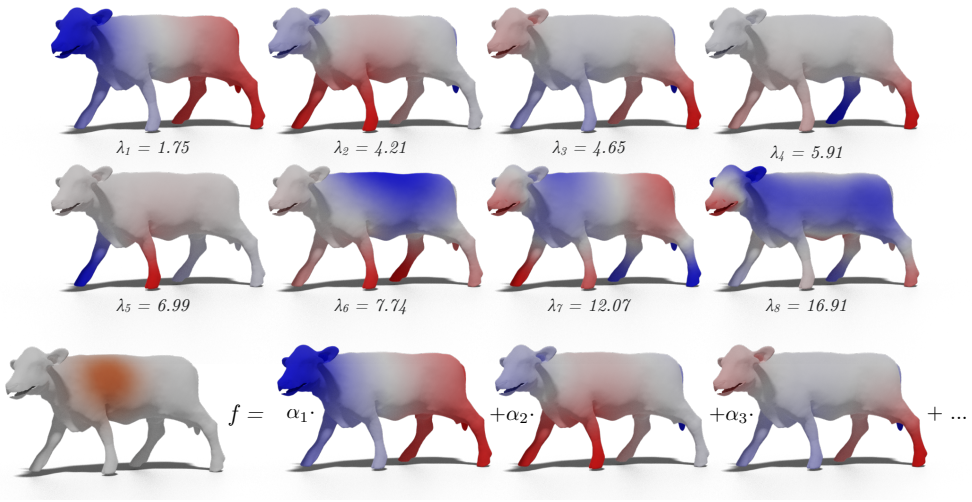


Figure 3.4: Eigenfunctions $(\Phi_i^M)_{[1,8]}$ of the Laplace-Beltrami operator, with the corresponding eigenvalues λ_i (the eigenfunction for $\lambda_0 = 0$ is not plotted as it is constant) on a mesh M. Red color corresponds to positive value, while blue is negative, and white is 0-value. We see that frequency rises as we take deeper functions in the basis. Any discrete function f can be written in this basis as $f = \sum_i \Phi_i^M$.

3.4.2 Laplace-Beltrami eigenbasis

Since $\Pi_{NM} : \mathcal{L}^2(N) \to \mathcal{L}^2(M)$ is a linear operator, it can be decomposed in functional bases on source and target shapes. One good choice of basis in the case of shape analysis, as noted in Section 3.2.3 is given by the *Laplace-Beltrami eigenbasis*, illustrated in Figure 3.4.

Definition 3.4.2. *Equipping $\mathcal{L}^2(M)$ and $\mathcal{L}^2(N)$ with their respective Laplace-Beltrami operators (denoted as Δ_M and Δ_N), and their eigenbasis (respectively $(\Phi_i^M)_{i\in\mathbb{N}}$ and $(\Phi_i^N)_{i\in\mathbb{N}}$). In this basis, we can re-write the operator Π_{NM}, resulting in a representation of the map φ in a so-called* spectral *basis, denoted as C_{NM}:*

$$\mathbf{C}_{NM} = \Phi_M^\dagger \Pi_{NM} \Phi_N$$

where $\Phi^\dagger$ is the Moore-Penrose pseudo-inverse of Φ.

In practice, it is shown in the original paper [106] that truncating these bases to a relatively low number, namely 100 or less, does not cut out much information in standard study cases (e.g. shapes separated by articulated motion). Therefore one can represent the map φ with a small matrix $\mathbf{C}$ of dimension ≤ 100.

Remark 3.4.1. *Note that a pullback operator inverts source and target:* $\mathbf{C}_{NM}$ *takes functions from the second shape* N*, to the first shape* M*. It is also possible to study the symmetric configuration, where a map* $\Psi : N \to M$ *results in the reduced functional map representation* $\mathbf{C}_{MN}$*.*
When the direction of the pull-back is obvious or unimportant, we may denote it simply as $\mathbf{C}$*.*

3.5 Functional Map Computation Pipeline

The functional map pipeline aims at estimating the truncated functional map $\mathbf{C}_{NM}$. To this end, one can rely on so called *descriptor functions*. Such functions are tailored to be stable under non rigid transformation of the underlying shape. They can for instance be based on intrinsic quantities such as Gaussian curvature or heat diffusion [151, 6]. Alternatively, other descriptors can be based on extrinsic or triangulation-related quantities, which is the case of the SHOT descriptor [158]. We display in A.4 some descriptor functions learned by the network described in Chapter 4. Robust descriptors are hard to compute in general as they need to be stable under non-rigid extrinsic deformation. However these functions are paramount to functional map initialization. The resulting map can then be refined to reduce aliasing.

More precisely, given a pair of 3D shapes, M, N represented as discrete meshes, and containing respectively n_M and n_N vertices, this pipeline aims at computing a functional map between them. The point-wise map can then be retrieved from the functional map.

Pre-Processing The first few eigenfunctions of the discrete Laplace-Beltrami operator are computed on each shape. Using k_M and k_N functions respectively, we get the bases $(\Phi_i^M)_{i\in[0,k_M-1]} \in \mathbb{R}^{n_M \times k_M}$ and $(\Phi_i^N)_{i\in[0,k_N-1]} \in \mathbb{R}^{n_N \times k_N}$.
The Laplacian operators are respectively denoted as Δ_M and Δ_N.

Descriptors Second, a set of d descriptor functions on each shape are computed. The key property of these descriptors (also called feature functions, or probe functions) resides in that they should be approximately preserved by the unknown map. In our case this means that descriptors should ideally be stable under *(a)* extrinsic non-rigid deformations (e.g. articulated movement, change of species in the case of animal shapes) *(b)* change in discretization (e.g. re-meshing, coarse triangulation).
The descriptor functions are then projected in the spectral basis, their coefficients in the respective basis are stored as columns in matrices $\mathbf{A}_M, \mathbf{A}_N \in \mathbb{R}^{k_M \times d} \times \mathbb{R}^{k_N \times d}$.

Map Initialization Third, the optimal functional map $\mathbf{C}_{NM}$ is then computed by solving the following optimization problem (as illustrated in Figure 3.5):

$$\mathbf{C}_{\text{opt}} = \arg\min_{\mathbf{C}} E_{\text{desc}}(\mathbf{C}) + \lambda E_{\text{reg}}(\mathbf{C}), \tag{3.2}$$

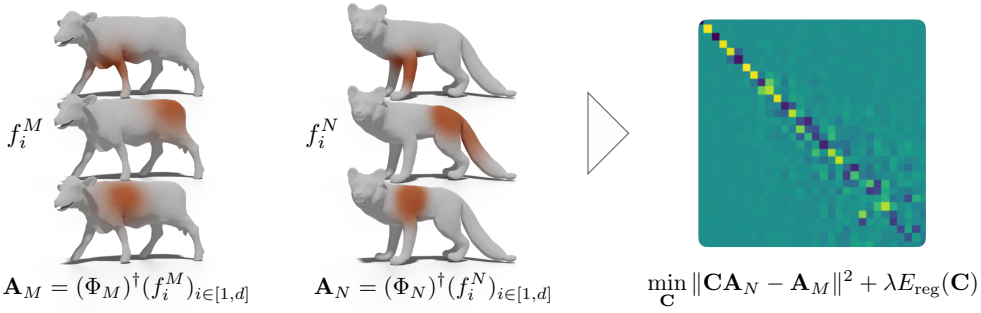


Figure 3.5: The shape matching pipeline is highly dependant on the *initial map estimation*, where a set of *descriptor functions* on each shape $(f_i^M)_{i\in[1,d]}$ and $(f_i^N)_{i\in[1,d]}$ are first projected on the corresponding Laplace-Beltrami eigenbases, and the map $\mathbf{C}$ is then estimated following Eq. (3.2).

where the first term enforces that preservation of the descriptors by the pull-back. It is common to use Fröbenius norm to compute the distance between matrices. The descriptor energy can be written:

$$E_{\text{desc}}(\mathbf{C}) = \left\|\mathbf{C}\mathbf{A}_N - \mathbf{A}_M\right\|^2 \tag{3.3}$$

This first energy is extremely sensitive to the quality of the input descriptors. It is thus usual to use a second term to *regularize* the map by promoting correctness of its overall structural properties.

Some of the most common regularizers are :

- *Orthogonality*. An orthogonal functional map corresponds to an *area-preserving* map φ. More precisely, φ is area-preserving if, and only if for all x measurable subsets of M, $\mu_M(x) = \mu_N(\varphi(x))$. The orthogonality loss can be written:

$$L_{\text{ortho}} = \left\|\mathbf{C}\mathbf{C}^T - \mathbf{I}\right\|^2, \tag{3.4}$$

- *Bijectivity*. For this loss to be computed, one needs to estimate functional maps in both directions, to subsequently enforce their being invert from each other, which corresponds to the condition $\varphi \circ \varphi^{-1} = id_M$:

$$L_{\text{bij}} = \left\|\mathbf{C}_{NM}\mathbf{C}_{MN} - \mathbf{I}\right\|^2, \tag{3.5}$$

- *Commutativity with Laplacian*. It is well-known that a map commuting with the Laplacian operators is an *isometry* [98]. An isometry preserves the local metric,

which entails the preservation of global geodesic distances. This property is also an interesting approximate invariant for shapes undergoing articulated motion, which is why it is used as such in some works [62, 46].
Note that isometries are in fact area-preserving *and* conformal maps. A conformal map is a map φ whose differential $\mathrm{d}\varphi$ preserves angles, as we will see in Chapter 5.
Finally, Laplacian operators in their own spectral basis become diagonal matrices with eigenvalues on the diagonal (denoted as $\mathbf{\Delta}_M$ and $\mathbf{\Delta}_N$). The Laplacian commutativity penalty therefore enforces a diagonal-like structure for the target functional map:

$$L_{\text{lap}} = \left\| \mathbf{C}\mathbf{\Delta}_N - \mathbf{\Delta}_M \mathbf{C} \right\|^2, \tag{3.6}$$

Note that descriptor energy (3.3), and the penalties (3.5), (3.6) are quadratic, while the orthogonality loss (3.4) is not quadratic.
Thus, if the orthogonality loss is not used, the minimization (3.2) leads to a simple least-squares optimization with $k_M \times k_N$ unknowns. It is significant that the number of unknowns *does not depend on the number of points of the shapes*, which can typically range from 10^3 to 10^6 points.
This spectral factorization is indeed, as noted in the previous section, the key advantage of the functional map pipeline.

Refinement As a last step, the estimated functional map $\mathbf{C}$, which maps across the spectral domains, is converted back to a point-to-point map. A post processing step called *refinement*, proposes to iteratively take the map from spectral to spatial domain, until it reaches a local optimum. This original process, similar to ICP iterations, has been improved in many follow-up works [128, 49, 124, 94, 127, 89].
More precisely, these two steps are described here :

- *Functional map to point-wise map*: To convert the functional map $\mathbf{C}_{NM}$ back to a point-wise map $\varphi : M \to N$, we consider the spectral eigenbases of the two shapes: $(\Phi_i^M)_{i\in[0,k_M-1]}$ and $(\Phi_i^N)_{i\in[0,k_N-1]}$. These spectral embeddings are virtually *aligned* by the functional map, the point-wise map can then be recovered as follows :

$$\varphi(p) = \arg\min_q \left\| \mathbf{C}_{NM}\Phi_N(q) - \Phi_M(p) \right\|, \quad \forall p \in M, \tag{3.7}$$

where in practice this operation can be performed using a nearest neighbor step. The search space for this nearest neighbor is k_M-dimensional.

- *Point-wise map to functional map*: This simply boils down to following the operations of Section 3.4.1 and 3.4.2. Namely, we write the point-wise map φ as a permutation matrix Π_{NM} with $\Pi_{NM}(i, j) = 1$ if $\varphi(i) = j$ and 0 otherwise, where i and j are vertex indices on shape M and N, respectively. This corresponds to a discretization of the pull-back operator introduced in Section 3.4.1. The conversion then simply boils down to writing the previous operator in the spectral basis:

$$\mathbf{C}_{NM} = \Phi_M^{\dagger} \Pi_{NM} \Phi_N \tag{3.8}$$

In this manuscript, we will often use the most efficient refinement method in the near-isometric case, called Zoomout [94], which only requires a small functional map (typically $k = 30$ or even less) as initialization. At each step of the iteration, Zoomout takes the map φ back to the spectral domain (Eq. (3.8)), then k is raised, thus getting a bigger, more precise functional map. This results in a strong regularization of all the principal submatrices of the refined functional map.

Discussion

Despite its simplicity and efficiency, being a sequential framework, the functional map estimation pipeline described above is fundamentally error prone, due to the initial choice of descriptor functions. Building descriptors that are robust through change in pose, character, or even in species in the case animals, poses a key challenge. To alleviate this dependence and address this bottleneck, several approaches have been proposed to *learn* an optimal transformation of initial descriptors from data, whether in a supervised or unsupervised manner [31, 85, 132].

These works principally aim at transforming a given set of descriptors so that the optimal computed map satisfies some desired criteria during training. This transformation can be learned so that output descriptors will be robust across the kind of transformations between shape pairs of the training set.

CHAPTER 4

Deep Geometric Functional Maps: Robust Feature Learning for Shape Correspondence

We present a novel learning-based approach for computing correspondences between non-rigid 3D shapes. Unlike previous methods that either require extensive training data or operate on handcrafted input descriptors and thus generalize poorly across diverse datasets, our approach is both accurate and robust to changes in shape structure. Key to our method is a feature-extraction network that learns directly from raw shape geometry, combined with a novel regularized map extraction layer and loss, based on the functional map representation. We demonstrate through extensive experiments in challenging shape matching scenarios that our method can learn from less training data than existing supervised approaches and generalizes significantly better than current descriptor-based learning methods. Our source code is available at: https://github.com/LIX-shape-analysis/GeomFmaps.

4.1 Introduction

Shape correspondence is a key problem in computer vision, computer graphics and related fields with a broad range of applications, including texture or deformation transfer and statistical shape analysis [18], among many others. While classical correspondence methods have been based on handcrafted features or deformation models [159], more recent approaches have focused on *learning* an optimal model from the data either in supervised [31, 168, 85, 55] or even unsupervised settings [61, 132, 56].

Despite significant progress in recent years, however, learning-based approaches

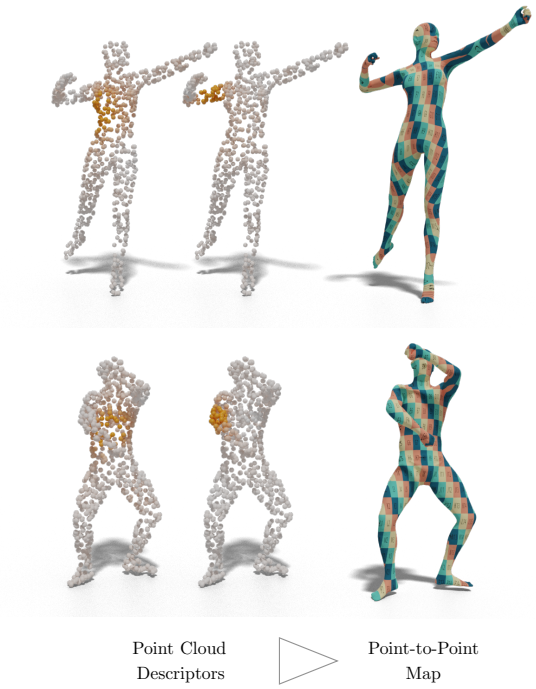


Figure 4.1: Given a pair of shapes, our approach builds consistent descriptors directly from the underlying point clouds (left), and automatically computes an accurate point-wise correspondence (right).

for shape correspondence typically require large amounts of training data in order to learn a model that generalizes well to diverse shape classes [168, 55]. Several existing methods address this challenge by learning a derived representation, through a non-linear transformation of pre-computed feature descriptors [31, 85, 61, 132], rather than on the geometry of the shapes themselves. Unfortunately, as we demonstrate below, this reliance on *a priori* hand-crafted descriptors makes the resulting learned models both less robust and less accurate leading to a significant drop in generalization power to new shape classes or instances.

In this work, we propose an approach that combines the power of learning directly from the 3D shapes with strong regularization based on a novel spectral correspondence extraction layer. Our method is inspired by recent learning techniques employing the functional map representation [85, 132]; however, we extend them to learn the features from 3D geometry rather than from some pre-computed descriptors. Furthermore, we introduce a regularizer into the functional map computation layer that greatly improves the speed and robustness of training. Finally, we demonstrate how the spectral loss based on the functional map representation in the reduced basis significantly reduces

over-fitting, while still leading to accurate correspondences coupled with recent post-processing techniques. As a result, our overall pipeline is both more robust and has greater generalization power than existing methods, while still being able to learn from limited training data.

4.2 Related Work

Computing point-to-point maps between two 3D discrete surfaces is a very well-studied area of computer vision. Below, we review those closest to our method, or with the best known results to serve as baselines, and refer to recent surveys [159, 16, 135] for an in-depth discussion.

Our method is built upon the functional map representation, which was originally introduced in [106] as a tool for non-rigid shape matching, and then extended in follow-up works [108]. The key property of this representation is being able to express maps as small matrices, encoded in a reduced basis, which greatly simplifies the associated optimization problems.

The original work used only a basic set of constraints on functional maps, which have been extended significantly in, e.g., [77, 2, 67, 47, 24, 129, 102, 68, 124] among many other works. These approaches both extend the generality and improve the robustness of the functional map estimation pipeline, by using regularizers, robust penalties and powerful post-processing of the computed maps.

A key challenge in all of functional map estimation techniques, however, is the strong reliance on given input *descriptors* used for computing the maps. Several approaches have suggested to use robust norms [77, 78], improved pointwise map recovery [128, 49] or more principled regularizers [125] which can help alleviate noise in the input descriptors to a certain extent but do not resolve strong inconsistencies in challenging cases.

More recent techniques have advocated learning optimal descriptors for functional map estimation directly from the data [31, 85]. These methods compute a transformation of given input descriptors so that the estimated functional maps are close to ground truth maps given during training. This idea was very recently extended to the *unsupervised* setting [61, 132] where the supervised loss was replaced with structural penalties on the computed maps.

Despite significant progress, however, in all of these cases, the descriptors are optimized through a transformation of *hand-crafted input features*, such as SHOT [158], Heat [151] or Wave kernel signatures [6]. This has two severe consequences: first, any information not present in the input features will be absent from the optimized descriptors, and second, such approaches generalize poorly across datasets as the input features can change significantly. This is particularly true of the commonly-used SHOT descriptors [158], which are sensitive to the triangle mesh structure and, as we show below, can

vary drastically across different datasets.

A number of other techniques have also been proposed for shape correspondence learning without using the functional map representation. These include approaches that exploit novel convolutional layers on triangle meshes [92, 21, 96, 113] and more general methods that use learning from depth-maps [168] or in some feature space [152, 29] among many others. Remarkably, relatively few methods aim to learn directly from the raw 3D shape geometry for shape correspondence, with the notable exceptions of [55, 36]. In large part this is due to the complexity of the correspondence problem, where unlike, e.g., shape segmentation, the number of labels can be unbounded. As a result, existing techniques address this either by learning from precomputed features, or relying on template-based matching and large training sets [55, 36], that might even require manual curation. Although PointNet [115] and its variants [116, 5, 156] achieve impressive results from raw point clouds for classification tasks, they are not yet competitive for shape correspondence task.

Contribution

In this chapter we show that feature learning for shape matching can be done directly from the raw 3D geometry even in the presence of relatively little training data, and without relying on a template or an *a priori* parametric (e.g., human body) model. Our main contribution is a end-to-end learnable pipeline that computes features from the 3D shapes and uses them for accurate dense point-to-point correspondence. We achieve this by introducing a novel map extraction layer using the functional map representation in a reduced basis, which provides a very strong regularization. Finally, we demonstrate that recent refinement techniques adapted to small functional maps [94], combined with our efficient learning pipeline jointly result in accurate dense maps at the fraction of the cost of existing methods.

4.3 Shape Matching and Functional Maps

One of the building blocks in our pipeline work is based on the functional map framework and representation. For completeness, we briefly review the basic notions for estimating functional maps, and refer the interested reader to a recent course [108] for a more in-depth discussion.

Basic Pipeline Given a pair of 3D shapes, M, N represented in a discrete setting as triangle meshes, and containing respectively n^M and n^N vertices, this pipeline aims at computing a map between them.

It consists in four main steps. First, the first few eigenfunctions of the discrete Laplace-Beltrami operator are computed on each shape, namely k_M and k_N functions respectively.

Second, a set of descriptor *functions* on each shape that are expected to be approximately preserved by the unknown map. For instance, a descriptor function can correspond to a particular dimension of the Heat or Wave Kernel Signatures [151, 6] computed at each point. Their coefficients are stored in the respective basis as columns in matrices $\mathbf{A}_M, \mathbf{A}_N$. In general and for the sake of readability, we will denote in bold letters quantities or tensors written in the reduced bases.

Third, the optimal *functional map* $\mathbf{C}$ is then computed by solving the following optimization problem:

$$\mathbf{C}_{\text{opt}} = \arg\min_{\mathbf{C}} E_{\text{desc}}(\mathbf{C}) + \lambda E_{\text{reg}}(\mathbf{C}), \tag{4.1}$$

where the first term aims at preserving the descriptors: $E_{\text{desc}}(\mathbf{C}) = \left\|\mathbf{C}\mathbf{A}_N - \mathbf{A}_M\right\|^2$, whereas the second term regularizes the map by promoting the correctness of its overall structural properties. It is common to use Fröbenius norm to compute the distance between these matrices. This Eq. (4.1) leads to a simple least-squares problem with $k_M \times k_N$ unknowns, independent on the number of points on the shapes.

As a last step, the estimated functional map $\mathbf{C}$, which maps across the spectral domains and converted to a point-to-point map. As a post processing step, called refinement, a number of advanced techniques are available [128, 49, 124, 94]. Most of them iteratively take the map from spectral to spatial domain, until it reaches a local optimum.

4.3.1 Deep Functional Maps

Despite its simplicity and efficiency, being a sequential framework, the functional map estimation pipeline described above is fundamentally error prone, due to the initial choice of descriptor functions. To alleviate this dependence, several approaches have been proposed to learn an optimal transformation of initial descriptors from data [31, 85, 132]. These works aim at transforming a given set of descriptors so that the optimal computed map satisfies some desired criteria during training. This transformation can be learned with a supervised loss, as in [31, 85], as well as with an unsupervised loss as in the more recent works of [61, 132].

More specifically, the FMNet approach proposed in [85] assumes to have as input, a set of shape pairs for which ground truth point-wise maps are known, and aims to solve the following problem:

$$\min_{\Theta} \sum_{(M,N)\in\text{Train}} l_F(\text{Soft}(\mathbf{C}_{\text{opt}}), \mathbf{C}^{gt}_{(M,N)}), \text{ where} \tag{4.2}$$

$$\mathbf{C}_{\text{opt}} = \arg\min_{\mathbf{C}} \|\mathbf{C}\mathbf{A}_{\mathcal{F}_\Theta(D_N)} - \mathbf{A}_{\mathcal{F}_\Theta(D_M)}\|^2 \tag{4.3}$$

Here, adopting the notation from [132] $\mathcal{F}_\Theta$ is a non-linear transformation, in the form of a neural network with parameters Θ, to be applied to some input descriptor functions D. The training set, called *Train* in Eq. (4.2) is the set of training *pairs* for which

ground truth correspondence $\mathbf{C}^{gt}_{(M,N)}$ is known. l_F is the *soft error loss*, which penalizes the deviation of the computed functional map $\mathbf{C}_{\text{opt}}$, after converting it to a soft map $\text{Soft}(\mathbf{C}_{\text{opt}})$ from the ground truth correspondence, and $\mathbf{A}_{\mathcal{F}_\Theta(D_S)}$ denotes the transformed descriptors D_S written in the spectral basis of shape $S \in \{M, N\}$. Thus, the FMNet framework [85] learns a transformation $\mathcal{F}_\Theta$ of descriptors $\mathcal{F}_\Theta(D_M)$, $\mathcal{F}_\Theta(D_N)$ based on a supervised loss that minimizes the discrepancy between the resulting soft map and the known ground-truth correspondence.

A related recent approach, SURFMNet [132] follows a similar strategy but replaces l_F with an *unsupervised loss* that enforces the desired structural properties on the resulting map, such as its bijectivity, orthonormality and commutativity with the Laplacian.

3D-CODED In contrast to the the methods described above that primarily operate in the *spectral* domain, there are also some approaches that never leave the spatial domain. With the recent works on point clouds neural networks, pioneered by PointNet [115], and significantly extended by [5, 156], to name a few, it is now possible to learn 3D features directly from point clouds. 3D-CODED [55, 36] is based on this approach, as it is a method built on a variational auto-encoder with a PointNet architecture for the encoder. Their method relies on a template that is supposed to be deformable in a non-rigid but isometric way to any of the shape of the datasets. It is a supervised method, and requires the knowledge of all ground-truth correspondences between any shape of the dataset and the deformable template. 3D-CODED is trained on 230K shapes, introduced in SURREAL [161], and generated with SMPL [88].

Motivation The two main classes of existing approaches have their associated benefits and drawbacks. On the one hand, spectral methods are able to use small matrices instead of all the points of the shape, and operate on *intrinsic* properties of the 3D surfaces, making them resilient to a change in pose, and allowing them to train on really small datasets. However, due to their use of input descriptors (typically SHOT [158]), they tend to overfit to the connectivity of the training set, which can lead to catastrophic results even in apparently simple cases. On the other hand, 3D-CODED shows extreme efficiency when trained on enough data, regardless of the connectivity, but with a small dataset, it is prone to overfitting and fails to generalize the training poses to predict the different poses of the test set.

Our method is a mix of the two approaches, and, as we show below, can obtain accurate results with little training data leading to state-of-the-art accuracy on a challenging recent benchmark of human shapes in different poses and with different connectivity [93].

4.4 Method

4.4.1 Overview

In this work, we introduce a novel approach to learn descriptors on shapes in order to get correspondences through the functional map framework. Our method is composed of two main parts, labeled as *Feat* and *FMReg* in Figure 4.2. The first aims at optimizing point cloud convolutional filters [5, 156] to extract features from the raw geometry of the shapes. These filters are learned using a Siamese network $\mathcal{F}_\Theta$ on the source and a target shapes by using shared learnable parameters Θ, in a similar way as in [85]. However, unlike that approach and follow-up works [61, 132] we learn the features directly from the geometry of the shapes rather than computing a transformation of some pre-defined existing descriptors. These learned descriptors are projected in the spectral bases of the shapes and fed to the second block of the method, which uses them in a novel regularized functional map estimation layer. Finally, we use a spectral loss, based on the difference between the computed and the ground truth functional maps. This makes our approach very efficient as it operates purely in the spectral domain, avoiding expensive geodesic distance matrix computations as in [85, 61] and moreover allows us to handle functional or soft ground truth input maps without requiring the training shapes to have the same number of points or fixed mesh connectivity.

We stress again that the two components: learning features directly from the shapes and using the functional map representation both play a crucial role in our setup. The former allows us to learn robust and informative features independently from the mesh structure, while the latter allows us to strongly regularize correspondence learning, resulting in a method that generalizes even in the presence of a relatively small training set.

4.4.2 Architecture

The novelty of our architecture lies in its hybrid character. The first part, which we will refer to as the *feature extractor* in the following, aims at computing point-wise features on the input shapes. It corresponds to the *Feat* block in Figure 4.2, and takes as input only the point clouds making it robust towards changes in connectivity.

The purpose of the second part is to recover robust functional maps using these learned features. This block is built according to the pipeline of [106], first taking the features to the spectral domain over the two shapes (which corresponds to the dot products blocks after the *Feat* blocks in Figure 4.2), and then computing the map by minimizing an energy. However, since our method is based on a neural network, this operation should be differentiable with respect to the features over the shapes for the back-propagation algorithm to work. We extend the previously proposed functional map layers [85] to also incorporate a differentiable regularizer, which results in the very robust map extraction, represented as *FMReg* in Figure 4.2.

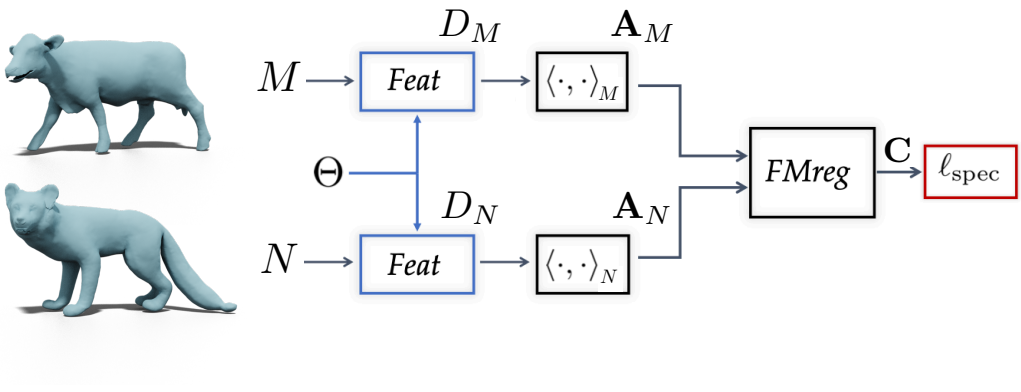


Figure 4.2: Overview of our approach: given a pair of shapes, we optimize for a point cloud convolutional model to get point-wise features for each shape, that we convert to a functional map using our FMReg block. The loss that we put forward penalizes maps according to their distance to the ground-truth map between the two shapes.

4.4.3 The feature extractor

The goal of this block is to learn functional characterizations of point clouds that will later be used to compute spectral descriptors and then functional maps. To this end, this network must be applied with the same weights to the source and target shapes, as represented in Figure 4.2, and must result in informative descriptors, extracted from the point clouds of the two shapes.

For this part, we chose the state of the art point cloud learning method KPConv [156], by extending the segmentation network proposed in that work. Our feature extractor is thus a Siamese version of the segmentation network described in KPConv, which we review for completeness in appendix A.1.

4.4.4 The regularized FMap layer

This block provides a novel fully differentiable way to compute a robust functional map from potentially low dimensional spectral descriptors.

The main goal is, as in Section 4.3, to recover the ground-truth bijection between M and N, on which we have the computed raw-data features D_M and D_N.

For this we first express the computed feature functions in the respective spectral basis, which we denote by Φ_M and Φ_N. This leads to the spectral descriptors $\mathbf{A}_M = (\Phi_M)^\dagger D_M$ and $\mathbf{A}_N = (\Phi_N)^\dagger D_N$, with $\Phi^\dagger$ the Moore pseudo-inverse of Φ. We stress again that this step is where we shift focus from spatial to spectral domain, and corresponds to the dot product blocks in Figure 4.2.

In the pipeline first introduced in [106] and then widely used in the follow-up works [108], the standard strategy is to compute the functional map $\mathbf{C}$ that optimizes the following energy:

$$\min_{\mathbf{C}} \left\|\mathbf{C}\mathbf{A}_N - \mathbf{A}_M\right\|^2 + \lambda\left\|\mathbf{C}\mathbf{\Delta}_N - \mathbf{\Delta}_M\mathbf{C}\right\|^2, \tag{4.4}$$

where λ is a scalar regularization parameter.

Remark that the optimization problem in Eq. (4.4) is quadratic in terms of $\mathbf{C}$ and can be solved e.g. via standard convex optimization techniques. However, in the learning context, we need to differentiate the solution with respect to the spectral features $\mathbf{A}_M, \mathbf{A}_N$, which is challenging when $\mathbf{C}$ is computed via an iterative solver. Alternatively, the problem in Eq. (4.4) can be written directly in terms of a large least squares system, by vectorizing the matrix $\mathbf{C}$ as was suggested in [106]. However, for a $k \times k$ functional map, this leads to a system of size $k^2 \times k^2$ which becomes prohibitive even for moderate values of k. To avoid these issues, previous learning-based approaches based on functional maps [85, 61, 132] have only optimized for $\mathbf{C}$ using the first part of the energy in Eq. (4.4): $\left\|\mathbf{C}\mathbf{A}_N - \mathbf{A}_M\right\|^2$. This results in a simple linear system for which the derivatives can be computed in closed form. This has two major limitations, however: first the linear system is only invertible if there are at least k linearly independent feature functions. This condition can easily be violated in practice, especially in the early stages of learning, potentially resulting in a fatal error. Furthermore, the lack of regularization makes the solved-for functional map very sensitive to inconsistencies in the computed descriptors, which leads to an overall loss of robustness.

In our work we address this problem by using the full energy in Eq (4.4) in a fully differentiable way. In particular, we use the fact that the operators Δ_M and Δ_N are diagonal when expressed in their own eigen-basis.

Indeed we remark that the gradient of the energy in Eq (4.4) vanishes whenever $\mathbf{C}\mathbf{A}_N\mathbf{A}_N^T + \lambda\mathbf{\Delta} \circ \mathbf{C} = \mathbf{A}_M\mathbf{A}_N^T$, where the operation $\circ$ represents the element-wise multiplication, and $\mathbf{\Delta}_{ij} = (\mu_j^N - \mu_i^M)^2$, where μ_l^M and μ_l^N respectively correspond to the l-th eigenvalue of Δ_M and Δ_N. It is then easy to see that this amounts to a separate linear system *for every row* c_i of $\mathbf{C}$:

$$(\mathbf{A}_N\mathbf{A}_N^T + \lambda\text{diag}((\mu_j^N - \mu_i^M)^2))c_i = \mathbf{A}_N b_i \tag{4.5}$$

where b_i stands for i^{th} row of $\mathbf{A}_M$.

