Structures for deep learning and topology optimization of functions on 3D shapes

Adrien Poulenard

INSTITUT POLYTECHNIQUE DE PARIS

NNT : 2020IPPAX007

Thèse de doctorat

Structures for deep learning and topology optimization of functions on 3D shapes

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 (IP Paris)
Spécialité de doctorat : Informatique, Données et Intelligence Artificielle

Thèse présentée et soutenue à Palaiseau, le 15/04/2020, par

ADRIEN POULENARD

Composition du Jury :

M. Pierre Alliez Directeur de recherche, Inria Sophia Antipolis	Président
M. Yaron Lipman Professeur, Weizmann Institute of Science	Rapporteur
M. Matthieu Cord Professeur, Sorbonne Université (LIP6)	Examinateur
Mme. Stefanie Wuhrer Chargé de recherche, INRIA Grenoble Rhône-Alpes	Examinateur
Mme. Julie Digne Chargé de recherche, CNRS	Examinateur
M. Maks Ovsjanikov Professeur, École Polytechnique (LIX)	Directeur de thèse

Contents

Chapter 1

Introduction

We live in an era where increasingly many tasks are being automated, drastically improving productivity. We can trace back this trend at least to the industrial revolution which introduced the massive use of machines to boost human productivity. However, until recently many tasks required human supervision. The progress of new digital technologies allows the automation of an increasing number of tasks. Nowadays many autonomous agents interacting with a complex 3D world are being developed. Robots are designed to automate the production, storage and delivery of goods in various industries and in agriculture. Self-driving cars and domestic robots might become a reality in the near future. These agents rely on various sensors to perceive their environment and require algorithms to process and analyse input signals in order to make decisions.

In recent years considerable progress has been made in the design of such algorithms, largely thanks to the progress of machine learning and especially of deep learning and its numerous successes. A particularly successful class of algorithms for image analysis is convolutional neural networks (CNN) popularized by the now famous AlexNet [64] algorithm who won the 2012 ImageNet competition [106] by a large margin. Since then, CNNs have been applied in many different fields such as medical image analysis [70], facial recognition [92], object classification or detection [64, 48, 101] to name a few, with new applications coming every year. The success of deep learning algorithms is due to two main factors: the increase of computational power with the use of modern GPUs to accelerate computation and the availability of massive data to train algorithms. In addition to the proliferation of image acquisition devices in recent years many 3D technologies previously reserved to experts in the industry have made their way to the consumer market. It is now possible to find relatively affordable depth field cameras like the Kinect, virtual reality devices like Oculus Rift and even 3D printers. For this reason it is expected that our digital world will become increasingly 3 dimensional in the coming years. The emergence and popularisation of new 3D sensor technologies allows autonomous agents to interact with the 3D environment in a much more complete way than previously authorized by standard acquisition devices like cameras. This together with the availability of 3D data with datasets like ShapeNet [23], Modelnet [137] and ABC [59] motivates the development of new algorithms for processing and analysing 3D signals.

Recently considerable research effort has been made to replicate the success of methods from image analysis to 3D shape analysis (see [17] for a survey of the state of the art in geometric deep learning prior to the work presented in this thesis). Despite the similarities between the two fields, shape analysis has its own unique challenges. Perhaps the biggest difference is the object of study. Image analysis studies signals over a fixed domain (an

image is a signal over a regular grid of pixels) while in shape analysis the object of study is the domain itself. This is especially challenging for learning since working on a fixed domain allows to associate statistics of the signal to a certain location and query statistics across locations in a consistent way, this is key to the success of CNNs. Moreover 3D shapes can be represented using different modalities such as point clouds, polygon meshes or even volumetric grids depending on the way they were acquired or generated. What makes learning on 3D data particularly challenging is that these representations are highly non canonical, the order of points in a point cloud or a mesh is arbitrary, different mesh or point cloud structures can represent the same object. Designing algorithms that can accommodate and are invariant to the shape representation is a challenging task on its own.

Another major difficulty comes from the fact that shapes might undergo various rigid transforms such as rotations and articulated motion or complex non rigid deformations. Algorithms for 3D shape analysis are often required to make predictions which are invariant to rigid motion of the objects. For instance in classification tasks the predicted class of an object shouldn't depend on its pose, similarly the part labels predicted by a segmentation algorithm should be invariant to rotations and translations. Many existing datasets of 3D shapes such as [23] and [137] are aligned meaning that the shapes are in a canonical pose. Classical networks like [76, 100, 73] trained on aligned data usually fail to generalise to rotated shapes. The classical solution consists in data augmentation in the form of random rotations at training. This can be computationally inefficient and the final performance still drops noticeably compared to the aligned case. Designing algorithms that are either invariant to these transforms or can quantify them such as equivariant algorithms whose output undergoes a predictable transform given certain transforms of the input is a challenging task.

In the following dissertation we propose three contributions to these challenges based on the following three publications forming the main chapters of the dissertation.

List of publications

1. Topological Function Optimization for Continuous Shape Matching, [97], Symposium on geometry processing (SGP), 2018.

2. Multi-directional Geodesic Neural Networks via Equivariant Convolution, [95], SIGGRAPH asia, 2018.

3. Effective Rotation-invariant Point CNN with Spherical Harmonics Kernels, [96], 3DV, 2019.

In Chapter 3 we introduce a new method for optimising topological properties of real valued functions. Our method is based on a descriptor of the topology of level sets of real valued functions coming from the theory of *persistent homology* [35] called *persistence diagram* [21, 20]. We show that the persistence diagram of a function is differentiable with respect to the function allowing to modify parametric functions to remove topological noise or to exhibit prescribed topological features using continuous

optimisation techniques. We present a method for aligning persistence diagrams of functions on different domains, without requiring correspondences between them based on the minimization of a "topological" energy term. We demonstrate the utility of these tools within the context shape matching with functional maps. Shape matching is a classical task which consists in finding correspondences between shapes that preserve geometric or semantic properties. The functional map pipeline [88] for shape matching is based on the observation that a map between two shapes induces a map between functions on these shapes in the opposite direction (via the pull-back operation). Given bases of functions on the two shapes it is possible to represent a map as a matrix and formulate the shape matching problem as a linear algebra optimization problem. One difficulty is to convert this matrix to a point-to-point map, since the standard process of conversion often results in discontinuous maps with matching errors. Existing methods for improving the continuity of functional maps-based correspondences often rely on a post processing step on the resulting point-to-point map. In [97] we propose a method to enforce continuity of the underlying point-to-point map directly during the functional map optimization phase by aligning the persistence diagrams of functions and their images via the functional map, effectively resulting in smoother and more accurate correspondences.

In Chapter 4 we consider the problem of learning on triangle meshes. The first challenge to overcome is to design a learning framework which is invariant to the particular structure of the mesh (connectivity, shape of triangles). We follow the approach of [75] who introduced the idea of using local polar coordinates on triangles meshes do define a convolution operator and use it in a deep learning pipeline. A major difficulty arising when using local polar coordinates is that they are canonical only up to rotation of each coordinate frame. The solution considered in [75] is to take the maximal response of the convolution operator over all choices of polar coordinates thus losing the local direction information. In [95] we propose a new direction aware convolution operator on triangles meshes which is *equivariant* with respect to rotations of the local polar coordinates and show an overall improvement in training and test accuracy in various experimental settings.

One limitation of the method we present in Chapter 4 is that it is only applicable to shapes represented as triangle meshes. On the other hand point clouds provide the least structured data representation, since any other representation can be converted to a point cloud via sampling. Thus any algorithm working on point clouds will generalise to other formats. Also 3D sensors usually produce point cloud data. This motivates the development of algorithms operating on point cloud data. In Chapter 5 we consider the problem of learning over point clouds, and specifically focus on rotation *invariance*. In [96] we propose a new rotation invariant deep learning framework for point clouds analysis. We show that our model is indeed robust to rotations and achieves better performance on classical classification and segmentation tasks, when tested on unaligned datasets.

In Chapter 6 we present a method for enforcing orientation preservation of functional maps [88] in view of resolving symmetry issues in shape matching applications of the functional map pipeline. Our method contributed to the paper:

4. Continuous and orientation-preserving correspondences via functional maps, [102], SIGGRAPH Asia, 2018.

This paper introduced a new method for improving bijectivity, continuity and orientation preservation of point-to-point maps obtained via the functional map pipeline.

Finally we conclude in Chapter 7 where we propose a survey of follow-up works based on the contributions presented in this thesis and propose directions for future work.

Chapter 2

Introduction en français

Nous vivons à une époque où de plus en plus de tâches sont automatisées, ce qui améliore considérablement la productivité. On peut retracer cette tendance au moins jusqu'à la révolution industrielle qui a introduit l'utilisation massive des machines pour accroître la productivité humaine. Cependant, jusqu'à récemment, de nombreuses tâches nécessitaient une supervision humaine. Les progrès des nouvelles technologies numériques permettent l'automatisation d'un nombre croissant de tâches. De nos jours, de nombreux agents autonomes interagissant avec un environnement 3D complexe sont en cours de développement. Des robots sont conçus pour automatiser la production, le stockage et la livraison de marchandises dans diverses industries ainsi que dans l'agriculture. Les voitures autonomes et les robots domestiques pourraient devenir une réalité dans un proche avenir. Ces agents s'appuient sur différents capteurs pour percevoir leur environnement et nécessitent des algorithmes pour analyser et traiter les signaux d'entrée afin de prendre des décisions.

Ces dernières années, des progrès considérables ont été réalisés dans la conception de ces algorithmes, en grande partie grâce aux progrès de l'apprentissage machine et surtout de l'apprentissage profond et à ses nombreux succès. Les réseaux neuronaux convolutifs (CNN) constituent une classe d'algorithmes particulièrement efficace pour l'analyse d'images. Popularisé par le désormais célèbre algorithme AlexNet [64] qui a remporté le concours ImageNet 2012 [106] avec une large marge. Depuis lors, les réseaux CNN ont été utilisés dans de nombreux domaines tels que l'analyse d'images médicales [70], la reconnaissance faciale [92], la classification ou détection d'objets [64, 48, 101] pour ne citer que ceux-là et de nouvelles applications viennent s'ajouter à la liste chaque année.

Le succès des algorithmes d'apprentissage profond est dû à deux facteurs principaux : l'augmentation de la puissance de calcul avec l'utilisation des GPU modernes pour accélérer le calcul et la disponibilité de données massives pour entrainer les algorithmes.

En plus de la prolifération des dispositifs d'acquisition d'images au cours des dernières années, de nombreuses technologies 3D auparavant réservées à des experts dans l'industrie ont fait leur chemin vers le marché grand public. Il est maintenant possible de trouver des caméras 3D relativement abordables comme le Kinect, des appareils de réalité virtuelle comme l'Oculus Rift et même des imprimantes 3D. C'est pourquoi on s'attend à ce que notre monde numérique devienne de plus en plus tridimensionnel dans les années à venir. L'émergence et la popularisation de nouvelles technologies de capteurs 3D permettent aux agents autonomes d'interagir avec l'environnement 3D d'une manière beaucoup plus complète qu'auparavant avec les dispositifs d'acquisition standard comme les caméras traditionnelles. Ceci, combiné à la disponibilité de données 3D avec des

ensembles d'apprentissage comme ShapeNet [23], Modelnet [137] et ABC [59] motive le développement de nouveaux algorithmes pour traiter et analyser des signaux 3D.

Récemment, d'importants efforts de recherche ont été faits pour transposer le succès de méthodes issues de l'analyse d'images à l'analyse de formes 3D (voir [17] pour un aperçu de l'état de l'art en apprentissage profond géométrique précédant les travaux présentés dans cette thèse). Malgré les similitudes entre les deux domaines, l'analyse de formes a ses propres défis. La plus grande différence est sans doute l'objet d'étude. L'analyse d'image étudie les signaux sur un domaine fixe (une image est un signal sur une grille régulière de pixels) alors que dans le cas de l'analyse de formes 3D l'objet d'étude est le domaine lui-même. Cette différence complique grandement l'apprentissage sur les formes 3D car travailler sur un domaine fixe permet d'associer des statistiques du signal à des emplacements identifiables et de comparer ces statistiques d'un emplacement à l'autre de manière cohérente, ce qui est une clé du succès des CNNs. De plus, les formes 3D peuvent être représentées selon différentes modalités telles que les nuages de points, les maillages polygonaux ou même les grilles volumétriques en fonction de la façon dont elles ont été acquises ou produites. Ce qui rend l'apprentissage sur les données 3D particulièrement difficile. Ces représentations sont hautement non canoniques, l'ordre des points dans un nuage de points ou un maillage est arbitraire, différentes structures de maillage ou de nuages de points peuvent représenter le même objet. La conception d'algorithmes qui peuvent s'adapter à ces différentes modalités et sont invariants à la représentation de la forme est une tâche difficile en soi.

Une autre difficulté majeure vient du fait que les formes peuvent subir diverses transformations rigides telles que des rotations et des mouvements articulés ou des déformations complexes non rigides. Les algorithmes d'analyse de formes 3D doivent souvent faire des prédictions qui sont invariantes par deplacement des objets. Par exemple, dans les tâches de classification, la classe prédite d'un objet ne doit pas dépendre de sa pose. De même le découpage prédit par un algorithme de segmentation devrait être invariant par rotations et translations. De nombreux ensembles de d'apprentissage de formes 3D telles que [23] et [137] sont alignés, ce qui signifie que les formes sont dans une pose canonique. Des réseaux classiques tels que [76, 100, 73] entrainés sur des données alignées ne parviennent généralement pas à faire des predictions généralisables à de nouvelles poses. La solution classique consiste à augmenter les données sous forme de rotations aléatoires lors de l'entraînement. Cela peut s'avérer inefficace en matière de temps de calcul et la performance finale chute tout de même sensiblement par rapport au cas aligné. Concevoir des algorithmes qui sont invariants par ces transformations ou qui peuvent les quantifier, comme des algorithmes équivariants dont la sortie subit une transformation prévisible compte tenu de certaines transformations de l'entrée, est une tâche difficile.

Dans le mémoire qui suit, nous proposons trois contributions à ces défis, basées sur les trois publications suivantes qui constituent les principaux chapitres du mémoire:

Liste des publications

1. Topological Function Optimization for Continuous Shape Matching, [97], Sympo-

sium on geometry processing (SGP), 2018.

2. Multi-directional Geodesic Neural Networks via Equivariant Convolution, [95], SIGGRAPH asia, 2018.

3. Effective Rotation-invariant Point CNN with Spherical Harmonics Kernels, [96], 3DV, 2019.

Dans le Chapitre 3 nous introduisons une nouvelle méthode pour optimiser les propriétés topologiques des fonctions à valeurs réelles. Notre méthode est basée sur un descripteur de la topologie des ensembles de niveaux de fonctions à valeurs réelles issues de la théorie de l'homologie persistante. [35] appelé *diagramme de persistance* [21, 20]. Nous montrons que le diagramme de persistance d'une fonction est différentiable par rapport à cette dernière permettant de déformer des fonctions paramétriques pour supprimer le bruit topologique ou pour présenter des caractéristiques topologiques prescrites en utilisant des techniques d'optimisation continue. Nous présentons une méthode pour aligner les diagrammes de persistance des fonctions sur différents domaines, sans nécessiter de correspondances entre ceux-ci basée sur la minimisation d'un terme d'énergie "topologique". Nous démontrons l'utilité de ces outils dans le contexte de la correspondance de formes utilisant la méthode des applications fonctionnelles (Functional maps [88]). La correspondance de formes est un problème classique qui consiste à trouver des correspondances entre des formes qui conservent des propriétés géométriques ou sémantiques. La méthode des applications fonctionnelles [88] pour la correspondance de formes est basé sur l'observation qu'une application entre deux formes induit une application entre des fonctions sur ces formes dans la direction opposée (via l'opération de "pull-back"). Étant donné des bases de fonctions sur les deux formes, il est possible de représenter une application sous la forme d'une matrice et de formuler le problème de correspondance de formes comme un problème d'optimisation d'algèbre linéaire. Une difficulté est de convertir cette matrice en une application point-à-point, le processus standard de conversion aboutit souvent à des applications discontinues avec des erreurs de correspondance. Les méthodes existantes pour améliorer la continuité des correspondances basées sur des applications fonctionnelles utilisent souvent un post-traitement sur l'application point-à-point résultante. Dans [97] nous proposons une méthode pour renforcer la continuité de l'application point-à-point sous-jacente directement durant la phase d'optimisation en alignant les diagrammes de persistance des fonctions et leur image par l'application fonctionnelle, ce qui permet d'obtenir des correspondances plus lisses et précises.

Dans le Chapitre 4 nous considérons le problème de l'apprentissage sur les maillages triangulaires. Le premier défi à relever est de concevoir un cadre d'apprentissage qui ne dépend pas de la structure particulière du maillage (connectivité, forme des triangles). Nous nous inspirons de l'approche de [75] qui a introduit l'idée d'utiliser les coordonnées polaires locales sur les surfaces triangulées pour définir un opérateur de convolution et l'utiliser dans un contexte d'apprentissage profond. Une difficulté majeure qui se pose lorsque l'on utilise les coordonnées polaires locales est qu'elles ne sont canoniques qu'a rotation près. La solution envisagée dans [75] est de considérer la réponse maximale

de l'opérateur de convolution sur tous les choix de coordonnées polaires perdant ainsi l'information directionnelle locale. Dans [95] nous proposons un nouvel opérateur de convolution opérant sur l'espace des directions locales sur les surfaces triangulées qui est équivariant par rapport aux rotations des coordonnées polaires locales. Nous montrons une amélioration globale de la précision de test et d'entraînement dans divers contextes expérimentaux.

Une limitation de la méthode que nous présentons dans le Chapitre 4 est qu'elle n'est applicable qu'aux formes représentées par des maillages triangulaires. D'autre part, le nuage de points est la représentation de données la moins structurée, toute autre représentation peut être convertie en nuage de points par échantillonnage, en particulier tout algorithme opérant sur des nuages de points se généralisera à d'autres formats. De plus, les capteurs 3D produisent généralement des données sous forme de nuages de points. Cela motive le développement d'algorithmes fonctionnant sur des données de ce type. Dans le Chapitre 4 nous examinons le problème de l'apprentissage sur les nuages de points, en nous concentrons spécifiquement sur l'invariance par rotation.

Dans [96] nous proposons un nouveau cadre d'apprentissage profond invariant par rotation pour l'analyse des nuages de points. Nous montrons que notre modèle est robuste aux rotations et qu'il atteint de meilleures performances comparé à l'état de l'art sur des tâches classiques de classification et de segmentation sur des ensembles de données non alignés.

Dans le Chapitre 6 nous présentons une méthode pour contraindre les applications fonctionnelles [88] à preserver l'orientation en vue de résoudre des problèmes de symétrie dans les applications de correspondance de forme basées sur la méthode des applications fonctionnelles. Notre méthode a contribué à la publication suivante:

4. Continuous and orientation-preserving correspondences via functional maps, [102], SIGGRAPH asia, 2018

qui introduit une nouvelle méthode pour améliorer la bijectivité, la continuité et la préservation de l'orientation des applications point-à-point obtenues par la méthode des applications fonctionnelles.

Enfin, nous concluons avec le Chapitre 7 où nous proposons un survol de travaux inspirés par les contributions présentées dans cette thèse ou presentant des perspectives interessantes sur ces dernières et nous suggerons des pistes pour de futurs travaux.

Chapter 3

Topological function optimisation for continuous shape matching

We present a novel approach for optimizing real-valued functions based on a wide range of topological criteria. In particular, we show how to modify a given function in order to remove topological noise and to exhibit prescribed topological features. Our method is based on using the previously-proposed *persistence diagrams* associated with real-valued functions, and on the analysis of the derivatives of these diagrams with respect to changes in the function values. This analysis allows us to use continuous optimization techniques to modify a given function, while optimizing an energy based purely on the values in the persistence diagrams. We also present a procedure for aligning persistence diagrams of functions on different domains, without requiring a mapping between them. Finally, we demonstrate the utility of these constructions in the context of the functional map framework, by first giving a characterization of functional maps that are associated with *continuous* point-to-point correspondences, directly in the functional domain, and then by presenting an optimization scheme that helps to promote the continuity of functional maps, when expressed in the reduced basis, without imposing any restrictions on metric distortion. We demonstrate that our approach is efficient and can lead to improvement in the accuracy of maps computed in practice.

3.1 Introduction

A core problem in geometry processing consists in quantifying similarity between shapes and their parts, as well as detecting detailed region or point-based correspondences [125, 122, 11]. A common approach for both shape comparison and correspondence consists in computing real-valued (for example, descriptor) functions defined on the shapes and comparing the shapes and their parts by comparing the values of such functions. This includes both computing correspondences by matching in descriptor space, and also, more recently, by computing linear transformations between spaces of real-valued functions using the so-called functional map framework [89, 90].

Many existing techniques for comparison of functions on the shapes directly rely on comparing function values, without analyzing the global structure of the functions involved. For example, a descriptor function computed on one shape can have several prominent maxima, whereas on another shape, it can be uniform or with low variance. Intuitively, pairs of functions with dissimilar structural properties can lead to large errors in the correspondence computation. This problem is especially prominent in the context of *functional maps* which are linear transformations between spaces of real-valued functions defined on different shapes. In this case, one is often interested in formulating

an objective which would promote mapping indicator functions of connected regions to other such indicator functions without knowing in advance which regions should match. At a high level, such an objective should promote the preservation of *the topological structure* of the functions before and after the mapping.

In this chapter, we show how such problems can be solved by efficiently optimizing the topological structure of real-valued functions defined on the shapes, either independently (to promote certain structural properties), or jointly (to enforce similarity between such properties), without resorting to combinatorial search or point-to-point maps. The key to our approach is the manipulation of *persistence diagrams* [21, 20]. These diagrams have been shown to summarize the properties of very general classes of topological spaces, including, most relevant to us, real-valued functions defined on the surfaces, and also enjoy several key properties such as being stable under a broad range of perturbations [30, 24]. Existing methods, however, concentrate on either *efficiently constructing* persistence diagrams from a given signal [21, 81, 25] or using them as a tool for, e.g. shape or image comparison [21, 66] or shape segmentation [114] among many others.

Our main insight is that it is possible to formulate optimization objectives on the persistence diagrams of real-valued functions, regardless of their underlying spatial domain, and to optimize a given function to improve such objectives, via continuous non-linear optimization. For this, we first show, how the derivative of a persistence diagram of a function can be computed with respect to the change in the function values, and then how this computation can be used to efficiently optimize various energies defined on persistence diagrams. Crucially, these computations can be performed in any functional basis, which can significantly improve stability and computation speed. Moreover, our approach allows us to jointly optimize the persistence diagrams of multiple functions, without assuming that they are defined over the same domain.