In total, if k is the number of eigenvectors used for representing the functional map, this operation amounts to inverting k different $k \times k$ matrices. Since inverting a linear system is a differentiable operation, which is already implemented e.g. in TensorFlow, this allows us to estimate the functional map in a robust way, while preserving differentiability.

4.4.5 The supervised spectral loss

Our method also uses a loss with respect to the ground truth functional map in the spectral domain. This is similar to the energy used in [31], but is different from the loss of the original FMNet work [85], which converted a functional map to a soft correspondence matrix and imposed a loss with respect to the ground truth point-wise map, relying on expensive geodesic distance matrix computation.

Specifically, calling $\mathbf{C}$ the functional map obtained by the FMap block, and $\mathbf{C}^{gt}$ the ground truth spectral map, our loss is defined as:

$$l_{\text{spec}} = \left\| \mathbf{C} - \mathbf{C}^{gt} \right\|^2$$

As mentioned above, we use a Fröbenius norm to compute the distance between matrices.

It is important to note that whenever a pointwise ground truth map is given it is trivial to convert it to the functional map representation. Conversely, the ground truth spectral map is more general than the point-wise ground truth correspondence. Indeed, with just a few precise landmarks one can recover a functional map accurate enough to make this loss efficient, for instance through the original pipeline of [106], but also with more recent follow-up works, such as [124] or [94], which we will further describe as baselines to our method in Section 4.5.

This is useful, e.g., in the case of re-meshed datasets. Indeed, complete ground truth correspondences between two shapes of these datasets are not fully known. One can only have access to the (often partial and not bijective) ground truth pointwise map from a template mesh $\mathcal{T}$ to each re-meshed shape $\mathcal{S}_i$. Each such map can be converted to a functional map $\mathbf{C}_i$ and a very good approximation of the spectral ground truth $C^{gt}_{i \to j}$ between $\mathcal{S}_i$ and $\mathcal{S}_j$ can be expressed as $\mathbf{C}_j^{\dagger}\mathbf{C}_i$.

4.4.6 Postprocessing

Once our model is trained, we can then test it on a pair of shapes and get a functional map between these shapes. This map can either directly be converted to a point to point map, or refined further. We use a very recent and efficient refining algorithm, called ZoomOut [94] based on navigating between spatial and spectral domains while progressively inceasing the number of spectral basis functions. This efficient postprocessing technique allows us to get state-of-the-art results, as described in Section 4.5.

4.4.7 Implementation

We implemented our method in TensorFlow [1] by adapting the open-source implementation of SURFMNet [132] and KPConv [156].

Our feature extraction network is based on a residual convolutional architecture of [156], consisting of 4 convolutional blocks with leaky linear units, with successive poolings and dimension augmentation from 128 to 2048, followed by a 4 up-sampling blocks with shortcuts from corresponding pooling layers, and dimension reduction from 2048 back to 128. Please see the supplementary materials, part A, in [156] for more details. Following the pipeline of KPConv, we start with a sub-sampled version of our point clouds with a grid subsampling of step 0.03. The pooling layers are therefore obtained with grid samplings of parameters 0.06, 0.12, 0.24 and 0.48.

Similarly to FMNet [85] and SURFMNet [132], our network is applied in a Siamese way on the two shapes, using the same learned weights for the feature extractor.

In the case of fully automatic spectral methods such as BCICP [124] and ZoomOut [94], or the deep learning based FMNet [85, 61] (supervised or unsupervised) and SURFMnet [132], all results are invariant by any rigid transformation of the input shapes. However, in the case of methods using the 3D coordinates of the points to generate information about the input shape, this does not remain true. Consequently, both 3D-CODED [55] and our method avoid this dependency through data augmentation to be as close as possible to the generality of fully spectral methods. To that end, assuming the shapes are all aligned on one axis (e.g. on human the natural up axis), both 3D-CODED and our method perform data augmentation by randomly rotating the input shapes along that axis.

4.4.8 Parameters

In addition to the architecture above, our method has two key hyper-parameters: the size of the functional basis and the regularizer λ in Equation 4.5. For the size of the basis, we discovered if this number is too high, for instance, with 120 eigenvectors as in FMNet and SURFMNet, it can easily lead to overfitting. However, by reducing this number to 30, the results of SURFMNet on FAUST re-meshed (here reported in Table 4.1) go from 0.15 to 4.5. As a consequence, we choose the number of eigenvectors to be 30 in all of our experiments on our method. Regarding the weight λ in Equation (4.5), we observed that setting it to $\lambda = 10^{-3}$ helps getting good results while drastically reducing the number of training steps, as pointed out in the ablation study. We use this value throughout all experiments.

We train our network with a batch size of 4 shape pairs for a number of epochs depending on the number of shapes in the dataset. We use a learning rate of 10^{-3} and gradually decreasing it to 10^{-4} with ADAM optimizer [42].

Method \ Dataset	F	S	F on S	S on F
BCICP	15.	16.	*	*
ZoomOut	6.1	7.5	*	*
SURFMNet	15.	12.	32.	32.
SURFMNet+icp	7.4	6.1	19.	23.
Unsup FMNet	10.	16.	29.	22.
Unsup FMNet+pmf	5.7	10.	12.	9.3
FMNet	11.	17.	30.	33.
FMNet+pmf	5.9	6.3	11.	14.
3D-CODED	2.5	31.	31.	33.
Ours	3.1	4.4	11.	6.0
Ours+zo	**1.9**	**3.0**	**9.2**	**4.3**

Table 4.1: Comparative results ($\times 100$) of the different methods on Experiment 1.

4.5 Results

Datasets

We test our method on a wide spectrum of human datasets: first, the re-meshed versions of FAUST dataset [18] containing 100 human shapes in 1-1 correspondence, and of SCAPE [3], made publicly available by Ren et al. [124]. These re-meshed datasets offer significantly more variability in terms of shape structures and connectivity, including for instance point sampling density, making them harder to match for existing algorithms. We also highlight that the SCAPE dataset is slightly more challenging since the shapes are less regular, and two shapes never share the same pose. This is not true for FAUST, wherein all the poses present in the test set also exist in the training set, with the variation coming from body type only, making the pose recovery easier at test time.

We also use the re-meshed version of the more recent SHREC'19 dataset [93], which, in theory, is the most challenging of the test sets, because of stronger distortions in the poses, the presence of an incomplete shape, and the number of test pairs (430 in total, so two times the number of test pairs of FAUST or SCAPE). At last, we also use the generic training dataset of 3D-CODED [55], originally consisting in 230K synthetic shapes generated using Surreal [161], with the parametric model SMPL introduced in [88]. We use it only for training purposes in our second experiment, to show that our method can generalize well to changes in connectivity, being able to train on a synthetic, very smooth, identical triangulation for the whole training set, and still produce results of excellent quality on re-meshed datasets.

Ablation study

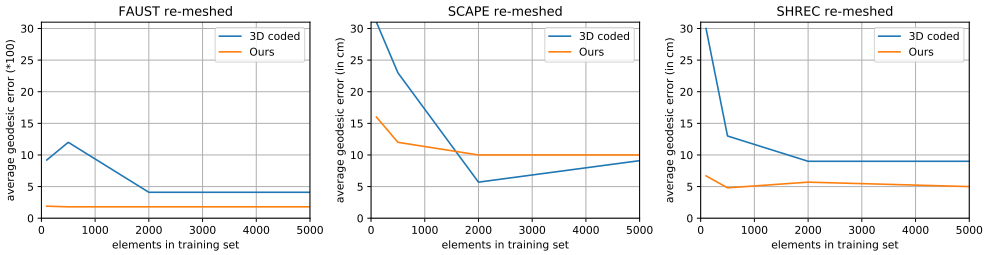


Figure 4.3: Comparison with 3D-CODED while varying training size of SURREAL dataset and simultaneously testing on other datasets.

Our method is built with a number of building blocks, all of which we consider essential to achieve optimal performance. To illustrate this, we provide an extensive ablation study of all the key components of our algorithm in appendix A.2.

Baselines

We compare our method to several state of the art methods: the first category includes fully automatic methods without any learning component [124, 94]. These methods are simply evaluated on the test sets without any training. The second category includes FMNet [85] and its unsupervised versions, referred to as Unsup FMNet [61] and SURFMNet [132], with and without post-processing (PMF [164] for FMNet, and standard functional map refinement [106], referred to as ICP, for SURFMNet). All these variants of FMNet give similar results, but SURFMNet is the only one to train within a few hours, without requiring too much space. This is due to the fact SURFMNet only operates in the spectral domain, in contrast to other methods. Lastly, we compare to the supervised 3D-CODED [55], described earlier in more details in Section 4.3. For conciseness, we refer to our method as Ours in the following text. We show our results with and without ZoomOut [94] refinement, referred to as ZO, in order to prove that our method stands out even without post processing. We compare these different methods in two main settings named Experiment 1 and Experiment 2 below.

Experiment 1 consists of evaluating the different methods in the following setting: we split FAUST re-meshed and SCAPE re-meshed into training and test sets containing 80 and 20 shapes for FAUST, and 51 and 20 shapes for SCAPE. We obtain results for training and testing on the same dataset, but also by testing on a different dataset. For instance, by training on SCAPE re-meshed train set and testing on FAUST re-meshed test set. This experiment aims at testing the generalization power of all methods to small re-meshed datasets, as well as its ability to adapt to a different dataset at test time.

Experiment 2 consists of sampling 100, 500, 2000, and 5000 shapes from the SURREAL dataset to be used for training. We then test the trained models on the test sets of FAUST re-meshed, SCAPE re-meshed, and SHREC19 re-meshed. This experiment

aims at testing the robustness and generalization power of the different methods in the presence of varying amounts of training data, as well as adaptability to train on a perfect synthetic triangulations and still get results on challenging re-meshed shapes.

Quantitative results

To evaluate the results, we use the protocol introduced in [72], where the per-point-average geodesic distance between the ground truth map and the computed map is reported. All results are multiplied by 100 for the sake of readability.

As we can see in Table 4.1, our method performs the best overall on Experiment 1. Fully automatic methods do not provide competitive results compared to the learning methods (except on crossed settings because they did not train on anything and are thus not influenced by the training shapes). As reported in the Section 4.3, this highlights that hand-crafted features can easily fail. It is noticeable that spectral methods (FMNet variations, and Ours as a hybrid method) get reasonable, or even good results in our case, with these small datasets. In comparison, 3D-CODED seems to fail in almost all cases. It is remarkable that it can learn on such a small dataset as the training set of FAUST re-meshed. One explanation for that is that FAUST contains the same set of poses in the test set as in the train set.

Contrary to other baselines, our method gives good results on all settings, *even without refinement*, showing good resilience to a really low number of shapes, even with re-meshed geometry. We would like to stress that no other method is able to achieve such a generalization with this low number of shapes.

For a fair comparison with 3D-CODED, we complete our study with a second experiment, in which the training set is now made of the same shapes 3D-CODED uses for training in their paper, namely SURREAL dataset. The aim of this experiment is to further showcase the generalization power of our method when compared to 3D-CODED. First, by training on a very smooth synthetic dataset, on which previous fully spectral methods tend to easily overfit due to the obvious mismatch in triangulation in training and test set. Our second goal is to observe the dependence of different methods on size of the training set.

We report the results (multiplied by 100) of 3D-CODED and Our method in Figure 4.3, as they are the only two competitive algorithms in Experiment 2. These results once again demonstrate that our method can achieve impressive results even with a low number of training shapes. On SHREC re-meshed, we achieve state of the art results with an average error of $\mathbf{0.048}$ with only 500 training shapes. We provide additional quantitative comparisons in appendix A.3.

It can be observed in Figure 4.3 that our results are consistent and unaltered even with the drop in number of training shapes. 3D-CODED, on the other hand, always suffers from a reduced training set.

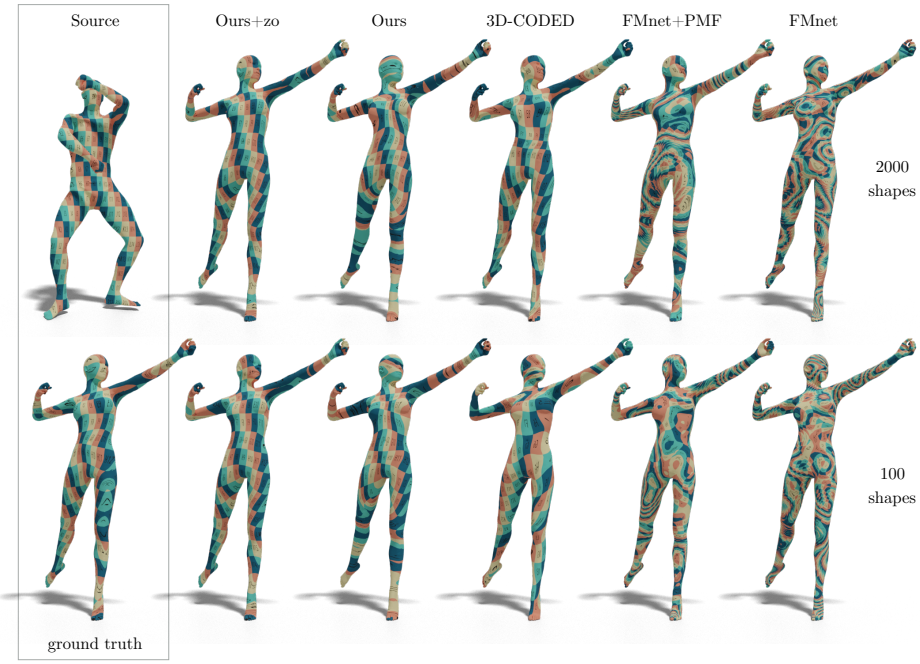


Figure 4.4: Qualitative results obtained with texture transfers for the different methods on Experiment 2, training on two different numbers of shapes in the SURREAL dataset, and testing on SHREC re-meshed shapes.

Qualitative results

In Figure 4.4 we show the results of our method (with and without ZoomOut refinement [94]), 3D-CODED [55], FMNet [85] (with and without PMF refinement [163]), trained on respectively 2000 and 100 shapes, as presented in Experiment 2, via texture transfer.

With 2000 training shapes, both our method and 3D-CODED lead to good or even excellent texture transfers, while fully spectral methods fail due to the change of connectivity from training to test set. However, with only 100 training shapes, 3D-CODED fails to get a good reconstruction in many cases, leading to bad texture transfer as in Figure 4.4. This highlights the fact that our method performs better than any other existing method when only a few training shapes are provided.

4.6 Conclusion, Limitations & Future Work

We presented a method for improving the robustness and reducing overfitting in learning shape correspondences. Key to our approach is a hybrid network structure, made of a raw-data feature extractor that learns descriptors on a pair of shapes, and a novel robust

functional map layer. Our network can thus operate in both the spectral and the spatial domain, thus taking advantages of both representations.

Our approach has several limitations: first, as a supervised method it requires at least partial correspondences (as discussed in Section 4.4.5) between the training shapes. Also, it requires data augmentation to be able to predict non-aligned shapes, which can be costly and unstable.

In the future, we plan to work towards an unsupervised spectral loss, similar in spirit to SURFMNet [132], while avoiding the symmetry ambiguity problem. We also plan to try other, invariant feature extractors such as [58], or [114] to avoid data augmentation.

Acknowledgements This work was supported by KAUST OSR Award No. CRG-2017-3426, a gift from Nvidia and the ERC Starting Grant No. 758800 (EXPROTEA). We are grateful to Jing Ren, Simone Melzi and Riccardo Marin for the help with re-meshed shapes, and SHREC'19 dataset.

CHAPTER 5

Complex Functional Maps: a Conformal Link Between Tangent Bundles

In this chapter, we introduce complex functional maps, which extend the functional map framework to conformal maps between tangent vector fields on surfaces. A key property of these maps is their *orientation awareness*. More specifically, we demonstrate that unlike regular functional maps that link *functional spaces* of two manifolds, our complex functional maps establish a link between *oriented tangent bundles*, thus permitting robust and efficient transfer of tangent vector fields. By first endowing and then exploiting the tangent bundle of each shape with a complex structure, the resulting operations become naturally orientation-aware, thus favoring *orientation and angle preserving correspondence* across shapes, without relying on descriptors or extra regularization. Finally, and perhaps more importantly, we demonstrate how these objects enable several practical applications within the functional map framework. We show that functional maps and their complex counterparts can be estimated jointly to promote orientation preservation, regularizing pipelines that previously suffered from orientation-reversing symmetry errors.

5.1 Introduction

Non-rigid shape matching is a well-established challenge in computer graphics, geometry processing and related fields [159, 136], with applications ranging from medical imaging to statistical shape analysis, to name a few.

One prominent direction for addressing this problem is given by the functional map

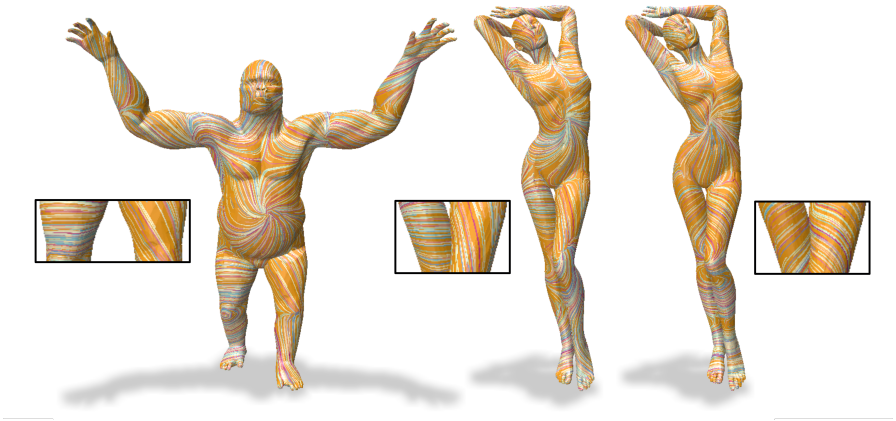


Figure 5.1: A comparison to [8] in a vector field transfer application. The transfer is done with a functional map mixing symmetries (see Section 5.5.1). Unlike the method of [8], our approach is robust to this type of noise and transfers the vector field correctly, without breaking its asymmetry.

framework [106]. This framework is based on representing correspondences as linear transformations between function spaces, and encoding them as matrices using a reduced basis. A key advantage of this construction is that it allows to both optimize for and to manipulate mappings by solving small-scale optimization problems, whose complexity is largely independent of the size of the underlying meshes. Furthermore, the continuous nature of this representation enables the use of differentiable optimization techniques, which has recently proven useful in learning pipelines, e.g., [85, 62, 52].

Despite the flexibility and simplicity of the functional map representation, it has several key limitations: first, while functional maps encode correspondences between points, they do not immediately provide access to maps between derived quantities such as the surface metric or tangent vector fields that require the notion of a map differential. Several attempts have been made to recover differential information in the functional map framework, e.g., [8, 33]. However, these approaches often lead to non-trivial optimization problems and, as we demonstrate below, can be prone to error especially when faced with approximate maps in the reduced basis (see Figure 5.1). Perhaps even more importantly, the functional map representation does not encode information about the *surface orientation*, which means that standard functional map optimization energies can easily lead to orientation-reversing correspondences that may arise, e.g., due to intrinsic symmetries. Existing methods try to tackle this challenge through a range of solutions including by using landmarks [103, 90], injecting orientation into descriptor-based energies [124] or alignment in the ambient space [126] among others. Unfortunately, these solutions often require additional user input and careful parameter tuning or incur significant computational cost.

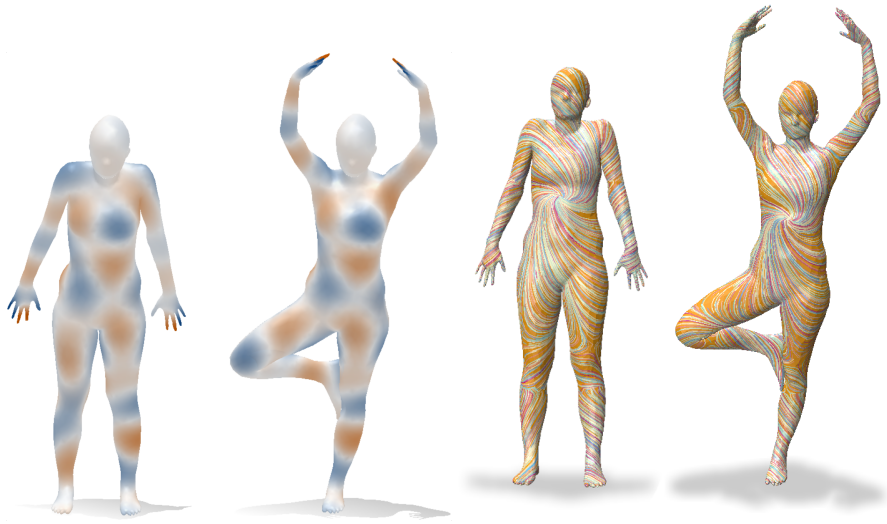


Figure 5.2: A visualization of function transfer (left) and vector field transfer (right) obtained respectively via a functional map and our *complex functional map*. The values of the functions on the left are encoded by a colormap, while the vector fields are visualized using line integral convolution. Both functions and vector fields can be decomposed in a Laplacian basis and be transferred using small matrices.

In this work, we introduce a novel construction that helps to address these challenges directly, without relying on user input or post-processing. Our key idea is to build a representation for correspondences that *only allows* orientation and angle-preserving (conformal) maps. Such a representation is, by its nature, more restrictive than the functional map representation, since only a subset of maps is allowed. However, as we demonstrate below, using this representation and especially the link with standard functional maps helps to regularize map computation and to improve accuracy in several applications without sacrificing expressive power.

To achieve this goal, we first observe that in the case of conformal maps, the push-forward (also called "map differential") is a *complex linear* operator between tangent bundles. As a result, it can be encoded as a small-sized complex-valued matrix, given a choice of basis for the tangent vector fields on each shape. Since a single vector field can be represented as *a complex function*, and the pushforward allows to transfer vector fields across shapes, we call these operators *complex functional maps*. As we highlight below, complex functional maps have several properties that are complementary to the standard functional map representation. Specifically, they provide a simple and robust tool for transferring vector fields; furthermore, they allow to disambiguate intrinsic symmetries and help to promote more accurate, smooth, orientation-preserving point-to-point correspondences.

Contributions To summarize, our main contributions include:

- We introduce a novel complex-linear operator acting on tangent vector fields, used as relaxation of the map differential.
- We demonstrate that this operator is naturally orientation-aware, and show how it can be used to regularize functional map estimation, especially with respect to symmetry errors.
- We highlight the utility of our construction in a range of applications, from robust vector field transfer to orientation-aware map refinement, leading to consistent improvement in accuracy.

5.2 Related Work

Computing maps or correspondences between non-rigid 3D shapes is a key challenge in geometry processing and computer vision. Below, we review some approaches in this area and focus especially on methods that we either build upon or use as baselines, and refer to recent surveys [159, 16, 136] for a more in-depth discussion.

Functional maps framework Our method heavily relies on the functional map representation, which was originally introduced in [106] as a tool for non-rigid shape matching. The key idea of this representation is to represent point-to-point maps as small matrices, encoded in a reduced basis, which greatly simplifies subsequent optimization problems. The original work used only a basic set of linear constraints on functional maps, which have been extended significantly in, e.g., [77, 2, 67, 48, 24, 129, 103, 68, 104, 124, 94] among many other works (see [108]). These approaches heavily exploit the compact and continuous nature of the functional map representation and have been used to improve the accuracy, speed and robustness of the resulting shape matching pipeline. However, the functional representation itself has two major limitations that limit its applicability, as we review below.

Functional representations of differential quantities As mentioned in the introduction, functional maps do not naturally provide information about derived quantities such as the map differential. Several attempts have been made to recover differential information in the functional map framework. This includes operator representations for tangent vector fields [8] and cross fields [11] as well as covariant derivatives and parallel transport [9]. These operator representations enable tasks such as computing vector field flow efficiently, which has been used both for matching functions on surfaces [10] and even for recovering continuous maps between shapes [32]. Furthermore, functional representations have been also proposed for extrinsic "deformation fields" [30] and for capturing the intrinsic shape metric [134, 33], among others (see also Chapter 6 in [108] and [14] for an overview of some recent approaches).

These constructions significantly extend the power and flexibility of the functional maps framework. At the same time, the basic concept of the map differential or pushforward, and thus reliable mapping of *tangent vector fields* is still cumbersome to define and use within this formalism. As we demonstrate below, in the special case of orientation-preserving conformal maps, however, this differential has a particularly convenient representation, which provides complementary information to standard functional maps.

Orientation preservation Another common challenge to using the functional map representation is that it does not encode surface orientation. This implies that typically-used optimization energies, e.g., based on preservation of intrinsic descriptors, such as the HKS or WKS [151, 6] can lead to undesirable orientation-reversing correspondences (also known as symmetry flips). One can tackle this issue by using extrinsic descriptors such as SHOT [158], but such descriptors can be very sensitive to discretization [112, 39], which can have dramatic effects on robustness. Existing methods have tried to address this challenge by using either segment [106, 75] or pointwise landmarks [103, 90], injecting orientation information with descriptor-based energies [124], factoring the functional space using symmetry information [107], alignment in the ambient space [44] or, most recently, using map space exploration strategies [126]. However, since the functional map representation itself does not encode orientation information, these solutions only address the problem indirectly, and, e.g., in [126] orientation-preserving maps are selected *a posteriori* among the set of candidates, using a set of filtering criteria.

In contrast, we demonstrate that by first endowing the tangent bundle with the complex algebra using the outward normals, and then defining the derived *complex functional maps* it becomes possible to directly promote orientation-preserving correspondences without any additional descriptor preservation constraints or post-processing.

Vector Field Map Representation Closest to our construction is the work of [167], where the authors extend the functional map representation to differential forms on manifolds. Similarly to our approach, that method encodes the pushforward as a linear operator acting on vector fields, while imposing orthogonality, which is a necessary condition to arise from a conformal map. However, crucially, the authors of [167] use $\mathbb{R}$-linear operators to encode the pushforward, whereas we use the outward normals to construct and exploit the *complex algebra* on the tangent space, leading to $\mathbb{C}$-linear operators. This difference is fundamental, as it implies that the representation in [167] *cannot* distinguish orientation-preserving from orientation-reversing maps. As we demonstrate in Section 6.5, this severely limits the scope of applications of that representation, which are, in contrast, enabled by our approach thanks, in particular, to its orientation-aware nature.

5.3 Tangent Bundle Map as Operator

In this section, we describe the theoretical aspects of our complex functional maps. At a high level, we follow the motivation behind the original functional map framework, by ultimately providing a linear relaxation of a particular geometric concept. While functional maps aim at representing diffeomorphisms as linear operators acting on functions, our goal is to represent the pushforward of a conformal map as a $\mathbb{C}$-linear operator acting on complex fields.

5.3.1 Notation

From now on, we consider a pair of compact Riemannian surfaces M, N embedded in $\mathbb{R}^3$. We use T_pM to denote the tangent plane at a point $p \in M$, while the *tangent bundle* $TM := \cup_{p \in M} T_pM$, is the disjoint union of all the tangent planes of M. We equip this space with a proper inner product (i.e. the Riemannian metric) $\langle \cdot, \cdot \rangle_{T_pM} : T_pM \times T_pM \to \mathbb{R}$, which depends smoothly on $p \in M$. For every pair of real-valued functions $f, g : M \to \mathbb{R}$, we use another, L^2 inner product, which is defined as $\langle f, g \rangle_M = \int_M f(p)g(p)d\mu_M(p)$, where $d\mu_M$ is the area element on the surface M. With respect to this inner product, we define $L^2(M) = \{f : M \to \mathbb{R} \; s.t. \; \langle f, f \rangle_M < +\infty\}$, as the space of square-integrable real-valued function defined over M. Similarly for every pair of vector fields $X, Y : M \to TM$ we consider their inner product $\langle X, Y \rangle_{TM} = \int_M \langle X(p), Y(p) \rangle_{T_pM} d\mu_M(p)$. Finally we denote by $L_M : TM \to TM$ the connection Laplacian acting on vector fields.

5.3.2 Pushforward in the smooth setting

A diffeomorphism $\varphi : M \to N$ bijectively maps points on a surface M to points on a surface N. The pushforward $\mathrm{d}\varphi : T_pM \to T_{\varphi(p)}N$ associated to φ maps the tangent space at point $p \in M$, denoted as T_pM, to the tangent space $T_{\varphi(p)}N$ and can be understood as the best linear approximation of the map at the given point p (see Figure 5.3 top). Thus, the pushforward contains two pieces of information: 1) which tangent plane of N corresponds to a given tangent plane of M and 2) *how* a tangent plane is deformed by the diffeomorphism φ. While the first is already contained in φ, the second is of very different nature, and is especially difficult to recover in the discrete setting as it requires to numerically differentiate the mapping.

For surfaces, tangent planes can be identified to the vector space $\mathbb{R}^2$. Given an arbitrary diffeomorphism, the pushfoward acts *linearly* on the tangent vectors (see [59], Chapter 2) so at that each point $p \in M$, $(\mathrm{d}\varphi)_p$ can be represented as a linear map between two Euclidean spaces. This map can be represented by a 2×2 matrix if each space is endowed with a local 2D basis formed by two linearly independent tangent vectors. For a more in-depth discussion of map differentials, we refer to [59, 80].

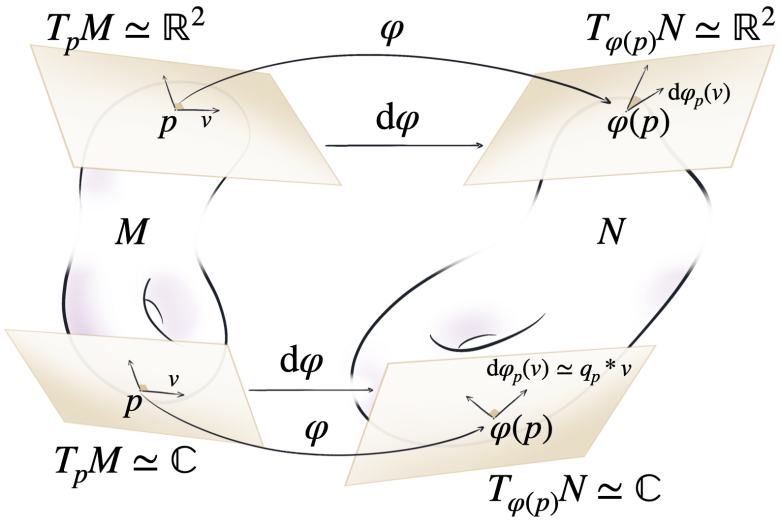


Figure 5.3: The pushforward $\mathrm{d}\varphi$ takes a tangent vector $v \in T_pM$ and transports it to the plane tangent to $\varphi(p) \in N$. In all generality (top) surface tangent planes identify to $\mathbb{R}^2$ and the pushforward is locally a $\mathbb{R}^2$-linear map. If φ is conformal (bottom), tangent planes identify to the complex plane $\mathbb{C}$ and $\mathrm{d}\varphi$ is locally a multiplication by a complex number $q_p \in \mathbb{C}$.

5.3.3 Pushforwards of conformal maps

In this chapter, we are especially interested in the pushforward of a conformal map. Conformal maps have been widely used in computer graphics notably for texture mapping [121, 149], parametrization [71, 101, 138] and shape matching [84, 73] among others. Their success is mostly due to their simple structure-preserving property: a conformal map preserves angles between tangent vectors. This means that, by definition, the pushforward of a conformal map is a *similarity transformation* (i.e., a combination of rotation and uniform scaling) of the tangent space at every point.

Each tangent plane on an orientable surface can also be naturally identified with the complex plane $\mathbb{C}$ by identifying an arbitrary fixed direction with the real axis, and using the outward normal to determine the $90°$ counter-clockwise rotations, associated with multiplication by $\imath$. Moreover, observe that for any fixed complex number q, the transformation $S : \mathbb{C} \to \mathbb{C}$ given by complex multiplication by q, and defined as $S(v) = q*v$, is a *similarity transformation* of $\mathbb{R}^2$. To see this, observe that in polar coordinates complex multiplication simply adds the arguments (angles) and multiplies the magnitudes. Conversely, it is easy to see that for any similarity transformation $S : \mathbb{R}^2 \to \mathbb{R}^2$, there exists a unique complex number q such that $S(v) = q * v$, where we implicitly identify $\mathbb{R}^2$ with $\mathbb{C}$ on both sides of the equality (see Figure 5.3 bottom). These observations can be therefore summarized in the following lemma:

Lemma 5.3.1. *Given a smooth map $\varphi : M \to N$ between surfaces, if each tangent plane T_p and $T_{\varphi(p)}$ is identified with the complex plane, then the map φ is conformal if and only if for every point $p \in M$ there exists a complex number q_p such that the pushforward $\mathrm{d}\varphi_p(v) = v * q_p$.*

Stated differently, a map is conformal if and only if its pushforward, which can be thought of as a mapping between (two copies of) the complex plane, can be represented, at every point, as multiplication by some fixed complex number.

In general the pushfoward is a real-linear map between tangent spaces identified with $\mathbb{R}^2$. However, in the special case of *conformal* mappings, the pushfoward is a $\mathbb{C}$-linear map. Crucially, while any $\mathbb{C}$-linear map is also real-linear, the converse, of course, does not hold. Remark also that up to technical conditions of differentiability, Lemma 5.3.1 is equivalent to the well-known *Cauchy-Riemann equations*, which can be stated compactly as saying that a mapping is conformal if and only if the associated map differential is $\mathbb{C}$-linear (see [123], p. 51 for a discussion).

5.3.4 Complex functional maps

Following a similar reasoning as in the functional map framework [106], in which a functional map is *any* linear transformation between functions spaces, we now consider the space of *all* complex linear maps between *tangent bundles*. As mentioned above, we implicitly assume that the tangent bundles are endowed with the complex structure given by some reference direction at every tangent space, and using the outward normal to define counter-clockwise rotation.

Thus we call *a complex functional map* any $\mathbb{C}$-linear operator Q that maps a complex field $X \in TM \simeq \cup_{p \in M} \mathbb{C}$ to a complex field $Q(X) \in TN \simeq \cup_{p \in N} \mathbb{C}$ on N, where we use $TM \simeq \cup_{p \in M} \mathbb{C}$ to denote the identification between tangent spaces and complex planes.

Remark that the complex structure allows us to represent any tangent vector field as a complex function, and, as a result, the operator Q enables the transfer of tangent vector fields defined on M to those defined on N. Crucially, by Lemma 5.3.1 we have that the pushforward of a conformal map must be a $\mathbb{C}$-linear operator and thus a complex functional map. The converse, however, does not hold, as not all $\mathbb{C}$-linear operators on the tangent bundle come from conformal pushforwards, in a similar way that not all linear functional mappings arise from pullbacks of pointwise correspondences.

In the remainder of the section we list the key properties of complex functional maps. In Section 5.3.5 we exhibit a necessary and sufficient condition for a complex functional map to represent the pushforward of a conformal mapping. Section 5.3.6 studies additional useful properties, notably a weaker necessary condition for Q to uniformly scale tangent planes and a characterization of isometric pushforwards as a subset of conformal maps.

5.3.5 Pushforwards vs. Complex-Linear Maps

As introduced above, a complex functional map $Q : TM \to TN$ is *any* $\mathbb{C}$-linear operator between tangent bundles. Our goal is to obtain a necessary and sufficient condition for such an operator to represent the pushforward (differential) of a conformal mapping. For this, we will use the following two key properties.

Connection to pullbacks First, suppose that $\varphi : M \to N$ is any diffeomorphism between two smooth surfaces. It is well-known that the pushforward $\mathrm{d}\varphi : T_pM \to T_{\varphi(p)}N$ is the unique linear map between tangent spaces that satisfies:

$$\langle X, \nabla(f \circ \varphi)\rangle_{T_pM} = \langle \mathrm{d}\varphi(X), \nabla f\rangle_{T_{\varphi(p)}N}, \tag{5.1}$$

for all $X \in TM, f \in L^2(N)$ (see [80], Chapter 3).

Orientability Second, consider any $\mathbb{C}$-linear mapping Q between tangent bundles. By linearity, for any complex number $q \in \mathbb{C}$ and any two complex fields $X, Y \in TM \simeq \cup_{p\in M}\mathbb{C}$, we have:

$$X = qY \quad \Longrightarrow \quad Q(X) = qQ(Y).$$

For instance, if $q = \imath$, the map Q preserves $90°$ rotation. As rotations are defined relative to the local basis of the tangent plane, they carry the information of the *orientation* of the manifold. Therefore **the operator Q cannot change the orientation of the tangent bundle**. This property of the complex functional maps is key in our shape matching applications: it forces the maps to remain orientation-preserving. Combining these observations, we conclude that complex functional maps can only represent the differential of an *orientation preserving* conformal map. Put differently, we have:

Theorem 5.3.1. *The complex-linear map Q is a pushforward if and only if there exists an* orientation-preserving *and conformal diffeomorphism $\varphi : M \to N$ satisfying:*

$$\langle X, \nabla(f \circ \varphi)\rangle_{T_pM} = \langle Q(X), \nabla f\rangle_{T_{\varphi(p)}N},$$

for all $X \in TM, f \in L^2(N)$.

Thm. 5.3.1 is a direct consequence of Lemma 5.3.1 and the uniqueness property of the map differential in Eq. (5.1). For completeness we provide the full argument in Appendix B.1.

This theorem highlights the importance of Eq. (5.1) to link the complex-linear map Q with the underlying pointwise map φ. Note that importantly, Eq. (5.1) only depends on the knowledge of the *pullback* associated with φ, and thus provides a natural link between Q and the standard functional map representation, which encodes pullbacks. In Section 5.4.2, we show how this property can be used to relate our complex and standard functional maps in practice.

5.3.6 Properties of complex functional maps

In this section we provide two additional structural properties of complex functional maps.

Orthogonality. First, we show a *necessary* condition for Q to represent the differential of a conformal map: Q must be an orthogonal operator. Interestingly, this is different from standard functional maps, which are orthonormal if and only if the underlying correspondence is locally area-preserving [106, 134].

Theorem 5.3.2. *If Q represents the pushforward of a conformal map φ between surfaces, then:*

$$Q^\star Q = I,$$

where $Q^\star$ is the adjoint operator uniquely defined by $\langle QX, Y\rangle_{TN} = \langle X, Q^\star Y\rangle_{TM}$.

Intuitively, this theorem comes from the fact the change of metric under the conformal map is given by some scaling factor at each tangent plane. When *integrating* the inner products on the surface, this scaling factor cancels out with the change of area measure. A similar result was shown in [134] for gradients of functions. We provide the complete proof in Appendix B.2.

Importantly, Theorem 5.3.2 only establishes a necessary condition: indeed, even if Q is an *orthonormal* $\mathbb{C}$-linear operator it is only guaranteed to represent the pushfoward of a conformal map, if it satisfies Eq. (5.1).

Isometries. Secondly, a pushforward is isometric if and only if it commutes with the Levi-Civita connection [25] (p.181). A similar statement can be made about the connection Laplacian recently used in geometry processing [143] to compute transport of vector fields along geodesic paths.

Theorem 5.3.3. *Let L be the connection Laplacian. If a complex functional map $Q : TM \to TN$ represents the pushforward $\mathrm{d}\varphi$ of a conformal map, then it satisfies:*

$$L_N \circ Q = Q \circ L_M$$

if and only if φ is an isometry.

Theorem 5.3.3, proved in detail in Appendix B.3, is very similar to that of functional maps [106].

5.3.7 Operators in a reduced basis

In order to improve the efficiency of our algorithms it is often desirable to consider operators acting on a subspace of smooth vector fields. Since tangent planes can be identified to complex planes, we will use the *complex inner product* defined as:

$$\langle X, Y\rangle_{\mathbb{C}_pM} := \langle X, Y\rangle_{T_pM} + \imath\langle \mathcal{J}X, Y\rangle_{T_pM} \in \mathbb{C},$$

where $\mathcal{J}$ is the $90°$ rotation around the normal. Note that for two vectors x, y in the same complex plane this inner product is equivalent to the standard $\overline{x}y$, where $\overline{x}$ is the complex conjugate of x.

Suppose that M is equipped with the family of vector fields $\{\Psi_i^M\}$ orthonormal with respect to the inner product $\langle ., . \rangle_{\mathbb{C}M}$. Then [122] (Thm II.6), any vector field X can be written as a linear combination:

$$X = \sum_i a_i \Psi_i^M,$$

where $a_i = \left\langle X, \Psi_i^M \right\rangle_{\mathbb{C}M} \in \mathbb{C}$. The transferred complex field $Q(X)$ can be decomposed in the basis of N:

$$\begin{aligned} Q(X) &= \sum_j \left\langle Q(X), \Psi_j^N \right\rangle_{\mathbb{C}N} \Psi_j^N \\ &= \sum_j \Psi_j^N \sum_i \left\langle Q(\Psi_i^M), \Psi_j^N \right\rangle_{\mathbb{C}N} a_i, \end{aligned}$$

where the second equality follows from the $\mathbb{C}$-linearity of Q. Thus the complex functional map can be understood as a matrix with coefficients $Q_{ji} = \left\langle \Psi_j^N, Q(\Psi_i^M) \right\rangle_{\mathbb{C}N}$, matching coefficients in basis on M to coefficients of the basis on N. Figure 5.2 provides a visual comparison between a classical functional map transferring functions *(left)* and a complex functional map transporting tangent vector fields *(right)*, both using a reduced basis of size 30.

5.4 Discrete Setting

In this section, we introduce complex functional maps in the discrete setting. Throughout, we consider oriented manifold triangle meshes (V, E, F). Our overall strategy is based on representing tangent vector fields as complex-valued pointwise functions, so that $X_i \in \mathbb{C}$ per vertex $i \in V$. This choice of a point-based representation for tangent vector fields (and not face-based, as in e.g. [8]) is motivated by our main application: disambiguating symmetries in non-rigid shape correspondence problems. Matching vertices between surface meshes is convenient as it directly allows to transfer texture coordinates or deformations.

We thus represent discrete complex functional maps Q as complex-valued matrices mapping complex-valued pointwise functions on M to complex-valued pointwise functions on N. In a similar spirit to functional maps, we improve computational efficiency by representing the operator Q as a small matrix in a reduced basis of vector fields. As a basis, we use the first eigenvectors of the *connection* Laplacian, discretized as in [143].

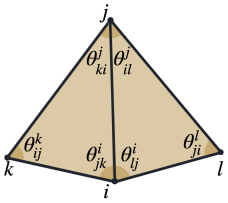
Below we introduce Laplacian operators and the necessary local complex structure, then we construct our complex functional map and translate each continuous property in the discrete setting.

5.4.1 Laplacian operators

Cotan-Laplacian The standard Laplacian operator $W \in \mathbb{R}^{|V|\times|V|}$ for a piecewise linear function $f \in \mathbb{R}^{|V|}$ is obtained by the well-known cotan-weight formula [22]:

$$(Wf)_i := \frac{1}{2} \sum_{(ij)\in E} \left(\cot \theta_{ij}^k + \cot \theta_{ji}^l\right) (f_i - f_j),$$

where the index notation is defined in the inset figure. We also define the diagonal $|V| \times |V|$ lumped mass matrix A:



$$A_{ii} := \frac{1}{3} \sum_{(ijk)\in T} a_{ijk},$$

where a_{ijk} are the areas of triangles adjacent to i. This matrix defines the scalar product in the space of piecewise linear functions. Namely if $f, g \in \mathbb{R}^{|V|}$ are two piecewise linear real-valued functions then:

$$\langle f, g\rangle_{L^2} := f^T A g.$$

Connection Laplacian Our discretization of tangent vector fields using complex numbers follows [76, 143]. Namely, we assign to each vertex an arbitrary reference unit vector orthogonal to the vertex normal. This reference direction represents the tangent vector, associated with the complex number $1 + \imath 0$. The outward normal dictates the orientation of the tangent planes, providing the additional axis $\imath$. Given this reference frame, any tangent vector can be represented as a complex number, and a tangent vector field as a complex-valued function.

In this context, the mass matrix A defines a complex scalar product in the space of tangent fields, represented as complex functions:

$$\langle X, Y\rangle_{\mathbb{C}M} := X^\star A Y,$$

where * represents the conjugate-transpose operation.

As mentioned above, in our applications we use a family of smooth vector fields given by the first k first eigenfunctions of the discrete *connection Laplacian*, as defined in [143]. The connection Laplacian is the matrix $L \in \mathbb{C}^{|V|\times|V|}$ uniquely defined by the Dirichlet energy:

$$X^\star L X = \frac{1}{2} \sum_{(ij)\in E} \left(\cot \theta_{ij}^k + \cot \theta_{ji}^l\right) |X_i - r_{ij} X_j|^2,$$

where the unit complex number r_{ij} are the rotations necessary to compare vectors expressed in two different bases.

By definition, L is a complex Hermitian matrix, and similarly to the cotangent Laplacian, on Delaunay meshes has only real non-negative eigenvalues. Therefore, L admits the generalized eigendecomposition:

$$L\Psi = A\Psi\Lambda,$$

where $\Psi^\star A\Psi = I$, Λ the diagonal matrix of eigenvalues, and Ψ is a set of complex-valued eigenfunctions. In particular, any tangent vector field U can be expressed as a linear combination *with complex coefficients* of the family $\{\Psi_i, i = 1, \ldots, |V|\}$ [66] (Thm. 4.1.5), *i.e.*: $U = \sum_i c_i\Psi_i$, where $c_i \in \mathbb{C}$. In practice we truncate this sum and typically use a fixed number k of the complex eigenfunctions, associated with the eigenvalues of smallest modulus.

5.4.2 Discrete complex functional map

In order to define a discrete equivalent of the pushforward, we simply discretize the continuous definition in Eq. (5.1). For meshes of same connectivity we are able to derive a closed-form expression for Q and a consistent notion of discrete conformality. For meshes with different connectivity or when the deformation is not exactly conformal we enforce this equation in the least-squares sense.

To discretize Eq. (5.1), we will need the pullback operator C, represented by a functional map [106], and the operator D_X often encountered in differential geometry to define tangent vectors [98] and introduced in geometry processing for vector field design in [8].

To define the discrete pullback, recall that given a map $\varphi : M \to N$, the associated functional map $C : L^2(N) \to L^2(M)$ can be discretized in the full, "hat", basis as a binary matrix $C_{NM} = \Pi_{MN}$, where $\Pi_{MN}(i, j) = 1$ if and only if $\varphi(i) = j$, while in the reduced basis we have $C_{NM} = (\Phi^M)^\top A^M \Pi_{MN} \Phi^N$ where Φ^M, Φ^N are matrices that store, as columns, the basis functions on the two shapes.

Vector field operator In addition, we will use the linear functional operator $D_X : L^2(M) \to L^2(M)$ describing the action of a vector field $X \in TM$ on a function $f \in L^2(M)$:

$$D_X(f)_p := \langle X, \nabla f\rangle_{T_pM}. \tag{5.2}$$

This operator uniquely characterizes a tangent vector field X on a manifold [98], and will allow us to write Eq. (5.1) as an equality between matrices. Azencot *et al.* [8] proposed a discretization for face-based vector fields, however, as previously stated, we use a different vertex-based discretization, that we describe in detail below.

We discretize the operator $D_X \in \mathbb{R}^{|V|\times|V|}$ as a vertex-wise scalar product at tangent planes:

$$(D_X f)_i := \langle X_i, \nabla f_i\rangle_{T_iM}, \tag{5.3}$$

where ∇f_i evaluates the gradient of a piece-wise linear function *at vertex* i (rather than at a face). In practice, we store the operator D_X into a complex sparse matrix.

Locally, the gradient of a function represents the best $\mathbb{R}^2$-linear approximation of this function in the tangent plane. The directional derivative of f along an edge vector e_{ij} is simply $(f_i - f_j)/|e_{ij}|$. We therefore ask ∇f to be the best approximation of all directional derivatives, namely:

$$\nabla f_i := \arg\min_{X \in \mathbb{R}^2} \sum_{(ij) \in E} \|e_{ij}^\top X - (f_i - f_j)\|^2. \tag{5.4}$$

This least squares optimization problem amounts to pseudo-inverting a $d_i \times 2$ matrix per vertex, where d_i is the degree of vertex i. This can be done exactly in pre-processing. As the gradient at vertex i only depends on its neighbors, it can be encoded as a complex *sparse* matrix whose non-zero coefficients per-line are equal to the degree of the vertex plus one.

Discrete definition of a pushforward By combining these tools, we obtain a simple discretization of the definition of the pushforward in Eq. (5.1), as a composition of functional operators, represented in the discrete setting as matrix multiplication:

$$D_X^M C_{NM} = C_{NM} D_{QX}^N, \quad \forall\, X \in \mathbb{C}^{|V_M|}. \tag{5.5}$$

Note that in this expression, consistently with Eq. (5.1), the pushforward Q maps vector fields in the opposite direction from the pullback C. In practice, we have found it simpler to work with an expression that assumes that the two operators map in the same direction, which leads to:

$$C_{MN} D_X^M = D_{QX}^N C_{MN}, \quad \forall\, X \in \mathbb{C}^{|V_M|}. \tag{5.6}$$

This expression can be obtained simply by pre- and post-multiplying Eq. (5.5) by C_{MN}, and assuming an invertible mapping.

In practice, mappings are usually not conformal, so we cannot hope to satisfy Eq. (5.6) exactly. However, we can define the energy E_{cq} measuring how close $Q : TM \to TN$ is to be the differential of the operator $C_{MN} : L^2(M) \to L^2(N)$ and evaluating the constraint in the least squares sense:

$$E_{cq}(C, Q) = \sum_i \|C D_{X_i}^M - D_{QX_i}^N C\|_F^2, \tag{5.7}$$

where $\{X_i\}$ is a family of vector fields in TM. Remark that Eq. (5.6) is linear with respect to X. Consequently, minimizing the energy of Eq. (5.7) will ensure that Eq. (5.6) is satisfied as well as possible on the subspace of TM generated by $\{X_i\}$. As above, we use the first eigenfunctions of the connection Laplacian operator for this family, and

thus the energy in Eq. (5.7) ensures that Eq. (5.6) is satisfied as well as possible for *smooth tangent vector fields*.

Our overall strategy thus consists in recovering Q given an arbitrary functional map C by solving the problem:

$$\min_{Q} E_{cq}(C, Q). \tag{5.8}$$

Eq. (5.8) defines a simple least squares system which can be efficiently optimized by solving a linear system of equations. We describe how to do in it more details in Appendix B.6. The solution of this problem is the best approximation of the map differential by an *orientation-preserving* conformal pushforward. The energy E_{cq} is fundamental in our experiments as it allows us to extract orientation information from any given functional map.

5.4.3 A closed-form expression for Q

While Eq. (5.7) plays a fundamental role in our experiments, we remark that in the case of meshes with the same connectivity, it is possible to obtain an intuitive closed-form expression for Q. Recall that a conformal pushforward is given by 1) an assignment between points and 2) a similarity transformation between matching tangent planes. Therefore, it is expected that the discrete $\mathrm{d}\varphi$, the pushfoward between piecewise linear complex fields, is simply the composition of a matrix Π_{MN}, assigning vertices of M to those on N, and a multiplication by a complex field $q : N \to \mathbb{C}$ performing the tangent plane deformation at each vertex. Using our discrete definition of the pushforward (Eq. (5.6)), we can recover this property when the two meshes have same the connectivity.

Theorem 5.4.1. *Given two meshes with same connectivity and given the permutation matrix Π describing the vertex-to-vertex correspondence, the solution of Eq. (5.8) has the form:*

$$Q = D(q)\Pi,$$

where $D(q)$ is a diagonal matrix with complex coefficients q_i. The complex numbers q_i are the best conformal alignement of the tangent planes.

The proof of Thm. 5.4.1 can be found in Appendix B.4.

5.4.4 Constraints on complex functional maps

Operator orthogonality While the discrete version of Thm. 5.3.2 holds exactly in our discrete setting, it is not very informative, since the discrete notion of conformality induced by Thm. 5.4.1 is too rigid, since the only deformations of a mesh that preserve angles at every triangle exactly are isometries with possible global scaling.

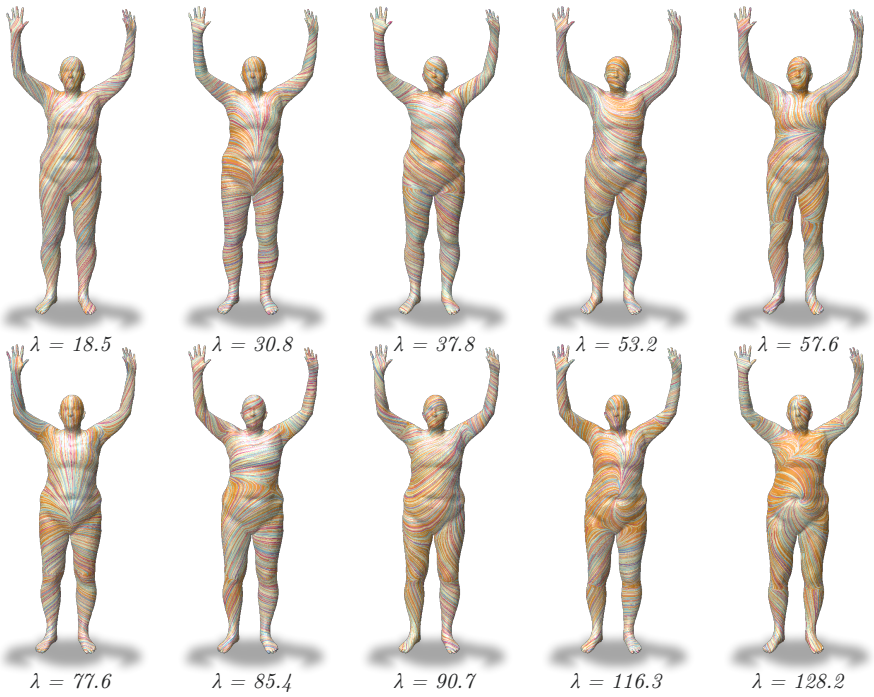


Figure 5.4: LIC visualization of the first 10 eigenvectors of the connection Laplacian (from left to right, top to bottom). We see that a higher eigenvalue λ gives an eigen tangent vector field whose frequency is higher, resulting in a less smooth flow. Nevertheless, all these tangent vector fields represent the generators of *the smoothest vector fields over the shape*. It is in this basis that we will decompose tangent vector fields, as described in Section 5.4.5.

However, orthonormality of Q is still a valuable constraint *in the reduced basis* and forces Q to be the approximation of a pushforward. Furthermore, the use of a reduced basis helps to avoid both the reliance on an exact mesh structure and the rigidity of exactly conformal maps. In our experiments this constraint proved to be very effective.

In practice, we therefore solve the following Procrustes problem:

$$\min_{Q^*Q=I} \quad E_{cq}(C, Q). \tag{5.9}$$

This is a standard problem that can be solved exactly using a Singular Value Decomposition. For completeness we provide the details in Appendix B.6.

Commutativity with the connection Laplacian The isometric constraint of Thm. 5.3.3 is still valid in the discrete setting. Interestingly, the isometry condition by enforcing the commutativity with the Laplacian is identical to the standard functional map framework. In particular, as remarked in [106], it implies the more isometric a mapping is, the more diagonal our matrix Q will be, when expressed in the reduced basis.

Theorem 5.4.2. *The conformal pushforward Q commutes with the connection Laplacian:*

$$L_N Q = Q L_M,$$

if and only if it represents an isometric map.

The proof is deferred to Appendix B.5.

5.4.5 Discrete operators in a reduced basis

In order to improve the efficiency of our algorithms we will not consider all piecewise linear tangent vector fields but only those spanned by a small number k of smooth vector fields stored in a complex $|V| \times k$-matrix Ψ.

For a known deformation across compatible meshes, the closed-form expression of Q is exhibited in Thm. 5.4.1 and can easily be rewritten in a reduced basis:

$$Q_{NM} = (\Psi^M)^\star A^M D(q) \Pi_{MN} \Psi^N. \tag{5.10}$$

Note that Eq. (5.10) is very similar to the expression of discrete *functional map* C introduced in [106]. For the function bases Φ^M and Φ^N a functional map reads:

$$C_{NM} = (\Phi^M)^\top A^M \Pi_{MN} \Phi^N. \tag{5.11}$$

All the constraints on Q in the "hat" basis can be simply rewritten by replacing each term by an operator projected in the reduced basis.

In theory any orthonormal basis could be considered. For the purpose of non-rigid 3D shape matching a basis smooth and stable under nearly-isometric deformations leads to better results. In our experiments, we use the k first eigenvectors of the connection Laplacian for complex functional maps, that we visualize via Line Integral Convolution in Figure 5.4. Indeed, as proved in Thm. 5.3.3 this operator is invariant under isometric deformations and moreover its discretization is easily implemented.

5.4.6 Point-to-point map conversion

In the functional map pipeline a key step is the conversion from a functional map to a vertex-to-vertex map. As originally described in [106] and extended in [109], one just needs to transfer Dirac functions on M using the adjoint of the functional map, and compute the closest Dirac function on N using a nearest neighbor search algorithm. Namely, the operation performed is $\Pi_{MN} = \mathrm{NNsearch}(\Phi^N, \Phi^M C_{NM})$, where Φ^M and Φ^N are the eigenfunctions from the standard Laplace-Beltrami operator on M and N.

In our case, extending this algorithm to "Dirac vector fields" is not straightforward as a single vector is not isotropic. Instead, we propose to use the divergence operator to

Random noise			
Method / level of noise	$s = 0$	$s = 0.2$	$s = 0.5$
Wang *et al.* [167]	**0.40**	2.2	5.5
Azencot *et al.* [8]	6.2	4.6	3.2
Ours	**0.40**	**0.43**	**0.74**
Symmetric noise			
Method / level of noise	$a = 0.3$	$a = 0.5$	$a = 0.6$
Wang *et al.* [167]	0.75	1.1	1.3
Azencot *et al.* [8]	4.6	1.8	4.2
Ours	**0.40**	**0.51**	**0.81**

Table 5.1: Average accuracy of the three vector field transfer algorithms on 20 random pairs of FAUST [17] for two types of noise, three noise levels. We use $k = 50$ eigenvectors for both real and complex Laplacian operators. For noisy input functional maps, our method is always the most accurate. For completeness, we report more results for $k = 30, 70, 150$ in Appendix B.7.

convert the vector field basis to functions and then use the standard functional point-to-point conversion scheme via nearest neighbor search. When Q is expressed in a reduced basis, this simply amounts to computing:

$$\Pi_{MN} = \text{NNsearch}(\text{div}_N \Psi_N, \text{div}_M \Psi_M Q_{NM}). \tag{5.12}$$

Where we define the discrete divergence to be the adjoint of the gradient operator matrix defined in Eq. (5.4).

This solution is not fully satisfying as it relies on the commutativity of the push-forward with the divergence operator, and thus is geared towards near isometries. This approach, however, proves to be sufficient for our shape matching applications in Section 5.5. Furthermore, in our current pipeline, Q is evaluated alongside C and thus, if needed, the conversion can be done using the standard functional map. We leave finding a robust and general conversion scheme for complex functional maps as interesting future work.

5.5 Results

In this section, we present several applications of our complex functional maps. We start by demonstrating that robust vector field transfer can be achieved with complex functional maps without requiring additional information or computation. Then, we demonstrate that the orientation-aware nature of complex functional maps can be used to eliminate the symmetry ambiguity in non-rigid near-isometric shape matching problems.

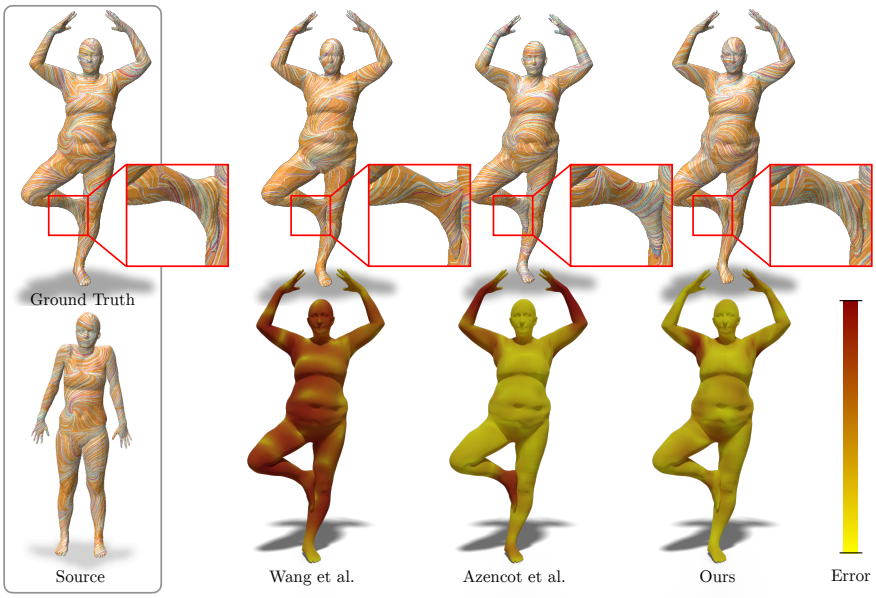


Figure 5.5: Comparison of three tangent vector field transfer methods. Top row: we display a LIC visualisation of the transferred vector field on the target shape. Bottom row: we show the transfer error, computed as the difference with the ground-truth transfer. For this transfer, we used a 50×50 ground-truth functional map blurred with a random noise of magnitude 0.1. Quantitative results can be found in Table 5.1.

Complex functional maps are easily added to standard functional map pipelines like map estimation from descriptors [106, 103, 124] or refinement methods like ZoomOut[94] and its follow-ups [126, 127].
An implementation of our method can be found at: github.com/nicolasdonati/QMaps.

5.5.1 Vector field transfer

The first direct application of complex functional maps is tangent vector field transfer. We compare three methods for vector field transfer using as only input an approximate functional map C. We demonstrate below that using our approach is more accurate, compared to existing methods [167, 8] especially in the presence of noise.

Vector field transfer with complex functional maps

Our representation allows for efficient and easy-to-use tangent vector field transfer. Indeed, given a functional map C_{MN}, we can easily recover a complex functional map Q by solving the Procrustes problem in Eq. (5.9). Then, transferring a vector field X defined on shape M to shape N, boils down to : 1) projecting X in the complex spectral basis $X \approx \Psi^M x$ (see Section 5.4.1), 2) transferring the spectral coefficients

Method (-/+ICP)	[103]	[124]
-	0.24 / 0.21	0.21 / 0.17
+ Ours	0.15 / 0.13	0.12 / 0.10
Method (-/+ICP)	[167]	[167] + [124]
-	0.31 / 0.23	0.23 / 0.13
+ Ours	0.16 / 0.13	**0.11 / 0.083**

Table 5.2: Average geodesic error on 190 FAUST remeshed shape pairs. Our complex functional map step always improves the correspondence quality for all four algorithms, even those already incorporating the orientation-aware operators from [124]. Detailed graph geodesic error vs. percentage of correspondences can be found in Fig. B.1 of the Appendix.

using: $y = Qx$ and 3) recovering the output as linear combination of the target basis: $Y = \Psi^M y$.

To assess the accuracy of our method, we compare it with two standard baselines: the transfer using Hodge decomposition from Wang *et al.* [167] and the transfer using the vector field operator proposed by Azencot *et al.* [8].

Hodge decomposition transfer On a shape M with a sphere-topology, any tangent vector field $X \in TM$ can be decomposed as the sum of a gradient and a rotated gradient. As remarked in [167], transferring vector fields with a conformal map can be done using only the functional map C by: 1) computing the Hodge decomposition, *i.e.* finding the functions $f, g \in \mathbb{R}^{|V|}$ such that $X = \nabla f + \imath \nabla g$, 2) transferring the functions f, g using C and 3) computing the gradient and rotated gradient on N.

Vector field operator Azencot *et al.* [8] use the representation of a tangent vector field $X \in TM$ via the associated functional operator $f \mapsto D_X(f)$, defined in Eq. (5.2) above. The method boils down to transferring X by solving :

$$\min_Y \|D_Y C - C D_X\|_F^2.$$

This method is similar to ours as their energy is also inspired by Equation (5.1). However, there are two key differences. Firstly, our method estimates the transfer for all low-frequency tangent vector fields of the source eigenbasis simultaneously by estimating Q, whereas the approach of Azencot *et al.* is limited to one vector field transfer at a time. Secondly, the approach in [8] is not limited to conformal deformations, making it more flexible but also more sensitive to noisy input functional maps.

Results Table 5.1 reports a quantitative comparison between the three methods described above on 20 random pairs taken from the original FAUST dataset [17]. These

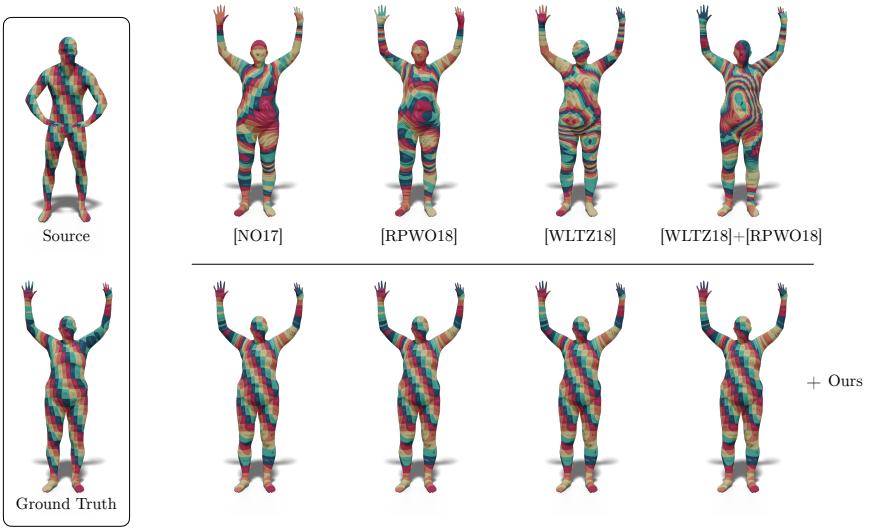


Figure 5.6: We visualize the influence of complex functional map step on some descriptor-based functional map pipelines. The quality of the computed maps is displayed with texture transfer on a pair of FAUST re-meshed shapes (91 as source, 89 as target). We see that for every of the considered pipeline, our method helped with continuity and left-right symmetry errors. Quantitative results can be found in Table 5.2.

meshes are in one-to-one vertex correspondence, allowing us to compute the ground-truth pushforward. To perform the comparison, we generate a smooth vector fields on the source, transfer it and then compute the L^2-distance with the exact transfer normalized by the norm of the input vector field. We use $k = 50$ eigenvectors for both the functional space and the vector field space. We assess the robustness of the three functional-based vector field transfer methods by adding two types of noise to the input ground-truth C_{gt} functional map:

- **Random noise:** the input ground-truth C_{gt} functional map is corrupted by adding a random matrix N whose entries are taken uniformly at random between $-s$ and s, s being a given threshold. We transfer tangent vector fields of the form $X = \nabla f + \imath \nabla g$ by randomizing the spectral coordinates of f, g such that they decrease in intensity as frequency goes up. The results are reported in the first half of Table 5.1.