We apply these insights to first provide a characterization of functional maps associated with continuous point-to-point maps, based on the preservation of persistence diagrams. Unlike previous results, this characterization does not assume that the map is isometric or area-preserving and holds for any continuous point-to-point map. We then propose an optimization scheme that helps to promote continuity of functional maps, even when they are expressed in a reduced basis.

3.2 Related Work

Persistent (Co)homology Topology is the study of connectivity and continuity, and persistent (co)homology is a natural language for describing it in an applied setting. Persistence has been widely studied [36] and has become a central tool for the rapidly developing area of topological data analysis. We provide an overview of persistence in Section 3.4. Most relevant to our work is the numerous applications it has found in geometry processing e.g. [21, 34, 114].

In this chapter, we are interested in optimizing functions to achieve certain prescribed topological criteria. Using persistence for modifying functions has been studied as topological simplification, which also served as one of the motivations of the original work on persistence [38]. The simplification problem has been primarily stated as a denoising

problem [5, 9], changing the function so that “persistent features" are preserved. Unlike these works, we approach this problem via continuous optimization, which allows us to explicitly modify the function, expressed in an arbitrary basis to optimize topological criteria. Our approach is inspired by Morse theory, which is deeply tied to persistence [37, 22].

Most related to our work is [44]. The main application of their work was the continuation of point clouds for dynamical systems and so the authors concentrate on a different class of complexes. They derive a similar chain rule result to ours, although both their application and perspective are very different. The key tool in their analysis is the study of the inverse map from the persistence diagram back to the underlying topological space. These maps have been studied in other contexts in applied topology, including persistence vineyards [31] as well as studying generalized minimum spanning trees on random complexes [115]. Finally, persistence diagrams have recently also been included in deep network architectures. In [71], the authors use persistence landscapes as a layer in their network architecture with a similar optimization step, but limited to one dimensional signals. In other work [136, 19], persistence diagrams are used as features with learned weights, whereas we optimize the underlying filtration, i.e. the diagrams themselves change during optimization.

Continuity for Functional Maps Our main application lies in using persistence diagrams to provide a characterization of functional maps associated with continuous point-to-point correspondences and then proposing a practical optimization scheme that helps to improve the accuracy of functional map computations. The functional map representation and the associated correspodence computation pipeline was first introduced in [89] and has since then been extended significantly in, e.g., [51, 62, 103, 39] among many others (see [90] for an overview). The key practical advantage of this representation is that it allows to encode correspondences between shapes as small-sized matrices that represent linear transformations between function spaces in some reduced basis. Moreover, computing functional maps can be done efficiently by leveraging tools from numerical linear algebra and manifold optimization, as shown in, e.g., [62, 68, 69]. One challenge with this approach, however, is that the space of linear functional transformations is much larger than that of point-to-point correspondences, which means that in many cases regularization is necessary to compute accurate maps. This has prompted work on characterizing how various properties of pointwise maps are manifested in the functional domain. For example, the original work showed that area-preserving correspondences lead to orthonormal functional maps [89] (Theorem 5.1). Other characterizations have been shown for conformal maps [107, 51], isometries [89, 62] and for partial correspondences [103, 68]. More recently, some works have exploited the relation of point-to-point maps to functional maps that preserve pointwise products of functions [86, 85]. One large missing piece, however, is to characterize *continuous* point-to-point maps purely in the functional domain, and without making assumptions on the metric preservation. In this work, we fill this gap by precisely characterizing functional maps that arise from continuous point-to-point maps and propose an optimization scheme that allows to promote this continuity.

We also note that another commonly used relaxation for matching problems is based on the formalism of optimal transport, which has recently been used for finding bijective and continous correspondences [117, 72, 126]. These techniques have benefited from the computational advances in solving large-scale transport problems, especially using the Sinkhorn method under entropic regularization [32, 116]. For example, several recent methods in this category [72, 127, 126] have been proposed to efficiently find bijective maps, while promoting continuity by iteratively solving optimal transport problems. While related to our work, in these techniques, continuity is measured using point-to-point correspondence or via metric distortion, most commonly by minimizing variance with respect to a previous solution in an iterative scheme. On the other hand, we show that continuity can be expressed in the functional domain directly. Ultimately, we believe that the combination of these techniques, in the spatial and functional domains, can be especially beneficial in tackling difficult problems with strong outliers and discontinuities.

3.3 Overview

The rest of the chapter is organized as follows: in Section 3.4 we give a brief overview of persistence diagrams and their computation, while concentrating on the specific case of real-valued functions defined on surfaces. In Section 3.5 we describe a simple algorithm for computing the derivatives of persistence diagrams with respect to function values and outline how functions can be optimized for using various energies based on persistence diagrams. In Section 3.6 we characterize functional maps associated with continuous point-to-point maps and describe an approach to promote continuity directly in the functional domain. Finally, Section 3.7 is dedicated to experimental results and practical validation of our methods, while Section 3.8 concludes the chapter. with a summary, a description of limitations and future work.

3.4 Persistence Diagrams of Real-Valued Functions

We begin with a brief overview of persistence diagrams and their computation [21, 20]. We omit a formal introduction as much as possible, referring the reader to [36] for a more complete discussion. In a nutshell, persistent homology takes as input a sequence of topological spaces and tracks topological features, specifically homology groups, as they appear and disappear in the sequence. Homology groups, and likewise persistent homology groups are defined for different dimensions, up to the dimension of the underlying space, representing topological features of each dimension.

We concentrate on the 0-dimensional homology associated with real-valued functions defined on surfaces, as this is the setting that is most directly related to our practical scenarios. Namely, throughout our discussion we assume that the topological space is a surface $\mathcal{M}$, represented as an embedded discrete triangle mesh, consisting of N vertices. We also assume that we are given a function $f : \mathcal{M} \to \mathbb{R}$, which is represented as just a N-dimensional vector of real values. To each such function, it is possible to associate a *persistence diagram* of the super-level (resp. sub-level) sets of f, which intuitively

captures the number and the relative prominence of all the local maxima (resp. minima) of f. In practice, we define the function on the vertices and linearly interpolate it to the rest of the mesh. The resulting *super-level* set filtration is equivalent (up to homotopy) to the *upper-star filtration* (see [36, Chapter VI.3]). Throughout the rest of the chapter, for simplicity we only consider filtrations of this type, although our constructions are not restricted to this choice.

More formally, we consider the connected components of $f^{-1}[\alpha, \infty)$ at various values of parameter α. To construct the persistence diagram of a function f, we consider the evolution of the connected components of $f^{-1}[\alpha, \infty)$ as α ranges from ∞ to $-\infty$. The number of points in the persistence diagram equals the number of local maxima of f, i.e., vertices x, such that $f(x) \geq f(y)$ for all y adjacent (in the triangle mesh) to x. For each such vertex x, we will construct a point p in the persistence diagram having two coordinates: its birth and its death. The *birth* of a local maximum x equals simply to $f(x)$. To *death* of x equals the smallest real value β such that $f(x) \geq f(y)$ for all y in the same connected component as x in $f^{-1}[\beta, \infty)$. Note that unless $f(x)$ is the global maximum of f there will always exist some value β such that $f^{-1}[\beta, \infty)$ contains a vertex in the same connected component as x such that $f(y) > f(x)$. Thus, the largest value for which this occurs is called the death of x. By convention, we also declare the death of the global maximum of f on a compact surface, to be the global minimum of f.

The persistence diagram is simply the collection of birth and death times. Each pair defines a point p_i with b_i and d_i coordinates, representing the birth and death time respectively. The *persistence* of the point p in the diagram is half of the difference between its birth and death values which is also the L_∞ distance to the diagonal. In this chapter, it will be convenient to view the persistence diagram as a map which takes a topological space and a function to a multi-set of points in $\mathbb{R}^2$.

$$P_f : (\mathcal{M}, f) \to \{(b_i, d_i)\}_{i \in \mathcal{I}} \tag{3.1}$$

We use $\mathcal{I}$ to denote the index on the points in persistence diagram. In our case, this can simply be an index up to the number of maxima. We can assume that this number is finite as we deal with “nice" functions on, especially discrete, surfaces.

Figure 3.1 illustrates the persistence diagrams of two functions defined on a 1-dimensional interval. A key result in the theory of persistence is the stability of the diagrams under small perturbations of the function values [30, 24], which holds in great generality. Intuitively, given two functions f and g, the distance between their associated persistence diagrams $d(P_f, P_g)$ is bounded by the difference between the functions themselves. Numerous distances between persistence diagrams have been proposed. One of the most commonly used distances is the p-Wasserstein distance:

$$d_{W^p}(P_f, P_g) = \left(\inf_{\substack{\mu : P_f \to P_g \\ \mu \in \text{bijections}}} \sum_{x \in P_f} \|x - \mu(x)\|_p^p \right)^{1/p} \tag{3.2}$$

This distance is based on the optimal transport between the two diagrams, specifically between the points in the two diagrams. Taking the limit, $p \to \infty$, we recover the

bottleneck distance:

$$d_B(P_f, P_g) = \inf_{\substack{\mu: P_f \to P_g \\ \mu \in \text{bijections}}} \max_{x \in P_f} \|x - \mu(x)\|_\infty \tag{3.3}$$

This is the smallest cost (length of the longest edge) of the perfect matching between the points in P_f and P_g, where each point is also allowed to match with the diagonal. The classical stability theorem [30] states that $d_B(P_f, P_g) < \|f - g\|_{L^\infty} = \max_x |f(x) - g(x)|$. For the diagrams in Fig. 3.1, the smallest cost perfect matching would associate p with p', and q with q' while the remaining two red points would be matched with the nearest points on the diagonal. The cost (length of the longest edge) of this matching would be the (relatively small) distance between q and q', capturing the proximity of these two functions.

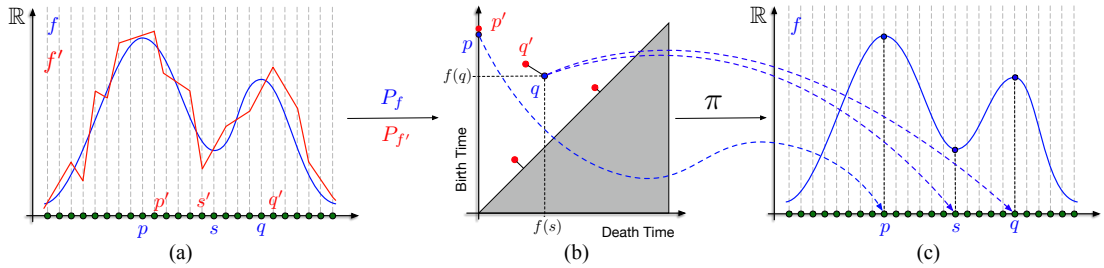


Figure 3.1 – (a) Two functions (f in blue and f' in red) defined on a 1-d interval. (b) Persistence diagrams of f, f'. (c) The map from the persistence diagram back to the underlying space for f.

Both the bottleneck and Wasserstein distances are realized by the matching μ between diagrams. This matching can be computed using the Hungarian algorithm, or any algorithm for computing the minimum weight matching of a bipartite graph.

3.4.1 Computation of Diagrams and their Distances

Efficiently computing persistence diagrams has been extensively studied, in e.g. [14] (see also [87] for a recent review of the state-of-the-art). The scalability and practicality of persistent homology computation has vastly increased over the last few years. Computing the 0-dimensional persistence diagram corresponding to the *upper-star filtration* of a piecewise-linear, real-valued function on a triangle mesh is particularly straightforward [38] and we include it here for completeness. The main steps of this computation are summarized in Algorithm 1. Given a function f, the persistence diagram is computed by first sorting the values of f and then processing each vertex x of the mesh in descending value of f. A new point in the persistence diagram is created whenever a new connected component of $f^{-1}[\alpha, \infty)$ appears, which occurs precisely when $f(x)$ is a local maximum. Otherwise, when two components are merged, the one associated with the smaller value of f dies, and is absorbed into the one associated with the larger value of f. This association between connected components and values of f is maintained in the data structure "parent", which points, for each vertex of the mesh, to the local maximum of f

connected to this vertex and having the highest value. Note that line 8. of Algorithm 1 can be implemented efficiently by using a Union-Find data structure, which only requires considering the immediate neighbors w of x. As a result the complexity of entire algorithm is $O(N \log N + N\alpha(N))$, with the former part arising from the sorting of f, while the latter corresponds to processing the vertices and maintaining the association between them and the local maxima, and $\alpha(N)$ is the inverse Ackermann function.

Input: Triangle mesh M, real-valued function f.
Output: Persistence diagram P_f of f.
1 **Initialization:** sort f in descending order.;
2 **for** *each vertex x, in descending order of $f(x)$* **do**
3 **if** *x is local maximum of f in M* **then**
4 add new point to P_f with birth value $f(x)$;
5 parent$[x] = x$.
6 **else**
7 let y be s.t. $f(y)$ is maximal among all parent$[w]$, w adjacent to x and $f(w) > f(x)$;
8 update parent$[w] = y, \forall\ w$ in the same connected component as x of $f^{-1}[f(x); \infty)$;
9 **if** *parent$[w]$ changed in the previous step* **then**
10 set death of w in P_f to $f(x)$.
11 **end**
12 **end**
13 **end**
14 set death of global max of f in P_f to global min of f.

Algorithm 1: Computing the persistence diagram of a function f on a triangle mesh.

A key aspect of this computation is that for a vertex x that is a local maximum of f, its birth value equals $f(x)$, whereas its death value equals to the value $f(w)$ of some "paired" vertex w which, when processed, merges the connected component of x with that of some other vertex y, where $f(y) > f(x)$. Therefore, when all values of f are distinct, if f is perturbed infinitesimally, the value of the point p in the persistence diagram will change by the amount related to the change at the local maximum and its paired vertex. In the following section, we describe this intuition in detail.

Let us also note that the persistence diagrams of two functions can be compared even if the functions are defined on different domains. In other words, when going from the function f to its diagram P_f all information about the underlying domain on which f is defined is removed. This can be useful in our applications, where we will use persistence diagrams as a way to compare and align functions defined on different domains. Finally, we note that persistence diagrams are *provably stable* under perturbations of the function values [30], which has motivated their use in many settings, including shape analysis, e.g., [114, 66].

3.5 Derivatives and Optimization of Persistence Diagrams

We now derive how the persistence diagram changes as we change the underlying functions. Our approach is based on the observation that persistence diagrams can be thought of, in the simplest case, as extensions of the max operator. We follow the general common approach used to compute derivatives of such operators, for example in the case of max pooling in convolutional neural networks [46]. These derivatives are most often discontinuous, but we show that generically, they are locally well defined, and, just like the max operator, can be successfully optimized for within more complex energies in practice. Finally, we note that all the results in this section can be applied to persistence diagrams of any dimension (as well as to any functionals of persistence diagrams).

Our goal is to understand how the diagram changes as we change the function. Let the filtration function f, depend on a set of parameters which we denote by α_j. To optimize these parameters, we must derive formulas for

$$\frac{\partial b_i}{\partial \alpha_j} \qquad \text{and} \qquad \frac{\partial d_i}{\partial \alpha_j},$$

for all points in the diagram (b_i, d_i). Our key tool is the existence of a map π from the points in the diagram to pairs of vertices in the space, i.e.

$$\pi : (b_i, d_i) \mapsto (v_b, v_d)$$

We first define a possibly non-unique map from points in the persistence diagram to pairs of simplices

$$\zeta : (b_i, d_i) \mapsto (\sigma, \tau)$$

A finite simplicial filtration can always be extended to a *total order*. For example, one way is to consider lexographical ordering of the vertices to order simplices which share the same function value. In the total ordering, each birth and death time are distinct, hence there is precisely one simplex which corresponds to any birth time (and precisely one simplex which corresponds to a death time respectively). Note that these times refer to the index in the total order. This correspondence defines ζ, which is simply the pairing returned by the standard persistence algorithm [21]. In our case, each birth time corresponds to the value of a specific vertex, whereas the death time is associated with the edge of the mesh, which merges two connected components when it is added.

Since we are considering a super-level set (upper-star) filtration, edges are added implicitly in the filtration, once both vertices are included. This allows us to define a map from each simplex to one of its vertices. In the case of vertices, this is simply the vertex itself, whereas in the case of edges, it is given by the vertex with the smaller function value:

$$\eta : \sigma \mapsto v_\sigma, \text{ where } \eta(\sigma) = \operatorname*{argmin}_{v \in \sigma} f(v)$$

In general this is not unique, but we can always choose an arbitrary fixed vertex. Finally, we define:

$$\begin{cases} \pi_b = \eta \circ \zeta(b_i) \\ \pi_d = \eta \circ \zeta(d_i) \\ \pi = (\pi_b, \pi_d) \end{cases}$$

Algorithm 1 implicitly computes this map (see Section 3.5.2 for details). This map depends on numerous choices, however as we saw in the previous section, we can assume that the function values are unique at each vertex. This can be achieved through an infinitesimal perturbation of the function, either implicitly or explicitly. Finally, while π is defined for each point in the diagram, by applying it to *every point*, we can induce a map from a diagram to a multi-set of vertices. We then obtain the following two results.

Lemma 3.5.1 *If the vertex function values are distinct, then π is unique.*

Proof As all critical points in a super-level set filtration occur at vertex values, there must be a unique vertex v corresponding to each homological critical value (i.e. any function value where the topology changes). Since the coordinates of each point in the persistence diagram correspond to a critical value the map $\pi(b_i, d_i) = (v_b, v_d)$ is unique. □

Note that this does not imply that there is a one-to-one correspondence between points in the diagram and pairs of vertices, as multiple points can map to the same vertex. However, we do have the following result.

Lemma 3.5.2 *If the vertex function values are distinct, there exists a neighborhood where π is constant.*

Proof Let ε denote the minimum separation between vertex function values. First note that by the assumption of distinct function values, there is a total ordering on the vertices. For any function within a $\varepsilon/2$ ball, i.e.

$$|f(x) - f'(x)| < \varepsilon/2 \qquad \forall x \in \mathcal{M}$$

this ordering remains the unchanged. This immediately implies that the map η from each simplex to a vertex is constant in this neighborhood. We first formally define the δ-neighborhood of a persistence diagram P_f. Recall that by Eq.3.1, a persistence diagram is a map from a topological space endowed with a real valued function (X, f) to a multi-set of points. Note that the map is completely determined by the space and the function. Therefore, we define a δ-neighborhood of P_f as all P_g of the form (X, g) such that $||f - g||_\infty$.

We next show that ζ is constant in a small enough neighborhood. Although ζ is defined per point in the diagram, this induces a map from a diagram to some collection of simplicies in the space. If the map is constant per point, it is constant in the neighborhood defined above.

In an upper-star filtration, the total ordering on the vertices can also be extended to the remaining simplices. Each vertex v has a set of simplices for which it determines the function value when the simplex enters the filtration. This is given by the preimage $\eta^{-1}(v)$. Generically, the preimages are disjoint sets of simplices, i.e. no simplex maps to two vertices. As η is constant in some neighborhood, this implies that the map ζ can also be chosen so that it is constant in this neighborhood. Each preimage can be extended to a total order independently. As the sets of preimages do not change, this extension can be kept constant. Finally, we note that the composition of two constant maps is again constant completing the proof. □

To define the neighborhood recall that we can consider the persistence diagram as a map from a pair (X, f) to a multi-set of points (Eq. 3.1). The neighborhood is defined as all the diagrams with the domain (X, g) such that $||f - g||_\infty$ is sufficiently small. We note that stability [30] implies that the corresponding multi-sets of points are close with respect to the bottleneck distance.

The existence of this neighborhood allows us to derive analytic expressions the derivatives. First, we observe that

$$f(\pi_b(b_i)) = b_i \qquad f(\pi_d(d_i)) = d_i \qquad \forall i \tag{3.4}$$

Since π is constant, it follows that

$$\frac{\partial b_i}{\partial \alpha_j} = \frac{\partial f(\pi_b(b_i))}{\partial \alpha_j} = \frac{\partial f(v_i)}{\partial \alpha_j} = \frac{\partial f}{\partial \alpha_j}(v_i) \tag{3.5}$$

In other words, ***the derivative is equivalent to the derivative of the function evaluated at the image of the map*** π_b. The derivation for d_i is analogous.

3.5.1 Application to optimization

In our application we would like to minimize some functional $\mathcal{F}$ of an input diagram. Using the the chain rule, we obtain

$$\begin{aligned}\frac{\partial \mathcal{F}}{\partial \alpha_j} &= \sum_{i \in \mathcal{I}} \frac{\partial \mathcal{F}}{\partial b_i} \cdot \frac{\partial b_i}{\partial \alpha_j} + \frac{\partial \mathcal{F}}{\partial d_i} \cdot \frac{\partial d_i}{\partial \alpha_j} \\ &= \sum_{i \in \mathcal{I}} \frac{\partial \mathcal{F}}{\partial b_i} \cdot \frac{\partial f}{\partial \alpha_j}(\pi_b(b_i)) + \frac{\partial \mathcal{F}}{\partial d_i} \cdot \frac{\partial f}{\partial \alpha_j}(\pi_d(d_i))\end{aligned}$$

$\frac{\partial \mathcal{F}}{\partial x_i}$ and $\frac{\partial \mathcal{F}}{\partial y_i}$ must be derived for each functional separately. We derive the complete formula for two specific cases. We first consider the bottleneck distance.

Lemma 3.5.3 *For almost all persistence diagrams P_f and P_g, the point $x \in P_f$ whose matching achieves the bottleneck distance is unique and constant in some ε-neighborhood of P_f (in the bottleneck metric).*

Proof Assuming generic P_f and P_g all pairwise distances between points in P_f and P_g are unique. This implies that the maximum of any matching is unique. This proves the first part of the lemma. As the values are distinct, it follows that there exists a neighborhood such that the pair of points achieving the maximum is constant. Finally, the continuity of persistence diagrams implies that any point which is sufficiently close to to the diagonal cannot achieve the bottleneck distance and so cannot change the matching. □

The neighborhood here again refers to the space of persistence diagrams (here we do not need to restrict the space of diagrams as in Lemma 3.5.2). If P_f and P_g are generic, the matching is unique in a sufficiently small neighborhood of nearby diagrams. Since nearby

diagrams may have different numbers of points, one needs to be careful to ensure the lemma holds (see Appendix).