- **Symmetric noise:** The input ground-truth map C_{gt} is mixed with an orientation-reversing functional map C_{sym} by linearly interpolating between the two maps $C = aC_{gt} + (1 - a)C_{sym}$. This kind of noise often arises when estimating maps from descriptors with the original pipeline introduced in [106]. We transfer a

tangent vector field of the form $X = \nabla f$, with f the extrinsic coordinate of the left-right axis. This results in an antisymmetric vector field that will only be transferred correctly if the method is robust to noise. The results are reported in the second half of Table 5.1.

In presence of noise, our method always outperforms the baselines. This is due to the fact that we first compute the pushforward closest to C in the least square sense, making it robust to random noise. By construction, our pushforward is *orientation-preserving*, so it is resistant to symmetric noise and is able to recover a well-oriented transfer as shown in Fig. 5.1. In comparison, the other two methods directly rely on the functional map and fail if C does not exactly represent a pointwise map. Moreover, if C is a mix of a direct and orientation-reversing map, it is not easy to recover the underlying orientation-preserving map and thus to be robust to such noise.

In Figure 5.5 we provide qualitative illustrations of vector field transfer obtained using our method compared to baselines. Note that even a small amount of noise in the input functional map can compromise the quality of the transfer performed by [167]. Our method only exhibits minor errors, even though it is only designed to handle conformal deformation. In fact, the only visible mis-transfer happens at the shoulder joint where the deformation is far from conformal. The approach of Azencot *et al.* [8] is clearly under-performing and always achieves the worst accuracy of our comparisons. However this method is more general and would give the best results in presence of perfect information *and in the full basis*, even with strongly non-isometric deformations.

In conclusion, complex functional maps can help to alleviate the errors in map orientation and allow to accurately transfer vector fields between near isometric shapes even if the deformation is not exactly conformal.

5.5.2 Disambiguating symmetry in functional maps computation

In our next application, we show that complex functional maps can be used within the standard descriptor-based functional map pipeline to significantly improve robustness and accuracy. The key issue that we consider is that, as remarked in prior works [107, 32] intrinsic descriptors [151, 6] are often *symmetric*, which can lead to poor correspondences, where a point is arbitrarily matched to the correct target point or its symmetric counterpart [126]. As we demonstrate below, injecting our orientation-preserving complex maps into the pipeline can help to resolve this issue efficiently.

To achieve this, we propose to *project* a given functional map into the space of orientation-preserving maps by using Q as an intermediary. This projection is done in two steps. First, we approximate the associated map differential Q by solving the Procrustes problem in Eq. (5.9). Secondly, we extract from Q the underlying point-to-point mapping using the algorithm described in Section 5.4.6. Since, by construction, Q is orientation preserving, the projection removes the orientation reversing component

Method / stats	Avg.	Med.	Min.
ZO	0.520	0.523	0.226
ZO + Ours	**0.320**	**0.328**	**0.037**
ZO + bij	0.508	0.447	0.106
ZO + bij + Ours	**0.367**	**0.382**	**0.040**
ZO + bij + conf	0.47	0.45	**0.025**
ZO + bij + conf + Ours	**0.225**	**0.078**	0.029
ZO + bij + iso	0.450	0.433	**0.025**
ZO + bij + iso + Ours	**0.198**	**0.081**	0.029

Table 5.3: Adding our complex functional map step in the pipeline of Ren *et al.* [127] always improves map accuracy. We report the average, median and minimal geodesic distance error on 50 shape pairs of the SMAL dataset. Detailed graph geodesic error vs. percentage of correspondences can be found in Fig. B.2 of the Appendix.

of the input map, and thus the resulting point-to-point mapping should be orientation preserving.

In a third optional step, one can reconstruct a new functional map from the point-to-point map using Eq. (5.11). This allows us to improve the mapping using post-processing technique like the spectral ICP refinement introduced in [106].

We compare this approach to four alternative algorithms for computing functional maps solely from 50 WKS [6] descriptors:

- The algorithm introduced by Nogneng *et al.* [103] where descriptors are converted into operators that must commute with the functional map.
- We add the orientation-promoting operators of [124] to the pipeline of Nogneng *et al.* [103].
- The method proposed in [167], in theory closest to ours, which proposes to also transfer differential 1-forms with functional maps. It differs from our method mainly because they rely on $\mathbb{R}^2 - linearity$ whereas our maps are $\mathbb{C}$-linear, and thus orientation-aware. In [167] the authors transform descriptor functions into tangent vector fields by taking the gradient.
- A combination of [167] with the orientation-aware operators of [124].

To assess the ability of Q to recover orientation information, we compare maps obtained by direct conversion of the functional map computed using each of approaches above, with maps obtained by conversion from Q, that has first been estimated from C.

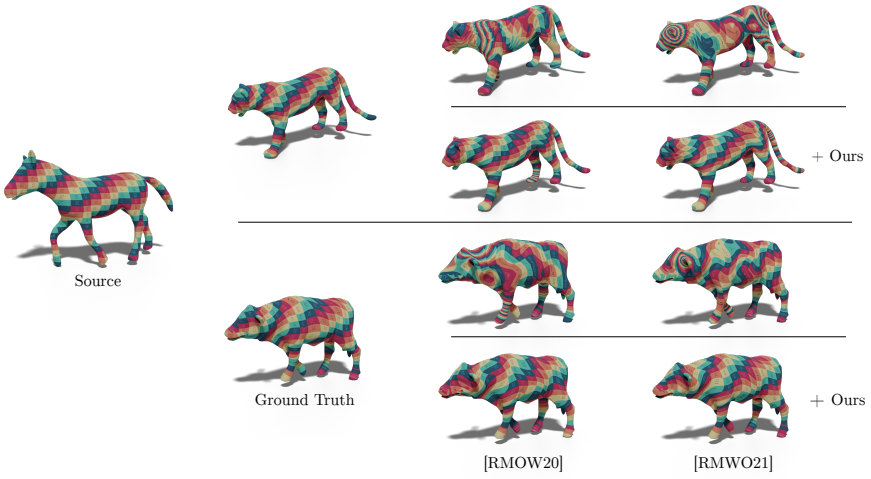


Figure 5.7: Texture transfer on SMAL re-meshed dataset [174] illustrating maps obtained with two methods based on ZOOMOUT: bijective ZOOMOUT[126] (third column) and bijective ZOOMOUT with isometric energy [127] (last column). Adding our Q-step (bottom sub-rows) considerably improves the accuracy of the map. For quantitative results see Table 5.3.

The results are computed on 20 shapes (which corresponds to 190 pairs) from the FAUST dataset [17], which were re-meshed in [124] in order to remove the bias of identical triangulations. We then compare the mean geodesic distance error obtained by these different methods – for each source point, we compute the geodesic distance on the target between the point mapped by the obtained maps and the point mapped by ground-truth. The results are reported in Table 5.2. Additionally, we display in Figure 5.6 a qualitative result where our additional step was able to disambiguate symmetry whereas the standard pipelines could not.

This experiment shows that our Q-map projection step significantly improves the accuracy of the correspondence *with all algorithms*, even in the presence of orientation-aware operators.

5.5.3 Orientation-preserving ZOOMOUT

Finally, our complex functional maps can also be used to improve more recent functional map algorithms based on spectral upsampling, inspired by ZOOMOUT[94]. The ZOOMOUT algorithm starts with an initial pointwise map Π_0 and alternates between two steps: estimating a new functional map C_n from Π_n and recovering the new pointwise map Π_{n+1} from C_n. In order to increase the map precision, the size of the spectral basis increases at each iteration.

Algorithm 1 Complex ZOOMOUT

Intput: Manifold meshes M and N
Initial pointwise maps Π_{MN}
Output: Refined maps Π_{MN}^{ref}
Parameters: The number of refinement steps J
An array $[k_j], j \in [1, J]$ with the (increasing) number of spectral coordinates to use at each refinement step
Preprocessing: Compute the Laplace Beltrami eigenbases Φ^M and Φ^N (used for function in spectral basis)
Compute the connection Laplacian complex eigenbases Ψ^M, Ψ^N (used for vector field spectral bases, see Section 5.4.5)
Compute the differential operators $D_{\Phi_i^M}$, and $D_{\Phi_i^N}$ for $i \in [1, k_J]$ (used for estimating Q from C, see Section 5.4.2)
Compute the reduced divergence operators div_M and div_N (used for conversion from Q to pointwise map, see Section 5.4.6)
for $k \in [k_1, ..., k_J]$ **do**
    $\Phi_M = \Phi^M_{[1,k]}, \Phi_N = \Phi^N_{[1,k]}$
    $C_{NM} = \Phi_M^\dagger \Pi_{MN} \Phi_N$
    $Q_{NM} = \arg\min_{Q \in \mathcal{O}(k)} \sum_{i=1}^{k} \|C_{NM} D_{\Psi_i^N} - D_{Q\Psi_i^N} C_{NM}\|_F^2$
    $\Pi_{MN} = \mathrm{NNsearch}(\mathrm{div}_N \Psi_N, \mathrm{div}_M \Psi_M Q_{NM})$
end for

Interestingly, we can modify the basic ZOOMOUT approach to incorporate complex functional maps and thus promote discovery of orientation preserving maps. Our new algorithm basically boils down to 3 steps instead of 2: 1) estimate the functional map C_n from Π_n. *2) **[Q-step]** Estimate the complex functional map Q_n from C_n, using Eq.* (5.9). 3) Estimate the new pointwise map Π_{n+1} *from* Q_n (instead of C_n like in classic ZOOMOUT) using Eq. (5.12). The pseudo-code can be found in Algorithm 1 where the lines that describe our modification are highlighted in bold.

We remark that this "Q-step" can easily be added to other algorithms built on top of ZOOMOUT. MapTree [126], which uses a tree structure to explore the space of maps, modifies ZOOMOUT to promote bijectivity. This idea was later extended to other properties like conformality or isometry by Ren *et al.* [127]. Like ZOOMOUT, these algorithms are based on spectral upscaling and conversions between spectral mappings and vertex-to-vertex maps. We describe their modification in Appendix B.9.

We demonstrate the beneficial effect of this Q-step by refining *random* functional maps with four versions of ZOOMOUT with and without our modification. We upscale the maps from 4 eigenfunctions up to 50, with a step of 1, and 10 inner loops per step. We perform this experiment on 50 shape pairs of SMAL [174] re-meshed. We report

the obtained quantitative results in Table 5.3. For the baselines, we used the standard ZOOMOUT algorithm [94], bijective ZOOMOUT[126], and the discrete optimization with first conformal *(conf)* and isometric energy *(iso)* [127]. For all of these baselines, we report both their overall geodesic error (mean, median and minimum error) and that of their modification with our method. We observe that our modification is always relevant in this case, resulting in a significant boost in overall accuracy on this dataset. More specifically, the discrete optimization approach [127] with complex functional maps performs really well (median error below 0.09) despite the fact it is randomly initialized, and does not use *any* descriptors.

In Figure 5.7 we provide qualitative results with two shape pairs with the same source, and report five maps per pair: the ground-truth, bijective ZOOMOUT[126], and bijective isometric ZOOMOUT[127], as well as their version with our modification. For both shape pairs and both methods, the original algorithms are affected by the left-right symmetry and converge to discontinuous maps.

In summary, in state-of-the-art refinement pipelines involving ZOOMOUT, our Q-step appears to promote orientation preservation and continuity, resulting in better convergence and more accurate results overall. This confirms that using complex functional maps in functional map pipelines is beneficial to estimate high quality correspondences from very low-frequency or extremely unreliable initialization.

5.6 Conclusion, Limitations & Future Work

In this chapter, we have introduced a new functional operator resulting from a $\mathbb{C}$-linear relaxation of the space of pushforwards: the complex functional map. This operator allows to robustly transfer tangent vector fields between non-rigid surfaces in 3D. Furthermore, the most prominent property of this new tool is that it reflects the complex structure of the surfaces and is thus orientation-aware. In our experiments, we exploited the orientation-aware property of complex functional maps in several shape matching tasks. This contribution to the functional framework considerably increases the applicability of intrinsic shape matching methods, which are often hindered by the presence of orientation-reversing intrinsic symmetries. This is particularly relevant for functional map-based algorithms, since they rely on a linear relaxation of diffeomorphisms, which can linearly blend direct and orientation-reversing maps, potentially resulting in large discontinuities.

However our framework is naturally restricted to differentials of *conformal* orientation-preserving mappings. This constraint is only enforced in a least squares sense in our approach and thus helps to *promote* conformality which can be beneficial for shape pairs satisfying this assumption. Besides, like most algorithms derived from the functional framework, our construction is dependent on the choice of reduced basis. This can limit its applicability in more general non-isometric shape matching. As follow-up work, we

would like to study other representations for orientation-preserving pushforwards and alleviate the dependency on the reduced basis.

In the future it would also be interesting to investigate the utility of complex functional maps in other applications that involve vector field transfer, including deformation or pose transfer, or synchronized convolution in the context of geometric deep learning. Indeed, restricting the search to well-oriented maps without additional supervision could lead to new efficient methods in unsupervised 3D deep learning.

Acknowledgements The authors would like to thank the anonymous reviewers for their helpful feedback and suggestions. Parts of this work were supported by the ERC Starting Grants No. 758800 (EXPROTEA) and No. 802554 (SPECGEO), and the ANR AI Chair AIGRETTE.

CHAPTER 6

Deep Orientation-Aware Functional Maps: Tackling Symmetry Issues in Shape Matching

State-of-the-art fully intrinsic networks for non-rigid shape matching often struggle to disambiguate the symmetries of the shapes leading to unstable correspondence predictions. Meanwhile, recent advances in the functional map framework allow to enforce orientation preservation using a functional representation for tangent vector field transfer, through so-called *complex functional maps*. Using this representation, we propose a new deep learning approach to learn orientation-aware features in a *fully unsupervised* setting. Our architecture is built on top of DiffusionNet, making it robust to discretization changes. Additionally, we introduce a vector field-based loss, which promotes orientation preservation without using (often unstable) extrinsic descriptors. Our code is available at: https://github.com/nicolasdonati/DUO-FM.

6.1 Introduction

Learning for non-rigid shape correspondence is a key problem in 3D shape analysis with applications ranging from statistical shape analysis [19, 111] to deformation or texture transfer [13]. Early approaches have focused either on learning informative features so that corresponding points have similar feature descriptors, e.g., [87], or modeling shape correspondence as a semantic segmentation problem. Approaches in the latter category, e.g., [92, 97, 112, 169] aim to predict, for every point on the surface, the corresponding vertex id on some ground truth template shape. Unfortunately both approaches impose very little consistency between individual point correspondence predictions, and can be

sensitive to the underlying shape discretization [144].

More recently, techniques have focused on both predicting and imposing a training loss on the entire *map* between each pair of shapes. This has been greatly facilitated by spectral approaches and especially the functional map representation [106], which encodes a map as a small matrix using the spectral (Laplacian) eigen-basis. A wide range of approaches based on both supervised [31, 85, 39] and unsupervised losses [132, 62] have been proposed using the functional map representation. Key to all of these methods is learning feature functions that are then used to predict the functional map *as a whole*. As was shown across multiple recent works, this reduces the amount of necessary training data (see chapter 4), provides strong regularization promoting smooth maps, makes the learned features robust to changes in discretization [144], and alleviates the requirement of the existence of a fixed template shape.

While using the compact functional map representation introduces a strong bias towards smooth approximately isometric correspondences, it nevertheless leaves room for both orientation-preserving and orientation-reversing correspondences. This orientation-agnostic property of functional maps can be useful, e.g., in symmetry detection tasks. However, in most practical scenarios, the underlying sought correspondence is expected to preserve orientation. Unfortunately, restricting to only orientation-preserving maps is not straightforward while using the functional map representation, and the maps obtained using this framework can easily introduce local and global symmetry flipping (i.e., left/right ambiguity present in many organic shapes). As a result, existing state-of-the-art learning networks require either a supervised loss [85, 39, 144], rigid pre-alignment [140], or rely on hand-crafted extrinsic descriptors to disambiguate symmetries.

In this chapter, we demonstrate that these limitations can be overcome by using the *complex functional map* representation introduced in chapter 5, that is based on alignment of tangent vector fields (represented as complex functions) rather than real-valued functions.

To achieve this, we propose the first architecture that uses complex functional maps and learns specific features that align tangent bundles on surfaces. The use of the complex structure makes our approach fully orientation-aware, and helps to restrict the space of allowed correspondences to only globally orientation-preserving maps, while regularizing the learning process. We introduce losses adapted to complex functional maps and demonstrate that our network can be trained in a fully unsupervised manner without relying on rigid pre-alignments or ground truth correspondences. More broadly, the vector-valued features learned by our approach provide a novel and informative signal for non-rigid shape analysis tasks.

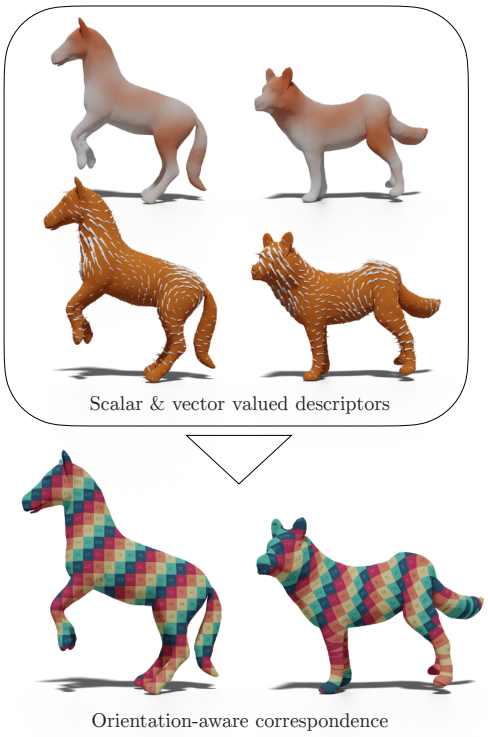


Figure 6.1: Our method aims at producing *orientation preserving* maps for non-rigid 3D shape matching in a fully unsupervised setting through the estimation of descriptors whose gradients also align on source and target shape.

6.2 Related Works

Non-rigid shape matching is a very rich and well-established research area. Below we review works that are most closely related to ours, focusing on learning-based, and especially unsupervised techniques. We also refer the interested readers to surveys including [16, 135] for a more in-depth overview.

Functional Maps Our method builds upon the functional map representation, which was originally introduced in [106] and then extended in a very wide range of follow-up works, e.g., [47, 103, 24, 167, 124, 166, 51, 146] among others. The key advantage of this framework is that it allows to represent and optimize for maps as small-sized matrices, enables strong linear-algebraic regularization, and can even be adapted to the partial setting [129, 86].

An essential step in works using this representation, are the "descriptor" (also known as "probe" [108]) functions that are used to estimate the underlying functional maps and that must be provided *a priori*. Early methods have exploited axiomatic descriptors such

as heat or wave kernel signatures [151, 6], with several attempts aiming to optimize the weights of such descriptors through optimization techniques [31].

Learning-based Methods Learning shape correspondence has also been done by treating it as a dense semantic segmentation problem, e.g., [92, 20, 97, 50, 112, 169], among many others, or *via* template alignment [55]. However, such works tend to require significant amount of training data, establish a map to a template, and can fail to generalize under connectivity changes [144].

More closely related to our approach are methods that use learning together with the functional map representation, thus evaluating the map as a whole and allowing to directly train and test on arbitrary shape *pairs*. This was first introduced in FMNet [85] , which proposed a method to *refine* given descriptor functions such as SHOT [158] with a deep neural network, by minimizing a supervised loss, given some ground truth correspondences. We extended this approach in chapter 4, where descriptor functions for functional map estimation are extracted directly from the shapes' geometry using point-based feature extractors, and a new regularized functional map estimation layer.

Unsupervised Spectral Learning Even closer to ours are spectral approaches that use *unsupervised learning* while exploiting the functional map representation. This was first done by replacing the supervised loss in FMNet with either geodesic distance preservation [62] or desirable structural properties in the spectral domain [132]. Other properties such as cycle consistency [52] or unsupervised alignment of heat kernels [7] have also been used to improve efficiency and accuracy.

These methods are attractive since they do not rely on manual supervision, are fully intrinsic and thus tend to generalize well across pose changes. At the same time, their fully intrinsic nature can cause ambiguities in the presence of intrinsic symmetries such as those present in human shapes. To alleviate this problem, previous functional maps methods, typically only *refine* given descriptors such as SHOT, which carry some extrinsic information [62, 132, 52, 7] but can unfortunately be highly unstable under connectivity changes. More recently, "weak supervision" was advocated in the form of rigid pre-alignment [140, 46] to resolve symmetry ambiguity. Finally, Deep Shells [45] performs SHOT feature refinement *jointly* with using the 3D embedding to guide unsupervised correspondence learning.

Unfortunately, despite significant effort, symmetry ambiguity remains a central problem in unsupervised learning for non-rigid shape matching. This is especially problematic since spectral methods tend to generalize much better to unseen poses compared to extrinsic methods such as [55]. As we argue in this chapter, the symmetry ambiguity problem is inherent to the functional maps approaches, as the losses used are fully intrinsic and thus cannot disambiguate orientation-preserving vs. orientation-reversing maps.

Complex Functional Maps This tool for geometry processing, introduced in chap-

ter 5 aims at aligning *tangent vector fields* rather than functions. Crucially, complex functional maps allow to remove orientation reversing maps from the space of allowed correspondences. As a result, as shown in 5.5.2, this can help to gain better control on both orientation, and ultimately symmetry in the computed maps. However, this approach still uses either axiomatic descriptors or an iterative procedure in its pipeline. Therefore, it is unclear how to incorporate this representation into a learning framework, while maintaining accuracy and efficiency.

Contributions Our main contributions are as follows:

1. We introduce a new *orientation-aware unsupervised loss*, using the complex functional maps representation of chapter 5, that exploits the properties of tangent vector fields.
2. We show that computing complex functional maps directly from gradients of learned features and then imposing an additional loss on these maps helps to regularize currently unstable pipelines with respect to symmetry aliasing.
3. By building upon a recent, robust feature extraction backbone [144], we introduce a fully unsupervised correspondence learning approach, without using extrinsic descriptors or coordinate information, while being robust to significant changes in triangulation.

6.3 Background and Motivation

6.3.1 Notation, Background & Motivation

Given a pair of non-rigid shapes, M, N, represented as triangle meshes, our main goal is to estimate a map $\varphi : M \to N$ in an unsupervised manner.

Functional Maps In this work, we use the functional map framework, which has recently given rise to state-of-the-art supervised [85, 39, 144] and unsupervised [132, 62, 45, 52] learning-based non-rigid shape correspondence methods.

The key idea behind the functional maps approaches is that any correspondence can be represented compactly as a small-sized matrix. Specifically, as described in Section 3.4.1, any map $\varphi : M \to N$ can be encoded as a binary matrix Π_{NM} s.t. $\Pi_{NM}(i, j) = 1$ if and only if $\varphi(i) = j$, where i and j are vertices on M and N respectively. The associated *functional map* $\mathbf{C}_{NM}$ is given as $\mathbf{C}_{NM} = \Phi_M^{\dagger} \Pi_{NM} \Phi_N$ where Φ_M, Φ_N are matrices storing as columns the first k eigenfunctions of the Laplace-Beltrami operators of shapes M, N, while $\dagger$ is the Moore-Penrose pseudo-inverse. Note that $\mathbf{C}_{NM}$ is of size $k \times k$ with k typically between 20 and 100, and is thus orders of magnitude smaller than Π_{NM}, since shapes usually contain thousands of points.

In addition to allowing to *represent* any map in a reduced basis, the functional map representation also allows to *recover* the underlying map Π_{MN} by exploiting $\mathbf{C}_{MN}$. Here and throughout this chapter we adopt the notation from [108] where objects in the reduced basis are denoted in bold.

The basic pipeline for map recovery, introduced in [106], assumes the presence of some descriptor functions that are expected to be preserved under the unknown mapping. If $\mathbf{A}_N, \mathbf{A}_M$ are the coefficients of descriptors in the basis Φ_M and Φ_N, the optimal functional map $\mathbf{C}_{MN}$ is computed as:

$$\min_{\mathbf{C}_{NM}} \|\mathbf{C}_{NM}\mathbf{A}_N - \mathbf{A}_M\|^2 + \lambda E_{reg}(\mathbf{C}_{NM}). \tag{6.1}$$

Here the first term promotes preservation of descriptor functions, whereas the second is a regularizer that promotes structural properties; e.g., $E_{reg}(\mathbf{C}_{NM}) = \|\mathbf{C}_{NM}\mathbf{\Delta}_N - \mathbf{\Delta}_M\mathbf{C}_{NM}\|^2$, where $\mathbf{\Delta}_M, \mathbf{\Delta}_N$ are diagonal $k \times k$ matrices of Laplacian eigenvalues.

The final point-to-point map $\varphi : M \to N$ can be extracted via nearest neighbor search between the rows of $\Phi_M\mathbf{C}_{NM}$ and those of Φ_N [109]. We refer to [108] for an overview of the functional map representation and its extensions.

Unsupervised Learning with Functional Maps The compactness of the functional map representation $\mathbf{C}_{MN}$ implies that the optimization problem in Eq. (6.1) reduces to a small scale least squares problem. On the other hand, the quality of the correspondence is intimately tied to the choice of the input descriptor functions. Early approaches have relied on hand-crafted features such as the Wave Kernel Signature [6]. However, more recent methods have focused on *learning* optimal features from data, first in the supervised setting [31, 85] and recently using unsupervised or weakly supervised deep learning, [132, 62, 45, 52, 140].

Our approach is directly inspired by methods in the latter category. The general approach, shared by *all* existing unsupervised or weakly supervised methods, is to train a neural network $\mathcal{F}_\Theta$ that, given a shape M can produce a set of d real-valued functions on M, $\mathcal{F}_\Theta(M) = \{f_1^M, f_2^M, ..., f_i^M\}$, where $f_i^M : M \to \mathbb{R}$.

At training time, the network $\mathcal{F}_\Theta$ is presented with a set of *pairs* of shapes M, N, and the extracted features $\mathcal{F}_\Theta(M), \mathcal{F}_\Theta(N)$ are used to estimate the functional map $\mathbf{C}_{NM}$ by first projecting the features onto the reduced basis and then solving the optimization in Eq. (6.1) (typically ignoring the regularization term E_{reg}). The network parameters Θ are then optimized by minimizing a *training loss* which penalizes some structural properties of the estimated functional map $\mathbf{C}_{NM}$.

The difference between existing methods [132, 62, 45, 52, 140] lies primarily in: *a)* The choice of feature extractor $\mathcal{F}_\Theta$ and *b)* The training losses used for learning.

Our first observation is that the vast majority of existing unsupervised learning methods have a fundamental limitation in the presence of shapes with intrinsic self-symmetries. We summarize our observation in the following theorem:

Theorem 6.3.1. *Given a set of shapes* $\{S_i\}$ *that all contain an* orientation reversing *isometric self-symmetry* $\{T_i : S_i \to S_i\}$, *s.t.* $d_{S_i}(x_j, x_k) = d_{S_i}(T_i(x_j), T_i(x_k))$, *then a generic neural network* $\mathcal{F}_\Theta$ *that is trained by any of the losses introduced in [132, 62, 52, 140, 7] has at least two possible solutions that both lead to the global optimum of the loss.*

Proof. See appendix C.1. □

In this theorem we call a neural network $\mathcal{F}_\Theta$ generic if it is capable of producing an *arbitrary* function on the shape. An orientation-reversing self symmetry is a map that is an intrinsic *reflection* such as the left-right symmetry of human shapes, and d_{S_i} is the geodesic distance on S_i.

A direct consequence of this theorem is that regardless of the neural network used, there must be at least two possible global optima, when training the networks using the unsupervised losses in the majority of existing works, in common settings involving symmetric shapes.

Existing methods have primarily tried to overcome this inherent limitation by restricting the power of the neural network, and training it, not from the shape geometry, but from some initial axiomatic features like the SHOT descriptors [158]. Unfortunately, as it has been observed in the past, e.g., [112, 144], and as we confirm in our extensive experiments, these descriptors are highly sensitive to the triangle mesh structure. Alternatively, some approaches [140] have relied on pre-aligning the shapes in 3D space (and enforcing a consistent forward direction) or used correspondences in 3D space to guide learning [45]. Such an approach, while useful for some categories, can be difficult to enforce for arbitrary non-rigid 3D shapes.

Finally, several solutions to intrinsic orientation problem have been proposed within the functional map framework. However, most of them are descriptor-based [124, 158] which are unreliable, and form very weak constraints: the map does not have to be orientation preserving but instead encouraged to follow (possibly noisy) descriptors.

6.3.2 Complex Functional Maps

In this work, we propose to address the limitations mentioned above by exploiting the *complex functional map* representation of chapter 5. The fundamental observation leading to this new tool lies in that it is challenging to recover orientation, a global signal, from a point-to-point mapping containing only local information. Therefore, the construction in chapter 5 relies on global analysis of the *pushforward* $\mathrm{d}\varphi : TM \to TN$ associated to φ whose local properties are related to normal orientation. By definition, a pushforward maps tangent vectors at $p \in M$ to tangent vectors at $\varphi(p) \in N$. $\mathrm{d}\varphi$ is also called the differential of φ and is the best linear approximation of the map at point p.

A complex functional map $Q : TM \to TN$ is a relaxed representation of the pushforward in the sense that it maps any tangent vector field on M to a tangent vector

field on N with only one constraint: it must be *complex linear*: $Q(zX) = zQ(X), z \in \mathbb{C}, X \in TM$. The adjective complex comes from the fact that tangent vector fields are represented by complex valued functions as in [143].

By definition, the pushforward contains the information of pointwise mapping (which tangent planes on N correspond to tangent planes on M) already carried by φ thus C and Q cannot be independent. Moreover, Q also contains local orientation information related to the orientation of the outward normals. Since Q is complex linear it can only represent *orientation preserving* maps. These fundamental properties have been summarized in Thm. 6.3.2 and proved in chapter 5.3.5.

Theorem 6.3.2. *The complex-linear map Q is a pushforward if and only if there exists an* orientation-preserving *and conformal diffeomorphism $\varphi : M \to N$ satisfying:*

$$\langle X, \nabla(f \circ \varphi)\rangle_{T_pM} = \langle Q(X), \nabla f\rangle_{T_{\varphi(p)}N}, \tag{6.2}$$

for all $X \in TM, f \in L^2(N), p \in M$.

Apart from this complex-linearity, the construction of complex functional maps (which we also call Q-maps) is analogous to standard functional maps described above. They can be written in the spectral basis $\{\Psi^i\}_{i\in(1,k)}$ of the connection Laplacian $\mathbf{L}$ introduced in [143]. In these reduced spaces, Q-maps are small matrices transferring coefficients in the basis Ψ_M to coefficients in Ψ_N. A point-to-point map is extracted from a nearly isometric $\mathbf{Q}$ using a nearest-neighbor search on Dirac functions: $\Pi_{MN} = \text{NNsearch}(\text{div}_M \Psi_M, \text{div}_N \Psi_N \mathbf{Q})$ (see chapter 5.4.6).

Moreover, a pushforwad $\mathbf{Q}$ is isometric if and only if it commutes with the connection Laplacian: $\mathbf{Q}\mathbf{L}_M = \mathbf{L}_N\mathbf{Q}$. Finally, as shown in chapter 5.3.6 a necessary condition for $\mathbf{Q}$ to represent a pushforward is that $\mathbf{Q}$ must be orthogonal *i.e.* $\mathbf{Q}^\star\mathbf{Q} = \mathbf{I}$ where * denotes the complex transposition. Here, unlike functional maps, orthogonality is not equivalent to area-preservation.

A complex functional map can be estimated by minimizing a simple optimization problem, similar to Eq. (6.1):

$$\mathbf{Q}_{MN} = \arg\min_{\mathbf{Q}} \|\mathbf{Q}\mathbf{B}_M - \mathbf{B}_N\|_F^2 + E_{\text{reg}}(\mathbf{Q}), \tag{6.3}$$

where $E_{\text{reg}}(Q) = w_{\text{Q-ortho}}\|\mathbf{Q}^\star\mathbf{Q} - \mathbf{I}\|_F^2 + w_{\text{Q-iso}}\|\mathbf{Q}\mathbf{L}_M - \mathbf{L}_N\mathbf{Q}\|_F^2$, and $\mathbf{B}$ are the coefficients of complex (tangent vector)-valued features expressed in spectral basis of the corresponding shape.