We consider the functional of the bottleneck distance to a fixed diagram, denoted by $\mathcal{F}_B$. Let $p_{max} = (b_{max}, d_{max})$ be the point in the diagram whose matching realizes the maximum in the bottleneck matching μ. If $p_i = p_{max}$ and $|b_{max} - \mu(b_{max})| > |d_{max} - \mu(d_{max})|$

$$\frac{\partial \mathcal{F}_B}{\partial b_i} = \begin{cases} -1 & b_i > \mu(b_i) \\ 1 & b_i < \mu(b_i) \end{cases}$$

and 0 otherwise. A similar expression holds for $\frac{\partial \mathcal{F}_B}{\partial d_i}$ if $|b_{max} - \mu(b_{max})| < |d_{max} - \mu(d_{max})|$. Note that if both are equal then the distance is 0, where the derivative can be defined as 0 (since the distance cannot be negative). Therefore, the derivative can be written as two cases: if $|b_{max} - \mu(b_{max})| > |d_{max} - \mu(d_{max})|$, then

$$\frac{\partial \mathcal{F}_B}{\partial \alpha_j} = \text{sgn}\,(\mu(b_{max}) - b_{max}) \frac{\partial f}{\partial \alpha_j}(\pi_b(b_{max})) \tag{3.6}$$

and if $|b_{max} - \mu(b_{max})| < |d_{max} - \mu(d_{max})|$,

$$\frac{\partial \mathcal{F}_B}{\partial \alpha_j} = \text{sgn}\,(\mu(d_{max}) - d_{max}) \frac{\partial f}{\partial \alpha_j}(\pi_d(d_{max})). \tag{3.7}$$

We also derive an equivalent result for the squared 2-Wasserstein distance. Again, we assume by genericity that the solution to the matching is unique and constant in a neighborhood. The squared 2-Wasserstein distance is given by

$$\begin{aligned} d^2_{W^2}(P_f, P_g) &= \inf_{\substack{\mu: P_f \to P_g \\ \mu \in \text{bijections}}} \sum_{p \in P_f} \|p - \mu(p)\|_2^2 \\ &= \inf_{\substack{\mu: P_f \to P_g \\ \mu \in \text{bijections}}} \sum_{p \in P_f} (p_b - \mu(p_b))^2 + (p_d - \mu(p_d))^2 \end{aligned}$$

Minimizing this distance gives the functional, $\mathcal{F}_{W^2}$, whose derivative is

$$\begin{aligned} \frac{\partial \mathcal{F}_{W^2}}{\partial b_i} &= \frac{\partial}{\partial b_i} \left(\sum_{i \in \mathcal{I}} (b_i - \mu(b_i))^2 + (d_i - \mu(d_i))^2 \right) \\ &= 2(b_i - \mu(b_i)) \end{aligned}$$

Putting this together,

$$\frac{\partial \mathcal{F}_{W^2}}{\partial \alpha_j} = 2 \sum_{i \in \mathcal{I}} (b_i - \mu(b_i)) \frac{\partial f(\pi_b(b_i))}{\partial \alpha_j} + (d_i - \mu(d_i)) \frac{\partial f(\pi_d(d_i))}{\partial \alpha_j} \tag{3.8}$$

where b_i and d_i are functions of α_j (see Eq. 3.4). We conclude this section by noting that most functionals for persistence diagrams map points in $\mathbb{R}^2$ to real valued functions. For most cases, it should be possible to compute a closed form for the derivative, for example in the case of persistence landscapes or specific choices of kernels on the space of persistence diagrams.

3.5.2 Computing Derivatives

The derivatives derived above can be efficiently computed by appropriately modifying Algorithm 1. We first need to compute π, the map from the persistence diagram to the vertices. Let p be the point added to P_f in line 4. When the point is created, we also add the pair (p, x) to π. Likewise in line 10, we again add the pair (p, x) to π. Note that these are two different vertices, as the x in Line 4 is a local maxima, while the x in line 10. connects two previously disjoint components (i.e. in our case a saddle point).

To compute the derivative, we must be able to evaluate the derivative of the function at a given vertex in the underlying space, i.e. $\mathcal{M}$. The derivative can be obtained by evaluating the derivative of the function at the image of π for that point. Therefore, by using the modified Algorithm 1 and computing the minimum weight matching μ with respect to the chosen distance, we can evaluate π and μ for every point in the persistence diagram P. For the bottleneck distance, we can then directly compute the derivative by evaluating Equations 3.6 or 3.7. The derivative for the 2-Wasserstein distance can likewise be computed by evaluating Equation 3.8.

In our derivation, we assumed a genericity condition by requiring the vertex function values be unique. While this is true for any one function (by infinitesemal perturbation), as we optimize the function, this assumption might be violated. However, in the case of non-uniqueness, we can make a choice randomly. In our implementation, the continuous optimization scheme such as L-BFGS explicitly checks that the energy decreases at every iteration, ensuring that the algorithm converges. Furthermore, in our applications we optimize over a reduced functional basis consisting of smooth functions, which significantly improves the robustness with respect to pathological behavior. In practice, the optimization schemes we use converge to local minima remarkably consistently. Nevertheless, we leave the theoretical study of guarantees of convergence as interesting future work.

3.6 Applications: Continuity in Functional Maps

We propose to apply the ideas presented above for improving functional maps. For this, we first provide a novel characterization of functional maps arising from *continuous* point-to-point maps, purely in the functional domain, i.e., without making reference to point-to-point correspondences. We also prove that our characterization is complete: i.e., all functional maps (linear transformations across function spaces) that satisfy our condition exactly must arise from continuous point-to-point maps. We then present an energy, which can be optimized using standard non-linear continuous optimization techniques, such as L-BFGS, and which promotes continuity directly in the functional domain, even when maps are expressed in a reduced basis. In this section, we describe these constructions in detail, and present the main experimental results in Section 3.7.

Our main observation is that the information encoded in persistence diagrams can be used to improve the computation of correspondences between shapes using the so-called functional map framework (see [90] for an recent overview). Works in this domain are based on the idea that when looking for a point-to-point map $T : \mathcal{M} \to \mathbb{N}$ between two shapes $\mathcal{M}$ and $\mathbb{N}$, it is often easier to consider the associated pull-back of real-valued

functions $T_F : \mathcal{F}(\mathbb{N}) \to \mathcal{F}(\mathcal{M})$, where $\mathcal{F}(\mathbb{N})$ is the space of real-valued functions on shape $\mathbb{N}$. For a given map T, T_F is defined simply via composition $T_F(f) = f \circ T$, where f is any real valued function $f : \mathbb{N} \to \mathbb{R}$. Since T_F is a linear operator between function spaces, whenever both the domain and range can be endowed with a functional basis, this operator can be written as a change of basis matrix $\mathbf{C}$, also called a functional map matrix, whose size depends on the size of the chosen basis.

The basic pipeline for computing functional maps introduced in [89], and extended in follow-up works, aims at recovering the functional map matrix $\mathbf{C}$ directly and then using it to estimate the underlying map T. The general approach follows the following steps (see Section 2.4.4 in [90]):

1. Construct a set of basis functions, consisting e.g. of the eigenfunctions corresponding to the $k_{\mathcal{M}}$ and $k_{\mathbb{N}}$ smallest eigenvalues of the Laplace-Beltrami (LB) operators on the source and target shapes $\mathcal{M}$ and $\mathbb{N}$, stored in matrices $\Phi^{\mathcal{M}}$ and Φ^{N} respectively.

2. Compute some k_d descriptor functions on the source and target shapes, express them in the corresponding bases and store the coefficients as columns of matrices $\mathbf{A}$ and $\mathbf{B}$, of size $k_{\mathcal{M}} \times k_d$ and $k_{\mathbb{N}} \times k_d$ respectively.

3. Compute the optimal *functional map* matrix $\mathbf{C}$ of size $k_{\mathbb{N}} \times k_{\mathcal{M}}$, that aligns the descriptor functions and commutes with the Laplace-Beltrami operators on the two shapes. That is: $\mathbf{C} = \text{argmin}_{\mathbf{X}} \|\mathbf{X}\mathbf{A} - \mathbf{B}\| + \epsilon \|\mathbf{\Lambda}^{\mathbb{N}}\mathbf{X} - \mathbf{X}\mathbf{\Lambda}^{\mathcal{M}}\|$. Alternatively, one can use several extensions, such as manifold-constrained optimization [62], robust regularization [69] or requiring the map to commute with multiplicative operators with respect to descriptor functions [86] among others.

4. Convert the functional map $\mathbf{C}$ to a point-to-point map.

A key advantage of the functional map representation is that the optimization step 3. can be solved efficiently using robust numerical linear algebra tools, and in the most basic case reduces to solving a simple linear system of equations. Unfortunately, without additional regularization, the computed functional map might not correspond to any "natural" point-to-point map. Therefore, several approaches have been proposed to enforce desired properties on functional maps. For example, it was shown in the original article [89] that functional maps arising from area-preserving point-to-point maps must be orthonormal.

More recently several works have used the fact that functional maps that correspond to pointwise maps must also preserve pointwise products between functions [86]. In practice, however, we are often interested in *continuous* pointwise correspondences and translating the continuity of the point-to-point map into a constraint on the functional map has not been straightforward. To this end, we first establish and then exploit the following theorem.

Theorem 3.6.1 *An invertible linear functional map T_F corresponds to a* continuous *bijective point-to-point map* if and only if *both T_F and its inverse preserve pointwise*

products of pairs of functions, and moreover both T_F and its inverse preserve the persistence diagrams of all real-valued functions. In other words:

$$d_B(P_f, P_{T_F(f)}) = 0, \ \forall \ f. \tag{3.9}$$

Note that preservation of products ensures that T_F corresponds to a point-to-point map, whereas preservation of persistence diagrams guarantees that the underlying map is continuous.

Proof Given a continuous bijection between two topological spaces, the induced pull-back of real-valued functions must clearly preserve products of functions. Similarly it must preserve persistence diagrams of real-valued functions since both the topological structure of the space and function values are preserved.

Conversely, consider any invertible linear functional map T_F. It is well-known that if T_F preserves pointwise products of functions, then it must correspond to a pull-back by a point-to-point map (e.g. corollary 2.1.14 in [112]). Moreover, since T_F is invertible, its inverse must also satisfy this property, meaning that the underlying point-to-point map is a bijection.

Finally, by assumption both T_F and its inverse preserve persistence diagrams of real-valued functions. Now, consider an indicator function of a region. By definition, its persistence diagram will have a unique local maximum, if and only if the region is connected. Thus preservation of persistence diagrams implies that the underlying point-to-point correspondence (and its inverse) maps connected regions to connected regions. Finally, a bijective map between two topological spaces is continuous if and only if both it and its inverse map connected sets to connected sets (see e.g. [123]). □

The key observation in the proof of Theorem 3.6.1 is that a functional map associated with a point-to-point bijection will correspond to a pull-back by a *continuous* point-to-point map if it maps indicators of functions of connected regions to indicators functions of connected regions. Moreover, persistence diagrams provide a very convenient way to test whether an indicator function corresponds to some connected region, by simply checking whether it has more than one prominent local maximum. The persistence of local maxima gives a stable way to test for this criterion for an *arbitrary* (not necessarily binary, or even positive) function, since an indicator function of a connected region, perturbed by noise, will still have a single prominent local maximum, with other points close to the diagonal on the persistence diagram. This stability directly follows from the stability guarantees of persistence diagrams, which have been established under very broad conditions [30, 24]. Therefore, we propose to exploit this observation by defining the following (non-linear) energy on functional maps, expressed in an arbitrary basis:

$$E_{\text{cont}}(\mathbf{C}) = \sum_r d_B(P_{\Omega_r}, P_{\mathbf{C}(\Omega_r)}), \tag{3.10}$$

where r is an index over some set of connected regions defined on the source shape $\mathcal{M}$, Ω_r is the characteristic (indicator) function of region r, and $\mathbf{C}(\Omega_r)$ is the image of indicator function onto the target shape $\mathbb{N}$ via the functional map C. For simplicity we write $\mathbf{C}(\Omega_r)$

instead of the expression in the reduced basis, which should read $\mathbf{\Phi}^{\mathbb{N}}\mathbf{C}((\mathbf{\Phi}^{\mathcal{M}})^{+}\Omega_r)$, where $+$ is the pseudo-inverse.

Of course, a symmetric energy can be optimized for using a map between the target and source shape.

With this energy at hand, Theorem 3.6.1 can be restated simply to say that a bijective point-to-point map T is continuous if and only if both the pullback T_F by T and its inverse satisfy $E_{\text{cont}}(T_F) = 0$, assuming the sum in Eq. 3.10 is over *all* connected regions r.

Intuitively, E_{cont} measures the uncertainty in converting a functional map to a point-to-point map. For example, the image of an indicator function of a connected region or even a delta function via a computed functional map can have multiple prominent local maxima, which means that during the point-to-point conversion, the correspondence can oscillate between multiple possible solutions creating a highly discontinuous pointwise map. This can especially occur when a functional map corresponds to a blending of multiple pointwise maps, which can arise e.g. due to the symmetry present in the shapes. On the other hand, when optimizing for functional map using the energy in Eq. 3.10 we expect that the image of an indicator function of a region corresponds to a function with exactly one prominent maximum, which eventually should lead to a more continuous point-to-point map. In practice, we observe that it can be more efficient to replace E_{cont} with a simplified energy:

$$E_{\text{persist}}(\mathbf{C}) = \sum_r \text{Pers}(P_{\mathbf{C}(\Omega_r)}), \tag{3.11}$$

where the sum is again over connected regions r on the source shape and $\text{Pers}(P)$ is defined as the sum of the squares of persistence values (i.e., the distances to the diagonal) of all points in the diagram P, except for the one with the highest persistence. Intuitively, $\text{Pers}(P_f)$ penalizes all but the prominent local maxima of the function f, with the strength equal to the square of the distance to the diagonal.

Our main objective therefore is to use the energy $E_{\text{persist}}(\mathbf{C})$ to optimize a functional map $\mathbf{C}$ and especially to use it to infer functional maps that arise from *continuous* point-to-point maps. For this, we observe that the ability to differentiate persistence diagrams allows us to compute the gradient of $E_{\text{persist}}(\mathbf{C})$ with respect to the entries of the matrix $\mathbf{C}$. This, in turn, allows us to use quasi-Newton methods such as BFGS, which, as we show below, provide an efficient way to improve functional maps. Specifically, we solve the following continuous optimization problem:

$$\min_{\mathbf{C}} \; E_{\text{persist}(\mathbf{C})}, \tag{3.12}$$

where the functional map is represented in the reduced LB basis. In general, this is highly non-convex problem, and we use the re-computed functional map as an initialization in an iterative quasi-Newton scheme. For robustness, we also add a constraint that the functional map $\mathbf{C}$ should map the indicator (constant $= 1$) function of the source shape to the indicator on the target, and use a projected L-BFGS solver to optimize Eq. (3.12).

The derivative in this case is a special case of the 2-Wasserstein case in Equation 3.8. If we match to a diagram with only one point corresponding to a global maximum, all the points map except the global maximum map to the diagonal so

$$\mu(b_i) = \mu(d_i) = \frac{b_i + d_i}{2}$$

Substituting into Equation 3.8, the derivative can be written as

$$\frac{\partial E_{\text{persist}}}{\partial \alpha_j} = \sum_{i \in \mathcal{J}} (d_i - b_i) \left(\frac{\partial f}{\partial \alpha_j}(\pi_d(d_i)) - \frac{\partial f}{\partial \alpha_j}(\pi_b(b_i)) \right)$$

where $\mathcal{J}$ represents all the points in the diagram except the most persistent one.

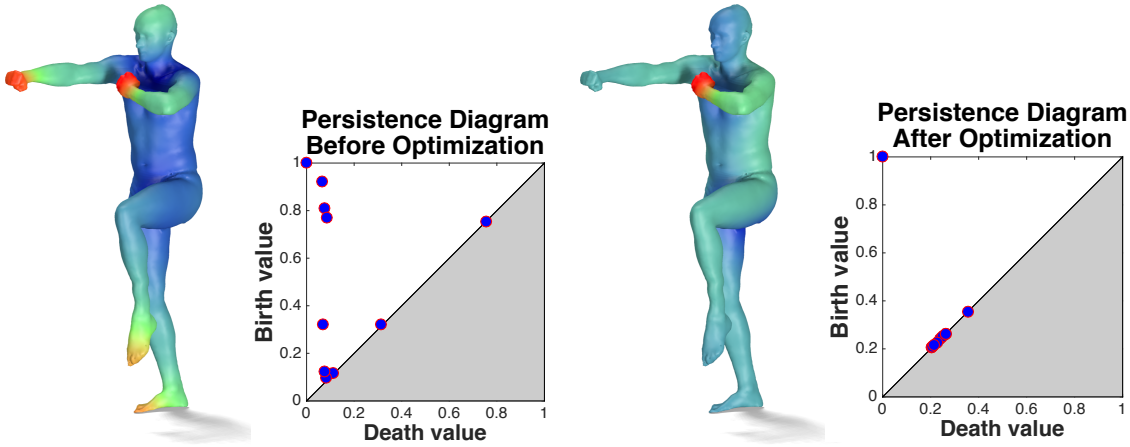
3.7 Experimental Results

In this section, we describe some practical aspects of the implementation of our topological optimization approach and show results in several applications.

Throughout all of our experiments we assume that shapes are represented as triangle meshes, and functions are defined on their vertices. We also use the standard discretization of the Laplace-Beltrami operator $L = A^{-1}W$, where A is the lumped area matrix and W is the matrix of cotangent weights [94, 80]. We use this operator to define a basis for real-valued functions by computing the eigenfunctions corresponding to the k smallest eigenvalues, $L\varphi_i = \lambda_i \varphi_i$. Then we express any real-valued function as a linear combination $f = \sum_{i=1}^{k} a_i \varphi_i$, where a_i are scalar weights. Thus, when optimizing for a function f, we consider the scalar weights a_i as the unknowns.

3.7.1 Topological Function Simplification

To illustrate the effect of our topological optimization, we first use it to "simplify" a real-valued function on a surface, by removing all but one most prominent local maxima. Namely, we first construct a real-valued function on one of the shapes from the SCAPE dataset [4] by computing the Heat Kernel Signature (HKS) [119] for a $t = 0.01$. This function and its associated persistence diagram are shown in Figure 3.2a. We then simplify this function by minimizing an energy, which penalizes the square of the persistence of all but first most prominent local maxima (which is equivalent to $\text{Pers}(P_f)$ introduced in Eq. 3.11 above), using $k = 40$ eigenfunctions of the Laplace-Beltrami operator as the basis for the function. The result of the optimization is shown in Figure 3.2b. Note that the resulting function has only one prominent local maximum. Remark also that our optimization does not impose the *location* of the final local maximum, but simply promotes its uniqueness. Finally, we note that our topological optimization does not assume the manifold structure of triangle meshes, and, once the functional basis is computed, can be applied to arbitrary graphs. The optimization converges after 81 iterations of L-BFGS in 2.26 seconds on a machine with 2.6 GHz Intel Core i7 CPU using a MATLAB implementation of L-BFGS with the analytic gradient of persistence diagrams described in Section 3.5.



(a) Function before optimization (b) Function after optimization

Figure 3.2 – Topological optimization for removing local maxima from a function: (a) a real-valued function on a surface and its persistence diagram, (b) function and its diagram after topological optimization, where we penalize the persistence of all, except the first most prominent maximum.

3.7.2 Topological Function Alignment

Next, we illustrate how our topological function optimization can be used to align the values of two functions defined on different domains without the knowledge of any (functional or point-to-point) correspondences. For this, we first compute two functions, f_1, f_2 corresponding to the HKS for the same value $t = 0.01$ on two different shapes, shown in Figure 3.3 (top, left and middle).

Note that since the shapes is not fully intrinsically symmetric, the function f_2 has two prominent local maxima on the hands that have different values, while the two most prominent maxima of f_1 are closer together, since since the undeformed shape is closer to being symmetric. We then modify f_2 so that the bottleneck distance $d_B(P_{f_1}, P_{f_2})$ is minimized, while keeping f_1 fixed. Let us stress that this energy does not require a map between the two domains, and we can optimize f_2 so that its persistence diagram aligns with that of f_1 by only considering the diagrams themselves. The result of this optimization, f_2^{opt} is shown in Figure 3.3 (right), where both the function becomes symmetric and the persistence diagrams align nearly perfectly. In this case, the optimization is done again, using $k = 40$ eigenfunctions of the Laplace-Beltrami operator, converges after 29 iterations of L-BFGS and takes 0.84 seconds.