6.4 Method

In this section, we describe our proposed network in detail. As mentioned in Section 6.3, a deep functional map pipeline can be decomposed into three different building blocks:

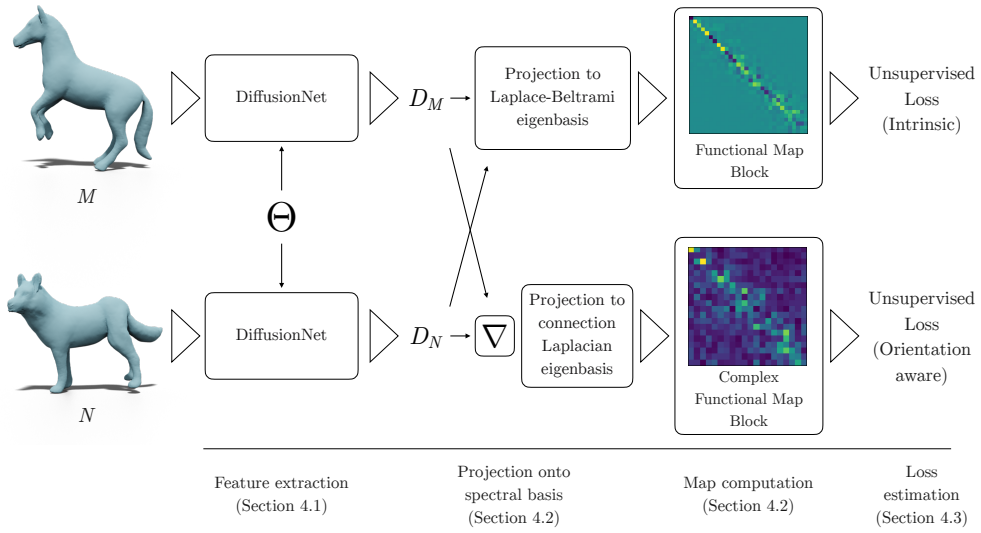


Figure 6.2: Overview of our unsupervised network. We extract source and target descriptors D_M and D_N using DiffusionNet [144] and then project descriptors onto the Laplace-Beltrami eigenbasis and descriptor gradients onto the connection Laplacian eigenbasis. This leads respectively to the Functional Map block (Section 6.4.2) and the Complex Functional Map block (Section 6.4.2). Losses are imposed on both of these maps (Section 6.4.3).

the feature extractor (Sec. 6.4.1), the non-learnable functional map layer (Sec. 6.4.2) and the loss (Sec. 6.4.3). We describe our design choices for each of these components, and how they permit orientation-aware unsupervised learning in the subsections below. We also provide a graphic representation of our whole approach in Figure 6.2.

6.4.1 Feature Extractor

The first major component of our network is the deep feature extractor. Its structure is that of a Siamese network extracting features for both source and target shapes. We use a recent surface feature extractor backbone DiffusionNet [144], and use the Wave Kernel Signature [6] (WKS) as input for the network, for its rotational invariance property. DiffusionNet then outputs feature vectors of dimension d on the source and target shapes (respectively composed of n_M and n_N vertices). We denote by $D_M, D_N \in \mathbb{R}^{n_M \times d} \times \mathbb{R}^{n_N \times d}$ the learned source and target features.

The use of DiffusionNet [144] makes our approach highly robust to changes in *shape triangulation* unlike previous unsupervised approaches relying on SHOT descriptors [158]. These methods usually fail if trained and tested triangulations are different, as demonstrated in Section 6.5. On the contrary, DiffusionNet is based on robust dif-

fusion, and is consequently largely independent on the choice of shape triangulation. Although learned diffusion is fully intrinsic, the network is aware of the shape orientation because of oriented gradient blocks, as described in [144] (Section 3.4). Therefore, our feature extractor can produce orientation-aware features that we use later to estimate orientation-preserving complex functional maps.

To sum up, we use DiffusionNet jointly with WKS inputs to build discretization-agnostic features. In the results Section 6.5, we exhibit shapes with anisotropic triangulations to illustrate that methods relying on SHOT descriptor refinement tend to overfit to the triangulation rather than learn relevant shape information.

6.4.2 The Functional Map Blocks

This block, first introduced in [85] estimates the functional map in a differentiable way from the source and target features estimated by the feature extractor.

Regularized Functional Map Block

The input features D are projected on spectral Laplace-Beltrami eigenbasis Φ_S to get spectral features $\mathbf{A}_S = \Phi_S^\dagger D_S$ with $S \in \{M, N\}$. The functional map $\mathbf{C}_{NM}$ is then estimated as the solution to the following least-squares problem:

$$\mathbf{C}_{NM} = \arg\min_{\mathbf{C}} \|\mathbf{C}\mathbf{A}_N - \mathbf{A}_M\|_F^2,$$

leading to:

$$\mathbf{C}_{NM} = \mathbf{A}_M \mathbf{A}_N^\dagger. \tag{6.4}$$

In this work, we use the *regularized* approach, introduced in chapter 4.4.4, which incorporates the Laplacian commutativity energy $\|\mathbf{C}_{NM}\mathbf{\Delta}_N - \mathbf{\Delta}_M\mathbf{C}_{NM}\|_F^2$ in a differentiable manner, directly in the functional map estimation step.

Complex Functional Map Block

The complex functional map estimation is analogous to that of the standard functional map. We first convert the feature functions D to vector fields using the discrete gradient operator G. We visualize these vector field descriptors in Figure 6.1 (where they are rotated by $\pi/2$ to better see singularities). These vector valued descriptors are then projected in the eigenbasis Ψ of the connection Laplacian. This leads to complex spectral feature vectors $\mathbf{B}_S = \Psi_S^\dagger G_S D_S$ with $S \in \{M, N\}$, and G_S the gradient operator on shape S.

The complex functional map $\mathbf{Q}_{MN}$ is then estimated as the solution to the following least-squares problem:

$$\mathbf{Q}_{MN} = \arg\min_{\mathbf{Q}} \|\mathbf{Q}\mathbf{B}_M - \mathbf{B}_N\|_F^2,$$

whose closed-form solution is given simply as:

$$\mathbf{Q}_{MN} = \mathbf{B}_N \mathbf{B}_M^{\dagger}. \tag{6.5}$$

In our work, we extend the in-network Laplacian regularization of chapter 4.4.4, and apply it to complex functional map estimation by modifying the least squares system in the same way as was done originally for real-valued functional maps.

Remark that as mentioned in Section 6.3.2, the complex functional map estimated from feature gradients is a pushforward if and only if the features themselves give rise to an orientation-preserving map. We elaborate on the relation between the two blocks in Section 6.4.3 below. Specifically, we demonstrate that although $\mathbf{C}$ and $\mathbf{Q}$ are estimated independently, they still satisfy the equation of Thm. 6.3.2.

6.4.3 Losses

From estimated $\mathbf{C}, \mathbf{Q}$ we build a loss inspired by SURFMNet [132].

Loss on C. SURFMNet [132] imposes the estimated functional map $\mathbf{C}$ to be orthogonal, resulting in the first loss L_{ortho}:

$$L_{\text{ortho}}(\mathbf{C}) = \|\mathbf{C}^{\top}\mathbf{C} - \mathbf{I}\|_F^2 \tag{6.6}$$

Moreover, they also propose to promote isometry through commutativity between $\mathbf{C}$ and the Laplace-Beltrami operators $\mathbf{\Delta}_M, \mathbf{\Delta}_N$, resulting in the second loss L_{iso}:

$$L_{\text{iso}}(\mathbf{C}) = \|\mathbf{C}\mathbf{\Delta}_N - \mathbf{\Delta}_M\mathbf{C}\|_F^2 \tag{6.7}$$

As stated previously, we remark that this isometric loss *is not necessary* if we estimate $\mathbf{C}$ with the Laplacian regularizer of chapter 4.4.4. Indeed, the regularizer only produces maps that have a low isometric loss. We therefore only use L_{ortho} in our implementation.

The association of these two losses is generally enough to estimate intrinsically an isometric map. The fundamental problem that we propose to remedy here is the fact that these two losses are not in themselves sufficient to rule out intrinsic symmetries. We stress again that many previous works rule out these symmetries based on triangulation only, using the SHOT feature extractor. However these methods are then biased towards the training triangulations.

Loss on Q. As demonstrated in chapter 5.3.6, a complex functional map will only encode a pointwise map (which will then be orientation preserving) if it is an orthogonal matrix. Hence the complex orthogonal loss $L_{\text{Q-ortho}}$:

$$L_{\text{Q-ortho}}(\mathbf{Q}) = \|\mathbf{Q}^{\star}\mathbf{Q} - \mathbf{I}\|_F^2 \tag{6.8}$$

Moreover, since we aim for maps which are as isometric as possible, we can also use a complex isometric loss $L_{\text{Q-iso}}$, evaluating the lack of commutativity with the connection Laplacians:

$$L_{\text{Q-iso}}(\mathbf{Q}) = \|\mathbf{Q}\mathbf{L}_M - \mathbf{L}_N\mathbf{Q}\|_F^2 \tag{6.9}$$

We observe that this isometric loss can also be avoided by computing Q using a Laplacian regularizer (4.4.4). We therefore only use $L_{\text{Q-ortho}}$ in our implementation.

Loss Function In summary, we use two losses that we combine to compute the final loss L_{final}:

$$L_{\text{final}}(\mathbf{C}, \mathbf{Q}) = w_{\text{ortho}} L_{\text{ortho}}(\mathbf{C}) + w_{\text{Q-ortho}} L_{\text{Q-ortho}}(\mathbf{Q})$$

These losses, jointly with the Laplacian regularizers ensure that the learned descriptors result in an isometric map and that this map is orientation-preserving. The whole pipeline remains both light and unsupervised.

Correlation of the Two Functional Map Blocks

Note that we never explicitly use Eq.(6.2) relating $\mathbf{C}$ and $\mathbf{Q}$ required by Theorem 6.3.2. However, we prove that when both functional maps are nearly isometric and estimated from the same features this relation is always verified.

Theorem 6.4.1. *Let M, N be two manifolds, and F_M, F_N surface features such that the functional map C estimated from these features is an isometry. Let Q be the complex functional map computed with the feature gradients as described in Section 6.4.2. Then the maps (C, Q) must satisfy Eq. (6.2), and C is an orientation-preserving isometry.*

Proof. See appendix C.1. □

The isometric assumption is not restrictive in the sense that deep spectral methods already implicitly make this assumption. Moreover, as we demonstrate below, our approach is robust even for non-isometric shape categories.

6.4.4 Implementation

We implemented our method with Pytorch 1.8 (this version is required to include complex Tensors in the differentiable pipeline) by adapting the open-source implementation of DiffusionNet [144] for the feature extractor and chapter 4.4.4 for the functional map block with Laplacian regularizer.

We use WKS descriptors [6] as input signal for the network. We use this descriptor because it is: *a)* Robust to changes in the shape triangulation, and captures the intrinsic geometry of the surface. As shown in the next section that guarantees that the network learns relevant surface information, rather than overfit to the mesh triangulation. *b)*

Independent of the embedding of the shape. This makes our approach fully rotationally invariant, and capable of predicting correspondences between arbitrarily rotated shapes. Indeed methods such as [140, 46] depend on pre-aligned datasets to work, which makes them only weakly supervised instead of fully unsupervised.

Our feature extraction network consists of 4 DiffusionNet blocks (a standard DiffusionNet setup [144]), where the 128-dimensional input WKS features are transformed by each block to learned features of same dimension 128, to finally produce 128-dimensional descriptors on source and target shape. As described in Section 6.4.1, our network is applied in a Siamese way on the two input shapes, using the same weights for feature extraction on source and target.

Parameters

In addition to the architecture above, our method has some key hyper-parameters: *a)* The size of both spectral basis: we use $k_C = 50$ for Laplace-Beltrami and $k_Q = 20$ for connection Laplacian *b)* The Laplacian regularizer from chapter 4 in the functional map blocks: we use $\lambda = 10^{-3}$ as recommended in chapter 4.4.8 *c)* The loss hyper-parameters: The loss if focused on map orthogonality since Laplacian-commutativity is enforced previously with the regularizers. We enforce both maps to be "equally" orthogonal, by setting $w_{\text{ortho}} = w_{\text{Q-ortho}} = 1$.

We train our network with a batch size of 1 for a number of epochs between 5 and 30. We use a learning rate of 10^{-3} with ADAM optimizer [42].

6.5 Results

In this section, we show that our network can outperform state-of-the-art deep shape matching architectures on standard datasets like FAUST (F_r) [19] and SCAPE (S_r) [3] as-well-as non-isometric datasets like SHREC'19 [93] and SMAL [175]. Following [124], all shapes are remeshed so that they do not share the same connectivity. Moreover, we introduce an anisotropic re-meshing of FAUST (denoted F_a) and SCAPE (denoted S_a), generated with Mmg [38, 34], to demonstrate how some methods overfit to mesh connectivity to disambiguate between intrinsic symmetries. We show our anisotropic remeshings in Figure C.1.
In all our tables, we denote by F/S a method trained on the dataset F and tested on S.

6.5.1 Quantitative Results

Baselines

We compare our method to:

- Axiomatic methods: BCICP [124] and ZoomOut [94] are very efficient for solving close to isometric matching. It should also be noted that these axiomatic

Meth / Data	F_r/F_r	F_r/F_a	S_r/S_r	S_r/S_a
SHOT+FMNet [85]	5.8	43.	7.0	41.
WKS+GeoFMap [39]	**2.0**	**2.6**	**2.2**	**2.3**
BCICP [124]	6.1	8.5	11.	14.
ZoomOut [94]	6.1	8.7	7.5	15.
SHOT+UnFMNet [62]	5.7	42.	9.9	44.
SHOT+DeepShells [45]	**1.7**	12.	**2.5**	10.
WKS+DeepShells [45]	8.2	9.5	8.3	20.
WKS+Ours	2.5	**3.0**	2.6	**2.7**
Cross Training				
Meth / Data	S_r/F_r	S_r/F_a	F_r/S_r	F_r/S_a
SHOT+FMNet [85]	14.	43.	11.	44.
WKS+GeoFMap [45]	**9.9**	**8.4**	**3.8**	**3.9**
SHOT+UnFMNet [62]	12.	44.	9.3	43.
SHOT+DeepShells [45]	**2.7**	15.	5.7	16.
WKS+DeepShells [45]	6.7	12.	9.2	21.
WKS+Ours	**2.7**	**3.1**	**4.2**	**4.4**

Table 6.1: Comparative results ($\times 100$) of all main baselines on FAUST and SCAPE remeshed and anisotropic. Deep Learning methods are displayed with the descriptor input they were fed during training time. The methods shown in the top group are supervised, while the ones below (separated by a double line) are axiomatic or unsupervised. Note that our approach outperforms all unsupervised baselines *without post-processing*, and achieves similar or better performance even to supervised ones.

methods are slower than a test pass of our method *which does not require post-processing*.

- Supervised methods: FMNet [85], and GeoFMap [39] (which is the network of Chapter 4) where we replaced KPConv [156] with DiffusionNet [144] as feature extractor.

- Unsupervised methods: Unsupervised-FMNet [62] (denoted as UnFMNet), and the state-of-the-art method Deep Shells [45].

For the most relevant baselines, we compare our method, which uses WKS descriptors, to both original networks (trained with SHOT descriptors) and their variant when trained with WKS as input. Consequently, we denote as "SHOT+Net" a Network trained with SHOT and "WKS+Net" its variant with WKS as input.

Anisotropic FAUST and SCAPE

Meth / Data	F_r/Sh_r	S_r/Sh_r	Sh_r/Sh_r
BCICP [124]	15.	15.	15.
ZoomOut [94]	21.	21.	21.
SHOT+DeepShells [45]	27.	24.	24.
WKS+DeepShells [45]	27.	29.	28.
WKS+Ours	**6.4**	**8.4**	**3.9**

Table 6.2: Comparative results ($\times 100$) on SHREC'19 re-meshed with different train sets, including SHREC'19 re-meshed itself. We compare unsupervised methods on this more challenging dataset, and keep the same notations as in Table 6.1. We see that our approach gives the best correspondences and their quality is relatively stable with respect to the training set.

For this experiment we train networks on FAUST and SCAPE re-meshed as in [124], and test them on both re-meshed and anisotropic. We report the mean geodesic errors in Table 6.1.

From the results shown in Table 6.1, we see that: *a)* Our method is robust to triangulation changes which make SHOT-based methods fail at test time (often because they mistake anisotropy for meaningful geometric information). Deep Shells [45] is the most robust SHOT-based approach, since its feature extractor uses spectral filters which helps filter out high-frequency overfitting. Still, the quality of the correspondence collapses on anisotropic datasets. *b)* Deep Shells, if presented with an intrinsically symmetric signal (here WKS [6]) as input, fails to learn accurate descriptors, resulting in overall poor correspondence. *c)* Our approach gives the best results among unsupervised networks. Moreover, the quality of the correspondence is often close to that achieved by the best supervised baseline WKS+GFM [39] (method presented in chapter 4).

SHREC'19 Re-meshed

For this second experiment, we train the networks respectively on FAUST, SCAPE

Meth / Data	SMAL_r
BCICP [124]	19.
ZoomOut [94]	35.
SHOT+DeepShells [45]	25.
WKS+DeepShells [45]	33.
WKS+Ours	**4.8**

Table 6.3: Comparative results ($\times 100$) on SMAL re-meshed dataset. This animal dataset exhibits strong non-isometries, as can be seen on the qualitative result in Figure 6.3, to which only our method proves to be robust.

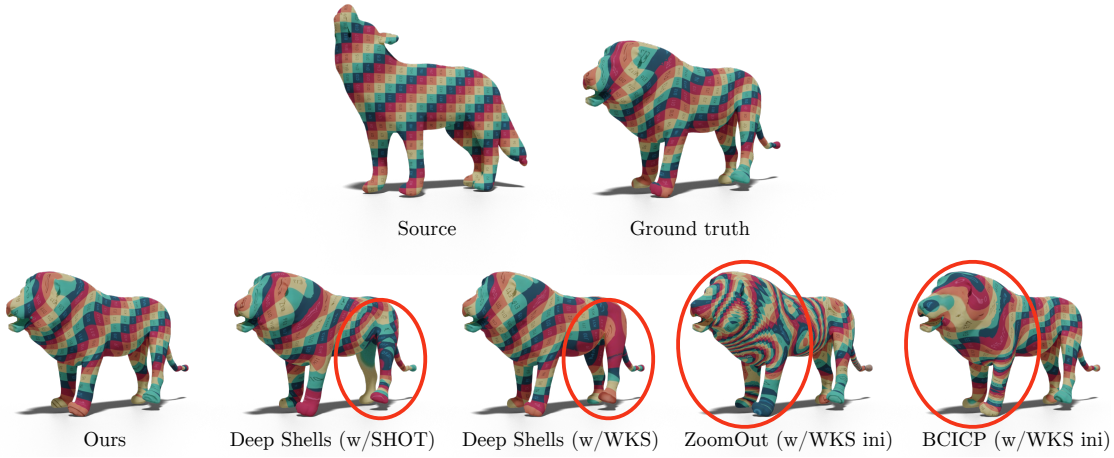


Figure 6.3: Qualitative results for baselines on the SMAL dataset. The areas where baselines gave the most wrong predictions are highlighted in red.

and SHREC'19 (denoted as F_r, S_r and Sh_r) re-meshed so that mesh connectivity are different and tested only on SHREC'19. We removed shape 40 from SHREC'19 in this experiment since it is the only partial non-closed shape and therefore outside the scope of this method.

We show in Table 6.2 that Deep Shells fails to generalize to this test set, even though all meshes have similar number of vertices and well-shaped meshes. In comparison, our method gives significantly better results, as it is a purely geometric approach tailored to exploit surface information rather than triangulation details and to produce well-oriented maps.

SMAL Re-meshed

We test unsupervised methods on SMAL shapes [175] (again re-meshed so that connectivity is different on every mesh as in [124]). The dataset, originally composed of 49 shapes, is split in 32 training shapes and 17 test shapes. This dataset constitutes the hardest of the three experiments, since its shapes are *often strongly non-isometric*. Indeed, they involve animal shapes of different species, which often poses significant challenges for existing (especially spectral) approaches.

The results of this experiment are reported in Table 6.3. Observe that our method is the *only* one that produces even reasonable correspondence, significantly outperforming the closest competitor. This highlights the significant additional robustness ensured by our combination of an accurate feature extractor with a range of well-founded geometric regularizers, which allow our approach to accommodate even for non-isometric shapes and learn from limited training data.

6.5.2 Qualitative Results

We additionally provide a qualitative result for our third experiment in Figure 6.3, comparing our method with baselines on SMAL. We see that our method yields a map very close to ground-truth, even on this challenging example with strong non-isometric distortions. Meanwhile, both axiomatic and learning baselines fail to predict accurate correspondences. We provide another qualitative comparison in Figure C.2.

6.6 Conclusion, Limitations & Future Work

To conclude, we introduced a new fully unsupervised way to efficiently learn accurate descriptors for shape matching. These descriptors are guaranteed by our complex functional map loss to be orientation aware in the sense that the resulting map is orientation preserving, making the pipeline robust to aliasing due to shape intrinsic symmetries. Besides, the use of DiffusionNet for a feature extractor enables robustness to changes in shape discretization.

Our approach has several limitations: the loss, which aims at as-isometric-as-possible functional maps is dependent on the fact that the input shapes are not too non-isometric. It would therefore be interesting to use spectral bases adapted to non-isometry, as in [91], which we leave as future work. Another current limitation of our method is that it needs manifold meshes as input. However, good Laplacian operators can be built on point clouds [141], and it would be interesting to check how robust our pipeline is to potentially noisy point cloud inputs. Finally, we believe it would be interesting to further leverage complex functional maps in other learning applications, while promoting general rotation-invariant and orientation-preserving maps.

Acknowledgements Parts of this work were supported by the ERC Starting Grant No. 758800 (EXPROTEA) and the ANR AI Chair AIGRETTE.

CHAPTER 7

Conclusion

In this thesis we have studied shape matching through the lens of the functional map representation. First we have reviewed and analyzed deep learning methods and found ways to make them more robust and generalizable. Secondly, we have introduced a new representation for maps between shapes that also falls in the spectral category, and deals with tangent bundles. The field of non-rigid shape matching still presents a variety of challenges and open problems, which we give an overview of in the next paragraphs.

Deep Learning in Shape Matching

Most of this thesis is focused on deep learning methods for shape matching. These techniques learn feature functions jointly on the source and target shapes to estimate a functional map on which a loss is subsequently designed. However, several parts of this deep shape matching pipeline could be upgraded.

Firstly, the type of input fed to the network. Indeed, we have experimented with point clouds and meshes, but it could be interesting to analyze the behavior of existing deep shape matching works with different input data, like 2D images or 3D voxel grids, implicit signed distance functions, or even other shape representation [100]. In this thesis we focused on spectral methods for deep shape matching, which require the possibility to build a Laplacian operator on the input data. In order to explore different input types, one would still need to define a Laplacian operator on them (which for instance has been explored on point clouds [141]).

Secondly, different signals can be designed on shapes as input for the feature extractor network. We have restricted ourselves to the study of: *(a)* the *extrinsic* 3D embed-

ding signal. This signal is particularly useful to disambiguate between intrinsic symmetries, but is not independent to rigid rotations or translations of the input shapes and therefore requires data augmentation. *(b)* The *intrinsic* wave kernel signature [6], which is independent to rigid displacement of the input shapes but is particularly affected by intrinsic symmetries. It would be interesting to explore in more details the different possibilities for input signals, ultimately combining extrinsic and intrinsic information in a way that brings together the best of both worlds.

Another possibility would be to let the network learn directly from the shapes themselves, without requiring any input signal design or data augmentation. This would require adapting the structure of current feature extractors. One potential network structure that has recently gained popularity and strives to tackle the aforementioned issue is that of equivariant networks ([114, 169, 35, 28]). These deep learning architectures are by construction equivariant to translation and rotation of the input shape, making this feature extractor particularly relevant in the case of deep shape matching. We believe that network structures can still undergo drastic improvement, especially when 3D data is concerned. Consequently, the feature extractor is one of the parts in this pipeline that could be upgraded. Additionally, we never tried to differ from the siamese structure introduced in FMNet [85]. Although this strategy has proved to be efficient and generalizable, some avenues remain unexplored.

It can also be noted that the features learned by deep shape matching networks, whether supervised or unsupervised, are theoretically very informative and invariant to learned non-rigid deformation. It could therefore be interesting to explore more fully transfer learning options for other shape analysis tasks, such as shape semantic segmentation or shape interpolation. In our current pipeline, we always use the Laplacian eigenbasis to project the learned features, which are then used for map computation. Although the Laplace-Beltrami operator has proven to be a prominent tool in discrete differential geometry and 3D shape analysis, it is possible that other operators, and functional bases are in fact more adapted to our problem. Alternatively, efficient methods to *learn* these bases jointly with the shape features such as [91] might be more adapted to strong non-isometries.

One of the most fundamental and critical building blocks of our method is the losses involved in the learning process. Indeed, we have relied on spectral losses, directly imposed on the small-scale functional maps for fast and stable computations. Yet, some works like Deep Shells [45] display efficient losses combining both spectral and spatial information for better shape alignment. Designing relevant losses for map between two shapes is a very active field in the geometry processing community [99, 46, 153, 82], and we have noticed throughout this thesis that efficient meaningful loss functions often translate to robust learning pipelines for shape matching.

Finally, we have only explored spectral methods for shape matching in our works. While these approaches are particularly well-suited for small-scale training by taking

an intrinsic and low-frequency point of view on shapes, they are not easily transferred to the non-isometric case or different types of 3D data.

Robust Representations for Shape Matching

The other part of this dissertation is dedicated to introducing complex functional maps. Previously, the functional map pipeline, as a fully intrinsic approach, could not rule out orientation-reversing maps efficiently. Using the complex structure of Riemannian surfaces, we proposed a map encoding for pushforwards between tangent bundles with given orientations. We used this map encoding to show that it is possible to filter out the orientation-reversing part of an input functional map. In previous works [124], orientation was taken into account through additional energies. We showed that our approach is easily plugged in any functional map pipeline, and always helps improve the quality of resulting correspondence. We also showed in chapter 6 that complex functional maps can be used to make unsupervised deep learning algorithms for shape matching. Additionally, complex functional maps open new possibilities for a lot of future work in shape matching.

Firstly, one major remaining challenge is to convert spectral complex maps back to pointwise maps in a meaningful way. Indeed, it should be theoretically possible to access a highly regular (differentiable) mapping by integrating the pushforward encoded in a complex functional map. Algorithms to retrieve continuous and bijective mappings [139, 154], despite being very precise are often slow and involve heavy optimization. We believe that spectral methods would help make these algorithms faster and more efficient.

Secondly, it would be interesting to extend this framework to more exotic tangent bundles (e.g. some discussed in [143]), containing more information. This would potentially give the possibility to include extrinsic data in a natural way to the spectral pipeline.

Finally, the use of tangent vector fields seems promising in other shape analysis tasks. For instance, the use of gradient features is central in the 3D mesh feature extractor DiffusionNet [144]. More generally, extrinsic vector fields can be integrated in vector field flows, which seem to be promising in diffusion-based techniques, such as PointFlow [170]. In essence, vector fields give access to motion of the underlying object, which can be key to problems such as shape interpolation or 3D scene understanding.

Consequently, we believe that the framework we introduced could potentially be extended to complex 3D tasks like deformation learning, high-precision map estimation, or feature tracking.

Over the course of this thesis and thanks to all the people who contributed to the field, shape matching methods have been significantly improved. While many modern

algorithm rely on deep learning to learn robust shape representations, some works [125, 139, 127, 89] are still fully axiomatic. However, it is undeniable that deep learning has changed the field of shape matching and had contributed to make it more robust to strong non-isometry [82]. More broadly, the field of geometric deep learning has also been through considerable development, as 3D data has become more available as well as met more success, with e.g. differentiable rendering [70] and perhaps more notably NERF [95, 83]. The field has come from point cloud or mesh analysis with classification and segmentation networks [92, 115, 116, 5, 112, 156] to more complex network structures and shape analysis tasks. Modern methods make it easier to go from point cloud to meshes [118, 142, 119], to interpolate between shapes using latent space exploration [55, 56, 117], even to generate 3D point clouds [170], potentially based on images [26, 27].

Still, there are many remaining challenges in geometry processing and geometric deep learning. For instance, the lack of a universally usable, large-scale 3D dataset to make data-based algorithms even more robust and general. It is still a hard problem to go from 2D images to high-detailed and accurate geometry. Meshing a point cloud with precise selection of outliers also remains a major issue. Going even further, unifying 3D data under a more general shape representation is still an important bottleneck. In shape matching, it is important to have access to *robust* shape representations, stable to non-rigid deformation in order to recover high-quality correspondence. The datasets used in deep shape matching still contain only a small number of shapes and more importantly, low shape variety, making current methods overfitting prone. A key challenge for modern shape matching algorithms is strong non-isometry, and to tackle this issue both wider datasets and new robust network regularizations will pave the way to more efficient and accurate methods.

Ethical Discussion

We made this work with the best of faith that it will not be used for malicious ends. We are nevertheless aware that 3D shape matching could potentially be used to develop algorithms for face tracking and more generally, population control. However, we firmly believe that there are many *useful* applications of 3D shape matching, one of which is medical imaging. AI has proved to produce robust diagnosis in this field, and modern geometric deep learning could make 3D-based predictions which make these methods more robust and accurate.

APPENDIX A

Deep Geometric Functional Maps: Supplementary Material

A.1 Additional details on KPConv

Here, we review briefly KPConv [156] method and describe the architecture we used in our implementation.

The input to this network is a 3D point cloud equipped with a signal, such as the 3D coordinates of the points. Let $\mathcal{P} \in \mathbb{R}^{N \times 3}$ be a point cloud in $\mathbb{R}^3$. Let $\mathcal{F} \in \mathbb{R}^{N \times D}$ be a D-dimensional feature signal over $\mathcal{P}$.

The goal of point cloud convolutional networks is to reproduce the architecture of convolutional neural networks on images. It boils down to transferring two key operations on the point cloud structure: the convolution and the pooling operators.

First, we define a convolution between $\mathcal{F}$ and a kernel g at point $x \in \mathbb{R}^3$. Since we only want a signal over the point cloud at each layer, we only need these convolutions at $x \in \mathcal{P}$.

The kernel will be defined as a local function centered on 0 depending on some learnable parameters, taking a D-dimensional feature vector as input and yielding a D'-dimensional feature vector.

More specifically, the kernel is defined in the following way : let r be its radius of action. Let $\mathcal{B}_3^r$ be the corresponding 3D ball. Let K be the number of points, thus the number of parameter matrices in this kernel. Let $\{z_k | k < K\} \subset \mathcal{B}_3^r$ be these points, and $\{W_k | k < K\} \subset \mathcal{M}_{(D,D')}(\mathbb{R})$ be these matrices. Then the kernel g is defined through the formula :

Method	No Ref	Ref
FMNet	17.	13.
PointNet	18.	14.
Old FMap	4.5	**1.9**
Ours	**3.4**	**1.9**

Table A.1: Comparative results for the different ablations of our method.

$$g(y) = \sum_{k<K} h(y, z_k) W_k$$

where we simply set $h(y, z) = \max(0, 1 - \frac{\|y-z\|}{\sigma})$, so each point of the kernel has a linear influence of range σ around it.

Then the convolution simply becomes :

$$(\mathcal{F} * g)(x) = \sum_{i | x_i \in \mathcal{B}_3^r} g(x_i - x) f_i$$

Where the learnable parameters are the matrices W_k. We set the points z_k of the kernel to be uniformly organized in $\mathcal{B}_3^r$, so as to better encompass the variations of the convoluted signal at a given point of the point cloud, and a given scale (see [156] supplementaries, Section B for more details).

For the pooling operator, we use a grid sampling that allows us to get the point cloud at an adjustable density. The network can then build hierarchical features over the point clouds by both adjusting the radius of influence of its kernels and the density of the mesh they are performed upon.

Once these two operations are set up, it is easy to build a convolutional feature extractor over the point cloud $\mathcal{P}$. In our work, we use the following architecture :

- Four strided convolutional blocks, each down-sampling the point cloud to half its density, and taking the feature space to another feature space (corresponding to higher-level features) two times larger.
- Four up-sampling layers, taking the signal back on the whole point cloud, through skipping connections followed by 1D convolutions.

A.2 Ablation study

This section presents the extensive ablation study of all the vital components of our algorithm.

We train all these ablations on 100 random shapes among the 230K proposed by 3D-CODED, as in experiment 2. We test them on the 20 test shapes of FAUST re-meshed, so that the connectivity differs from train to test. The different parts to ablate are :

- The point cloud feature extractor : as explained in the previous sections, it learns descriptors from raw data without relying too much on connectivity. The first ablation consists of our method, but with FMNet [85] feature extractor instead of ours. Similar to FMNet, we use SHOT [158] descriptors. However, we use the same number of eigenvectors as in our method, namely 30. As a general remark, we noticed that lowering this number can often help prevent overfitting in the case of FMNet-based architectures.

- The choice of KPConv [156]. The second ablation study replaces KPConv sampling and feature extractor block with that of PointNet [115]. For this ablation, we use random sampling to 1500 points instead of KPConv grid sampling. Indeed, grid sampling does not provide any guarantee on the number of points after sampling, so it can only be used in a network built to overcome this issue, with batches of adaptable size, which is not the case of PointNet.

- The regularized functional map layer. This third ablation simply consists in replacing our regularized functional map layer by the old functional map layer originally introduced by FMNet. Our layer is in theory mildly heavier than the original one, but in practice for less than 50 eigenvectors the computation times remain the same.