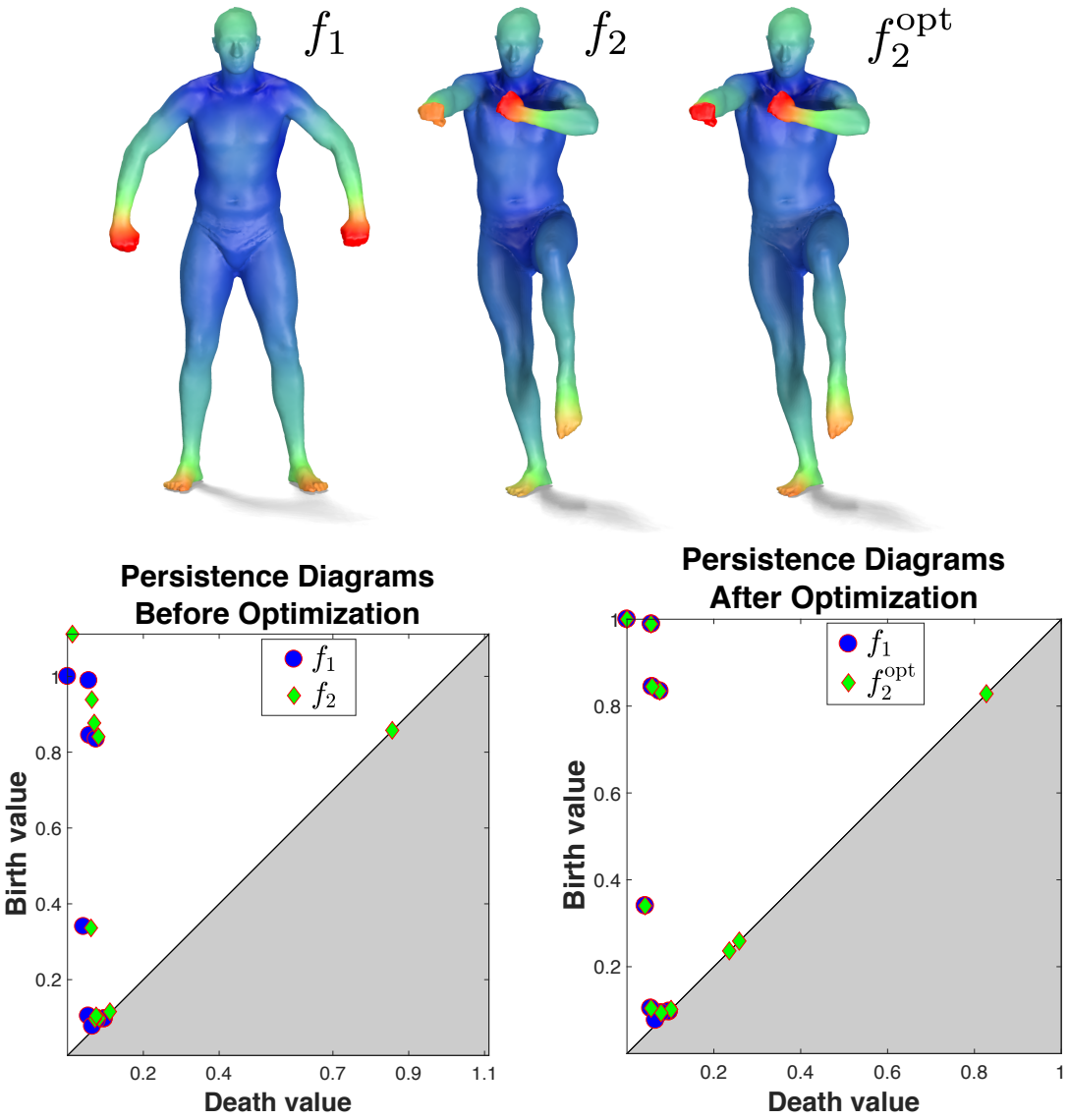
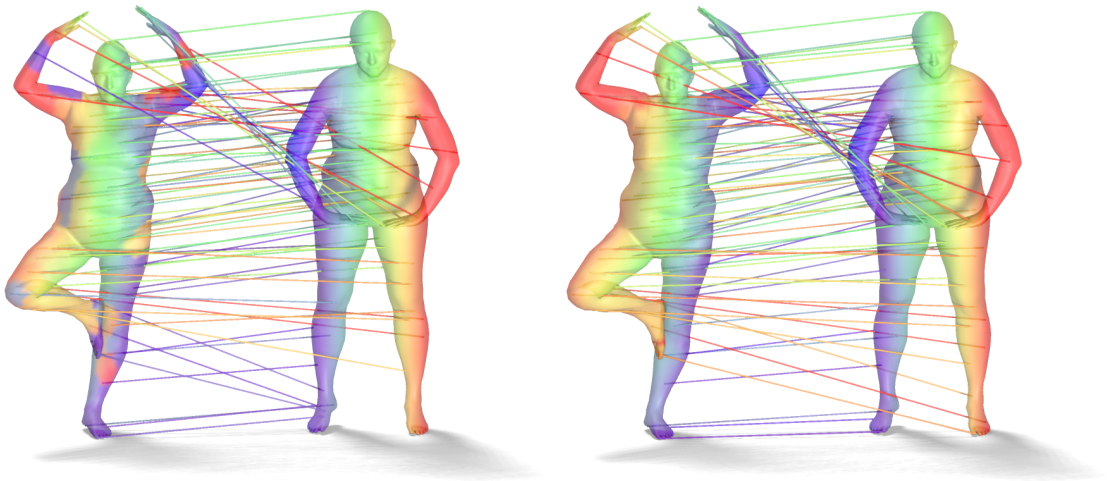


Figure 3.3 – Topological function alignment. Left and middle: initial functions f_1, f_2 on two shapes (top), and their persistence diagrams (bottom). Right: function f_2^{opt} on the second shape after aligning its persistence diagram with that of f_1, without any cross-shape correspondence (top) and the resulting aligned diagrams (bottom). Note the change in values on the hands.

3.7.3 Functional Map Improvement

In our next experiment, we show how topological function optimization can be used to improve functional maps and to promote the continuity of the recovered point-to-point maps directly in the functional space, without enforcing conditions on area preservation or conformality. For this, we first consider a noisy point-to-point map, which is obtained by a strong perturbation of the symmetric correspondence (i.e., mapping left to right)

between a pair of shapes from the FAUST dataset [12]. The initial correspondence is shown in Figure 3.4a. We then convert this map to a functional map representation using $k = 80$ eigenfunctions of the Laplace-Beltrami operator, resulting in a matrix $\mathbf{C}$ of size 80×80. We then optimize this functional map using L-BFGS on the energy E_{persist} described in Eq. 3.11. To construct the connected regions, required in the sum in Eq. 3.11, we randomly sample n points on the source shape and construct their intrinsic Voronoi diagram. We then use an iterative scheme, where we optimize a functional map with an increasing number of regions $n = 5, 15, ..., 45$. Between iterations we project a functional map to a point-to-point one to remain close to the desired solution space. This procedure is reminiscent to the ICP refinement approach proposed in the original functional maps work [89], but instead of promoting area preserving maps, it aims to promote continuous maps, without enforcing any constraints on the metric distortion.



(a) Initial correspondence (b) Correspondence after optimization

Figure 3.4 – (a) Noisy input functional map converted to a pointwise correspondence (b) Same map after topological optimization. Note the drastic reduction in discontinuities. Colors encode the x coordinate function of the target shape and its pull-back on the source.

The final functional map, converted to a point-to-point map, is shown in Figure 3.4b. Note that the resulting map does not have the large patch discontinuities present in the original one. We also evaluated the quality of the initial and optimized maps with respect to the ground truth map. Figure 3.5 shows the percent of point-to-point correspondences below a certain threshold, following the evaluation protocol introduced in [56]. We also plot the errors of in the ground truth map, which are caused by its representation as a reduced-size functional map. Note the significant improvement of the map after topological optimization, especially with respect to large errors, which are eliminated by the topological optimization.

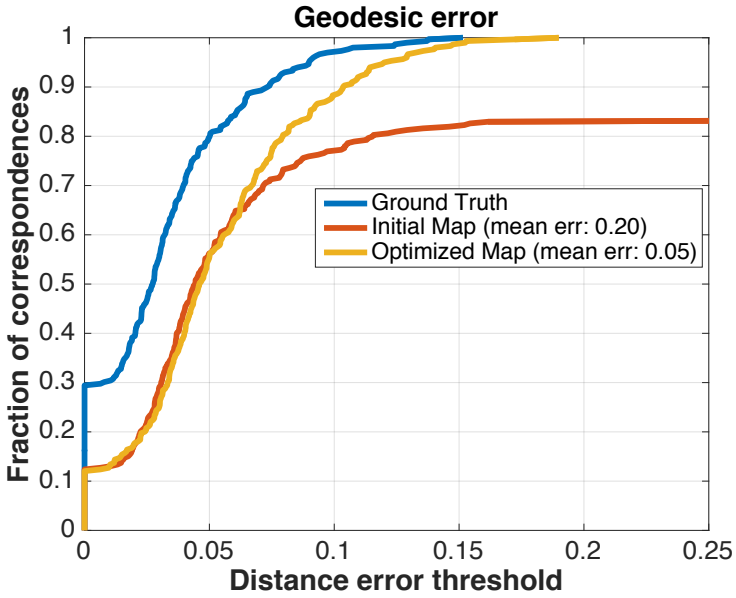
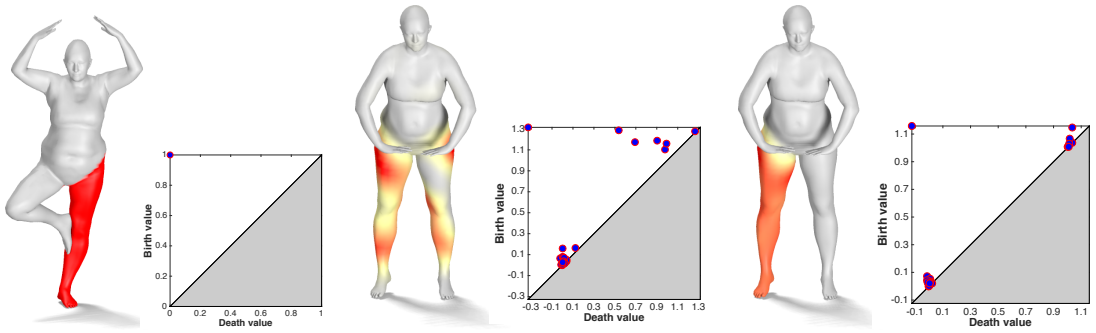


Figure 3.5 – Quantitative evaluation of the functional maps before and after topological optimization, shown in Fig. 3.4.

Finally, to illustrate the effect of the optimization, we consider the indicator function of a connected region on the source shape, and its associated persistence diagram, shown in Figure 3.6a. We then show its image on the target shape via the initial functional map in Figure 3.6b, which contains multiple significant local maxima. Finally, we show the image of the same function onto the target but via the functional map after our optimization in Figure 3.6c. Note the intuitive "connectivity" of this function and its unique prominent local maximum.



(a) Indicator function of a region (b) Image before optimization (c) Image after optimization

Figure 3.6 – Topological optimization for improving functional maps: (a) indicator function of a region on a source shape and its persistence diagram, (b) the image of this function via the initial functional map onto the target shape and its persistence. Notice multiple prominent local maxima. (c) Image of the same function after optimizing the functional map.

3.7.4 Improvement of Computed Functional Maps

We also use our topological optimization procedure to improve functional maps computed using descriptor preservation constraints, as described in [86]. For this, we consider 100 pairs of shapes, taken at random from the FAUST dataset [12], and for each shape pair introduce one landmark point (in the middle of the right leg). We then compute descriptor functions, which consist of Wave Kernel Signatures [7] and Wave Kernel Map (to enforce the landmark) sampled at 20 different times. Finally, we compute the functional map using a linear system of equations based on multipliciative operators described in [86]. We then optimize the resulting functional maps using our topological optimization approach by minimizing Eq. 3.11. During topological optimization, we again use an iterative procedure where we sample 5 points and compute their intrinsic Voronoi diagram. We then optimize Eq. 3.11 for 12 iterations of L-BFGS and convert the optimized functional map to a point-to-point one. We found that the last step helps to improve the quality of the map, as it restricts the optimization from drifting too far from point-wise maps. An alternative would be, e.g., to enforce preservation of pointwise products of functions, as suggested in [86]. Figure 3.7 shows the quantitative evaluation of the functional maps converted to a point-to-point maps (using the basic procedure introduced in [89]) on 100 shape pairs before and after topological optimization. Note the significant improvement in quality across all categories in the dataset.

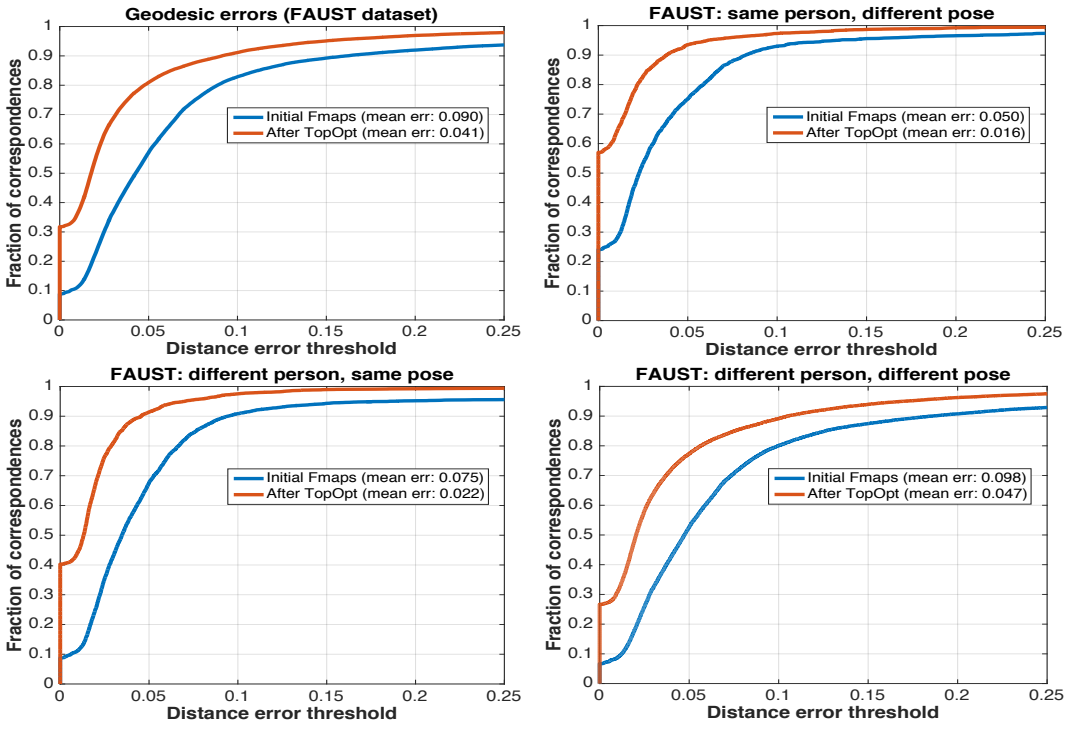


Figure 3.7 – Quantitative evaluation of correspondences computed on 100 random pairs of shapes from FAUST dataset [12], using the basic functional maps pipeline in [86] before and after our topological functional map optimization.

We also illustrate the quality of the computed functional maps converted to point-to-point maps in Figures 3.8 via texture transfer. Note the improvement in the continuity in the maps after topological optimization. We stress that unlike previous works, such as [72, 127], we never enforce continuity of point-to-point maps. Instead, the optimization is done purely in the "functional domain" by minimizing Eq. 3.11 directly. Moreover, we do not enforce any prior on the metric distortion, such as requesting the maps to be isometric, conformal or area-preserving, but instead only promote continuity, while operating with functional maps in a reduced basis. Since we use a basic approach for converting functional maps to point-to-point ones, some high-frequency noise can still be present in the resulting correspondence. Nevertheless, post-processing of these maps using techniques such as [72] can certainly improve the results further.

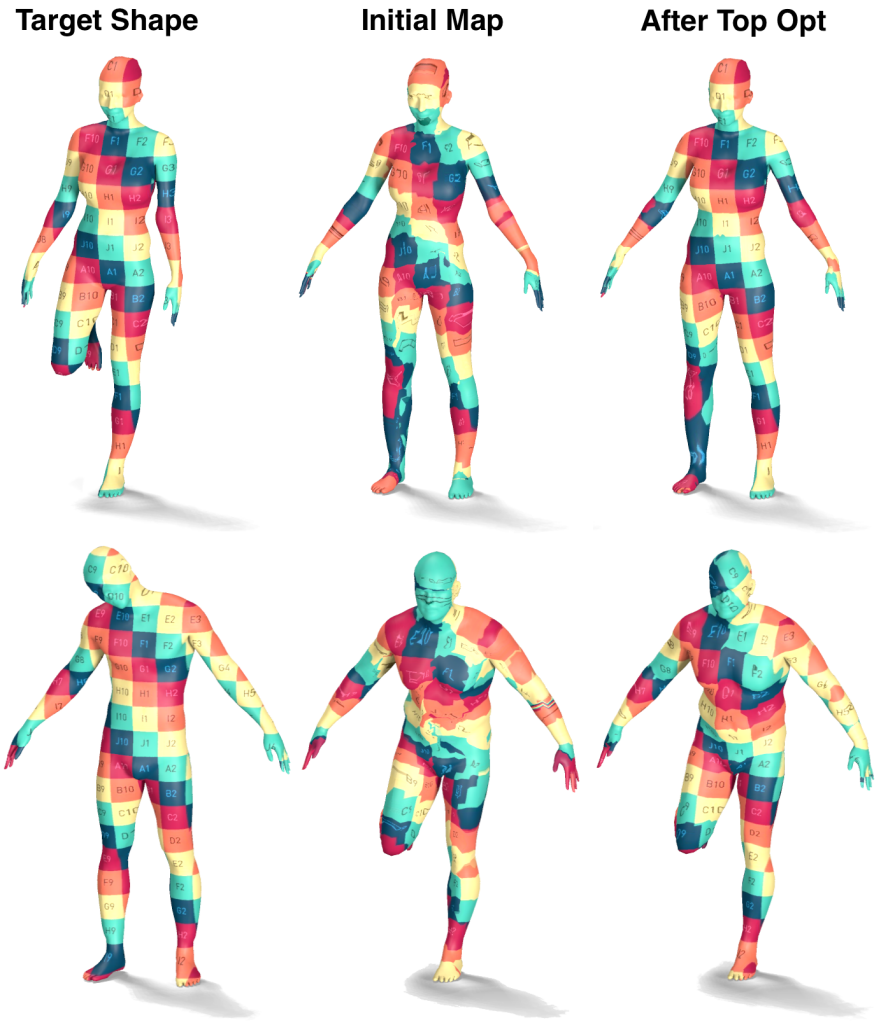


Figure 3.8 – Texture transfer from a target shape (left) onto the source using a functional map converted to a point-to-point one before (middle) and after (right) topological optimization.

3.8 Conclusion, Limitations & Future Work

In this chapter we presented an approach for optimizing real-valued functions defined on shapes, based on a wide variety of topological criteria. Our main observation is that previously proposed persistence diagrams can be differentiated with respect to the changes in function values, and as such optimized for using continuous optimization techniques. We use this procedure for both aligning diagrams of functions defined on possibly different domains, i.e., reducing their bottleneck or Wasserstein distances, and also for simplifying a given function to remove undesired topological features. Finally, we show how this analysis can be used in the context of functional map computations, first by characterizing *continuous* point-to-point maps directly in the functional domain and then presenting an optimization scheme that helps to promote continuity of functional maps in the reduced basis.

In the future, it would be very interesting to provide rigorous stability guarantees and bounds on how our characterization behaves for *approximately* continuous maps, especially when expressed as functional maps in the reduced basis. It would also be interesting to explore other functionals on persistence diagrams both in shape analysis and in the more broad data analysis applications, for example on images or graphs. Moreover, our functional map improvement procedure is not well-adapted to partial shapes and it would be interesting to see how it can be used jointly with objectives such as promoting bijectivity or partiality, or with existing approaches that promote continuity of pointwise maps, e.g. [72, 127]. In addition, it would be interesting to study questions of convergence of our optimization problems, depending on the choice and size of the basis, as well as stability with respect to changes in the shape, taking advantage of the theoretical stability guarantees available for persistence diagrams.

Finally, we believe that the interaction between topological data analysis techniques, including persistence diagrams, with function-based approaches, including the functional maps framework, is a very fruitful and largely unexplored area for future work in general.

Chapter 4

Multi-directional Geodesic Neural Networks via Equivariant Convolution

We propose a novel approach for performing convolution of signals on curved surfaces and show its utility in a variety of geometric deep learning applications. Key to our construction is the notion of directional functions defined on the surface, which extend the classic real-valued signals and which can be naturally convolved with with real-valued template functions. As a result, rather than trying to fix a canonical orientation or only keeping the maximal response across all alignments of a 2D template at every point of the surface, as done in previous works, we show how information across all rotations can be kept across different layers of the neural network. Our construction, which we call *multi-directional geodesic convolution*, or *directional convolution* for short, allows, in particular, to propagate and relate directional information across layers and thus different regions on the shape. We first define directional convolution in the continuous setting, prove its key properties and then show how it can be implemented in practice, for shapes represented as triangle meshes. We evaluate directional convolution in a wide variety of learning scenarios ranging from classification of signals on surfaces, to shape segmentation and shape matching, where we show a significant improvement over several baselines.

4.1 Introduction

The success of convolutional neural networks (CNNs) for image processing tasks [64] has brought attention from the geometry processing community. In recent years multiple techniques have been developed to reproduce the success of CNNs in the context of geometry of curved surfaces with applications including shape recognition [118], segmentation [53, 73] or shape matching [15], among many others. A key aspect of CNNs is that they rely on convolution operations. In the Euclidean domain the notion of convolution is well-defined whereas on non-Euclidean spaces, in general, there is no direct analogue that satisfies all the same properties.

Different approaches have been proposed to overcome this limitation. Perhaps the simplest and most common consist in either performing convolution directly on the surrounding Euclidean 3D space (using so-called volumetric approaches [76], [99]), or constructing multiple projections (views) of an embedded object from different angles and applying standard CNNs in 2D [118]. This idea has also been extended to using more general mappings onto canonical domains, including the plane [113, 40], or e.g. toric

domains which admit a global parameterization and where convolution can be defined naturally [73]. Unfortunately, such mappings can induce significant distortion and might be restricted to only certain topological classes. On the other hand, several *intrinsic* approaches have been proposed to define analogues of convolution directly on the manifold [15, 75, 16]. These techniques aim at learning local template (or kernel) functions, which can be mapped onto a neighborhood of each point on a surface and convolved with signals defined on the shape. These methods are general, can be applied regardless of the shape topology and moreover are not sensitive to the changes in shape embedding, making them attractive in non-rigid shape matching applications, for example. Unfortunately, general surfaces lack even *local* canonical coordinate systems, which means that the mapping of a template onto the surface is defined only up to the choice of an orthonormal basis of the tangent plane at every point. To overcome this limitation, previous methods either only consider the maximal response over a certain number of rotations [15] or aim to resolve these ambiguities using principal curvature directions [75, 16]. Unfortunately such approaches can either lead to more instabilities or, in the case of angular max pooling, lose the relative orientation of the kernels across multiple convolutional layers.

In this chapter, we propose to overcome this key limitation of intrinsic methods by aligning the convolutional layers of the network. Our idea is to consider *directional* (or, equivalently, *angular*) functions defined on surfaces, and to define a notion of convolution for them which results, again, in *directional functions* without loss of information. This allows us to impose specific canonical relations across the layers of a neural network, lifting the directional ambiguity to only the last layer. We then need to take the maximal response over all rotations *only on the last layer*. This allows us to better capture the relative response at different points, leading to an overall improvement in training and test accuracy.

4.2 Related Work

Geometric Deep Learning is an emerging field aimed at applying machine learning techniques in the context of geometric data analysis and processing. Below we review the techniques most closely related to ours, and especially concentrate on various ways to define and use convolution on geometric (3D) shapes and refer the interested reader to several recent surveys, e.g. [138, 17].