- Lastly, we remove the post-processing refinement step. Here we show the results of every ablation with and without refinement, thus proving it helps in getting

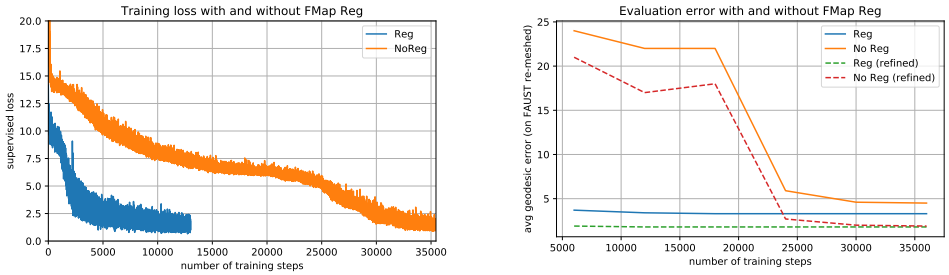


Figure A.1: Comparison of convergence speed with and without Laplacian Regularization in the FMap block. *Left*: Training loss evolution, *Right*: Evolution of geodesic error on test set with the number of epochs. Notice how the regularized fmap layer helps drastically with the convergence speed. It gives optimal results within only 500 epochs.

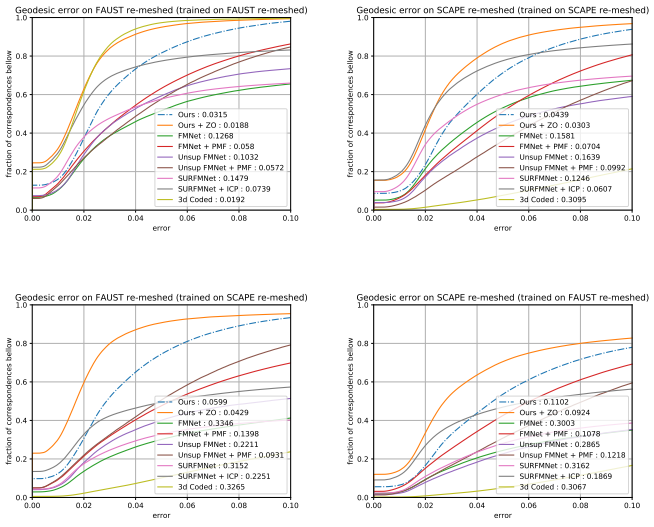


Figure A.2: Quantitative results of the different methods using the protocol introduced in [72], on all the settings of Experiment 1

better results. As a refinement method, we use the state-of-the-art ZoomOut [94], as mentioned in Chapter 4.5.

Table A.1 shows the ablation study of our method. It demonstrates the importance of all individual blocks and ascertains that all these components are needed to achieve optimal performance with our solution.

However, just looking at the results of the ablation study one does not see the importance of the FMap Reg addition. To prove its efficiency, we compare the learning and evaluation curves of our method, with and without this addition. As can be seen in Figure A.1, the models converge much faster with our regularized functional map layer. The models are trained on 100 shapes of the surreal dataset of 3D-CODED as in Experiment 2, and tested on FAUST re-meshed.

In addition, our regularized functional map layer is more robust, and does not result in a Cholesky Decomposition fatal error when computing the spectral map. In comparison, the previous functional map layer gave that fatal error in some experiments, and the model had to be relaunched.

Graphically, as reported in the original functional map paper [106], a natural functional map should be funnel-shaped. Our regularized functional map layer naturally computes maps that almost commute with the Laplacians on the shapes. These maps will naturally be close to diagonal matrices (as are funnel shaped maps) in the eigenbasis of the Laplacians, thus reducing the space of matrices attainable by this layer. We believe it helps the feature extractor block focus on setting the *diagonal* coefficients of

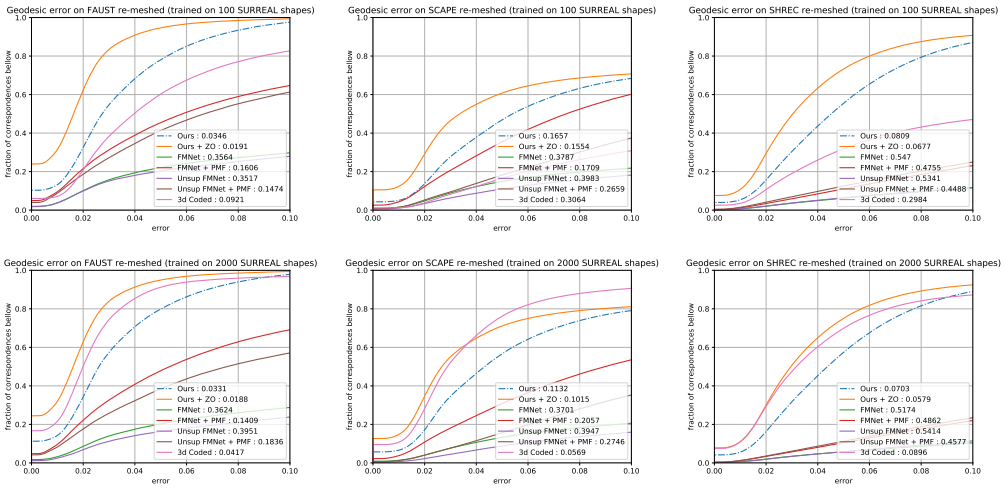


Figure A.3: Quantitative results of the different methods using the protocol introduced in [72], on two of the settings of Experiment 2. Top row: 100 shapes (low number). Bottom row: 2000 shapes (high number).

the functional map as in the ground truth, rather than on trying to get its funnel shape in the first thousand iterations, which is what a model with the original functional map layer does.

A.3 More quantitative results

Figures A.2 and A.3 summarize the accuracy obtained by our method and some baselines on the different settings of the two experiments we conducted, using the evaluation protocol introduced in [72]. Note that in all cases but one (trained on 2000 shapes of SURREAL, tested on SCAPE re-meshed), our network achieves the best results even compared to the state-of-the-art methods. As explained more thoroughly in the main manuscript, this proves our method is able to learn point cloud characterizations with only a small amount of data, and by projecting these descriptors in a spectral basis can retrieve accurate correspondences from them. Our method does not need any template and is thus more general than 3D-CODED, in addition to the fact that it trains faster and does not need a big training set.

We even believe the superiority of our method with a low number of training shapes is partially due to this fact that 3D-CODED uses a template and operates in the spatial domain, unlike our approach which is template-free, and partly operates in the spectral domain, making it easier to adapt to any new category of 3D shapes.

The relatively low performance of our method on SCAPE in Experiment 2 (see

Figure A.3) is due to the presence of back-bent shapes in this dataset. These shapes are seen by the network through their truncated spectral approximation, as discussed in Section A.4.1, making it unable to exploit refined features such as the face or hands, that could help getting descriptors able to differentiate left from right. Consequently, as there are no back-bent shapes in the training sets of this experiment, these shapes are often mapped with a left-to-right symmetry, resulting in a huge error for these particular shapes, increasing the mean error for the whole SCAPE test set.

A.4 More qualitative results

In this section we provide more qualitative results for the experiments described in 4. More specifically, we show a visualization of some point cloud descriptors learned by our method on SHREC'19 shapes. Then we provide another texture transfer obtained

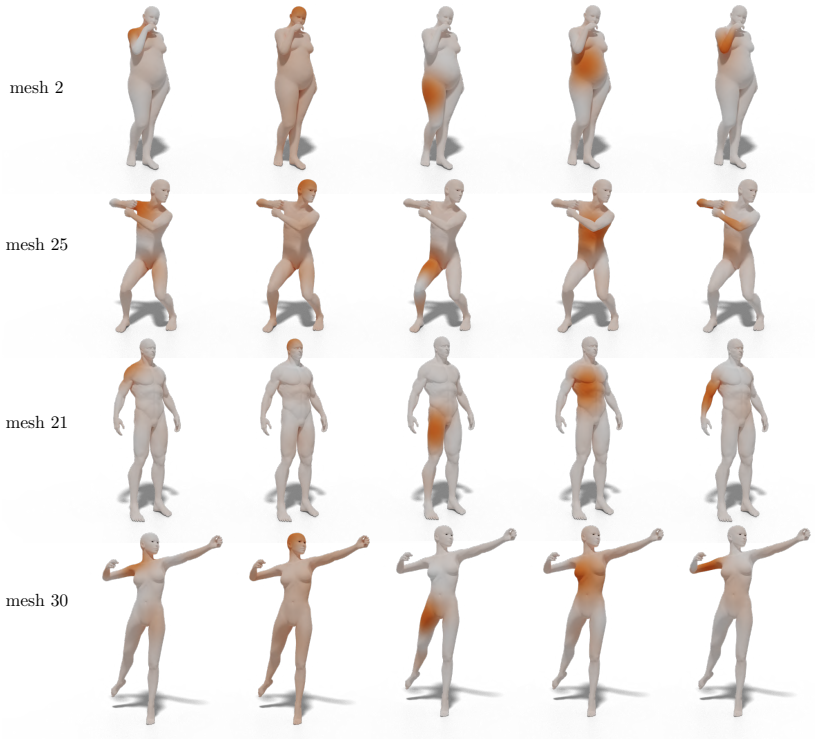


Figure A.4: Visualization of spectral descriptors learned by our method (with 2000 surreal shapes) on a test pair of SCAPE re-meshed. The source shape is shown in the first row, and the target shape in the bottom row. Notice how the descriptors are localized and seem to highlight one specific part of the body (first column for shoulder, second for scalp, third for right thigh, fourth for right side of the torso, fifth for elbow).

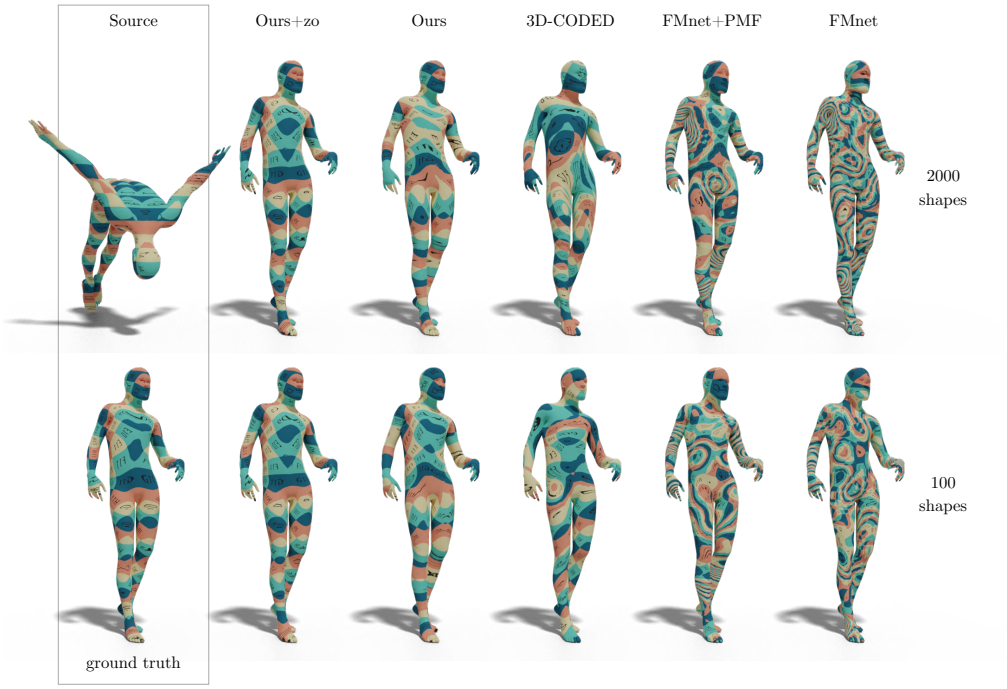


Figure A.5: Qualitative results on Experiment 2 through texture transfer, showing cases where our method is the only one that can give good correspondence with only 100 training shapes.

on Experiment 2 (also on SHREC'19 test shapes), which demonstrates our approach has

A.4.1 Visualization of some descriptors learned by our method

Our method aims at building descriptors on both input shapes (that are often labeled source and target shapes) from their raw point cloud data. These descriptors are then projected on the eigen basis of the respective Laplace-Beltrami operators of the source and the target shapes. We output these projections, that we call spectral descriptors, and we visualize some of them in Figure A.4.

It is remarkable that the descriptors learned on a parametric dataset such as the one used in 3D-CODED still generalize well to shapes with entirely different mesh connectivity and number of points. This is made possible by two components of our method. Firstly, it down-samples the input shapes through grid sampling before building these descriptor functions with convolutional neural networks. This allows for regularity in

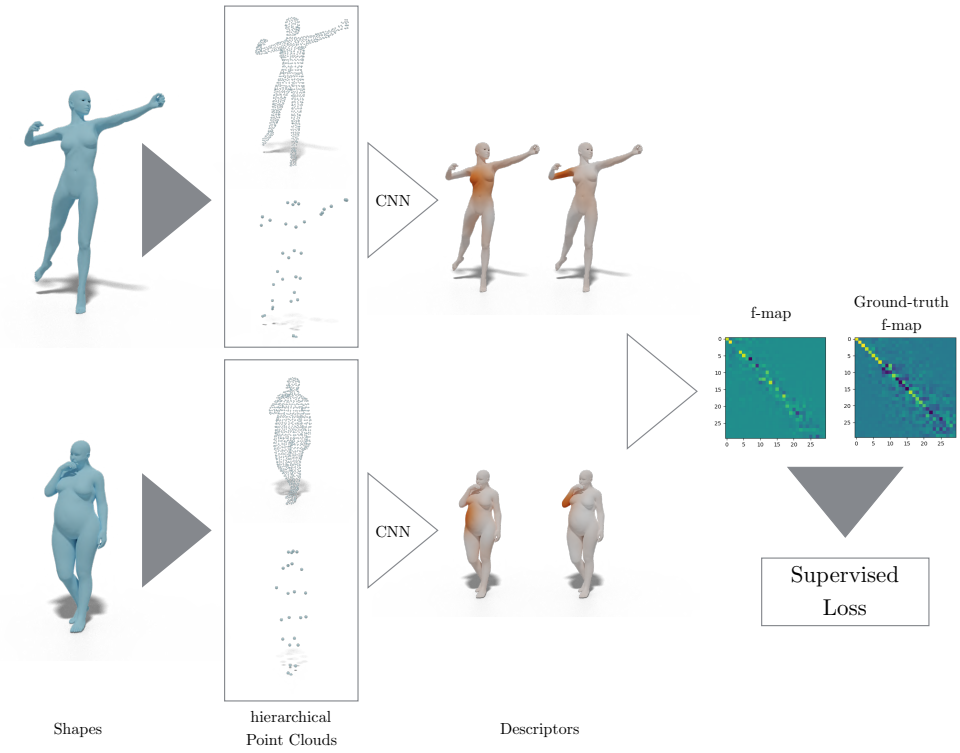


Figure A.6: Pipeline of our method: 1) Down-sample source and target shapes with grid sampling (providing the pooling at different scales). 2) Learning point cloud characterizations and project them in the Laplace-Beltrami eigen basis. 3) Compute the functional map from source and target spectral descriptors, with our regularized FMap layer. 4) Compute the loss by comparing the computed functional map with the ground truth map.

the input point clouds at all different hierarchies (see Figure A.6 for an example of such grid sampling). Secondly, the spectral projections take these point cloud descriptions to the shapes intrinsic space, adding some comprehensive surface-related information *without depending too much on the connectivity*, like with SHOT descriptors. Without this intrinsic translation, the network could have trouble differentiating two geometric components close in euclidean space, such as for instance the arms in mesh 25 of Figure A.4.

Additionally, these descriptors seem to capture some segmentation information, such as for instance head, arms, body and legs for humans, as can be observed in Figure A.4. More precise or complex descriptors such as hand or facial descriptors can not appear with only 30 eigen vectors. It is due to the fact that a spectral reconstruction of a human shape with only 30 eigenvectors does not show small details such as hands, feet

or facial features. One would need to push the number of eigenvectors above 100 to see such descriptors appear, and used correctly by the algorithm to produce even better correspondences. However, this could also more easily lead to overfitting.

A.4.2 Additional Texture transfer on SHREC'19 re-meshed

In Figure A.5 are shown additional qualitative results of our method (with and without Zoomout refinement [94]), 3D coded [55], FMNet [85] and Unsupervised FMNet [61] (with and without PMF refinement [163]), trained on respectively 2000 and 100 shapes, as presented in Experiment 2 of Chapter 4.5.

These results show again the failure of FMNet, due to the change in connectivity. It can be seen more thoroughly in the quantitative graphs provided in Figure A.3.

This pair of shapes in Figure A.5 represents a challenging case for both 3D-CODED and our method. Indeed, *these networks are not rotation invariant*, as discussed in the implementation section of Chapter 4.4.7. Here, the source shape is bent over and its head is really low compared with the rest of the body. 3D-CODED and our method are made robust to rotation around the vertical axis through data augmentation, but here the source shape is slightly rotated along another axis. As we can see, this resulted in poor reconstructions in the case of 3D-CODED algorithm, even with 2000 training shapes, whereas our method was able to yield good results with both a high and a low number of shapes.

A.4.3 General Pipeline

We also provide a visual illustration of our general pipeline in Figure A.6 to complement the textual description of our method provided in Chapter 4. In particular, we display: *a)* different hierarchical grid-poolings performed by the KPConv feature extractor on both source and target shapes, *b)* some learned descriptors on source and target shapes, *c)* the estimated functional map and the ground-truth functional map.

APPENDIX B

Complex Functional Maps: Supplementary Material

B.1 Proof of Theorem 5.3.1

Necessary condition: Let Q be the differential of an orientation-preserving conformal diffeomorphism φ. Then by Lemma 5.3.1 Q must be $\mathbb{C}$ linear and by virtue of being a differential, $Q = \mathrm{d}\varphi$ must satisfy Eq. (5.1) (see [80], Chapter 3).

Sufficient condition: Let Q be a $\mathbb{C}$ linear operator and suppose that there exists a diffeomorphism $\varphi : M \to N$ such that Q satisfies Eq. (5.1). Since the pushforward $\mathrm{d}\varphi : TM \to TN$ is the *unique* operator satisfying Eq. (5.1) (see [80], Chapter 3), we must have $Q = \mathrm{d}\varphi$. Now, since Q is a $\mathbb{C}$-linear pushforward, it preserves both the angles between the vectors and the orientation of the tangent bundle. Therefore, the map φ must then be an orientation-preserving conformal map.

B.2 Proof of Theorem 5.3.2

Let $\varphi : M \to N$ be a conformal diffeomorphism and $Q = \mathrm{d}\varphi : TM \to TN$ its corresponding pushforward. By definition of conformality there exists a log-scale factor $u : M \to \mathbb{R}$ relating the metric tensors of both surfaces $\varphi^\star \mathbf{g}^N = e^{2u}\mathbf{g}^M$ and their volume form $\varphi^\star \mathrm{d}\mu_N = e^{2u}\mathrm{d}\mu_M$. Thus, the L^2 scalar product between vector fields $X, Y \in TM$

is preserved by the pushforward:

$$\begin{aligned}\int_N \mathbf{g}_p^N\left(Q(X),Q(Y)\right)\mathrm{d}\mu_N(p) &= \int_N \mathbf{g}_p^N\left(\mathrm{d}\varphi(X),\mathrm{d}\varphi(Y)\right)\mathrm{d}\mu_N(p)\\ &= \int_M \mathbf{g}_{\varphi(q)}^N\left(\mathrm{d}\varphi(X),\mathrm{d}\varphi(Y)\right)e^{-2u}\mathrm{d}\mu_M(q)\\ &= \int_M (\varphi^\star\mathbf{g}^N)_q\left(X,Y\right)e^{-2u}\mathrm{d}\mu_M(q)\\ &= \int_M \mathbf{g}_q^M\left(X,Y\right)\mathrm{d}\mu_M(q)\end{aligned}$$

This this holds for arbitrary X, Y, we obtain that $Q^\star Q = I$.

B.3 Proof of Theorem 5.3.3

Let us assume that $L_M = Q^{-1} \circ L_N \circ Q$. As shown in [15, 143], the diffusion kernel of the connection Laplacian is, at first order, the parallel transport along a geodesic and its magnitude is identical to the decay of the scalar heat kernel. Since the pushforward preserves the connection Laplacian, it also preserves the scalar heat kernel, therefore it must be an isometry [151].

As proved in [25] (p.181) the pushforward of the Levi-Civita connection by a conformal mapping is itself the Levi-Civita connection if and only if the map is an isometry. Therefore, if φ is an isometry, the pushfoward of the connection Laplacian $Q^{-1} \circ L_N \circ Q$ is equal to the connection Laplacian on M.

B.4 Proof of Theorem 5.4.1

Let $\varphi : M \to N$ be the permutation associated to the matrix Π.

The energy of Eq. (5.8) for vertex-based vector field must be evaluated for X a basis of complex field and f a basis of functions. We choose the "hat" basis for the fields $X^i \in \mathbb{C}^{|V_M|}$ where $X^i_j = z\delta_{ij}, z = 1$ or $\imath$ and $f^i \in \mathbb{R}^{|V_M|}$ where $f^i_j = \delta_{ij}$. The symbol δ denotes the Kronecker delta. Explicitly writing the coefficients of the matrices, boils down to:

$$(CD^M_{X^i})_{kj} = \begin{vmatrix} \langle z, (\nabla f^j)_i\rangle, & k = \varphi(i) \\ 0, & k \neq \varphi(i)\end{vmatrix}$$

$$(D^N_{QX^i}C)_{kj} = \begin{vmatrix} \left\langle Q(X^i)_k, (\nabla f^{\varphi(j)})_k\right\rangle, & (k\varphi(j)) \in E_N \text{ or } k = \varphi(j) \\ 0, & (k\varphi(j)) \notin E_N \text{ and } k \neq \varphi(j)\end{vmatrix}$$

An immediate conclusion is that Q must be zero everywhere except at $Q_{\varphi(i),i}$ so there exists vector $q \in \mathbb{C}^{|V_M|}$ such that:

$$Q = D(q)\Pi.$$

Algorithm 2 Complex Bijective ZOOMOUT

Intput: Manifold meshes M and N
Initial pointwise maps Π_{MN} and Π_{NM}
Output: Refined maps Π_{MN}^{ref} and Π_{NM}^{ref}
Parameters: The number of refinement steps J
An array $[k_j], j \in [1, J]$ with the (increasing) number of spectral coordinates to use at each refinement step
Preprocessing: Compute the Laplace Beltrami eigenbases Φ^M and Φ^N (used for function in spectral basis)
Compute the connection Laplacian complex eigenbases Ψ^M and Ψ^N (used for vector fields in spectral basis, see Section 5.4.5)
Compute the differential operators $D_{\Phi_i^M}$ and $D_{\Phi_i^N}$ for $i \in [1, k_J]$ (used for estimating Q from C, see Section 5.4.2)
Compute the reduced divergence Operators div_M and div_N (used for conversion from Q to pointwise map, see Section 5.4.6)
for $k \in [k_1, ..., k_J]$ **do**
    $\Phi_M = \Phi^M_{[1,k]}, \Phi_N = \Phi^N_{[1,k]}$
    $C_{MN} = \Phi_N^\dagger \Pi_{NM} \Phi_M, \quad C_{NM} = \Phi_M^\dagger \Pi_{MN} \Phi_N$
    $Q_{MN} = \arg\min_{Q \in \mathcal{O}(k)} \sum_{i=1}^k \|C_{MN} D_{\Psi_i^M} - D_{Q\Psi_i^M} C_{MN}\|_F^2$
    $Q_{NM} = \arg\min_{Q \in \mathcal{O}(k)} \sum_{i=1}^k \|C_{NM} D_{\Psi_i^N} - D_{Q\Psi_i^N} C_{NM}\|_F^2$
    $\Pi_{MN} = \mathrm{NNsearch}(\mathrm{div}_N \Psi_N, \mathrm{div}_M \Psi_M Q_{NM})$
    $\Pi_{NM} = \mathrm{NNsearch}(\mathrm{div}_M \Psi_M, \mathrm{div}_N \Psi_N Q_{MN})$
    $C_{MN} = \begin{pmatrix} \Phi_N \\ \Pi_{MN}\Phi_N \end{pmatrix}^\dagger \begin{pmatrix} \Pi_{NM}\Phi_M \\ \Phi_M \end{pmatrix}$
    $C_{NM} = \begin{pmatrix} \Phi_M \\ \Pi_{NM}\Phi_M \end{pmatrix}^\dagger \begin{pmatrix} \Pi_{MN}\Phi_N \\ \Phi_N \end{pmatrix}$
    $\Pi_{MN} = \mathrm{NNsearch}(\begin{pmatrix} \Phi_N C_{NM} & \Phi_N C_{MN} \end{pmatrix}, \begin{pmatrix} \Phi_M & \Phi_M \end{pmatrix})$
    $\Pi_{NM} = \mathrm{NNsearch}(\begin{pmatrix} \Phi_M C_{MN} & \Phi_M C_{NM} \end{pmatrix}, \begin{pmatrix} \Phi_N & \Phi_N \end{pmatrix})$
end for

Now we can go back the least-squares problem and find each coefficient of q individually. Using the fact that z form a basis of $\mathbb{C}$, at a vertex $i \in M$ the best conformal deformation of the tangent plane $q_{\varphi(i)}$ is solution of:

$$q_{\varphi(i)} = \arg\min_{x \in \mathbb{C}} \sum_j \left| \left\langle x, (\nabla f^{\varphi(j)})_{\varphi(i)} \right\rangle - \left\langle 1, (\nabla f^j)_i \right\rangle \right|^2 .$$

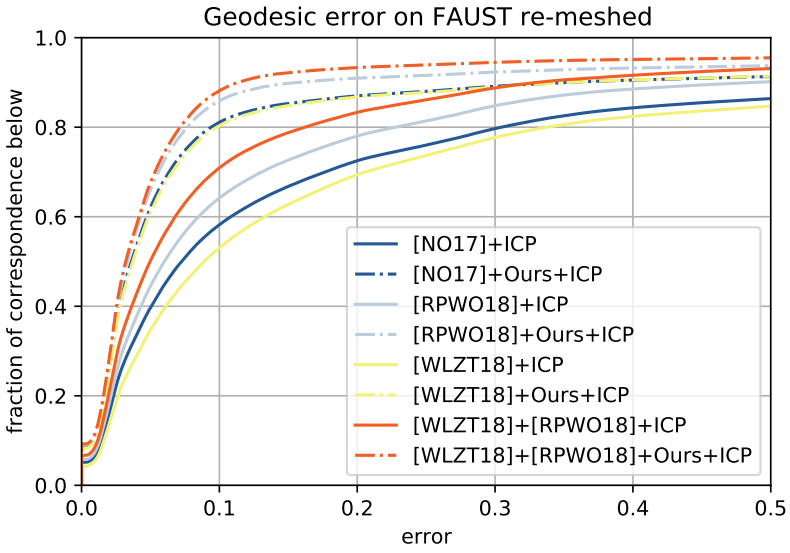


Figure B.1: Geodesic error of different methods with and without our Q-step, then refined with ICP, on 190 shape pairs of FAUST re-meshed.

B.5 Proof of Theorem 5.4.2

- Suppose that Q is an isometric pushforward then the commutativity with the Laplacian immediately holds as the connection Laplacian, like the usual cotangent Laplacian, is preserved by isometric changes.
- Assume that Q commutes with the Laplacian then thanks to Thm. 5.4.1, the cotan-weights are preserved under the mapping. Therefore, the deformation is an isometry [57].

B.6 Solving the least-squares problem to estimate Q

Here we propose to complete Section 5.4.2 by giving one potential way to solve Eq. (5.8) and Eq. (5.9) explicitly. To that end we first proceed to rewrite Eq. (5.6), by switching from functional to vector field operator. More precisely, for a shape M, and $f \in L^2(M), X \in TM$, we define the linear operator $D_f \in TM \to L^2(M)$:

$$D_f(X)_p = \langle X, \nabla f \rangle_{T_pM}$$

With this operator we adopt the dual point of view from D_X. Indeed $\forall f \in L^2(M), X \in TM, D_f(X) = D_X(f)$. Consequently, the discretization is almost identical, except that

Algorithm 3 Complex Discrete Optimization

Intput: Manifold meshes M and N
Initial pointwise maps Π_{MN}
Output: Refined maps Π_{MN}^{ref}
Parameters: The number of refinement steps J
An array $[k_j], j \in [1, J]$ with the (increasing) number of spectral coordinates to use at each refinement step
Preprocessing: Compute the Laplace Beltrami eigenbases Φ^M and Φ^N (used for function in spectral basis)
7: Compute the connection Laplacian complex eigenbases Ψ^M and Ψ^N (used for vector fields in spectral basis, see Section 5.4.5)
8: Compute the differential operators $D_{\Phi_i^M}$ and $D_{\Phi_i^N}$ for $i \in [1, k_J]$ (used for estimating Q from C, see Section 5.4.2)
9: Compute the reduced divergence Operators div_M and div_N (used for conversion from Q to pointwise map, see Section 5.4.6)
for $k \in [k_1, ..., k_J]$ **do**
    $\Phi_M = \Phi^M_{[1,k]}, \Phi_N = \Phi^N_{[1,k]}$
12: $Q_{NM} = \arg\min_{Q \in \mathcal{O}(k)} \sum_{i=1}^{k} \|C_{NM} D_{\Psi_i^N} - D_{Q\Psi_i^N} C_{NM}\|_F^2$
13: $\Pi_{MN} = \mathrm{NNsearch}(\mathrm{div}_N \Psi_N, \mathrm{div}_M \Psi_M Q_{NM})$
    $C_{NM} = \Phi_M^\dagger \Pi_{MN} \Phi_N$
    $\Pi_{MN} = \mathrm{NNsearch}(\mathrm{div}_N \Psi_N, \mathrm{div}_M \Psi_M Q_{NM})$
end for

since this operator takes vector fields as input, we choose to encode it as a complex operator $\mathbf{D}_f \in \mathbb{C}^{|V|\times|V|}$. Namely, $\mathbf{D}_f$ is a diagonal matrix such that $(\mathbf{D}_f)_{ii} = \overline{\nabla f_i}$. One can then retrieve $D_f(X)$ by taking the real part $Re(\mathbf{D}_f X)$. It can also be noted that $Im(\mathbf{D}_f X) = Re(\mathbf{D}_f \cdot \imath X) = D_f \cdot \imath X$, so that the complex matrix $\mathbf{D}_f$ also stores information for rotated gradients.

Let M and N be two manifolds. Switching from D_X to D_f in Eq. (5.6), we get Eq. (B.1):

$$C_{MN} D_f^M = D^N_{C_{MN} f} Q_{MN}, \quad \forall\, f \in \mathbb{R}^{|V_M|}. \tag{B.1}$$

We then discuss how to minimize, for a fixed input functional map C and a family of smooth functions f_i the following energy. Similarly to Eq. (5.7), f_i is chosen to be the truncated eigenbasis of the Laplace-Beltrami operator.

$$E_q(Q) = \sum_i \|C D_{f_i}^M - D^N_{Cf_i} Q\|_F^2, \tag{B.2}$$

Regular problem In this first paragraph we ignore the constraint $QQ^\star = I$, which

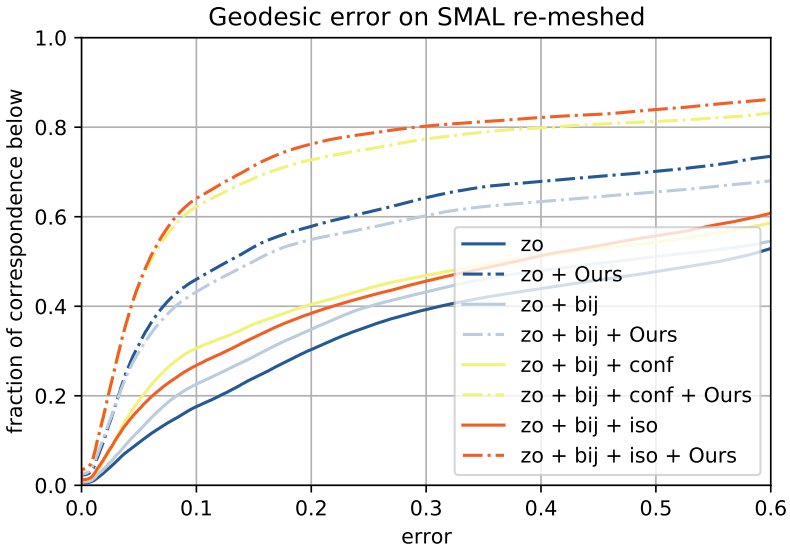


Figure B.2: Geodesic error of different versions of ZOOMOUT with and without our Q-step, on 50 shapes pairs of SMAL re-meshed.

corresponds to Eq. (5.8).

Consider an input functional map C, and reference functions $f_i, i \in [1, k_f]$. The minimum to Eq. (B.2) can be written explicitly: by concatenating $CD_{f_i}, i \in [1, k_f]$ in a big matrix $A \in \mathbb{C}^{(|V_N| \times k_f) \times |V_M|}$, and $D_{Cf_i}, i \in [1, k_f]$ in another matrix $B \in \mathbb{C}^{(|V_N| \times k_f) \times |V_N|}$, we can re-write Eq. (B.1) as $Q = B^\dagger A$, where $B^\dagger$ is the Moore pseudo-inverse of B. In practice, we express these operators in a reduced spectral basis (See Section 5.4.5) before computing the pseudo-inverse, to improve computation time. Indeed in the reduced basis, operators A and B are respectively of size $k_N k_f \times k_M$ and $k_N k_f \times k_N$ if we choose to truncate the tangent vector field eigenbasis at k_M on M and at k_N on N. This results in a $k_N \times k_M$ reduced complex operator for Q.

Procrustes problem Considering the same setting and keeping the last notations as in last paragraph, minimizing Eq. (B.2) with the constraint $QQ^\star = I$, which corresponds to Eq. (5.9), is a Procrustes problem. As such, it boils down to a Singular Value Decomposition: writing $\Omega = B^\star A$, we use its SVD $\Omega = U\Sigma V$ to retrieve the orthogonal matrix $Q = UV$ minimizing Eq. (5.9). As stated in the previous paragraph, one benefits from writing these equations in the reduced basis.

B.7 Vector Field transfer with more spectral values

In this appendix, we complete the vector field transfer experiment of Section 5.5.1 by showing the evolution of vector field transfer accuracy with the number of spectral coordinates involved in both the Laplace-Beltrami and the connection Laplacian operator (the latter is only used in Azencot *et al.* and our method). Table B.1 reports this comparison in detail. Let us recall that only a number below 1 shows a result with reasonable accuracy, above this threshold the error is higher that the norm of the input vector field.

We observe that the transfer from Wang *et al.* is always very sensitive to noise, leading to high inaccuracies when noise corrupts the high frequencies. Notice also that the approach of Azencot *et al.* only starts to give reasonable transfer with high spectral values, namely $k = 150$. In comparison, our method almost doesn't suffer from noise, whether it is random or symmetric, and is accurate even for low spectral values.

Our method is the only one to recover from strong symmetric noise ($a \geq 0.5$). Indeed the vector field we transfer for this second type of noise is antisymmetric as described in Section 5.5.1, but the input (blurred) functional map projects antisymmetric functions to 0. This results in baselines transferring the input vector field to a vector field close to 0 on the target shape, and thus errors above or close to 1.

B.8 Geodesic distance curves for Table 5.2 and 5.3

In Figure B.1 and B.2 we respectively display the results of Table 5.2 and 5.3 with more precision using the Princeton graphs first introduced in [73]. With these curves it is easier to assert the finer quality of the correspondence computed with the help of complex functional maps.

B.9 Complex bijective ZoomOut algorithms

In this section we describe precisely how to implement the different versions of ZoomOut to which we add our Q-step. These algorithms are used to generate the correspondence whose quality are reported in Figure 5.7 and Table 5.3.

Complex bijective ZoomOut We first display the bijective ZoomOut algorithm used in [126], modified to include our Q-step. This results in Algorithm 2, where we bold the line numbers where change occurs. The idea behind bijective ZoomOut is to optimize for maps in both directions (from M to N *and* from N to M). The energy to minimize is the so-called bijective energy $E_{bij} = \|C_{MN}C_{NM} - I\|_F^2 + \|C_{NM}C_{MN} - I\|_F^2$, and [126] proposes to optimize it with the following steps : Remarking that $C_{NM} = \Phi_M^\dagger \Pi_{MN} \Phi_N$, we have $\|\Phi_M\|_F^2 \|C_{NM}C_{MN} - I\|_F^2 = \|\Pi_{MN}\Phi_N C_{MN} - \Phi_M\|_F^2$. Thus, knowing Π_{MN}

Random noise			
Method / level of noise	$s = 0$	$s = 0.2$	$s = 0.5$
$k = 30$			
Wang *et al.* [167]	**0.42**	5.3	14
Azencot *et al.* [8]	11	12	12
Ours	0.54	**0.57**	**0.80**
$k = 70$			
Wang *et al.* [167]	**0.41**	14	33
Azencot *et al.* [8]	2.6	2.4	2.4
Ours	0.46	**0.48**	**0.77**
$k = 150$			
Wang *et al.* [167]	**0.37**	27	68
Azencot *et al.* [8]	0.68	0.81	1.0
Ours	0.44	**0.47**	**0.81**
Symmetric noise			
Method / level of noise	$a = 0.3$	$a = 0.5$	$a = 0.6$
$k = 30$			
Wang *et al.* [167]	0.87	1.13	1.29
Azencot *et al.* [8]	10.19	6.84	10.59
Ours	**0.54**	**0.62**	**0.76**
$k = 70$			
Wang *et al.* [167]	0.81	1.09	1.26
Azencot *et al.* [8]	2.93	1.99	2.55
Ours	**0.39**	**0.47**	**0.81**
$k = 150$			
Wang *et al.* [167]	0.77	1.05	1.22
Azencot *et al.* [8]	0.69	1.04	1.43
Ours	**0.37**	**0.41**	**0.56**

Table B.1: Average accuracy of the three vector field transfer algorithms of Section 5.5.1 on 20 random pairs of FAUST [17] for two types of noise, *and three noise levels*. We use $k = 30, 70, 150$ eigenvectors for both real and complex Laplacian operators.

and Π_{NM}, one can optimize for C_{MN} in a bijective fashion with:

$$C_{MN} = \arg\min_{C} \|\Phi_N C - \Pi_{NM}\Phi_M\|_F^2 + \|\Pi_{MN}\Phi_N C - \Phi_M\|_F^2$$

And the same operation can be performed for C_{NM}.

Afterwards, one can perform a symmetric trick to get Π_{MN} knowing C_{MN} and C_{NM},

by additionally noting that in a bijective orthogonal setting, $C^{-1} = C^T$:

$$\Pi_{MN} = \arg\min_{\Pi} \|\Pi\Phi_N C_{NM}^T - \Phi_M\|_F^2 + \|\Pi\Phi_N C_{MN} - \Phi_M\|_F^2$$

And the same operation can be performed for Π_{NM}.

In the modified version of this algorithm, the pointwise map directly comes from our complex functional maps, and thus is orientation-aware. Its translation to functional map will also carry that information, making Algorithm 2 more robust than its original version to symmetry errors.

Complex Discrete Optimization Secondly, we modify discrete optimization algorithms [127] (again, the necessary modifications bear bold line numbers). These algorithms consist in reducing a continuous energy with the same kind of trick used in bijective ZOOMOUT. It is thus adaptable to all kind of energies. Besides, this method proved more efficient than continuous solvers to reduce natural energies such as conformality or isometry. One simply defines either $E(\Pi_{MN})$ (respectively $E(\Pi_{MN}, \Pi_{NM})$ in the case of a bijective energy), rewrites a continuous energy using the pointwise map trick $C_{MN} = \Phi_N^\dagger \Pi_{NM} \Phi_M$, and minimizes it for Π_{MN} given $\mathbb{C}_{NM}$. Adding in our Q-step, we get Algorithm 3. This algorithm is both orientation-aware *and* optimizes for desirable maps such as isometries, often resulting in the best map as shown in Table 5.3 and Figure 5.7.

APPENDIX C

Deep Complex Functional Maps: Supplementary Material

C.1 Proof of Theorem 6.3.1, 6.3.2, 6.4.1

Theorem 1. *Given a set of shapes $\{S_i\}$ that all contain an* orientation reversing *isometric self-symmetry $\{T_i : S_i \to S_i\}$, s.t. $d_{S_i}(x_j, x_k) = d_{S_i}(T_i(x_j), T_i(x_k))$, then a generic neural network $\mathcal{F}_\Theta$ that is trained by any of the losses introduced in [132, 62, 52, 140, 7] has at least two possible solutions that both lead to the global optimum of the loss.*

Proof. The spectral losses L defined in [132, 62, 52, 140, 7] are fully intrinsic, thus they are invariant under shape isometric changes *i.e.* $L \circ T_i = L$. So, if all shapes admit an isometric self-symmetry, the solution composed with the isometry will have the same loss value. □

Theorem 2. *The complex-linear map Q is a pushforward if and only if there exists an* orientation-preserving *and conformal diffeomorphism $\varphi : M \to N$ satisfying:*

$$\langle X, \nabla(f \circ \varphi)\rangle_{T_pM} = \langle Q(X), \nabla f\rangle_{T_{\varphi(p)}N}, \tag{C.1}$$

for all $X \in TM, f \in L^2(N), p \in M$.

Proof. See Theorem 3.1 in chapter 5.3.5. □

Theorem 3. *Let M, N be two manifolds, and F_M, F_N surface features such that the functional map C estimated from these features is an isometry. Let Q be the complex functional map computed with the feature gradients as described in the main manuscript. Then the maps (C, Q) satisfy Eq. (C.1), and C is an orientation-preserving isometry.*

Proof. By assumption the functional map $C : L^2(N) \to L^2(M)$ represents the isometric map $\varphi : M \to N$ and exactly transfers the descriptors i.e. $C(F_N) = F_M$. Moreover the complex functional map $Q : TM \to TN$ transfers the gradient of the descriptors $Q(\nabla_M F_M) = \nabla_N F_N$ and is complex-linear.

To recover Eq. (C.1), we take the inner product of the gradient transfer with the gradient of an arbitrary function $f : N \to \mathbb{R}$:

$$\mathbf{g}_p^N \left(Q(\nabla_M F_M), \nabla_N f\right) = \mathbf{g}_p^N \left(\nabla_N F_N, \nabla_N f\right).$$

This equation easily simplifies using the properties of an isometric map: the metric is preserved by the pullback ($\varphi^\star \mathbf{g}^N = \mathbf{g}^M$) and the pushforward commutes with the gradient ($\mathrm{d}\varphi^{-1}\left(\nabla_N f\right) = \nabla_M C(f)$), yielding:

$$\begin{aligned}
&\mathbf{g}_p^N \left(Q(\nabla_M F_M), \nabla_N f\right) \\
&= \left((\varphi^\star)^{-1} \mathbf{g}^M\right)_{\varphi^{-1}(p)} \left(\mathrm{d}\varphi^{-1}\left(\nabla_N F_N\right), \mathrm{d}\varphi^{-1}\left(\nabla_N f\right)\right) \\
&= \mathbf{g}^M_{\varphi^{-1}(p)} \left(\nabla_M C(F_N), \nabla_M C(f)\right)
\end{aligned}$$

So Q and C satisfy Eq. (C.1) for all complex-linear combination of the gradient descriptors. Therefore, following Thm. 6.3.2, Q is the pushforward associated to φ and C must be orientation preserving. □

C.2 Ablation Study

This section presents an ablation study for the most vital components of our approach, namely: *a)* The input signal fed to the network, *b)* The orientation-aware feature extractor, *c)* The orientation-aware loss. We test these ablations on our third experiment, on the SMAL dataset [175] (see Chapter 6.5 for more details), and to a maximum of 20 epochs. We compare these three ablations to our approach and report the results in Table C.1. The ablations are commented in details in the sections below.

C.2.1 The WKS Descriptors as Input Features

As stated in Chapter 6, many unsupervised deep learning for non-rigid 3D shape matching rely on SHOT descriptors [158] as input signal for the neural network to produce correspondences between shapes [62, 132, 45]. This descriptor is orientation-aware but

Meth / Data	SMAL_r
xyz input-3 axis	25.
xyz input-1 axis	5.9
nonOA-FE	34.
no Q-maps (epoch 3)	5.8
no Q-maps (epoch 15)	8.1
Ours (epoch 3)	**4.8**
Ours (epoch 15)	5.1

Table C.1: Comparative results ($\times 100$) for the different ablations of our method.

very sensitive to the input triangulation, resulting in overfitting to the training triangulation as demonstrated in chapter 4, and also in the first experiment of 6, with anisotropic remeshings.

Therefore we use an input descriptor that is agnostic to the input triangulation so as to not overfit to it: the WKS descriptor [6], which is built using the eigenvectors and eigenvalues of the Laplace-Beltrami operator. These descriptor functions $(h_k)_{k\in[1,d]}$ are therefore fully intrinsic, and will display the same intrinsic self-symmetries as the shapes themselves. Namely, with the notations of Theorem 6.3.1, $h_k \circ T = h_k$.

Another commonly used option for an input signal is the 3-dimensional extrinsic coordinates of the shape points, as done in chapter 4. However, this input signal is dependent on the shape position in space. With this input signal, the method is no longer fully intrinsic and therefore potentially unstable to rotations of the input shapes. Consequently, the input data needs to be centered, and augmented by adding randomly rotated poses.

For this first ablation experiment, we train our method with this input signal (denoted as "xyz input" in Table C.1) instead of WKS descriptors. To make the experiment more complete, we report the results with two different data augmentations: *a)* The general case, where there is no prior on the shapes' alignment, so the data need to be augmented with all 3D rotations (3 parameters space). We denote this data augmentation as "3 axis" in Table C.1. *b)* The special case where the input shapes are all aligned to one axis, but potentially rotated around this axis, so the data needs to be augmented around this axis (1 parameter space). We denote this data augmentation as "1 axis" in Table C.1. We stress the fact that this kind of prior on the shapes rigid alignment already makes the method weakly supervised.

We see in Table C.1 that even with the prior of shapes aligned to one axis, our method is better (and more general) when trained with WKS descriptors as an input signal.

Up axis
Anisotropy

back-front axis
anisotropy

Left-right axis
anisotropy

Figure C.1: SCAPE [3] dataset remeshed in an anisotropic fashion, used in the first experiment of Chapter 6.5.

C.2.2 The Orientation-aware Feature Extractor

To make our approach unsupervised, it is crucial that the feature extractor should be orientation-aware. Indeed, since we train on shapes exhibiting an isometric self symmetry (the left-right symmetry present in most organic shapes), the only way to disambiguate between left and right is through orientation, since the symmetric map reverses this orientation. DiffusionNet [144] uses gradient features to incorporate this orientation information into potentially symmetric inputs (e.g. WKS descriptors in our case). For this second ablation, we propose to show that without this orientation-aware feature extractor, the method fails to produce informative descriptors, and report the results in Table C.1, on row "nonOA-FE" (standing for non orientation-aware feature extractor).

To that end, we deactivate the gradient-based blocks of DiffusionNet, which results in a new orientation-agnostic feature extractor which can still produce excellent results [144]. We then train our method using this feature extractor and WKS as input signal. We see in Table C.1 that this ablation greatly impairs the method.

C.2.3 The Complex Functional Maps Block and the Orientation-aware Loss

We remove the complex functional map block from the loss by setting $w_{\text{Q-ortho}} = 0$. As discussed in Theorem 6.3.1, the resulting network is not guaranteed or encouraged to produce orientation-preserving correspondence. We report the result of this ablation in Table C.1, on rows "no Q-maps".

We observe that this ablation still seems to converge to well oriented maps in this

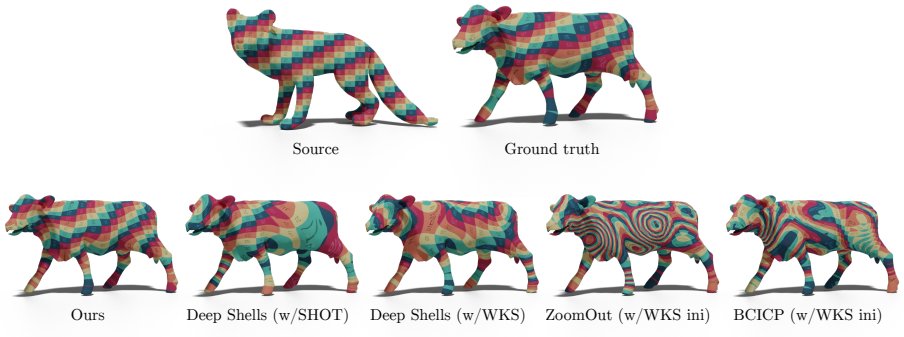


Figure C.2: Qualitative comparison to baselines on the SMAL dataset, using texture transfer from source to target shape. Only our method gives accurate correspondence, whereas in this challenging case baselines completely fail to predict the map.

case. This may be explained by the fact that DiffusionNet can produce non-symmetric descriptors from symmetric inputs like WKS, using shape orientation through gradients. Therefore, if two input shapes are consistently oriented, the symmetric input signal will be "taken in the same direction" by DiffusionNet gradient-based blocks. Conversely, if two shapes are non-consistently oriented (e.g. one with inward normals, one with outwards normals), the symmetric input will be "taken in opposite directions". In fact, using this remark one can retrieve symmetrized output descriptor functions (by symmetrized, here we mean composited with the intrinsic symmetric map T_i of the shape S_i) generated by DiffusionNet from symmetric descriptors such as WKS, by simulating a change in shape orientation (which corresponds to a conjugation operation on the tangent bundle structure, or more practically to setting `gradY = -gradY` in DiffusionNet gradient operator entries).

In practice, a second beneficial effect of our complex functional map loss is the reduction of overfitting. Indeed, in the experiment reported in Table 6.3 of the main manuscript, the train set is made of 32 SMAL re-meshed shapes and the test set is made of 17 shapes other SMAL re-meshed shapes. Learning methods are thus liable to overfit to their training set. We see in Table C.1 (where we report the geodesic error at epoch 3 and epoch 15 both with and without the complex functional map layer/loss) that without the complex functional map loss, the method is more prone to overfitting, as it looses generality if trained for too many epochs. To summarize, our complex functional map block and loss theoretically guarantee our approach to be orientation-preserving, and in practice also improve the pipeline stability with respect to overfitting.

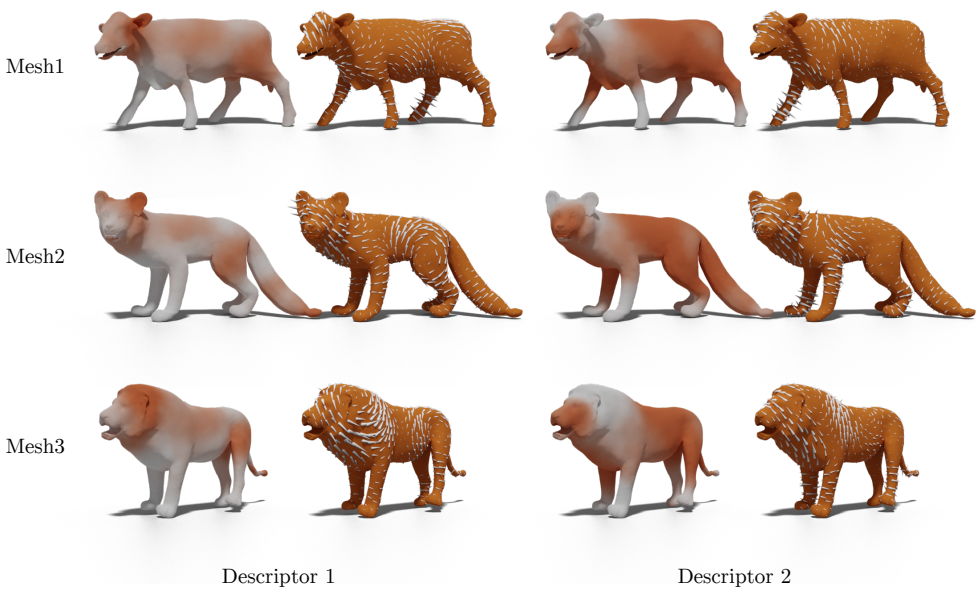


Figure C.3: Visualization of two different scalar descriptors learned by our method, along with their vector valued counterparts, on 3 meshes from the SMAL dataset. Contrary to the descriptors produced by the network of chapter 4, these descriptors are fully intrinsic and generally not localized. However, we see that both our scalar and vector valued descriptors are robust to strong distortions.

C.3 More Quantitative Results

For completeness, we report in Figure C.4 the accuracy of our method and some baselines on the third experiment we conducted in Chapter 6.5 (trained on 32 of SMAL remeshed shapes, tested on 17 other SMAL remeshed shapes), using the evaluation protocol introduced in [72]. We see that our method gives the best correspondence quality by far, as in this case it always predicts well-oriented maps for the test pairs (we see the tail of the error curve quickly reaches the $y = 1$ line, which is equivalent to saying most predicted correspondences are extremely close to ground-truth).

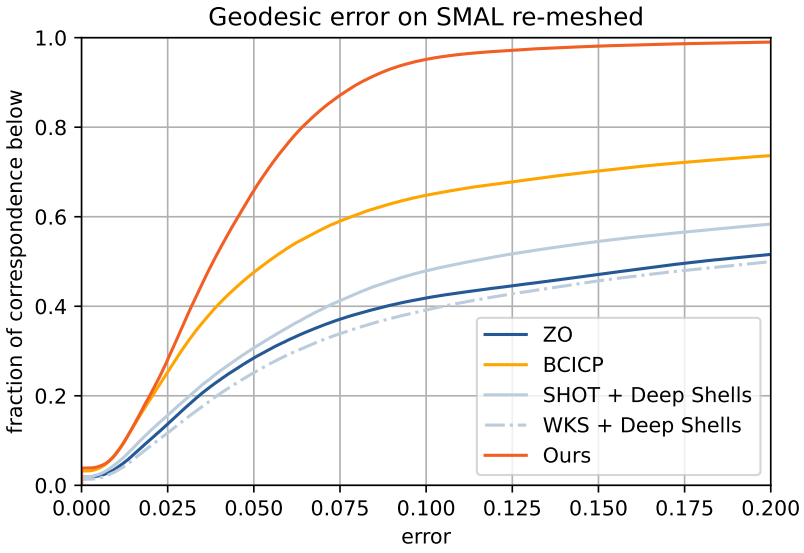


Figure C.4: Quantitative results of the different methods using the protocol introduced in [72], on the SMAL remeshed test set (third experiment of Chapter 6.5).

C.4 More Qualitative Results

C.4.1 Anisotropic Remeshing

In Figure C.1, we show the anisotropic remeshing of SCAPE dataset [3], generated with Mmg [38, 34]. We use this anisotropic remeshing in the first experiment of Chapter 6.5 to show that SHOT [158] based learning methods do not generalize to unseen triangulation. Specifically, we see in Figure C.1 that the triangle scale is a function of the element coordinate. For the first seven shapes of SCAPE test set, we constrain the triangle size to be dependent on the position on the up axis. For the next seven shapes, we constrain the triangle size to be dependent on the position on the back-front axis. For the remaining six shapes, we constrain the triangle size to be dependent on the position on the left-right axis. With this remeshing, a network overfitting to the triangulation combinatorics will most likely fail to predict the desired map. Our method, which is triangulation agnostic, remains almost unaltered, as shown in the first experiment Chapter 6.5.

C.4.2 Another Qualitative Comparison on SMAL

We report in Figure C.2 a second texture transfer performed by baselines and our method on two of the SMAL test shapes. On this example the distortion between the two shapes is even stronger than on the example displayed in Chapter 6.5. However, our method

still manages to predict accurate correspondences, while baselines fail to produce even a reasonable mapping in this case. Despite the fact that our method is spectral based, we see it can still produce accurate maps in challenging non-isometric cases.

C.4.3 Visualization of the Scalar & Vector Valued Descriptors Learned by our Method

Lastly, we propose to visualize some descriptors learned by our network, also on the SMAL dataset, in Figure C.3. Since our method also exploits the gradients of the scalar descriptors learned by DiffusionNet, we visualize these gradients (here rotated by $\pi/2$ to better make singularities stand out) which in fact correspond to the tangent vector field descriptors used to compute the complex functional map. Our method enforces learned descriptors *and their gradients* to correspond between source and target shapes, which was not done in any previous work to the best of our knowledge. Consequently, the features obtained with our method are all the more robust, since their gradients are also well-preserved under shape non-rigid deformation.

Indeed we notice that both the scalar-valued and the vector-valued part of the two descriptors displayed in Figure C.3 correspond well on the three chosen meshes, despite the strong distortions involved between these shapes.

Bibliography

[1] M. Abadi, A. Agarwal, and P. B. et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL https://www.tensorflow.org/. Software available from tensorflow.org. (Cited on page 55).

[2] Y. Aflalo, A. Dubrovina, and R. Kimmel. Spectral generalized multi-dimensional scaling. *International Journal of Computer Vision*, 118(3):380–392, 2016. (Cited on pages 48 and 66).

[3] D. Anguelov, P. Srinivasan, D. Koller, S. Thrun, J. Rodgers, and J. Davis. SCAPE: Shape completion and animation of people. *ACM Transactions on Graphics*, 24(3):408–416, July 2005. ISSN 0730-0301. (Cited on pages 57, 103, 137, and 140).

[4] S. Attaiki, G. Pai, and M. Ovsjanikov. Dpfm: Deep partial functional maps. *International Conference on 3D Vision (3DV)*, 2021. (Cited on pages 5 and 16).

[5] M. Atzmon, H. Maron, and Y. Lipman. Point convolutional neural networks by extension operators. *ACM Transactions on Graphics (TOG)*, 37(4):1–12, 2018. (Cited on pages 7, 18, 49, 51, 52, and 112).

[6] 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*, pages 1626–1633. IEEE, 2011. (Cited on pages 6, 7, 17, 18, 41, 48, 50, 67, 84, 85, 94, 96, 99, 102, 105, 110, and 136).

[7] M. Aygün, Z. Lähner, and D. Cremers. Unsupervised dense shape correspondence using heat kernels. In *2020 International Conference on 3D Vision (3DV)*, pages 573–582. IEEE, 2020. (Cited on pages 94, 97, and 134).

[8] O. Azencot, M. Ben-Chen, F. Chazal, and M. Ovsjanikov. An operator approach to tangent vector field processing. In *Proceedings of the Eleventh Eurographics/ACMSIGGRAPH Symposium on Geometry Processing*, pages 73–82. Eurographics Association, 2013. (Cited on pages 64, 66, 73, 75, 80, 81, 82, 84, and 131).

[9] O. Azencot, M. Ovsjanikov, F. Chazal, and M. Ben-Chen. Discrete derivatives of vector fields on surfaces–an operator approach. *ACM Transactions on Graphics (TOG)*, 34(3):1–13, 2015. (Cited on page 66).

[10] O. Azencot, O. Vantzos, and M. Ben-Chen. Advection-based function matching on surfaces. In *Computer Graphics Forum*, volume 35-5, pages 55–64. Wiley Online Library, 2016. (Cited on page 66).

[11] O. Azencot, E. Corman, M. Ben-Chen, and M. Ovsjanikov. Consistent functional cross field design for mesh quadrangulation. *ACM Trans. Graph.*, 36(4), July 2017. ISSN 0730-0301. doi: 10.1145/3072959.3073696. URL https://doi.org/10.1145/3072959.3073696. (Cited on page 66).

[12] X. Bai, Z. Luo, L. Zhou, H. Chen, L. Li, Z. Hu, H. Fu, and C.-L. Tai. Pointdsc: Robust point cloud registration using deep spatial consistency. *CVPR*, 2021. (Cited on pages 4 and 15).

[13] I. Baran, D. Vlasic, E. Grinspun, and J. Popović. Semantic deformation transfer. In *ACM SIGGRAPH 2009 papers*, pages 1–6. ACM, 2009. (Cited on page 91).

[14] M. Ben-Chen and O. Azencot. Operator-based representations of discrete tangent vector fields. In *Handbook of Numerical Analysis*, volume 20, pages 117–147. Elsevier, 2019. (Cited on page 66).

[15] N. Berline, E. Getzler, and M. Vergne. *Heat kernels and Dirac operators*. Springer Science & Business Media, 2003. (Cited on page 125).

[16] S. Biasotti, A. Cerri, A. Bronstein, and M. Bronstein. Recent trends, applications, and perspectives in 3d shape similarity assessment. In *Computer Graphics Forum*, volume 35-6, pages 87–119, 2016. (Cited on pages 48, 66, and 93).

[17] F. Bogo, J. Romero, M. Loper, and M. J. Black. FAUST: Dataset and evaluation for 3D mesh registration. In *Proc. CVPR*, pages 3794–3801, Columbus, Ohio, 2014. IEEE. (Cited on pages 80, 82, 86, and 131).

[18] F. Bogo, J. Romero, M. Loper, and M. J. Black. FAUST: Dataset and evaluation for 3D mesh registration. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, Piscataway, NJ, USA, June 2014. IEEE. (Cited on pages 46 and 57).

[19] 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*, pages 3794–3801. ACM, 2014. (Cited on pages 91 and 103).

[20] D. Boscaini, J. Masci, E. Rodolà, and M. Bronstein. Learning shape correspondence with anisotropic convolutional neural networks. In *Advances in neural information processing systems*, pages 3189–3197, 2016. (Cited on page 94).

[21] D. Boscaini, J. Masci, E. Rodola, and M. M. Bronstein. Learning shape correspondence with anisotropic convolutional neural networks. In *Proc. NIPS*, pages 3189–3197, 2016. (Cited on pages 7, 18, and 49).

[22] M. Botsch, L. Kobbelt, M. Pauly, P. Alliez, and B. Lévy. *Polygon mesh processing*. CRC press, 2010. (Cited on pages 32, 33, 37, 38, and 74).

[23] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017. (Cited on pages 5 and 16).

[24] O. Burghard, A. Dieckmann, and R. Klein. Embedding shapes with Green's functions for global shape matching. *Computers & Graphics*, 68:1–10, 2017. (Cited on pages 48, 66, and 93).

[25] M. P. d. Carmo. *Riemannian geometry*. Birkhäuser, 1992. (Cited on pages 72 and 125).

[26] E. R. Chan, M. Monteiro, P. Kellnhofer, J. Wu, and G. Wetzstein. pi-gan: Periodic implicit generative adversarial networks for 3d-aware image synthesis. *CVPR*, 2021. (Cited on page 112).

[27] E. R. Chan, C. Z. Lin, M. A. Chan, K. Nagano, B. Pan, S. D. Mello, O. Gallo, L. Guibas, J. Tremblay, S. Khamis, T. Karras, and G. Wetzstein. Eg3d: Efficient geometry-aware 3d generative adversarial networks. *CVPR*, 2022. (Cited on page 112).

[28] H. Chen, W. C. Shichen Liu and, H. Li, and R. Hill. Equivariant point network for 3d point cloud analysis. *CVPR*, 2021. (Cited on page 110).

[29] M. Chen, C. Wang, and H. Qin. Jointly learning shape descriptors and their correspondence via deep triplet cnns. *Computer Aided Geometric Design*, 62: 192–205, 2018. (Cited on page 49).

[30] E. Corman and M. Ovsjanikov. Functional characterization of deformation fields. *ACM Transactions on Graphics (TOG)*, 38(1):1–19, 2019. (Cited on page 66).

[31] E. Corman, M. Ovsjanikov, and A. Chambolle. Supervised descriptor learning for non-rigid shape matching. In *Proc. ECCV Workshops (NORDIA)*, 2014. (Cited on pages 7, 18, 44, 46, 47, 48, 50, 55, 92, 94, and 96).

[32] E. Corman, M. Ovsjanikov, and A. Chambolle. Continuous matching via vector field flow. *Computer Graphics Forum*, 34(5):129–139, 2015. (Cited on pages 7, 18, 66, and 84).

[33] E. Corman, J. Solomon, M. Ben-Chen, L. Guibas, and M. Ovsjanikov. Functional characterization of intrinsic and extrinsic geometry. *ACM Transactions on Graphics (TOG)*, 36(2):1–17, 2017. (Cited on pages 64 and 66).

[34] C. Dapogny, C. Dobrzynski, and P. Frey. Three-dimensional adaptive domain remeshing, implicit domain meshing, and applications to free and moving boundary problems. *Journal of computational physics*, 262:358–378, 2014. (Cited on pages 103 and 140).

[35] C. Deng, O. Litany, Y. Duan, A. Poulenard, A. Tagliasacchi, and L. Guibas. Vector neurons: A general framework for so(3)-equivariant networks. *CVPR*, 2021. (Cited on page 110).

[36] T. Deprelle, T. Groueix, M. Fisher, V. G. Kim, B. C. Russell, and M. Aubry. Learning elementary structures for 3d shape generation and matching. *arXiv preprint arXiv:1908.04725*, 2019. (Cited on pages 49 and 51).

[37] P. Dhariwal and A. Nichol. Diffusion models beat gans on image synthesis. *arXiv:2105.05233*, 2021. (Cited on pages 2 and 12).

[38] C. Dobrzynski and P. Frey. Anisotropic delaunay mesh adaptation for unsteady simulations. In *Proceedings of the 17th international Meshing Roundtable*, pages 177–194. Springer, 2008. (Cited on pages 103 and 140).

[39] N. Donati, A. Sharma, and M. Ovsjanikov. Deep geometric functional maps: Robust feature learning for shape correspondence. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8592–8601, 2020. (Cited on pages 5, 9, 16, 21, 67, 92, 95, 104, and 105).

[40] N. Donati, E. Corman, S. Melzi, and M. Ovsjanikov. Complex functional maps: A conformal link between tangent bundles. *Computer Graphics Forum*, 41(1):317–334, 2022. doi: https://doi.org/10.1111/cgf.14437. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.14437. (Cited on pages 10 and 21).

[41] N. Donati, E. Corman, and M. Ovsjanikov. Deep orientation-aware functional maps: Tackling symmetry issues in shape matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022. (Cited on pages 10 and 21).

[42] J. B. D.P. Kingma. Adam: A method for stochastic optimization. In *ICLR*, 2015. (Cited on pages 56 and 103).

[43] M. Edl, M. Mizerak, and J. Trojan. 3d laser scanners: History and applications. *International Scientific Journal about Simulation*, 2018. (Cited on pages 4 and 14).

[44] M. Eisenberger, Z. Lahner, and D. Cremers. Smooth shells: Multi-scale shape registration with functional maps. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12265–12274, 2020. (Cited on page 67).