4.2.1 Extrinsic and Volumetric Techniques

Perhaps the most common approach for exploiting the power of Convolutional Neural Networks (CNNs) on 3D shapes is to transform them into 2D images, for example by rendering multiple views of the object. Some of the earliest variants of this idea include methods that represent shapes as unordered collections of views (or range images) [118, 130, 53] or exploit the panorama image representation [111, 110] among others, as well as techniques based on Geometry Images [113], which represent the 3D geometry by mapping the coordinates onto the plane.

Another common set of methods considers 3D shapes as volumetric objects and defines convolution simply in the Euclidean 3D space, e.g. [137, 76] among others. Due to the potential memory complexity of this approach, several efficient extensions have been proposed, e.g., [128, 58], and a comparison between view-based and volumetric approaches has been presented in [99].

Other recent techniques analyze 3D shapes simply as collections of points and define deep neural network architectures on point clouds, including, most prominently PointNet [98], PointNet++ [100] and several extensions such as PointCNN [67] and Dynamic Graph CNN [129] for shape classification and segmentation, and PCPNet [47] for normal and curvature estimation, among others.

Despite their efficiency and accuracy in certain situations, these techniques rely directly on the embedding of the shapes, and are thus very sensitive to changes in shape pose, which can limit their use, for example in non-rigid shape matching.

4.2.2 Intrinsic and Graph-based Techniques

To overcome this limitation, several *intrinsic* methods have been proposed, for defining and exploiting convolution directly on the surface of the shape. This includes spectral methods, which exploit the relation between convolution and multiplication in the spectral domain [15], and which have also been applied on general graphs [33]. A similar technique, treating shapes as graphs, has been used for analyzing arbitrary shape collections [139], while synchronizing their laplacian eigenbases with functional maps [88]. Another recent method [40] consists in optimizing an embedding of the shape onto the planar domain using intrinsic metric alignment [117]. Finally, a very recent Surface Networks approach [61] is based on stacking layers consiting of combinations of features and their images by the Laplace or Dirac operator to exploit intrinsic and extrinsic information.

More closely related to our approach are techniques based on *local* shape parameterization, which define convolution of a signal with a learned kernel on a region of the shape surface. A seminal work in this direction was done in [75] where the authors defined Geodesic Convolutional Neural Networks (GCNNs), which locally align a given kernel with the shape surface at each point, and perform convolution in the tangent plane. Unfortunately, the absence of canonical coordinate systems on surfaces leads to a one-directional ambiguity in the alignment. To rectify this, the authors of [75] proposed to take the maximal response across all possible alignments. Several later extensions of this approach have used different local patch parameterizations, [16, 82] and also used prinicipal curvature directions to resolve the directional ambiguity. Unfortunately, principal curvature directions can be highly unstable, and not uniquely defined even on basic domains such as the sphere and the torus, which can lead to over-fitting in the training. Finally, a recent approach has been proposed for defining convolution via mapping onto a toric domain [73], which admits a global parameterization.

4.2.3 Contribution

In our work, we show that the directional ambiguity that exists when mapping a template (kernel) onto the surface, as done in [75, 16, 82] *can be maintained* across the layers of

the deep neural network without relying on a canonical direction choice or only keeping the maximal response across all directions. To achieve this, we first extend real-valued signals to more general *directional functions*. We then show that a directional function can be convolved with a template to produce another directional function, and can thus be stacked in a deep neural network. As a result the directional ambiguity is lifted up to the last layer, where it can be resolved by taking the maximum response only once. We demonstrate through extensive experiments that this leads to overall improvement in accuracy and robustness in a range of applications. Finally, we extend previous approaches such as [16, 82] by adding spatial pooling layers through mesh simplification and exploiting residual learning (ResNet) blocks [48] in the architecture.

4.3 Convolution over manifolds

Throughout our work, we assume that we are dealing with 3D shapes, represented as oriented (manifold) 2D surfaces, embedded in 3D. For simplicity of the discussion, we also assume that the shapes are without boundary, although our practical implementation does not have this limitation.

Throughout our discussion, the input signal is assumed to be a tuple of real-valued functions defined on each shape in the collection. These can either represent some geometric descriptors, or even simply the 3D coordinates of each point. In this section, we describe how the convolution operation is applied to a given signal for a fixed template. Our approach follows the general structure proposed in [75], but we highlight the key differences, arising from our use of *directional convolution*. Finally, let us note that for simplicity we concentrate on oriented two-dimensional manifolds, although most of our discussion can be adapted easily to more general settings.

Recall that in the standard two-dimensional Euclidean setting, the cross-correlation or convolution of a real-valued function f by a smooth compactly supported (template) function $k : \mathbb{R}^2 \to \mathbb{R}$ is defined for all $x \in \mathbb{R}^2$ by:

$$(f * k)(x) := \int_{\mathbb{R}^2} f(t)k(t + x)dt.$$

Convolution is a way to compare the function f to a template function k at every point. We can reinterpret convolution in the following way: first, for every point x we identify the tangent space to x with a copy of $\mathbb{R}^2$ whose origin has been shifted to x. Then, translation by x can be seen as linear isometry:

$$\tau_x : T_x\mathbb{R}^2 \to \mathbb{R}^2 \simeq T_0\mathbb{R}^2, \quad \tau_x(p) = p + x \ \forall \ p,$$

where $\mathbb{R}^2 \simeq T_0\mathbb{R}^2$ simply means that the tangent plane at the origin can be identified with the whole of $\mathbb{R}^2$. Moreover, the identity map "centered at x" can be seen as another isometry:

$$\mathrm{Id}_x : T_x\mathbb{R}^2 \to \mathbb{R}^2, \quad \mathrm{Id}_x(p) = p \ \forall \ p.$$

Therefore can rewrite the convolution of f by k at x, as:

$$(f * k)(x) = \langle \mathrm{Id}_x^* f, \tau_x^* k \rangle_{L^2},$$

where the supperscript * means the pull-back of the function with respect to the map, and $\langle , \rangle_{L^2}$ is the standard L^2 inner product. We can now generalize this construction to two-dimensional manifolds by generalizing the maps Id_x^* and τ_x^*. At every point x of a Riemannian manifold X the exponential map:

$$\exp_x^X : T_x X \to X$$

generalizes the previous map $\mathrm{Id}_x : T_x\mathbb{R}^2 \to \mathbb{R}^2$ in the sense that $\exp_x^{\mathbb{R}^2} = \mathrm{Id}_x$. Following [75], we assume that the template function k is defined over a copy of $\mathbb{R}^2$, denoted by $T_0\mathbb{R}^2$, and generalize the map τ_x by isometrically aligning this copy with the tangent plane $T_x X$ at x. That is, the map $\tau_x : T_x X \to T_0\mathbb{R}^2$ must be a linear isometry. This map is uniquely-defined given a choice of the correspondence of the coordinate axes of $\mathbb{R}^2$ with a coordinate frame of $T_x X$. For oriented surfaces, this reduces to the alignment of one coordinate axis. If $e_1 := (1, 0)$ is the first coordinate axis on $\mathbb{R}^2$, this means that the inverse of the map τ must send e_1 to all tangent spaces, and that k can be pulled-back onto $T_x X$ for any x, via $\tau_x^* k$.

Since unlike the Euclidean case, there is no global choice of reference direction on a surface, an arbitrary choice of the pre-image of e_1 on every tangent space $T_x X$ can lead to biased results, which furthermore will not generalize across different (e.g., training and test) shapes.

To resolve this ambiguity, the authors of [75] consider the family of maps $\tau_{x,v}$ parameterized by a choice of a unit vector v in the tangent plane of x, which is mapped to the first coordinate axis in $\mathbb{R}^2$, i.e. $\tau_{x,v}(v) = e_1$. They then define geodesic convolution by taking the maximum response of the signal to the template function k mapped via $\tau_{x,v}$ across all choices of v.

$$(f \circledast k)(x) = \max_v \langle (\exp_x^X)^* f, \tau_{x,v}^* k \rangle_{L^2}. \tag{4.1}$$

The two main advantages of this procedure are that: 1) it does not depend on the choice of reference direction in the tangent planes, and 2) the output of $f \circledast k$ is again a real-valued function, so that convolutions can be applied successively within a deep network.

Unfortunately, only keeping the maximum response also results in a loss of directional information, which can make it more difficult to detect certain types of features in the signal. In particular, since the maximum is applied independently at every point, directional information of the response of the signal to the template is not shared across nearby points.

4.3.1 Convolution of Directional Functions

We propose to address these limitations by considering convolution of a template function with a more general notion of *directional functions* defined on arbitrary surfaces.

We call a directional function, any function $\varphi(x, v)$ that depends on both the point x on a surface X, and on the unit direction v in the tangent plane $T_x X$ at x. Clearly any real-valued function $f : X \to \mathbb{R}$ can be lifted to a directional function $\tilde{f}$, simply by ignoring the directional argument and setting $\tilde{f}(x, v) = f(x)$ for any v.

Our key observation is that the convolution of a *directional function* φ with respect to a template $k : T_0\mathbb{R}^2 \to \mathbb{R}$ can be defined naturally, so that the result of the convolution is, once again, a directional function. For this we first complete the exponential map, so that for any point $p \in T_xX$ in the tangent plane of x, it produces both a point y on X and a unit direction in the tangent plane of y. To achieve this we define the completed exponential map:

$$\overline{\exp}_x^X(p) = (\exp_x^X(p),\ \Gamma_{x,p}(p/\|p\|)), \tag{4.2}$$

where $\Gamma_{x,p}(p/\|p\|)$ is the parallel transport of the unit vector $p/\|p\|$ from the tangent plane of x to the tangent plane of $y = \exp_x^X(p)$ along the geodesic between x and y with initial velocity $p/\|p\|$ (see Figure 4.1).

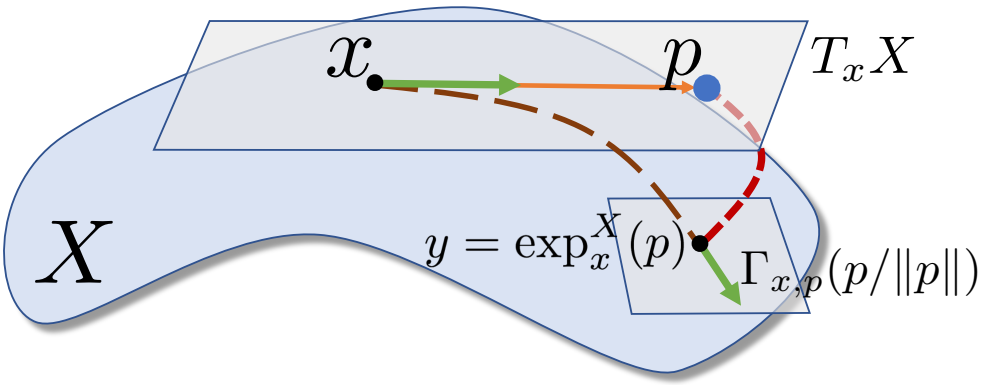


Figure 4.1 – The *completed* exponential map at a point $x \in X$ sends any non null tangent vector $p \in T_xX$ to the couple $(y = \exp_x^X(p), v = \Gamma_{x,p}(\frac{p}{||p||}))$ where $v \in T_yX$ is the result of parallel transport of $\frac{p}{||p||}$ along the geodesic from x to y, with the initial direction given by p.

Thus, for any point p in the tangent plane of x, $\overline{\exp}_x^X(p)$ outputs a point y on the manifold and a vector in the tangent plane of y. Moreover, since parallel transport along geodesics preserves the norms of vectors, $\Gamma_{x,p}(p/\|p\|)$ must also be a unit vector. This map is well-defined everywhere, except at the origin $p = 0$. This does not pose a problem in our setting, however, since we only use this map inside an integral. We now define *multi-directional geodesic convolution*, or *directional convolution* for short, of a template $k : T_0\mathbb{R}^2 \to \mathbb{R}$ with a directional function φ by:

$$(\varphi \star k)(x, v) = \langle (\overline{\exp}_x^X)^* \varphi, \tau_{x,v}^* k \rangle_{L^2}. \tag{4.3}$$

Note that, $\psi_x = (\overline{\exp}_x^X)^* \varphi$ is a real-valued function in the tangent plane of x, where $\psi_x(p) = \varphi(\overline{\exp}_x^X(p))$ (see Figure 4.2). Moreover, note that the result of a convolution of a directional function φ and a template k is once again a directional function, as it depends both on the point x and on the direction v. We also remark that directional convolution

can be extended to regular-valued functions. Thus, for any $f : X \to \mathbb{R}$ we simply set: $f \star k := \tilde{f} \star k$ where $\tilde{f}$ is the lifting of f to a directional function, as described above.

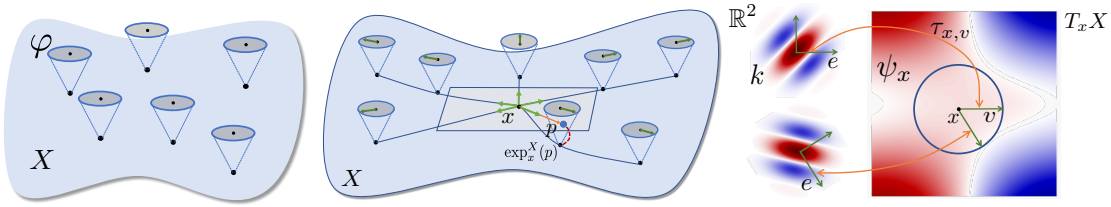


Figure 4.2 – Overview of our directional convolution: **(left)** we start with a surface X and a directional function φ, which assigns a real value to every unit direction in the tangent plane of every point of X, shown here as a cone above some points. **(middle)** given a point $x \in X$, we pull back φ onto the tangent plane of x by computing for every point $p \in T_xX$, its image $y \in X$ via the exponential map, and the transport of the unit vector $p/\|p\|$ to the tangent plane of $y = \exp_x^X(p)$. We assign to p the value of φ at y and the transported vector. By repeating this for every p we obtain a real-valued function ψ_x over the tangent plane of x. **(right)** Given a unit vector v tangent at x there is a unique orthogonal orientation-preserving linear map $\tau_{x,v}$ from T_xX to $\mathbb{R}^2$ sending v to $e := (1, 0)$. We pull back a kernel k defined over $\mathbb{R}^2$ to T_xX and take its L^2 product with ψ_x giving us a real-number for each unit tangent vector at x. Repeating this procedure for every $x \in X$ we obtain a new directional function which is the directional convolution of φ by the kernel k.

4.3.2 Directional vs. Geodesic Convolution

As suggested in the previous section, geodesic convolution introduced in [75] and our directional convolution are closely related. Indeed, below we show that directional convolution is strictly more informative than geodesic convolution. The following proposition shows that geodesic convolution can be factorized by taking the maximal directional response of directional convolution thus losing the directional information.

Proposition 4.3.1 *Let f a function on X and k a template. Denote by $\tilde{f} : (x, v) \mapsto f(x)$ the directional function obtained via $\tilde{f}(x, v) = f(x)$ for all unitary $v \in T_xX$. Then:*

$$f \circledast k(x) = \max_{v \in T_xX} (\tilde{f} \star k)(x, v)$$

Proof We have:

$$\begin{aligned}(\overline{\exp}_x^X)^* \tilde{f}(p) &= \tilde{f}(\exp_x^X(p), \Gamma_{x,p}(p/||p||)) \\ &= f(\exp_x^X(p)) := ((\exp_x^X)^* f)(p)\end{aligned}$$

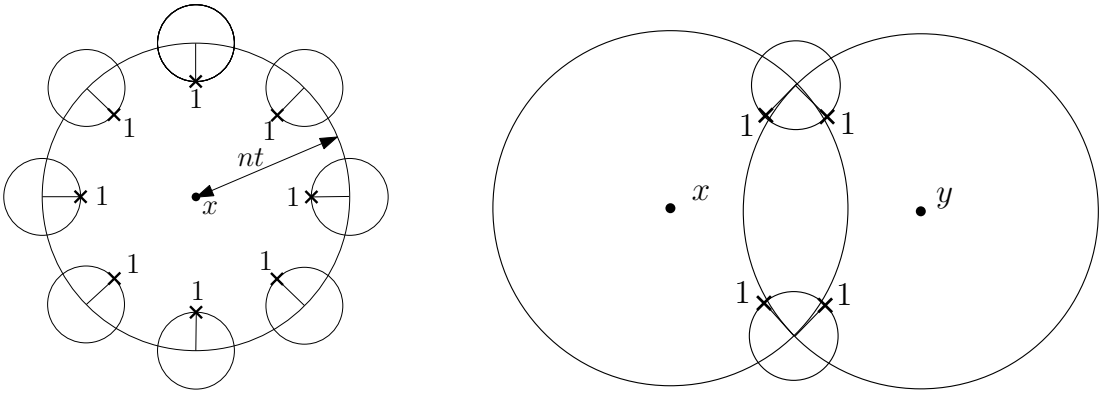
thus:

$$\begin{aligned}\max_{v \in T_xX} (\tilde{f} \star k)(x, v) &= \max_{v \in T_xX} \langle (\overline{\exp}_x^X)^* \tilde{f}, \tau_{x,v}^* k \rangle_{L^2}. \\ &= \max_{v \in T_xX} \langle (\exp_x^X)^* f, \tau_{x,v}^* k \rangle_{L^2}. =: f \circledast k(x)\end{aligned}$$

□

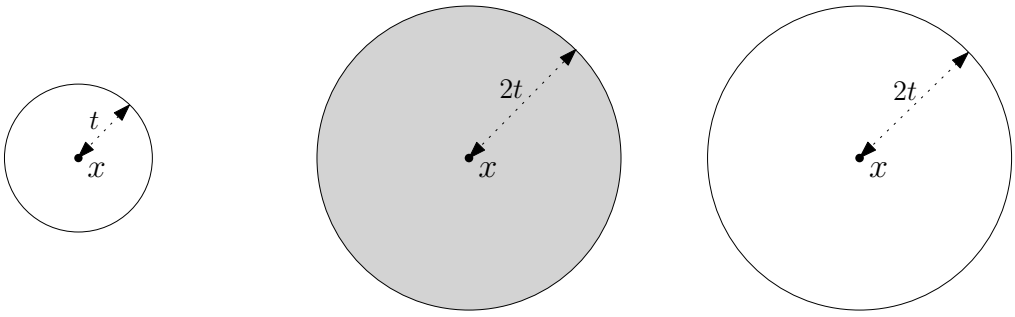
Intuitively, applying directional convolution allows to keep track of the direction that the signal comes from. To illustrate this, we consider the directional convolution of the indicator function $\mathbf{1}_x$ of a point x, by a shifted Dirac kernel $\delta_{(t,0)}$ for some $t > 0$. It is easy to see that the result of $\mathbf{1}_{(x,v)} \star \delta_{(t,0)}$ is the indicator function of the set of couples (y, v) such that $y \in X$ is at (geodesic) distance t from x and $v \in T_y X$ points in the direction of x (see Figure 4.3 for an illustration). Moreover, as also illustrated in Figure 4.3, applying directional convolution by $\delta_{(t,0)}$ multiple times will propagate the signal along geodesics from the source point x while maintaining the directional information attached to it. In contrast, after angular max-pooling directional information to the source point is lost. This means that when applying the basic geodesic convolution repeatedly (e.g. when stacking multiple convolution layers in a neural network) the signal does not necessarily travel along geodesic paths. Thus, for example, $(\mathbf{1}_x \circledast \delta_{(t,0)}) \circledast \delta_{(t,0)}$ is a geodesic *ball* instead of a circle of radius $2t$ around x (see Figure 4.4). This might result in a loss of information and makes stacks of filters based on geodesic convolution less efficient in estimating the distance between features or their relative position, compared to directional convolution shown in Figure 4.3b. We also note briefly that directional geodesic convolution does not admit an identity kernel but identity can be obtained as a limit of convolutions by shifted Dirac kernels since:

$$\lim_{t \to 0} \overline{\exp}_x^X(tv) = (x, v).$$



(a) n dir-conv, $\mathbf{1}_x \star \delta_t \star \cdots \star \delta_t$ (b) n dir-conv, $(\mathbf{1}_x + \mathbf{1}_y) \star \delta_t \star \cdots \star \delta_t$

Figure 4.3 – (left) The result of applying n times the directed convolution by the shifted Dirac kernel $\delta_{(t,0)}$ to the indicator function $\mathbf{1}_x$ of point x. We obtain a response along the circle of radius nt in the directions pointing to x. This allows to detect the presence of a feature at a given distance and to point in the direction of that feature. (right) The response of the convolution with two isolated signals at x and y is located along two circles, if they intersect the intersection points store information about the relative angle and distance between the features at x and y.



(a) 1 conv + amp, $\mathbf{1} \circledast \delta = \max_v \mathbf{1} \star \delta(\bullet, v)$

(b) 2 geod-conv, $(\mathbf{1} \circledast \delta) \circledast \delta$

(c) 2 dir-conv + amp, $\max_v (\mathbf{1} \star \delta) \star \delta(\bullet, v)$

Figure 4.4 – Comparison between repeated geodesic (geod, middle) and directional convolution (dir, right) of the indicator of a point $\mathbf{1}$ by a shifted Dirac kernel $\delta = \delta_t$. (b) With directional convolution we obtain a response along the circle of radius $2t$ indicating the presence of a feature at *exactly* distance $2t$ (c) With geodesic convolution, we obtain a response along the full disc of radius $2t$ only indicating the presence of a feature at distance $\leqslant 2t$.

4.3.3 Directional Convolution in Angular Coordinates

The expression in Eq. (4.3) is coordinate-free, however to implement it we must choose a particular coordinate system to represent directional functions. In practice, we represent tangent vectors in polar coordinates in their tangent plane. This representation implies an arbitrary choice of reference direction $e_x \in T_x X$ in each tangent plane. For any angle $\theta \in [0, 2\pi)$ we denote by $e_x(\theta) = R_\theta e_x$, the vector obtained by rotating e_x by angle θ in the tangent plane at x. Instead of operating with directional functions, we then work with *angular functions* as follows: given a directional function φ, and a choice of reference direction e_x for each point x, we define an angular function φ_e by:

$$\varphi_e(x, \theta) := \varphi(x, e_x(\theta)).$$

that is, φ_e is just the function φ expressed in polar coordinates in each tangent plane with respect to the reference direction e. Note also that given reference directions $e_x \in T_x X$ for all $x \in X$, both the exponential map and the parallel transport can be expressed for points and vectors described in polar coordinates in each tangent plane (see Appendix). Finally, given the family of reference directions $e = (e_x)_{x \in X}$ and the angular function φ_e we denote by $\varphi_e \star k$ its directional convolution with kernel k w.r.t. e. To define this operation, we first convert φ_e to its directional counterpart, using the reference directions e and then simply apply the definition given in Eq. (4.3) above and convert it back to an angular function using the same reference directions:

$$\varphi_e \star k := (\varphi \star k)_e$$

The following proposition shows that the result of directional convolution of angular functions is *equivariant* with respect to the change of the reference directions. Namely:

Proposition 4.3.2 *Let $e_x \in T_xX$ be a family of unit tangent vectors defined at every point $x \in X$ then for any family of rotations R_x of angle θ_x at point x we have:*

$$(\varphi_{Re} \star k)(x, \theta) = (\varphi \star k)_{Re}(x, \theta) = (\varphi_e \star k)(x, \theta + \theta_x)$$

Where $(Re)_x := R_x e_x$ at each point x.

Proof The first equality holds directly by definition. Namely, by applying the definition of directional convolution of the angular function φ_{Re} with reference direction Re we have:

$$\varphi_{Re} \star k := (\varphi \star k)_{Re}.$$

To prove the second equality observe that Re_x is simply the rotation of e_x by angle θ_x therefore:

$$\begin{aligned}(\varphi \star k)_{Re}(x, \theta) &:= (\varphi \star k)(x, Re_x(\theta)) \\ &= (\varphi \star k)(x, e_x(\theta + \theta_x)) \\ &=: (\varphi \star k)_e(x, \theta + \theta_x) \\ &=: (\varphi_e \star k)(x, \theta + \theta_x)\end{aligned}$$

which proves the proposition using the coordinate free definition of the directional convolution operator. In the appendix section 4.8 we provide an alternative proof when the directional convolution operator $\star$ is defined using arbitrary local angular coordinates. □

This proposition guarantees that even if we pick an arbitrary reference direction in the tangent plane of every point x, the result of possibly multiple directional convolution steps is the same up to an angle offset. Moreover, this offset is fixed and is the angle difference between the two reference directions.

When directional convolution operator $\star$ is defined directly in angular coordinates, and angles are discretized in angular bins, as they are in our practical implementation, this implies that changing the reference direction at a given point by applying a circular permutation to the angular bins' indices will lead to the same permutation of the result of directional convolution. Thus, no further ambiguity is introduced between layers of directional convolution, and the initial rotational ambiguity is lifted to the last layer. We resolve it by applying angular max pooling. This is crucial to learning since it allows to learn the same features independently of the reference directions used to define the angular coordinates.

4.3.4 Multi-Directional Convolutional Neural Networks

Our main motivation for introducing directional convolution is to use it as a layer inside a deep neural network for shape analysis tasks. We define a directional convolution layer simply by replacing the standard convolution (resp. geodesic convolution) operator and regular functions by directional functions in the convolutional layer definition. Crucially, since directional convolution by a template kernel results in another directional function, we can stack *directional convolutional layers* in a neural network.

In the simplest setting a (multi-directional)-convolutional neural network consists of $N+1$ layers stacked sequentially. The output y_i of the i-th layer is a collection of $d_i > 0$ directional functions $(y_i)_{p\in[\![1,d_i]\!]}$. The network takes a map $f : X \to \mathbb{R}^{d_0}$ as input where $d_0 > 0$ is the number of input descriptor functions. Then f is converted into a directional function $\tilde{f}$ defined by $\tilde{f}(x, v) := f(x)$ for all $x \in X$ and unit vector $v \in T_xX$. The network has stacks of learnable template kernels $(k^i_{pq})_{p\in[\![1,d_i]\!],[\![1,d_{i+1}]\!]} : \mathbb{R}^2 \to \mathbb{R}$, a collection of learnable bias vectors $b^i \in \mathbb{R}^{d_{i+1}}$, and activation functions ξ_i for $i \in [\![1, N]\!]$. The outputs $(y_i)_{i\in[\![0,N]\!]}$ of the network's layers are defined recursively by:

$$\begin{cases} y_0 := \tilde{f} \\ (y_{i+1})_p := \xi_i(\sum_q((y_i)_q \star k^i_{pq}) + b^i_p) \end{cases}$$

Our goal during training is to learn the parameters k^i_{pq} and b^i so that the output function minimizes some error over a set of examples. In practice, we consider more complex architectures, as described in Section 4.6. In all cases, angular max pooling is applied to the last layer y_N to obtain a regular signal over X. We call networks based on directional convolution MDGCNN for Multi Directional Geodesic Convolutional Neural Networks.

4.4 directional convolution over Discrete Surfaces

We assume that all shapes are represented as connected manifold triangle meshes, possibly with boundary. In this section we describe our general approach for implementing directional convolution in practice. To simplify the presentation, we describe only the main steps necessary for the implementation, and defer the exact implementation details to Section 4.8.1 in the Appendix.

To implement directional convolution, we need to decide on the discrete representation of template functions, angular functions on the surface, the exponential map and parallel transport. These are described as follows:

4.4.1 Template Functions

To discretize template functions of the form $k : \mathbb{R}^2 \to \mathbb{R}$, we represent them via windows of discrete polar coordinates in the plane: (ρ_i, θ_j), where $\rho_i := \frac{(i+1)R}{N_\rho+1}$ is a set of radii, for i varying between 0 and $N_\rho - 1$ bounded by the window radius R and $\theta_j := \frac{2j\pi}{N_\theta}$ is a uniform discrete sampling of angles $[0, 2\pi)$. This means that each k associates a real value to each pair (ρ_i, θ_j), and can therefore be stored as a matrix of size N_ρ, N_θ. Note that during training these are the unknowns that need to be trained.

4.4.2 Angular Functions

Angular functions are real-valued functions that depend on the point on a given surface and on a direction in its tangent plane. For a mesh with N_v vertices, we represent these as matrices of size (N_v, N_θ), where N_θ is defines a discrete angular sampling, in the same way as for template functions above, but with respect to some arbitrary reference direction in the tangent plane of each point.

4.4.3 Exponential Map

To discretize the exponential map, similarly to previous approaches, we use Geodesic Polar Coordinates (GPC), which associate a window to every point, and represent other points in its neighborhood, through the geodesic distance ρ and angle θ to the base point in its window. We discretize GPC using the same set of discrete polar pairs (ρ_i, θ_j) as for the template functions. This means, in particular, that given a vertex i, the points in its GPC might not fall on the vertices of the mesh. We therefore use barycentric coordinates inside triangles to interpolate values at points in the GPC windows, based on the values at the vertices of the mesh. This procedure is crucial since it helps us gain resilience against changes in mesh structure. We can model the GPC using a tensor E of size $(N_v, N_\rho, N_\theta, N_v)$, such that E_{vijw} stores, the barycentric coordinates with respect to vertex w of the point having polar coordinates ρ_i, θ_j in the GPC of the window of vertex v. Note that by definition, generically, E_{vijw} will have exactly three non-zero values for fixed v, i, j.

4.4.4 Parallel Transport

Finally, we need to discretize the parallel transport operation in order to define the discrete analogue of the (completed) exponential map $\overline{\exp}$ defined in Eq. (4.2). Note that a key feature of our definition is that parallel transport needs to be defined only for unit vectors connecting a point p in the tangent plane of v to the origin, which correspond, in our discretization, to the angular polar coordinate θ of p. This means that we need to compute the angle difference between θ in the window of v and the corresponding angle in the GPC of the exponential map of p. However due to angular discretization the transported angle does not necessarily fall into the angles $(\theta_j)_j$ of the window at p. We therefore interpolate between consecutive discrete angles to compute angular functions. Similarly to the exponential map above, the parallel transport can be discretized as a $5D$ tensor Γ of size $(N_v, N_\rho, N_\theta, N_v, N_\theta)$ where Γ_{vijwk} stores the interpolating coordinate with respect to the angle θ_k at vertex w of the geodesic parallel transport of angle θ_j to the point with polar coordinates (ρ_i, θ_j) in the GPC of the window of vertex v. The (completed) angular exponential map $\overline{\exp}$ can then be interpolated in both the spatial and angular domains as a 5D tensor $\overline{E}$ of size $(N_v, N_\rho, N_\theta, N_v, N_\theta)$ such that:

$$\overline{E}_{vijwk} := E_{vijw}\Gamma_{vijwk}.$$

With these constructions at hand, discretizing the generalized convolution, defined in Eq. (4.3) can be done simply by matrix multiplication. Note that aligning the template function with the tangent plane at x via a map τ_x simply corresponds to aligning the angular coordinate of the template with the angular coordinate of the GPC at x. While aligning a rotated template function via $\tau_{x,v}$ corresponds to applying a circular permutation to the angular indexing of k. We define geodesic convolution of a template K and a function f by:

$$(f \circledast k)_v := \max_k \sum_{ij} f_w E_{vijk} K_{i,(j+k) \bmod N_\theta}$$

Likewise directional convolution of an angular function φ by K is defined by:

$$(\varphi \star K)_{vl} = \sum_{ij} \varphi_{wk} \overline{E}_{vijwk} . K_{i,(j+l) \bmod N_\theta}.$$

We also note that the above definitions extend to the case of multidimensional filters and matrix-valued templates. Remark that only the triangles supporting the window points will contribute to the result of the convolution, while the value of the signal at the central point will not be taken into account. To achieve this, we add the result of a dense layer to the convolution. That is we multiply the signal at the central point (and direction for dir-conv) by the some fixed (learned) matrix and add the resulting vector to the result of the convolution at every point.

4.4.5 Spatial Pooling

In traditional CNNs pooling layers refer to a way of sampling the signal. They reduce the resolution of the image by mapping groups of pixels of the original image to a single pixel of a reduced image. The advantage is two-fold: first it allows to summarize the information over a group of pixels and to achieve robustness to local perturbations of the signal and second it reduces computation time and space. A common option called max-pooling is to separate the original image into consecutive square blocks of pixels, where each block is mapped to a single pixel of the reduced image by taking the maximal response over the block in each channel.

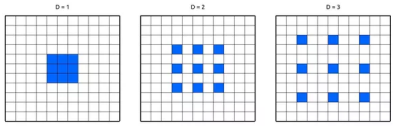


Figure 4.5 – dilated convolution on images with convolution kernel in blue. Spacing of 0 (left), 1 (middle) and 2 (right).

Another option for reducing the resolution, which is easily generalizable to meshes is the notion of dilated convolution (Figure 4.5). It consists of spacing the pixels of the filter window by $D-1$ pixels and applying it every D pixels thus reducing the output resolution by a factor D. We can see dilated convolution as a regular convolution applied to the sub-sampled image. For meshes we define a similar notion by transferring the signal to a coarser mesh and then applying geodesic convolution on the new mesh. In practice, we simplify the original mesh using the classic quadratic edge collapse approach [45], which also produces a mapping between the original and the simplified meshes. We use this to transfer both a signal from the original to the simplified mesh (pooling), or from the simplified mesh to the original (un-pooling) by simply picking the value stored at the closest vertex, among those collapsed to it. To define pooling for directional (angular) functions we need to also transfer angles from the original mesh to its simplified counterpart. To transfer local angles from a mesh M to a mesh N given a map from M to N we first compute the $3D$ rotation sending the normal at each vertex to the normal at its image, this allows us to compare the reference directions on both shapes in the tangent plane to N and to deduce the oriented angular offset between them. We then simply transfer the angles by adding the offset to all discretized angles and interpolating the result between consecutive discrete bins, similarly to our construction of parallel transport.

4.5 Geodesic polar coordinates and parallel transport

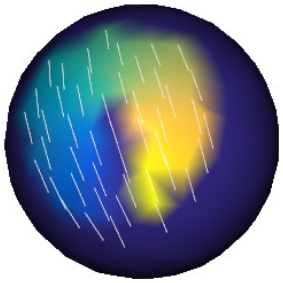


Figure 4.6 – Parallel transport of reference direction at origin and angular coordinate of the GPC.

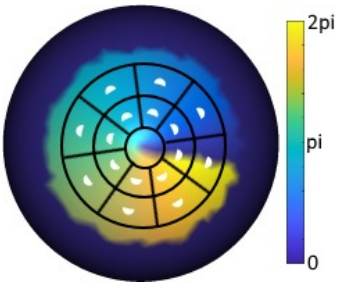


Figure 4.7 – GPC window with 2 radial and 8 angular bins and its contributing points (white)

To compute Geodesic polar coordinates (GPC) at each vertex of the mesh we used the algorithm proposed by [78] which is a variation of the fast marching algorithm [109] computing the angular coordinate as well as the geodesic distance. We extended the algorithm of [78] to also compute parallel transport of angles along geodesics. Fast marching-like algorithms propagate information along meshes, they rely on a local transfer subroutine. A vertex is selected among a set of candidates based on a priority criterion, the information stored at the vertex is then propagated to some of its neighbors based on an update criterion using the local transfer subroutine. The algorithm stops once a stopping condition is met. In our case a vertex is updated until the radius exceeds a certain threshold $R_{\max} > 0$. In practice, we follow the basic approach of [78] for propagating information, but in addition to updating the geodesic distance and polar angle we also keep track of the difference in angles between the reference directions at the source and target points. The original algorithm [78] has a subroutine for updating the GPC angle at a vertex inside a triangle given estimates at the two other vertices. We use the same subroutine to update the transported angle given estimates at the two other vertices. Since the representation of directions is different at each vertex, we transport estimates to every new vertex along edges. To transport an angle from vertex i to a neighbor j along the edge e_{ij} we first apply rotations to the GPCs of i and j so that e_{ij} gives the reference direction at i and e_{ji} gives reference direction at j. The rotated GPCs at i and j have a relative angular offset of π which allows to deduce the angular offset between the original GPCs. We transfer the angle at i to j along e_{ij} by adding the angular offset between the GPCs at i and j. We use the same edge transfer to transfer the reference direction at the source to its neighbors in order to initialize the algorithm candidates set. This modified version allows us to compute the angle difference between the initial and final reference directions, which in turn provides the estimate for the parallel transport of the unit directions along geodesics between points. Figure 4.7 illustrates the angular coordinates of the GPC window via color coding, and the parallel transport of a particular direction from the center vertex to other vertices in the window and Figure 4.6 illustrates our discretization of the angular and radial bins. In addition to this high level description of the algorithm we give a more detailed description in the appendix Section 4.8.3 which was not included in the original publication [95] to save space.

4.6 Evaluation

4.6.1 Architecture

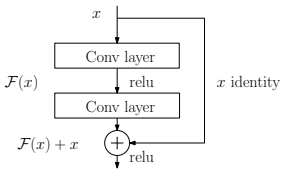


Figure 4.8 – ResNet block

We implemented our deep learning pipeline with Keras [26] using Tensorflow backend [1]. We based our architecture on Residual networks [48]. For classification tasks we used an architecture organized in stacks of ResNet blocks (Figure 4.8). Each stack is the composition of a fixed number of ResNet blocks. After each stack the mesh is down-sampled by a factor 4 using a pooling layer. The radius and number of filters is multiplied by two to preserve time and space complexity across stacks (Figure 4.9). For classification tasks we apply angular max pooling and average the signal over the shape we then apply softmax classifier on the resulting vector. For segmentation tasks we need to produce a point-wise prediction therefore the signal needs to be up-sampled back to the original shape. We combined our ResNet architecture with U-net [104]. Our U-ResNet architecture (Figure 4.10) consists of two blocks an encode and a decode block. The encode block is a copy of our ResNet architecture, the decode block is similar but pooling is replaced by un-pooling. After each stack the window radius is divided by 2, the signal dimension is divided by two and its up-sampled by 4. Shortcut connection are added to help keeping spatial localization information:

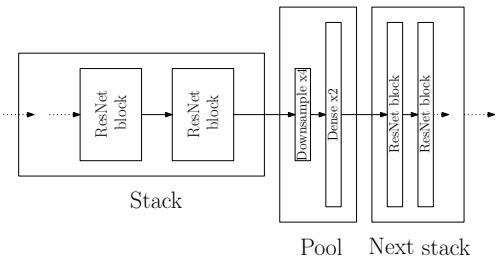


Figure 4.9 – Our ResNet architecture

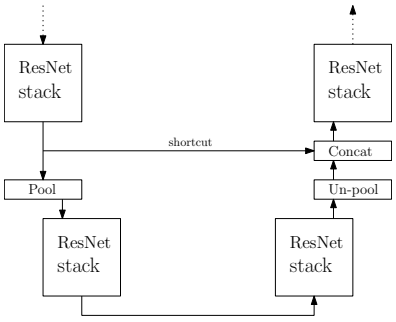


Figure 4.10 – U-ResNet architecture for point-wise prediction

4.6.2 Experiments

In our experiments we compared MDGCNN to GCNN [75] and PointCNN [67] in image classification and to GCNN, PointNet++ [100] and Dynamic Graph CNN [128] in shape segmentation and shape matching tasks using various input features. We trained all networks using ADAM [57] optimizer with learning rate 0.001. Since MDGCNN and GCNN are closely related we used the same architectures and window radii for both to make the comparison as fair as possible. For experiments with varying domains we first center the shapes and normalize them so that they have unit variance, then we

use a fixed initial radius for the whole dataset. The memory complexity is linear in the number of vertices, the number of radial bins and the number of directional bins of the windows. MDGCNN and GCNN have similar time and memory complexity but MDGCNN uses more complex tensor indexing to align local windows and performs directional interpolation of the result of convolution. Therefore our implementation of MDGCNN is slightly slower in practice. For example in our image classification experiment on 50000 images mapped to spheres with 3000 vertexes we observed times of 7 min 13s for one epoch with GCNN and 10 min 13 with MDGCNN using a GTX 1080 graphics card. This speed advantage however is compensated by a the significantly faster convergence of MDGCNN as shown in Figure 4.13.

4.6.3 Image classification

In our first experiment we compare MDGCNN, GCNN and PointCNN on the CIFAR-10 image classification benchmark [63] on different domains. We do not try to achieve state-of-the-art performance. Our purpose is to demonstrate that MDGCNN is able to learn complex signals over mesh domains and is superior to GCNN. The CIFAR-10 dataset consists of 60000 32 by 32 RGB images in 10 classes, (airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck) with 6000 images per class. There are 50000 training images and 10000 test images. We mapped the images to two different meshes, a regular grid and a sphere. [41] (shown in Figure 4.11) to map the images to both hemispheres.

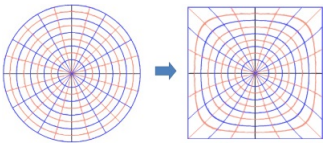


Figure 4.11 – Elliptical mapping from a disc to a square

In the case of the sphere, we parametrized two opposite hemispheres in polar coordinates and then used elliptical mapping Once the mapping is computed every mesh vertex is equipped with 2D coordinates which allows us to pull back the images using bilinear interpolation inside pixels. The resulting image on the sphere is then linearly interpolated inside each triangle as shown on Figure 4.12. Let us note that in both the grid and the sphere case, using principal curvature directions to fix the reference orientation, as done in [16, 82], would not be meaningful as every point is an umbilic on these domains, so that *every* direction is a principal one.

Figure 4.12 – Examples of CIFAR-10 images mapped to a sphere using elliptical mapping

We trained our ResNet architecture (Figure 4.9) in batch of 10 images with 3 ResNet stacks, one ResNet block (Figure 4.8) per stack and 16 filters for the first stack, we chose an initial radius of 1.8 pixels, that is 1.8 in the grid case and $(32 \times 1.8)/\pi$ on the sphere. The network converged after 50 epochs. The results scores in Table 4.1 show a clear advantage for MDGCNN over GCNN, we also observe similar scores for MDGCNN on different domains despite the important stretching introduced by the elliptical mapping of images to the sphere and the irregularity of the sphere meshing compared to a grid mesh. In their recent work [67] Li et al. applied PointCNN to the CIFAR-10 classification benchmark. The results summarized in Table 4.1 suggest that our method compares favorably to PointCNN in the image classification context.

Table 4.1 – Classification accuracy on the CIFAR-10 dataset using different methods on different domains.

CIFAR 10 - classification		
Method	Domain	Accuracy
GCNN	sphere	0.6712
MDGCNN	sphere	**0.7706**
GCNN	grid	0.6767
MDGCNN	grid	**0.7932**
PointCNN	grid	0.7669

4.6.4 Shape segmentation

In our second experiment we compared our MDGCNN against GCNN and several other state-of-the-art methods: Toric cover CNN [73], PointNet++ [100] and Dynamic graph CNN [129]. We evaluated all methods on the human segmentation benchmark proposed by [73]. This dataset consists of 370 models from SCAPE, FAUST, MIT and Adobe Fuse [2]. All models are manually segmented into eight labels, three for the legs, two for the arms, one for the body and one for the head. The test set is the 18 models from the SHREC07 dataset in human category. We used our U-ResNet architecture with 2×2 of two blocks, 16 filters on the first layer and an initial radius of 0.1. Since we did not have access to the same features as used in [73], we used different inputs, SHOT [108] and WKS [8] descriptors as well as the 3D coordinates of the shape, we denote by SHOT_k the 64-dimensional SHOT descriptor with window radius equal to k percent of the shape area. For the experiments taking the 3D coordinates of the shape as input we first apply a random rotation and scaling between 0.85 and 1.15 to learn features that are robust to global transformations. Table 4.2 summarizes the scores obtained by different methods. In addition to improving accuracy, we have also observed that training MDGCNN can be significantly more stable compared to GCNN. In Table 4.3 we report the standard deviation of the validation accuracy of MDGCNN and GCNN across 5 independent runs with different number of epochs. Here the training data is fixed and the variance is only due to the stochastic nature of the optimization procedure. Usual shape descriptors often carry more local information about the points for example SHOT relies on local histograms counting the mesh vertexes and normals into bins, while WKS relies on diffusion processes along the surface. We observe close performance in favor of MDGCNN when using WKS. For SHOT we notice that GCNN performance considerably degrades when reducing the radius of the SHOT windows while MDGCNN

is able to maintain its performance much better. PointNet++ and Dynamic Graph CNN behaved similarly on this benchmark we observed slightly slower convergence compared to MDGCNN but slightly better final results (see Figure 4.13). We observe similar accuracy between MDGCNN and the Toric Cover method of [73]. We note, however, that the Toric Cover method is based on non-canonical mappings from the torus to the surface and requires considerable data augmentation by examining many such mappings. This results in long training and prediction computation times. According to [73] it takes about 5 hours for the Toric Cover method to complete one epoch at training using 6 Nvidia K80 GPUs. Using 50 different mappings, it takes 45 minutes to calculate predictions on the human class of SHREC07 while it takes 1 min 10s to train our MDGCNN network on one epoch using a single Nvidia TITAN Xp card and 2.2317s to compute predictions on the test set.