[45] M. Eisenberger, A. Toker, L. Leal-Taixé, and D. Cremers. Deep shells: Unsupervised shape correspondence with optimal transport. *Advances in Neural Information Processing Systems*, 34, 2020. (Cited on pages 5, 9, 16, 21, 94, 95, 96, 97, 104, 105, 110, and 135).

[46] M. Eisenberger, D. Novotny, G. Kerchenbaum, P. Labatut, N. Neverova, D. Cremers, and A. Vedaldi. Neuromorph: Unsupervised shape interpolation and correspondence in one go. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. (Cited on pages 5, 9, 16, 20, 43, 94, 103, and 110).

[47] D. Eynard, E. Rodola, K. Glashoff, and M. M. Bronstein. Coupled functional maps. In *3D Vision (3DV)*, pages 399–407. IEEE, 2016. (Cited on pages 48 and 93).

[48] D. Eynard, E. Rodolà, K. Glashoff, and M. M. Bronstein. Coupled functional maps. In *2016 Fourth International Conference on 3D Vision (3DV)*, pages 399–407, Oct 2016. (Cited on page 66).

[49] D. Ezuz and M. Ben-Chen. Deblurring and denoising of maps between shapes. *Computer Graphics Forum*, 36(5):165–174, 2017. (Cited on pages 43, 48, and 50).

[50] M. Fey, J. Eric Lenssen, F. Weichert, and H. Müller. Splinecnn: Fast geometric deep learning with continuous b-spline kernels. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 869–877, 2018. (Cited on page 94).

[51] A. Gehre, M. Bronstein, L. Kobbelt, and J. Solomon. Interactive curve constrained functional maps. In *Computer Graphics Forum*, volume 37-5, pages 1–12. Wiley Online Library, 2018. (Cited on page 93).

[52] D. Ginzburg and D. Raviv. Cyclic functional mapping: Self-supervised correspondence between non-isometric deformable shapes. In *European Conference on Computer Vision*, pages 36–52. Springer, 2020. (Cited on pages 64, 94, 95, 96, 97, and 134).

[53] R. Girshick. Fast r-cnn. *CVPR*, 2015. (Cited on pages 2 and 12).

[54] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. *NeurIPS*, 2014. (Cited on pages 2 and 12).

[55] T. Groueix, M. Fisher, V. G. Kim, B. C. Russell, and M. Aubry. 3d-coded: 3d correspondences by deep deformation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 230–246, 2018. (Cited on pages 46, 47, 49, 51, 56, 57, 58, 60, 94, 112, and 122).

[56] T. Groueix, M. Fisher, V. G. Kim, B. C. Russell, and M. Aubry. Unsupervised cycle-consistent deformation for shape matching. In *Computer Graphics Forum*, volume 38-5, pages 123–133. Wiley Online Library, 2019. (Cited on pages 46 and 112).

[57] X. D. Gu, R. Guo, F. Luo, and W. Zeng. Discrete laplace-beltrami operator determines discrete riemannian metric. *arXiv preprint arXiv:1010.4070*, 2010. (Cited on page 127).

[58] P. Guerrero, Y. Kleiman, M. Ovsjanikov, and N. J. Mitra. PCPNet: Learning local shape properties from raw point clouds. *Computer Graphics Forum*, 37(2): 75–85, 2018. doi: 10.1111/cgf.13343. (Cited on page 61).

[59] V. Guillemin and A. Pollack. *Differential topology*, volume 370. American Mathematical Soc., 2010. (Cited on page 68).

[60] C. Guo, J. Gardner, Y. You, A. G. Wilson, and K. Weinberger. Simple black-box adversarial attacks. *36th International Conference on Machine Learning, PMLR*, 97:2484–2493, 2019. (Cited on pages 3 and 13).

[61] O. Halimi, O. Litany, E. Rodol'a, A. Bronstein, and R. Kimmel. Unsupervised learning of dense shape correspondence. In *CVPR*, 2019. (Cited on pages 46, 47, 48, 50, 52, 54, 56, 58, and 122).

[62] O. Halimi, O. Litany, E. Rodola, A. M. Bronstein, and R. Kimmel. Unsupervised learning of dense shape correspondence. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4370–4379, 2019. (Cited on pages 5, 7, 9, 16, 18, 20, 43, 64, 92, 94, 95, 96, 97, 104, 134, and 135).

[63] J. Han, L. Shao, D. Xu, and J. Shotton. Enhanced computer vision with microsoft kinect sensor: A review. *IEEE transactions on cybernetics*, 43:1318–1334, 2013. (Cited on pages 4 and 15).

[64] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. (Cited on pages 3 and 14).

[65] J. Ho, C. Saharia, W. Chan, D. J. Fleet, M. Norouzi, and T. Salimans. Diffusion models beat gans on image synthesis. *arXiv:2106.15282*, 2021. (Cited on pages 2 and 12).

[66] R. A. Horn and C. R. Johnson. *Matrix analysis*. Cambridge university press, 2012. (Cited on page 75).

[67] Q. Huang, F. Wang, and L. Guibas. Functional map networks for analyzing and exploring large shape collections. *ACM Transactions on Graphics (TOG)*, 33(4): 36, 2014. (Cited on pages 48 and 66).

[68] R. Huang and M. Ovsjanikov. Adjoint map representation for shape analysis and matching. In *Computer Graphics Forum*, volume 36-5, pages 151–163. Wiley Online Library, 2017. (Cited on pages 48 and 66).

[69] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. *CVPR*, 2017. (Cited on pages 2 and 12).

[70] H. Kato, D. Beker, M. Morariu, T. Ando, T. Matsuoka, W. Kehl, and A. Gaidon. Differentiable rendering: A survey. *arXiv:2006.12057v2*, 2020. (Cited on page 112).

[71] L. Kharevych, B. Springborn, and P. Schröder. Discrete conformal mappings via circle patterns. *ACM Transactions on Graphics (TOG)*, 25(2):412–438, 2006. (Cited on page 69).

[72] V. G. Kim, Y. Lipman, and T. Funkhouser. Blended intrinsic maps. In *ACM Transactions on Graphics (TOG)*, volume 30-4, page 79. ACM, 2011. (Cited on pages 59, 117, 118, 139, and 140).

[73] V. G. Kim, Y. Lipman, and T. Funkhouser. Blended intrinsic maps. In *ACM Transactions on Graphics (TOG)*, volume 30-4, page 79. ACM, 2011. (Cited on pages 69 and 130).

[74] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv pre-Print*, 2013. (Cited on pages 2 and 12).

[75] Y. Kleiman and M. Ovsjanikov. Robust structure-based shape correspondence. In *Computer Graphics Forum*. Wiley Online Library, 2018. (Cited on page 67).

[76] F. Knöppel, K. Crane, U. Pinkall, and P. Schröder. Globally optimal direction fields. *ACM Transactions on Graphics (ToG)*, 32(4):1–10, 2013. (Cited on page 74).

[77] A. Kovnatsky, M. M. Bronstein, A. M. Bronstein, K. Glashoff, and R. Kimmel. Coupled quasi-harmonic bases. In *Computer Graphics Forum*, volume 32-2pt4, pages 439–448, 2013. (Cited on pages 48 and 66).

[78] 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*, pages 905–914, 2015. (Cited on page 48).

[79] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *NeurIPS*, 2012. (Cited on pages 3 and 14).

[80] J. M. Lee. Smooth manifolds. In *Introduction to Smooth Manifolds*, pages 1–31. Springer, 2013. (Cited on pages 68, 71, and 124).

[81] L. Li, S. Zhu, H. Fu, P. Tan, and C.-L. Tai. End-to-end learning local multi-view descriptors for 3d point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1919–1928, 2020. (Cited on pages 4 and 15).

[82] L. Li, N. Donati, and M. Ovsjanikov. Learning multi-resolution functional maps with spectral attention for robust shape matching. *NeurIPS*, 2022. (Cited on pages 10, 22, 110, and 112).

[83] C.-H. Lin, W.-C. Ma, A. Torralba, and S. Lucey. Barf: Bundle-adjusting neural radiance fields. *ICCV*, 2021. (Cited on page 112).

[84] Y. Lipman and T. Funkhouser. Möbius voting for surface correspondence. *ACM Transactions on Graphics (TOG)*, 28(3):1–12, 2009. (Cited on page 69).

[85] O. Litany, T. Remez, E. Rodolà, A. M. Bronstein, and M. M. Bronstein. Deep functional maps: Structured prediction for dense shape correspondence. *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 5660–5668, 2017. (Cited on pages 5, 6, 7, 16, 17, 18, 44, 46, 47, 48, 50, 51, 52, 54, 55, 56, 58, 60, 64, 92, 94, 95, 96, 100, 104, 110, 116, and 122).

[86] O. Litany, E. Rodolà, A. M. Bronstein, and M. M. Bronstein. Fully spectral partial shape matching. In *Computer Graphics Forum*, volume 36-2, pages 247–258. Wiley Online Library, 2017. (Cited on page 93).

[87] R. Litman and A. M. Bronstein. Learning spectral descriptors for deformable shape correspondence. *IEEE transactions on pattern analysis and machine intelligence*, 36(1):171–180, 2013. (Cited on page 91).

[88] M. Loper, N. Mahmood, J. Romero, G. Pons-Moll, and M. J. Black. Smpl: A skinned multi-person linear model. *ACM Trans. Graph.*, 34(6):248:1–248:16, Oct. 2015. ISSN 0730-0301. doi: 10.1145/2816795.2818013. URL http://doi.acm.org/10.1145/2816795.2818013. (Cited on pages 51 and 57).

[89] R. Magnet, J. Ren, O. Sorkine-Hornung, and M. Ovsjanikov. Smooth non-rigid shape matching via effective dirichlet energy optimization. *3DV*, 2022. (Cited on pages 43 and 112).

[90] R. Marin, S. Melzi, E. Rodolá, and U. Castellani. Farm: Functional automatic registration method for 3d human bodies. *Computer Graphics Forum*, 39(1): 160–173, 2020. (Cited on pages 64 and 67).

[91] R. Marin, M.-J. Rakotosaona, S. Melzi, and M. Ovsjanikov. Correspondence learning via linearly-invariant embedding. *Advances in Neural Information Processing Systems*, 2020. (Cited on pages 107 and 110).

[92] J. Masci, D. Boscaini, M. Bronstein, and P. Vandergheynst. Geodesic convolutional neural networks on riemannian manifolds. In *Proceedings of the IEEE international conference on computer vision workshops*, pages 37–45, 2015. (Cited on pages 5, 7, 16, 18, 49, 91, 94, and 112).

[93] S. Melzi, R. Marin, E. Rodolà, U. Castellani, J. Ren, A. Poulenard, P. Wonka, and M. Ovsjanikov. Matching Humans with Different Connectivity. In S. Biasotti, G. Lavoué, and R. Veltkamp, editors, *Eurographics Workshop on 3D Object Retrieval*. The Eurographics Association, 2019. ISBN 978-3-03868-077-2. doi: 10.2312/3dor.20191070. (Cited on pages 51, 57, and 103).

[94] S. Melzi, J. Ren, E. Rodolà, A. Sharma, P. Wonka, and M. Ovsjanikov. Zoomout: Spectral upsampling for efficient shape correspondence. *ACM Transactions on Graphics (TOG)*, 38(6):155:1–155:14, Nov. 2019. ISSN 0730-0301. (Cited on pages 43, 44, 49, 50, 55, 56, 58, 60, 66, 81, 86, 88, 103, 104, 105, 117, and 122).

[95] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. Representing scenes as neural radiance fields for view synthesis. *ICCV*, 2021. (Cited on page 112).

[96] F. Monti, D. Boscaini, J. Masci, E. Rodolà, J. Svoboda, and M. M. Bronstein. Geometric deep learning on graphs and manifolds using mixture model cnns. In *CVPR*, pages 5425–5434. IEEE Computer Society, 2017. (Cited on page 49).

[97] F. Monti, D. Boscaini, J. Masci, E. Rodola, J. Svoboda, and M. M. Bronstein. Geometric deep learning on graphs and manifolds using mixture model cnns. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5115–5124, 2017. (Cited on pages 91 and 94).

[98] S. Morita. *Geometry of Differential Forms*. American Mathematical Society, 2001. (Cited on pages 27, 30, 32, 42, and 75).

[99] L. Morreale, N. Aigerman, V. G. Kim, and N. J. Mitra. Neural surface maps. *CVPR*, 2021. (Cited on page 110).

[100] L. Morreale, N. Aigerman, P. Guerrero, V. G. Kim, and N. J. Mitra. Neural convolutional surfaces. *CVPR*, 2022. (Cited on pages 5, 16, and 109).

[101] P. Mullen, Y. Tong, P. Alliez, and M. Desbrun. Spectral conformal parameterization. In *Computer Graphics Forum*, volume 27-5, pages 1487–1494. Wiley Online Library, 2008. (Cited on page 69).

[102] D. Nogneng and M. Ovsjanikov. Informative descriptor preservation via commutativity for shape matching. *Computer Graphics Forum*, 36(2):259–267, 2017. (Cited on page 48).

[103] D. Nogneng and M. Ovsjanikov. Informative descriptor preservation via commutativity for shape matching. *Computer Graphics Forum*, 36(2):259–267, 2017. (Cited on pages 7, 18, 64, 66, 67, 81, 82, 85, and 93).

[104] D. Nogneng, S. Melzi, E. Rodolà, U. Castellani, M. Bronstein, and M. Ovsjanikov. Improved functional mappings via product preservation. In *Computer Graphics Forum*, volume 37-2, pages 179–190. Wiley Online Library, 2018. (Cited on pages 7, 18, and 66).

[105] D. W. Otter, J. R. Medina, and J. K. Kalita. A survey of the usages of deep learning for natural language processing. *arXiv:1807.10854*, 2019. (Cited on pages 2 and 12).

[106] 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(4):30, 2012. (Cited on pages 6, 7, 17, 18, 40, 48, 52, 54, 55, 58, 64, 66, 67, 70, 72, 75, 78, 79, 81, 83, 85, 92, 93, 96, and 117).

[107] M. Ovsjanikov, Q. Mérigot, V. Pătrăucean, and L. Guibas. Shape matching via quotient spaces. *Computer Graphics Forum*, 32(5):1–11, 2013. (Cited on pages 7, 8, 18, 19, 67, and 84).

[108] 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*, SIGGRAPH '17, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450350143. doi: 10.1145/3084873.3084877. URL https://doi.org/10.1145/3084873.3084877. (Cited on pages 48, 49, 54, 66, 93, and 96).

[109] G. Pai, J. Ren, S. Melzi, P. Wonka, and M. Ovsjanikov. Fast sinkhorn filters: Using matrix scaling for non-rigid shape correspondence with functional maps. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 384–393, 2021. (Cited on pages 79 and 96).

[110] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami. The limitations of deep learning in adversarial settings. *arXiv:1511.07528*, 2015. (Cited on pages 3 and 13).

[111] L. Pishchulin, S. Wuhrer, T. Helten, C. Theobalt, and B. Schiele. Building statistical shape spaces for 3d human modeling. *Pattern Recognition*, 67:276–286, 2017. (Cited on page 91).

[112] A. Poulenard and M. Ovsjanikov. Multi-directional geodesic neural networks via equivariant convolution. *ACM Transactions on Graphics (TOG)*, 37(6):1–14, 2018. (Cited on pages 5, 7, 16, 18, 67, 91, 94, 97, and 112).

[113] A. Poulenard, P. Skraba, and M. Ovsjanikov. Topological function optimization for continuous shape matching. In *Computer Graphics Forum*, volume 37-5, pages 13–25. Wiley Online Library, 2018. (Cited on pages 7, 18, and 49).

[114] A. Poulenard, M.-J. Rakotosaona, Y. Ponty, and M. Ovsjanikov. Effective rotation-invariant point cnn with spherical harmonics kernels. *arXiv preprint arXiv:1906.11555*, 2019. (Cited on pages 5, 7, 16, 18, 61, and 110).

[115] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 1(2):4, 2017. (Cited on pages 5, 6, 7, 16, 17, 18, 49, 51, 112, and 116).

[116] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pages 5105–5114, USA, 2017. Curran Associates Inc. ISBN 978-1-5108-6096-4. URL http://dl.acm.org/citation.cfm?id=3295222.3295263. (Cited on pages 5, 6, 7, 16, 17, 18, 49, and 112).

[117] M.-J. Rakotosaona and M. Ovsjanikov. Intrinsic point cloud interpolation via dual latent space navigation. *ECCV*, 2020. (Cited on pages 5, 16, and 112).

[118] M. J. Rakotosaona, V. L. Barbera, P. Guerrero, N. J. Mitra, and M. Ovsjanikov. Pointcleannet: Learning to denoise and remove outliers from dense point clouds. *CVPR*, 2021. (Cited on pages 4, 5, 15, 16, and 112).

[119] M.-J. Rakotosaona, P. Guerrero, N. Aigerman, N. J. Mitra, and M. Ovsjanikov. Learning delaunay surface elements for mesh reconstruction. *CVPR*, 2021. (Cited on pages 4, 5, 15, 16, and 112).

[120] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen. Hierarchical text-conditional image generation with clip latents. *arXiv:2204.06125*, 2022. (Cited on pages 2 and 12).

[121] N. Ray and B. Lévy. Hierarchical least squares conformal map. In *11th Pacific Conference onComputer Graphics and Applications, 2003. Proceedings.*, pages 263–270. IEEE, 2003. (Cited on page 69).

[122] M. Reed and B. Simon. *Methods of Modern Mathematical Physics: Functional Analysis; Rev. ed.* Academic press, 1980. (Cited on page 73).

[123] R. Remmert. *Theory of complex functions*, volume 122. Springer Science & Business Media, 1991. (Cited on page 70).

[124] J. Ren, A. Poulenard, P. Wonka, and M. Ovsjanikov. Continuous and orientation-preserving correspondences via functional maps. *ACM Transactions on Graphics (TOG)*, 37(6), 2018. (Cited on pages 7, 18, 43, 48, 50, 55, 56, 57, 58, 64, 66, 67, 81, 82, 85, 86, 93, 97, 103, 104, 105, 106, and 111).

[125] J. Ren, M. Panine, P. Wonka, and M. Ovsjanikov. Structured regularization of functional map computations. In *Computer Graphics Forum*, volume 38-5, pages 39–53. Wiley Online Library, 2019. (Cited on pages 48 and 112).

[126] J. Ren, S. Melzi, M. Ovsjanikov, and P. Wonka. Maptree: Recovering multiple solutions in the space of maps. *ACM Trans. Graph.*, 39(6), Nov. 2020. (Cited on pages 64, 67, 81, 84, 86, 87, 88, and 130).

[127] J. Ren, S. Melzi, P. Wonka, and M. Ovsjanikov. Discrete optimization for shape matching. In *SGP*, 2021. (Cited on pages 43, 81, 85, 86, 87, 88, 112, and 132).

[128] E. Rodolà, M. Moeller, and D. Cremers. Point-wise map recovery and refinement from functional correspondence. In *Proc. Vision, Modeling and Visualization (VMV)*, 2015. (Cited on pages 43, 48, and 50).

[129] E. Rodolà, L. Cosmo, M. M. Bronstein, A. Torsello, and D. Cremers. Partial functional correspondence. In *Computer Graphics Forum*, volume 36-1, pages 222–236. Wiley Online Library, 2017. (Cited on pages 48, 66, and 93).

[130] R. Roriz, J. Cabral, and T. Gomes. Automotive lidar technology: A survey. *IEEE Transactions on Intelligent Transportation Systems*, 23(7):6282 – 6297, 2013. (Cited on pages 4 and 15).

[131] J.-M. Roufosse, A. Sharma, and M. Ovsjanikov. Unsupervised deep learning for structured shape matching. *arXiv preprint arXiv:1812.03794*, 2018. (Cited on pages 5 and 16).

[132] J.-M. Roufosse, A. Sharma, and M. Ovsjanikov. Unsupervised deep learning for structured shape matching. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1617–1627, 2019. (Cited on pages 7, 9, 18, 20, 44, 46, 47, 48, 50, 51, 52, 54, 55, 56, 58, 61, 92, 94, 95, 96, 97, 101, 134, and 135).

[133] A. Roy, J. Sun, R. Mahoney, L. Alonzi, S. Adams, and P. Beling. Deep learning detecting fraud in credit card transactions. *Systems and Information Engineering Design Symposium (SIEDS)*, 2018. (Cited on pages 2 and 13).

[134] R. M. Rustamov, M. Ovsjanikov, O. Azencot, M. Ben-Chen, F. Chazal, and L. Guibas. Map-based exploration of intrinsic shape differences and variability. *ACM Transactions on Graphics (TOG)*, 32(4):72, 2013. (Cited on pages 66 and 72).

[135] Y. Sahillioğlu. Recent advances in shape correspondence. *The Visual Computer*, pages 1–17, 2019. (Cited on pages 48 and 93).

[136] Y. Sahillioğlu. Recent advances in shape correspondence. *The Visual Computer*, 36(8):1705–1721, 2020. (Cited on pages 63 and 66).

[137] W. S. Sarle. Stopped training and other remedies for overfitting. *Computing science and statistics*, 1996. (Cited on pages 3 and 13).

[138] R. Sawhney and K. Crane. Boundary first flattening. *ACM Transactions on Graphics (ToG)*, 37(1):1–14, 2017. (Cited on page 69).

[139] P. Schmidt, M. Campen, J. Born, and L. Kobbelt. Inter-surface maps via constant-curvature metrics. *SIGGRAPH*, 2020. (Cited on pages 111 and 112).

[140] A. Sharma and M. Ovsjanikov. Weakly supervised deep functional maps for shape matching. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 19264–19275. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/dfb84a11f431c62436cfb760e30a34fe-Paper.pdf. (Cited on pages 9, 20, 92, 94, 96, 97, 103, and 134).

[141] N. Sharp and K. Crane. A Laplacian for Nonmanifold Triangle Meshes. *Computer Graphics Forum (SGP)*, 39(5), 2020. (Cited on pages 107 and 109).

[142] N. Sharp and M. Ovsjanikov. Pointtrinet: Learned triangulation of 3d point sets. *ECCV*, 2020. (Cited on pages 4, 5, 15, 16, and 112).

[143] N. Sharp, Y. Soliman, and K. Crane. The vector heat method. *ACM Trans. Graph.*, 38(3), 2019. (Cited on pages 72, 73, 74, 98, 111, and 125).

[144] N. Sharp, S. Attaiki, K. Crane, and M. Ovsjanikov. Diffusionnet: Discretization agnostic learning on surfaces, 2020. (Cited on pages 5, 6, 9, 16, 17, 21, 92, 94, 95, 97, 99, 100, 102, 103, 104, 111, and 137).

[145] J. Shen, S. Zafeiriou, G. G. Chrysos, J. Kossaifi, G. Tzimiropoulos, and M. Pantic. The first facial landmark tracking in-the-wild challenge: Benchmark and results. *ICCV*, 2015. (Cited on pages 2 and 12).

[146] M. Shoham, A. Vaxman, and M. Ben-Chen. Hierarchical Functional Maps between Subdivision Surfaces. *Computer Graphics Forum*, 2019. (Cited on page 93).

[147] C. Shorten and T. M. Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of Big Data*, 2019. (Cited on pages 3 and 13).

[148] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. (Cited on pages 3 and 14).

[149] B. Springborn, P. Schröder, and U. Pinkall. Conformal equivalence of triangle meshes. In *ACM SIGGRAPH 2008 papers*, pages 1–11. ACM, 2008. (Cited on page 69).

[150] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 2014. (Cited on pages 3 and 13).

[151] J. Sun, M. Ovsjanikov, and L. Guibas. A concise and provably informative multi-scale signature based on heat diffusion. *Computer graphics forum*, 28(5):1383–1392, 2009. (Cited on pages 6, 7, 17, 18, 41, 48, 50, 67, 84, 94, and 125).

[152] Z. Sun, Y. He, A. Gritsenko, A. Lendasse, and S. Baek. Deep spectral descriptors: Learning the point-wise correspondence metric via siamese deep neural networks. *arXiv preprint arXiv:1710.06368*, 2017. (Cited on page 49).

[153] R. Sundararaman, G. Pai, and M. Ovsjanikov. Implicit field supervision for robust non-rigid shape matching. *European Conference on Computer Vision (ECCV)*, 2022. (Cited on pages 5, 16, and 110).

[154] K. Takayama. Compatible intrinsic triangulations. *ACM Transactions on Graphics*, 41(4), 2022. (Cited on page 111).

[155] T. Takikawa, J. Litalien, K. Yin, K. Kreis, C. Loop, D. Nowrouzezahrai, A. Jacobson, M. McGuire, and S. Fidler. Neural geometric level of detail: Real-time rendering with implicit 3d shapes. *CVPR*, 2021. (Cited on pages 5 and 16).

[156] H. Thomas, C. R. Qi, J.-E. Deschaud, B. Marcotegui, F. Goulette, and L. J. Guibas. Kpconv: Flexible and deformable convolution for point clouds. *Proceedings of the IEEE International Conference on Computer Vision*, 2019. (Cited on pages 5, 6, 7, 16, 17, 18, 19, 49, 51, 52, 53, 55, 56, 104, 112, 114, 115, and 116).

[157] C. Tian, L. Fei, W. Zheng, Y. Xua, W. Zuo, and C. W. Ling. Deep learning on image denoising: An overview. *Elsevier, Neural Networks*, 131:251 – 275, 2020. (Cited on pages 2 and 13).

[158] F. Tombari, S. Salti, and L. Di Stefano. Unique signatures of histograms for local surface description. In *Proc. ECCV*, pages 356–369. Springer, 2010. (Cited on pages 6, 7, 8, 17, 18, 19, 41, 48, 51, 67, 94, 97, 99, 116, 135, and 140).

[159] O. Van Kaick, H. Zhang, G. Hamarneh, and D. Cohen-Or. A survey on shape correspondence. In *Computer Graphics Forum*, volume 30-6, pages 1681–1707, 2011. (Cited on pages 46, 48, 63, and 66).

[160] D. Varmedja, M. Karanovic, S. Sladojevic, M. Arsenovic, and A. Anderla. Credit card fraud detection - machine learning methods. *International Symposium on INFOTEH-JAHORINA (INFOTEH)*, 2019. (Cited on pages 2 and 13).

[161] G. Varol, J. Romero, X. Martin, N. Mahmood, M. J. Black, I. Laptev, and C. Schmid. Learning from synthetic humans. In *CVPR*, 2017. (Cited on pages 51 and 57).

[162] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. *NeurIPS*, 2017. (Cited on pages 2 and 12).

[163] 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 pages 60 and 122).

[164] M. Vestner, R. Litman, E. Rodolà, A. Bronstein, and D. Cremers. Product manifold filter: Non-rigid shape correspondence via kernel density estimation in the product space. In *Proc. CVPR*, pages 6681–6690, 2017. (Cited on page 58).

[165] R. Walecki, O. Rudovic, V. Pavlovic, B. Schuller, and M. Pantic. Deep structured learning for facial action unit intensity estimation. *CVPR*, 2017. (Cited on pages 2 and 12).

[166] L. Wang, A. Gehre, M. M. Bronstein, and J. Solomon. Kernel functional maps. In *Computer Graphics Forum*, volume 37-5, pages 27–36. Wiley Online Library, 2018. (Cited on page 93).

[167] Y. Wang, B. Liu, K. Zhou, and Y. Tong. Vector field map representation for near conformal surface correspondence. In *Computer Graphics Forum*, volume 37-6, pages 72–83. Wiley Online Library, 2018. (Cited on pages 67, 80, 81, 82, 84, 85, 93, and 131).

[168] L. Wei, Q. Huang, D. Ceylan, E. Vouga, and H. Li. Dense human body correspondences using convolutional networks. In *Proceedings of the IEEE Conference on*

Computer Vision and Pattern Recognition, pages 1544–1553, 2016. (Cited on pages 46, 47, and 49).

[169] R. Wiersma, E. Eisemann, and K. Hildebrandt. Cnns on surfaces using rotation-equivariant features. *ACM Transactions on Graphics (TOG)*, 39(4):92–1, 2020. (Cited on pages 91, 94, and 110).

[170] G. Yang, X. Huang, Z. Hao, M.-Y. Liu, S. Belongie, and B. Hariharan. Pointflow: 3d point cloud generation with continuous normalizing flows. *arXiv:1906.12320*, 2019. (Cited on pages 111 and 112).

[171] W. Yang, X. Zhang, Y. Tian, W. Wang, J. H. Xue, and Q. Liao. Deep learning for single image super-resolution: A brief review. *IEEE Transactions on Multimedia*, 21(12):3106 – 3121, 2019. (Cited on pages 2 and 13).

[172] Y. Yu, X. Si, C. Hu, and J. Zhang. A review of recurrent neural networks: Lstm cells and network architectures. *Neural Computation*, 2019. (Cited on pages 2 and 13).

[173] J. Zhou, L.-Y. Luo, Q. Dou, H. Chen, C. Chen, G.-J. Li, Z.-F. Jiang, and P.-A. Heng. Hierarchical text-conditional image generation with clip latents. *JMRI*, 2019. (Cited on pages 2 and 12).

[174] S. Zuffi, A. Kanazawa, D. Jacobs, and M. J. Black. 3D menagerie: Modeling the 3D shape and pose of animals. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2017*, pages 5524–5532, Piscataway, NJ, USA, July 2017. IEEE. (Cited on pages 86 and 87).

[175] S. Zuffi, A. Kanazawa, D. Jacobs, and M. J. Black. 3D menagerie: Modeling the 3D shape and pose of animals. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2017*, pages 5524–5532, Piscataway, NJ, USA, July 2017. IEEE. (Cited on pages 103, 106, and 135).

ECOLE DOCTORALE

Titre : Représentations robustes pour la mise en correspondence de formes 3D via apprentissage supervisé et non-supervisé

Mots clés : géometrie, nuage de points, maillage, réseau de neurones, correspondance, champ de vecteur

Résumé : L'analyse de données 3D est un problème fondamental de la science moderne, et les avancées récentes telles que l'apprentissage profond ont ouvert la voie à de nouvelles possibilités dans ce domaine. Cependant, l'analyse de formes 3D présente des problèmes complexes en raison de la structure particulière des données mises en jeu. Dans cette thèse, nous nous concentrons principalement sur la mise en correspondance de formes 3D dont le but est d'étudier et de calculer des applications entre formes, habituellement représentées par des maillages triangulaires. Cette correspondance entre deux formes peut être utilisée pour transférer de l'information (segmentation, texture). Afin d'aborder ce problème complexe et non linéaire, nous adoptons un point de vue fonctionnel qui permet une représentation plus simple et efficace pour ces applications entre formes. Certains réseaux de neurones profonds on été conçus pour produire des signaux descriptifs sur des nuages de points ou des maillages, pour la segmentation par exemple, ainsi nous proposons d'utiliser ces signaux pour produire des correspondances précises dans le cadre de la méthode des "applications fonctionnelles", sur laquelle nous nous basons pour implémenter de nouveaux algorithmes de correspondance. Nous abordons notamment plusieurs problèmes-clé à l'intersection entre la mise en correspondance de formes et l'apprentissage profond. Tout d'abord, l'écueil du surapprentissage, très récurrent particulièrement dans les jeux de données 3D qui ne sont pas aussi variés que pour les images. Ensuite, nous proposons d'incorporer l'information d'orientation de formes dans le cadre des applications fonctionnelles en utilisant l'analyse de champs de vecteurs tangents, que nous utilisons pour résoudre des défauts de symétrie. Enfin, nous proposons une solution globale pour apprendre sur des formes 3D pour une mise en correspondance robuste et non supervisée. Notre travail propose des méthodes efficaces pour explorer l'espace des applications entre les formes 3D en exploitant la structure particulière des surfaces pour construire des régularisateurs pour les réseaux d'apprentissage profond visant à établir des correspondances.

Title : Robust representations for supervised and unsupervised 3D shape matching

Keywords : geometry, point cloud, mesh, neural network, shape matching, vector field, functional maps

Abstract : 3D data analysis is a fundamental problem in modern science, and recent advances such as deep learning have opened the door to new algorithms and possibilities in this field. Nevertheless, 3D shape analysis presents difficult problems due to its particular structure. Indeed, deep neural networks and more specifically convolutional neural networks were originally tailored to tackle grid-like structures like images. Consequently, the challenge is to adapt deep learning to more complex structures like 3D point cloud or meshes. In this thesis, we focus the problem of non-rigid 3D shape matching, whose objective is to analyze and compute maps between shapes, typically represented as triangle meshes. Correspondence between a pair of shapes can be used to transfer information such as texture, or segmentation, from one shape to the other. To tackle this hard non-linear problem, we adopt a functional point of view allowing for a simpler and more efficient representation of maps between shapes. Since deep neural networks have been designed to produce feature functions on point clouds or meshes, e.g. for segmentation, we propose to use these network features to match functional spaces using the so-called functional map framework, which we build upon to implement new algorithms for shape matching. In particular, we tackle several key problems which lie at the intersection between shape matching and deep learning. Firstly, we propose a method that helps to address the well-known problem of overfitting, which is a very recurrent problem particularly for 3D data. Secondly, we propose a new way to incorporate orientation information into the functional map pipeline using tangent vector field analysis. We use this novel representation to solve some symmetry issues, difficult to address because of the intrinsic nature of functional maps. Lastly, we propose a global solution that learns features for efficient and robust shape matching, in an unsupervised way. Overall our work proposes efficient methods to explore the space of maps between shapes by exploiting the particular structure of 3D surfaces to build robust regularizers for deep learning correspondence networks.

Institut Polytechnique de Paris
91120 Palaiseau, France