Human body segmentation				
Method	Input feat.	# feat.	epochs	accuracy
Toric cover	WKS, AGD, curv.	20+2+4	20	0.88
Pointnet++	3D coords	3	200	**0.9077**
DynGraphCNN	3D coords	3	200	0.8972
GCNN	3D coords	3	200	0.7649
MDGCNN	3D coords	3	50	**0.8861**
GCNN	WKS, curv.	20+4	50	0.8489
MDGCNN	WKS, curv.	20+4	50	**0.8612**
GCNN	$SHOT_6$	64	50	0.3888
MDGCNN	$SHOT_6$	64	50	**0.8530**
GCNN	$SHOT_9$	64	50	0.7410
MDGCNN	$SHOT_9$	64	50	**0.8879**
GCNN	$SHOT_{12}$	64	50	0.8640
MDGCNN	$SHOT_{12}$	64	50	**0.8947**

Table 4.2 – Segmentation accuracy of several methods on the human body dataset introduced in [73].

Human body segmentation			
method	Input	epochs	std
MDGCNN	3D	50	**0.01059**
GCNN	3D	50	0.11487
MDGCNN	3D	100	**0.02831**
GCNN	3D	100	0.08007
MDGCNN	3D	200	**0.00454**
GCNN	3D	200	0.05513

Table 4.3 – Standard deviation (std) of test accuracy of MDGCNN and GCNN across 5 independent runs.

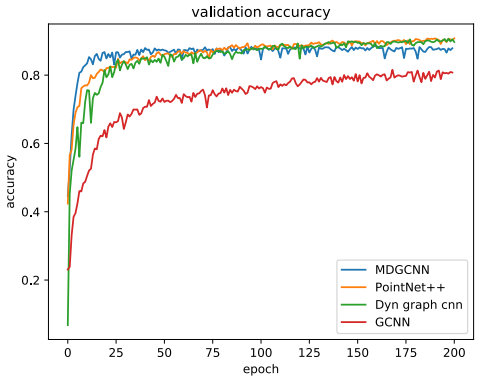


Figure 4.13 – Validation accuracy per epoch on the human body segmentation benchmark introduced in [73] using global 3D coordinates as input.

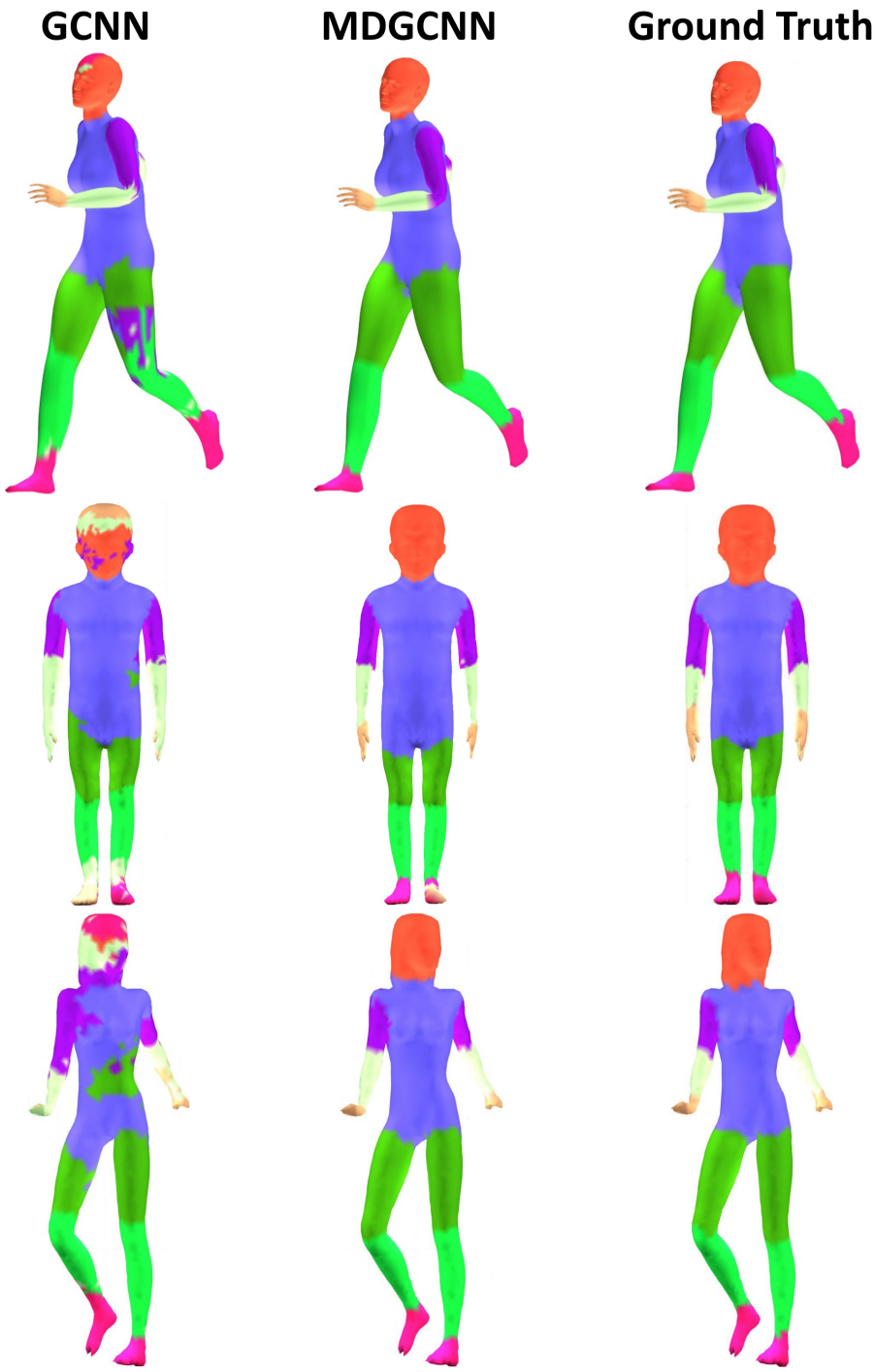


Figure 4.14 – Human shapes segmentation comparison between standard GCNN and MDGCNN (ours) using the 3D coordinates of the shapes as input. The data is augmented by random rotations and scaling at training to ensure rigid motion invariance and improve robustness of the learning.

We observe in Figure 4.14 that MDGCNN is better than GCNN at learning features that are invariant to rigid motion directly from the 3D coordinates of the shape without *a priori* descriptors assumed in the data, simply via data augmentation. Learning global features from the 3D coordinates of the shape vertices requires aggregating information between possibly distant points. Since directional convolution allows better communication between distant points, MDGCNN noticeably outperforms GCNN when using 3D coordinates as input, producing much smoother results. This is also illustrated in the qualitative results shown in Figure 4.14.

4.6.5 Shape matching

Finally, we also applied our pipeline in the context of non-rigid shape matching on the FAUST dataset, used in [75]. In this experiment, goal is to predict the index corresponding to each vertex in the 0-th shape of the dataset. The original experiment in [75] used the GCNN architecture using SHOT descriptors as input. In order to remove the bias present in the data, due to all meshes sharing the same connectivity, we also re-meshed the FAUST shapes from 6890 vertexes to 5000 vertexes to evaluate the robustness of both algorithms. We used our U-ResNet architecture with 2×2 stacks of two blocks with 16 filters on the first layer and an initial radius of 0.1 taking SHOT_{12} as input. The network was trained for 100 epochs for MDGCNN and 200 for GCNN. We measured the geodesic error between the predicted labels and the ground truth on the 0-th shape for both the original FAUST shapes and the re-meshed ones (5k) (Figure 4.15). Contrary to [75] we do not use post processing on the predictions of the algorithm, and measure the accuracy directly on the output of the networks. As a baseline we also measured the geodesic error of correspondences obtained on 100 random pairs of the test set using nearest neighbors in the space of SHOT descriptors. We see that learning techniques vastly outperform the baseline. On the original set with shapes having the same connectivity MDGCNN and GCNN behave similarly. On the re-meshed set GCNN noticeably degrades while MDGCNN is able to maintain its precision. Figure 4.16 shows several examples of correspondences between pairs of shapes computed using different methods. Figure 4.15 also shows a comparison with PointNet++ [100] and Dynamic graph CNN [129] using 3D coordinates as input (trained for 200 epochs). Note that these methods are designed for segmentation tasks and are not suited for shape matching in their current form.

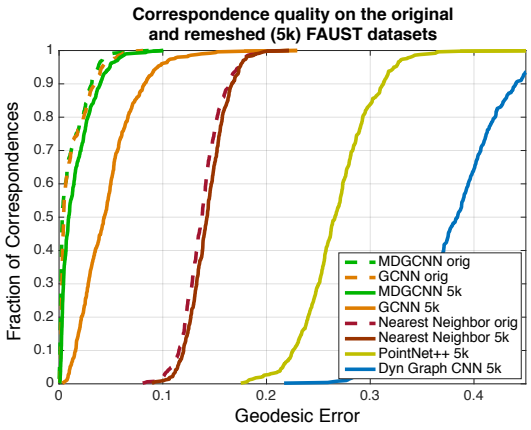


Figure 4.15 – Performance of shape correspondence on the FAUST dataset and its re-meshed version (5k) evaluated by plotting the fraction of correspondences within a geodesic radius of the ground truth. Higher curve corresponds to better performance.

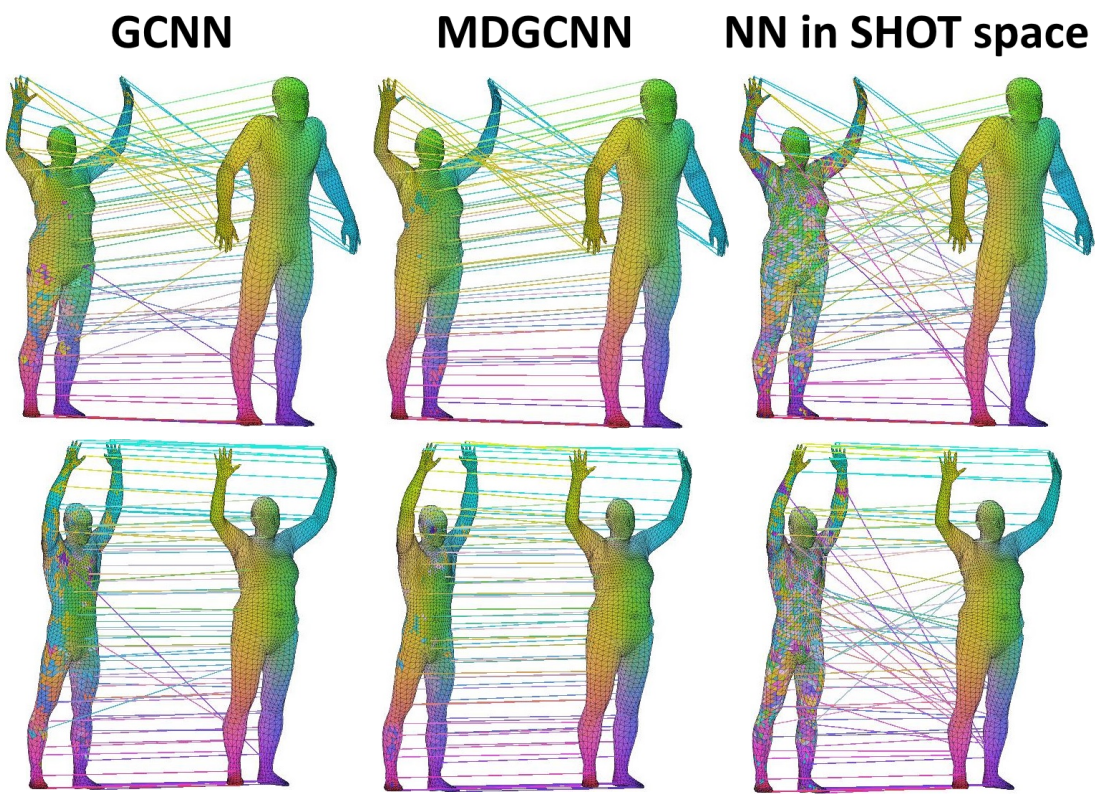
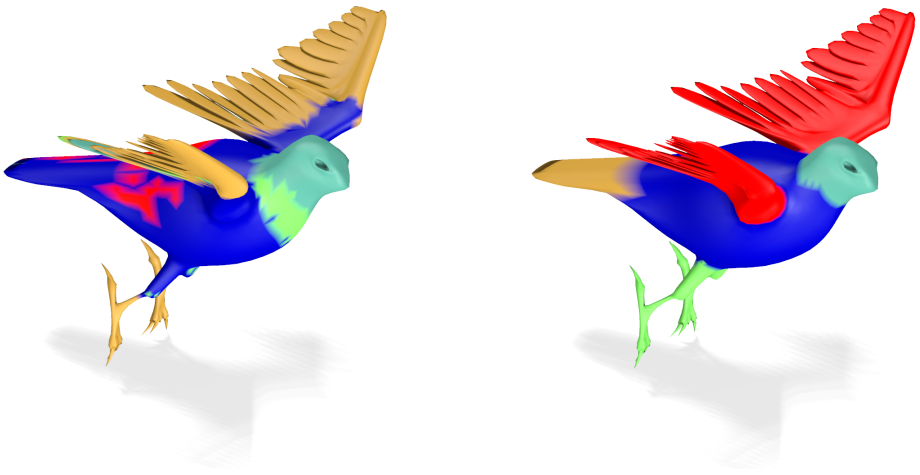


Figure 4.16 – Shape correspondence on the remeshed (5k) version of the FAUST dataset using : (left) GCNN [75] (middle) Our method MDGCNN, and (right) using nearest neighbors in the SHOT descriptors space.

4.6.6 Limitations & Future Work

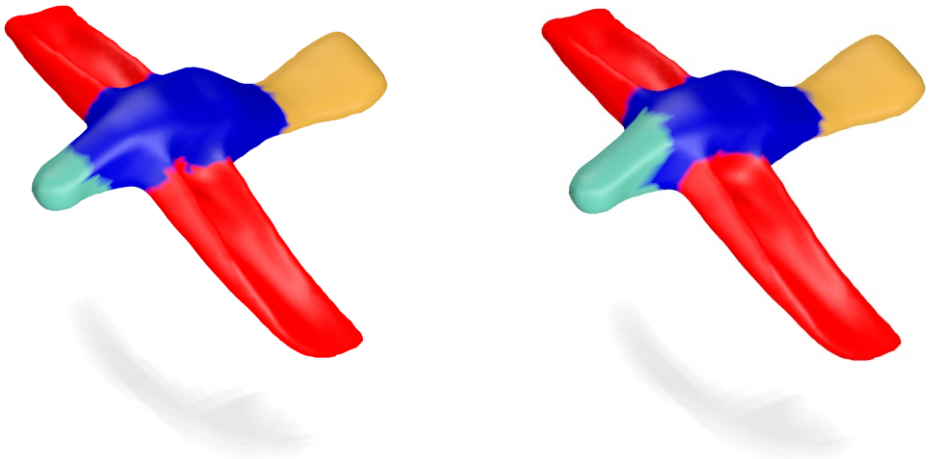
Our pipeline still has important limitations. The learning process depends on the construction of local coordinate systems which might not be suited to describe certain types of patterns possibly introducing a bottleneck to the learning. More specifically constructions such as geodesic polar coordinates and parallel transport are purely intrinsic based on the metric of the surface, therefore some areas that are close in the embedding space might be considered far in this representation. A typical limiting case of purely intrinsic pipelines such as ours is when some region of the shape is made of multiple parts that seem to merge in a single object, they might very well fail to recognize it as as a such. We illustrate this by applying our segmentation pipeline to the bird class of PSB dataset, shown in Figure 4.17. From a purely topological perspective the bird's wings are locally disconnected as they are made of many feathers. As we can see in Figure 4.17 our pipeline most likely recognized each independent feather as a bird tail, failing to recognize the wing in its entirety. Another limiting case of pipelines based on local coordinates is the presence of very thin nearly degenerate parts as the bird's legs and claws. Two dimensional systems of coordinates might not be appropriate to model near one dimensional parts.



(a) MDGCNN 3D (b) Ground truth

Figure 4.17 – Limiting case of purely intrinsic pipelines

On the other hand, Figure 4.18 shows that in the absence of such limiting cases a bird shape can be properly segmented by MDGCNN.



(a) MDGCNN 3D (b) Ground truth

Figure 4.18 – Successful segmentation with our MDGCNN approach.

Another limitation shared by CNNs for image processing is the choice of scale (size) of local windows. The same features can have different meaning depending on the scale at which they are detected. Since our current implementation only uses a single fixed scale, this may limit the generalization power of our method in situations where relative proportions of object parts are different from the examples in the training set. In image processing the scale issue has been addressed e.g. by using the notion of Inception modules [121]. We leave implementation of inception modules within the MDGCNN

framework for future work.

Our discretization also has some issues. For example, the parallel transport of a direction might fall between angular bins attached to the target vertex. For this reason we used linear interpolation between the two adjacent directions. However this still introduces a non-negligible error which grows with the number of layers especially for low number of angular bins. A possible alternative is to use the Fourier basis to represent directional signals at all points. This would allow exact direction transfer since rotations act linearly on the basis functions. However activation functions and operations such as angular max pooling would be harder to perform.

Perhaps the most immediate, and relatively straightforward extension of our work would be to use our multi directional approach in the context of local parameterizations via anisotropic diffusion kernels [16], but without assuming a canonical reference direction at every point. More generally, it would be interesting to use multi-scale approaches with different ways of computing the coordinate systems and transporting the information depending on the scale possibly using extrinsic information to help the network learning different semantic interpretations across different scales in order to improve the overall robustness. Finally other constructions of directional convolution with potentially stronger properties are possible, and we give an example in the Appendix Section 4.8.2.

4.7 Conclusion

In the this work we presented a novel approach to define convolution over curved surfaces that do not admit global or canonical coordinate systems. Namely, we proposed a way to align *local* systems of coordinates allowing to build and learn consistent filters that can then be naturally used across different domains. Our approach is built on the notion of directional functions, which generalize real-valued signals. We proposed a technique to convolve such directional functions with learned template (or filter) functions to produce new directional functions. This allows us to compose these convolution operations, without any loss of directional information across layers of a neural network.

We showed that our new approach compares favorably to its most direct analogues producing smoother and more robust results and can compete with more recent techniques, even when using "weak" input signals such as the 3D coordinates of the points. We believe that idea of multi-directional convolution can be generalized and can open the door to addressing many other situations where the data representation is ambiguous by allowing a neuron to have different ways of interpreting its input and to communicate with its contributors. For example, in our case a neuron can process its input depending on the choice of reference direction and propagate this information to its contributors. However, other types of alignment between layers can be thought of across different types of ambiguities in the signal as well.

4.8 Appendix

Alternative Proof of Proposition 4.3.2 Below, we provide an alternative proof showing that proposition 4.3.2 still holds when the directional convolution operator $\star$ is defined in the angular coordinate setting, which thus provides a more direct link to the practical setting. The family of unit vectors $e_x \in T_xX$ defines polar coordinate systems on each tangent plane T_xX. A tangent vector $p \in T_xX$ is then represented by a tuple (r, θ) where r is its radius and θ is the angle between e_x and $p = re_x(\theta)$ where $e_x(\theta)$ denote the direct rotation of e_x by angle θ. We represent unit vectors only by their angle. Adapting the notations of Eq. (4.3) in the polar coordinate systems defined by e we denote:

$$\begin{aligned}\tau^e_{x,\theta} &:= \tau_{x,e_x(\theta)}\\ \exp^X_{x,e_x}(r,\theta) &:= \exp^X_x(r.e_x(\theta))\end{aligned}$$

and $\Gamma^e_{x,(r,\theta_1)}(\theta_2)$ is the angle between $e_{\exp^X_{x,e_x}(r,\theta)}$ and $\Gamma_{x,re_x(\theta_1)}(e_x(\theta_2))$ i.e.

$$e_{\exp^X_{x,e_x}(r,\theta)}(\Gamma^e_{x,(r,\theta_1)}(\theta_2)) := \Gamma_{x,re_x(\theta_1)}(e_x(\theta_2))$$

We first observe that:

$$\begin{aligned}&\varphi_{R.e}(x,\theta) = \varphi_e(x,\theta+\theta_x),\\ &\tau^{R.e}_{x,\theta} = \tau^e_{x,\theta+\theta_x},\\ &\exp^X_{x,R_x.e_x}(r,\theta) = \exp^X_{x,e_x}(r,\theta+\theta_x).\\ &\Gamma^{R.e}_{x,\theta_1}(\theta_2) = \Gamma^e_{x,\theta_1+\theta_x}(\theta_2+\theta_x)\end{aligned}$$

We have:

$$\begin{aligned}&(R.e)_{\exp^X_{x,R.e_x}(r,\theta_1)}(\Gamma^{R.e}_{x,(r,\theta_1)}(\theta_2)) =\\ &e_{\exp^X_{x,e_x}(r,\theta_1+\theta_x)}(\Gamma^e_{x,(r,\theta_1+\theta_x)}(\theta_2+\theta_x)) =\\ &R.e_{\exp^X_{x,R.e_x}(r,\theta_1)}(\Gamma^e_{x,(r,\theta_1+\theta_x)}(\theta_2+\theta_x) - \theta_{\exp^X_{x,e_x}(r,\theta+\theta_x)})\end{aligned}$$

Thus:

$$\Gamma^{R.e}_{x,(r,\theta_1)}(\theta_2) = \Gamma^e_{x,(r,\theta_1+\theta_x)}(\theta_2+\theta_x) - \theta_{\exp^X_{x,e_x}(r,\theta+\theta_x)}$$

Therefore

$$\begin{aligned}&(\overline{\exp}^{X,R.e}_x)^*\varphi_{R.e}(x,(r,\theta))\\ &:= \varphi_{R.e}\big(\exp^X_{x,R_x.e_x}(r,\theta), \Gamma^{R.e}_{x,(r,\theta)}(\theta)\big)\\ &= \varphi_{R.e}(\exp^X_{x,e_x}(r,\theta+\theta_x), \Gamma^e_{x,(r,\theta+\theta_x)}(\theta+\theta_x) - \theta_{\exp^X_{x,e_x}(r,\theta+\theta_x)})\\ &= \varphi_e(\exp^X_{x,e_x}(r,\theta+\theta_x), \Gamma^e_{x,(r,\theta+\theta_x)}(\theta+\theta_x))\\ &=: (\overline{\exp}^{X,e}_x)^*\varphi_e(x,(r,\theta+\theta_x)).\end{aligned}$$

So that:

$$\begin{aligned}&(\varphi_{R.e}\star k)(x,\theta)\\ &= \langle(\overline{\exp}^{X,R.e}_x)^*\varphi_{R.e}, (\tau^{R.e}_{x,\theta})^*k\rangle.\\ &= \langle(\overline{\exp}^{X,e}_x)^*\varphi_e(\bullet,\bullet+\theta_x), (\tau^e_{x,\theta+\theta_x})^*k\rangle\\ &= (\varphi_e\star k)(x,\theta+\theta_x)\end{aligned}$$

4.8.1 Details on the implementation of multi-directional geodesic convolution

Our practical implementation of geodesic convolution relies on dense tensors for efficiency optimization. We describe it in the following subsection.

Let X be a triangle mesh with N_v vertexes. The exponential map over X is given by geodesic polar coordinates (GPC) around each vertex. We model it as two N_v by N_v matrices r and θ where r_{ij} and θ_{ij} represent the radius and angle at vertex j of the GPC centered at vertex i. The associated Euclidean coordinates $(x_{ij}, y_{ij})_{ij}$, extend inside triangles by linear interpolation. We use the GPC to construct windows at each vertex along which we can transfer signals on the mesh. The windows are defined by their radius R. The window attached to vertex i consists of the points of polar coordinates are $(\frac{j.R}{N_\rho}, \frac{2k.\pi}{N_\theta})_{j\in[|1,N_\rho|],k\in[|1,N_\theta|]}$ in the GPC at i. Each window vertex lies inside a triangle we denote by E_{ijkl} the index of the l-th vertex of the triangle containing p_{ijk} the jk-th point of i-th window and by W_{ijkl} the associated barycentric coordinate. That is:

$$p_{ijk} = \sum_{l=1}^{3} W_{ijkl}(x_{i,E_{ijkl}}, y_{i,E_{ijkl}}).$$

A a-dimensional signal f on the mesh consist of a N_v by a matrix, it can be pulled back to the window system by the following formula:

$$E^* f_{ijkl} := \sum_{m=1}^{3} W_{ijkm} f_{E_{ijkm},l}$$

this can be seen as a discretization of the pull back by exponential map. Template functions are stacked in ab-polar kernel tensors of shape (N_ρ, N_θ, a, b) to be convolved with a-dimensional signals. In our context we define the geodesic convolution of a a dimensional signal f by the a ab-polar kernel K as the b-dimensional directional signal:

$$(f \circledast K)_{ijk} := \sum_{r,m,l} E^* f_{irml} K_{r,(m+j) \bmod N_\theta,l,k}$$

We adapt the original definition of geodesic convolutional layer:

Definition 4.8.1 (geodesic convolutional layer) *The geodesic convolutional layer of ab-kernel K, central kernel C and bias vector B and activation function ξ transforms any a-dimensional signal f into the b-dimensional signal:*

$$\mathrm{gc}_{K,C,B,\xi}(f)_{ij} := \max_k \xi(f \circledast K_{ikj} + \sum_l C_{jl} f_{il} + B_j)$$

We model the parallel transport as a N_v by N_v matrix γ where γ_{ij} is the angular offset in the angular coordinates θ of the GPC induced by the parallel transport along the geodesic joining vertex i to vertex j. We discretize the transport of angles by the 4D tensor Γ of shape $(N_v, N_\rho, N_\theta, 3)$ defined by:

$$\Gamma_{ijkl} := \frac{\gamma_{i,E_{ijkl}}}{2\pi} + \frac{k}{N_\theta}$$

is the normalized angle representing the unit radial vector at jk-vertex of i-th window in its GPC. The tensor Γ stores exact angular values however since discretized directional functions are defined using N_θ evenly spaced angles we consider the lower and upper transport tensors $\lfloor\Gamma\rfloor$ and $\lceil\Gamma\rceil$. We can interpolate directional signals inside the resulting angular sectors by using the fractional part $\{\Gamma\} := \Gamma - \lfloor\Gamma\rfloor$. We define the discrete pull back of a directional signal φ and by the "completed" exponential map as the tensor:

$$\varphi(E,\Gamma)_{ijkl} := \sum_m W_{ijkm}\big((1-\{\Gamma\}_{ijkm})\varphi_{E_{ijkm},\lfloor\Gamma\rfloor_{ijkm},l} + \{\Gamma\}_{ijkm}\varphi_{E_{ijkm},\lceil\Gamma\rceil_{ijkm},l}\big)$$

We define the discrete directional geodesic convolution of a a-dimensional directional signal φ by the ab-polar kernel K as the b-dimensional directional signal:

$$\varphi \star K_{ijk} := \sum_{r,m,l} \varphi(E,\Gamma)_{irml} K_{r,(m+j) \bmod N_\theta,l,k}$$

Definition 4.8.2 (Directional geodesic convolutional layer) *The directional geodesic convolution of ab-polar kernel K, central kernel C and bias vector $B \in \mathbb{R}^b$ and activation function ξ transforms any a-dimensional directional signal φ to the b-dimensional directional signal:*

$$\mathrm{dir}_{K,C,B,\xi}(\varphi)_{ijk} := \xi(f \star K_{ijk} + \sum_l C_{kl} f_{ijl} + B_k)$$

4.8.2 A Stronger Notion of Directional Convolution

Several definition of directional convolution are possible we chose to transport tangent directions along geodesic to produce a local radial vector field on the surface which is rotation invariant this simplifies implementation. There is another natural choice to pull-back directional signals to the tangent plane. We can set:

$$\overline{\exp}^X_{x,v}(p) = (\exp^X_x(p),\ \Gamma_{x,p}(v)),$$

and define convolution of a directional function φ over X by a template k by:

$$(\varphi \star_X k)(x,v) = \langle(\overline{\exp}^X_{x,v})^*\varphi, \tau^*_{x,v}k\rangle$$

On the plane ($X = \mathbb{R}^2$) the above formula can be simplified, we have:

$$\overline{\exp}^{\mathbb{R}^2}_{x,v}(p) = (x+p, v)$$

To ease notation we identify $\mathbb{R}^2$ with $\mathbb{C}$. The map τ simply rotates the template:

$$\tau^*_{x,e^{i\theta}}k := k(e^{-i\theta}\bullet)$$

therefore for any directional function φ over $\mathbb{R}^2$:

$$(\varphi \star_{\mathbb{R}^2} k)(x, e^{i\theta}) = (\varphi(\bullet, e^{i\theta}) * k(e^{-i\theta}\bullet))(x)$$

Let f be a function over $\mathbb{R}^2$ denote by $(x, v) \mapsto \tilde{f}(x, v) := f(x)$ the associated directional function then:

$$(\tilde{f} \star_{\mathbb{R}^2} k)(x, e^{i\theta}) = (f * k)(e^{-i\theta}\bullet)(x)$$

The resulting directional function essentially stores the result of the usual convolution with the rotated kernel in each direction. Thanks to the above observation we can show that an image based CNN using this notion of directional convolution is equivalent to taking the maximal response under rotations of the kernels with its standard CNN counterpart as shown by the following proposition. For simplicity we consider one dimensional signals although the property still holds in higher dimensions.

Proposition 4.8.1 *Let f a function over $\mathbb{R}^2$, $(K_l)_l$ a sequence of template kernels, $(b_l)_l$ a sequence of real numbers $(\xi_l)_l$ a sequence of activation functions. We define the sequence $(\varphi_l)_l$ of directional functions over $\mathbb{R}^2$ and the sequence $(f_l^\theta)_l$ of function over $\mathbb{R}^2$ by:*

$$\begin{cases} \varphi_0 = \tilde{f} \\ \varphi_{l+1} = \xi_l \circ \left(\varphi_l \star_{\mathbb{R}^2} K_l + b_l\right) \end{cases} \quad \begin{cases} f_0^\theta = f \\ f_{l+1}^\theta = \xi_l \circ \left((f_l^\theta * K_l(e^{-i\theta}\bullet) + b_l\right) \end{cases}$$

Then for all n we have:

$$\varphi_n(x, e^{i\theta}) = f_n^\theta(x).$$

Proof We proceed by recurrence. The property is true for $n = 0$ by definition, suppose it is true for $n = k$. We have:

$$\begin{aligned} &\varphi_{k+1}(x, e^{i\theta}) \\ &= \xi_k(\varphi_k \star_{\mathbb{R}_2} K_k(x, e^{i\theta}) + b_k) \\ &= \xi_k((\varphi_k(\bullet, e^{i\theta}) * K_k)(e^{-i\theta}\bullet)(x) + b_k) \\ &= \xi_k((f_k^\theta * K_k(e^{-i\theta}\bullet))(x) + b_k) \\ &:= f_{k+1}^\theta(x) \end{aligned}$$

which proves the property for $n = k + 1$ and concludes the proof. □

4.8.3 Algorithm for computing geodesic polar coordinates (GPC) and parallel transport on triangle meshes

Below we give a formal description of the algorithm [78] for computing GPC and how we extended it for computing parallel transport, complementing the high level description from Section 4.5.

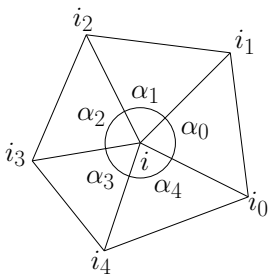


Figure 4.19 – One ring neighborhood of vertex i

To initialize the algorithm we must specify polar coordinates and parallel transport on the immediate neighborhood of each vertex i. Denote by $\{i_0, \dots i_{|\deg(i)|-1}\}$ the set of neighbours of i which is ordered. For each $j \in [|0, \deg(i) - 1|]$ we denote by α_{ij} the oriented angle between the halfedges e_{ii_j} and $e_{i,i_{j+1\% \deg(i)}}$ (Figure 4.19) and define:

$$\theta_{ii_j} := \frac{2\pi}{\sum_{j=0}^{\deg(i)-1} \alpha_{ij}} \sum_{k=0}^{j} \alpha_{ij}$$

For adjacent vertices $i \sim j$ we denote by r_{ij} the euclidean distance between i and j and $r_{ii} = 0$. If i and j are not adjacent we set $\tilde{r}_{ij} = +\infty$.

The parallel parallel transport of an angle u at vertex i to the a neighbour vertex j expressed in their respective polar coordinate systems is given by:

$$\rho_{ij}(u) := u + \theta_{ji} - \theta_{ij} - \pi$$

thus the offset $\rho_{ij} := \rho_{ij}(0)$ fully characterizes the parallel transport along the halfedge e_{ij}. The parallel transport of γ the angular coordinate is initialized in the neighborhood of i by setting:

$$\gamma_{ii_j} = \theta_{ii_j} + \rho_{ii_j}, \ \forall j \in [|0, \deg(i) - 1|].$$

The algorithm of [78] computes an extension $(\tilde{r}, \tilde{\theta})$ of the polar coordinates (r, θ) beyond the immediate neighborhood of a source vertex s, we extend it to compute an extension $\tilde{\gamma}$ of the parallel transport γ as well. Like other fast marching algorithms it iteratively updates vertices farther and farther away from the source based on a priority queue on vertices based on their distance to s (see Algorithm 3). Once a vertex is selected for update a subroutine is called to update its neighbourhood. Let i be the selected vertex for any neighbour $k \sim i$ and any of the two common neighbours $j \sim i, k$ such that $\tilde{r}_{sj} < +\infty$ the subroutine proposes updates for $\tilde{r}_{sk}, \tilde{\theta}_{sk}$ and $\tilde{\gamma}_{sk}$, call them r_{sijk}, θ_{sijk} and γ_{sijk}. To compute these updates, first embed the immediate neighborhood of vertex k into the plane using the polar coordinates (r, θ) for any $l \sim k$ this gives a point $v_l \in \mathbb{R}^2$ of polar coordinates (r_{kl}, θ_{kl}). Then a pseudo-source point $s' \in \mathbb{R}^2$ is built, uniquely defined by the conditions:

$$\begin{cases} l \in \{i, j\} \\ \|v_l - s'\|_2 = \tilde{r}_{s,t_{ik}} \\ ||v_k - s'||_2 \text{ is maximal} \end{cases}$$

The pseudo source s' is considered as the new origin and the coordinates of the $(v_l)_l$'s are updated accordingly. Two cases may occur as shown on Figure 4.20 below:

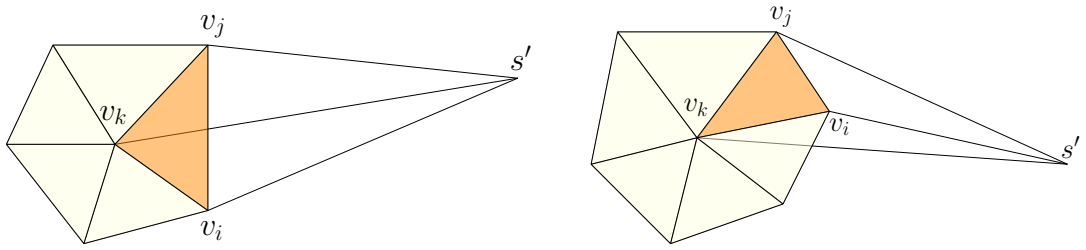


Figure 4.20 – First case (left): v_k is in the positive cone spanned by v_i and v_j. Second case (right) v_k is in not the positive cone spanned by.

In the fist case define $(\tilde{r}_{sijk}, t_{sijk})$ to be the polar coordinates of v_k otherwise set:

$$\begin{cases} \tilde{r}_{sijk} := \min_{l \in i,j} \tilde{r}_{sl} + r_{kl} \\ t_{sijk} := \tilde{\theta}_{sl}, \text{ for } \tilde{r}_{sijk} = \tilde{r}_{sl} + r_{kl} \end{cases}$$

Let t_i be the angular coordinate of v_i and t_j be the angular coordinate of v_j, we may assume for simplicity that $t_i < t_j$ otherwise swap i and j, furthermore we assume $|t_j - t_i| \leqslant \pi$ otherwise add 2π to t_i and swap i and j, finally assume $t_{sijk} \in [t_i, t_j]$ otherwise add 2π. Then set $\alpha = \frac{t_{sijk} - t_i}{t_j - t_i}$ we have:

$$t_{sijk} = (1 - \alpha)t_i + \alpha t_j.$$

Algorithm 2 below computes convex combination of angles:

Input: Angles a, b and $t \in [0, 1]$
Output: Calculate $((1-t)a + tb) \mod 2\pi$

```
if a > b then
    swap(a, b);
    t ← 1 − t;
end
if b − a > π then
    a+ = 2π;
    swap(a, b);
    t ← 1 − t;
end
return ((1 − t)a + tb) mod 2π
```

Algorithm 2: Computing a convex combination of two angles a and b.

The angular coordinate update is given by:

$$\tilde{\theta}_{sijk} := (1 - \alpha)\tilde{\theta}_{si} + \alpha\tilde{\theta}_{sj} \mod 2\pi$$

We define the parallel transport update similarly by:

$$\tilde{\gamma}_{sijk} := (1 - \alpha)(\tilde{\gamma}_{si} + \rho_{ik}) + \alpha(\tilde{\gamma}_{sj} + \rho_{jk}) \mod 2\pi$$

where the above expressions are understood according to Algorithm 2.

The general algorithm for computing the GPC and parallel transport is given by:

Input: GPC and parallel transport on the immediate neighborhoods r, θ, γ and a starting vertex s.

Output: Extensions $\tilde{r}, \tilde{\theta}, \tilde{\gamma}$ of r, θ, γ beyond the immediate neighborhood of s

Initialization:
for *each vertices* i, j **do**
  if $i \sim j$ *or* $i = j$ **then**
    $\tilde{r}_{ij} \leftarrow r_{ij},\ \tilde{\theta}_{ij} \leftarrow \theta_{ij},\ \tilde{\gamma}_{ij} \leftarrow \gamma_{ij}$;
  else
    $\tilde{r}_{ij} \leftarrow +\infty,\ \tilde{\theta}_{ij} \leftarrow 0,\ \tilde{\gamma}_{ij} \leftarrow 0$;
  end
end
candidates $= [j,\ j \sim s]$; `/* sort` s `neighbors w.r.t.` $\tilde{r}$ `*/`
while candidates.notEmpty() **do**
  $i =$ candidates.pop(); `/* pick smallest vertex w.r.t.` r `and remove it from candidates */`
  for $k \sim i$ **do**
    let $j \sim i, k$ st $\tilde{r}_{si}, \tilde{r}_{sj} < +\infty$ minimising $\tilde{r}_{sijk}$;
    `/* precision` $\epsilon > 0$ `*/`
    if $\tilde{r}_{sk} > (1+\epsilon).\tilde{r}_{sijk}$ **then**
      `/* update estimates */`
      $\tilde{r}_{sk} \leftarrow \tilde{r}_{sijk}$;
      $\tilde{\theta}_{sk} \leftarrow \tilde{\theta}_{sijk}$;
      $\gamma_{sk} \leftarrow \tilde{\gamma}_{sijk}$; `/* extend transport */`
      `/* if stopping criterion is not met */`
      if $\tilde{r}_{sk} \leqslant R_{\max}$ **then**
        candidates.push(k); `/* add` k `to candidates */`
      end
    end
  end
end

Algorithm 3: Compute discrete Geodesic Polar Coordinates (GPC) $(\tilde{r}_s, \tilde{\theta}_s)$ and parallel transport $\tilde{\gamma}_s$ along geodesics on a mesh starting from vertex s.

Chapter 5

Effective Rotation-invariant Point CNN with Spherical Harmonics Kernels

We present a novel rotation invariant architecture operating directly on point cloud data. We demonstrate how rotation invariance can be injected into a recently proposed point-based PCNN architecture, on all layers of the network. This leads to invariance to both global shape transformations, and to local rotations on the level of patches or parts, useful when dealing with non-rigid objects. We achieve this by employing a spherical harmonics-based kernel at different layers of the network, which is guaranteed to be invariant to rigid motions. We also introduce a more efficient pooling operation for PCNN using space-partitioning data-structures. This results in a flexible, simple and efficient architecture that achieves accurate results on challenging shape analysis tasks, including classification and segmentation, without requiring data-augmentation typically employed by non-invariant approaches[1].

5.1 Introduction

Analyzing and processing 3D shapes using deep learning approaches has recently attracted a lot of attention, inspired in part by the success of such methods in computer vision and other fields. While early approaches in this area relied on methods developed in the image domain, e.g. by sampling 2D views around the 3D object [118], or using volumetric convolutions [137], recent methods have tried to directly exploit the 3D structure of the data. This notably includes both mesh-based approaches that operate on the surface of the shapes [74, 83] or the approach we presented in chapter 4, and point-based techniques that only rely on the 3D coordinates of the shapes without requiring any connectivity information [98, 100].

Point-based methods are particularly attractive, being both very general and often more efficient, as they do not require maintaining complex and expensive data-structures, compared to volumetric or mesh-based methods. As a result, starting from the seminal works of PointNet [98] and PointNet++ [100], many point-based learning approaches have been proposed, often achieving remarkable accuracy in tasks such as shape classification and segmentation among many others. A key challenge when applying these methods in practice, however, is to ensure *invariance* to different kinds of transformations, and especially to rigid motions. Common strategies include either using spatial transformer blocks

[1] Code and data are provided on the project page https://github.com/adrienPoulenard/SPHnet.

[52] as done in the original PointNet architecture and its extensions, or applying extensive *data augmentation* during training to learn invariance from the data. Unfortunately, when applied to shape collections that are not pre-aligned, these solutions can be expensive, requiring unnecessarily long training. Moreover, they can even be incomplete when *local* rotation invariance is required, e.g. for non-rigid shapes, undergoing articulated motion, which is difficult to model through data augmentation alone.

In this chapter, we propose a different approach for dealing with both global and local rotational invariance for point-based 3D shape deep learning tasks. Instead of learning invariance from data, we propose to use a different kernel that is theoretically guaranteed to be invariant to rotations, while remaining informative. To achieve this, we leverage the recent PCNN by extension operators [6], which provides an efficient framework for point-based convolutions. We extend this approach by introducing a rotationally invariant kernel and making several modifications for improved efficiency. We demonstrate on a range of difficult experiments that our method can achieve high accuracy directly, without relying on data augmentation.

5.2 Related work

A very wide variety of learning-based techniques have been proposed for 3D shape analysis and processing. Below we review methods most closely related to ours, focusing on point-based approaches, and various ways of incorporating rotational invariance and equivariance in learning. We refer the interested readers to several recent surveys, e.g. [138, 83], for an in-depth overview of geometric deep learning methods.

Learning in Point Clouds. Learning-based approaches, and especially those based on deep learning, have recently been proposed specifically to handle point cloud data. The seminal PointNet architecture [98] has inspired a large number of extensions and follow-up works, notably including PointNet++ [100] and Dynamic Graph CNNs [129] for shape classification and segmentation. More recent works include PCPNet [47] for normal and curvature estimation, PU-Net [141] for point cloud upsampling, and PCN for shape completion [142] among many others.

While the original PointNet approach is not based on a convolutional architecture, instead using a series of classic MLP fully connected layers, several methods have also tried to define and exploit meaningful notions of convolution on point cloud data, inspired by their effectiveness in computer vision. Such approaches notably include: basic pointwise convolution through nearest-neighbor binning and a grid kernel [50]; Monte Carlo convolution, aimed at dealing with non-uniformly sampled point sets [49]; learning an $\mathcal{X}$-transformation of the input point cloud, which allows the application of standard convolution on the transformed representation [67]; and using extension operators for applying point convolution [6]. These techniques primarily differ by the notion of neighborhood and the construction of the kernels used to define convolution on the point clouds. Most of them, however, share with the PointNet architecture a lack of support for invariance to rigid motions, mainly because their kernels are applied to point coordinates, and defining invariance at the level of individual points is generally

not meaningful.

Invariance to transformations. Addressing invariance to various transformation classes has been considered in many areas of Machine Learning, and Geometric Deep Learning in particular. Most closely related to ours are approaches based on designing *steerable filters*, which can learn representations that are equivariant to the rotation of the input data [133, 132, 3, 135]. A particularly comprehensive overview of the key ideas and results in this area is presented in [60]. In closely related works, Cohen and colleagues have proposed group equivariant networks [27] and rotation-equivariant spherical CNNs [28]. While the theoretical foundations of these approaches are well-studied in the context of 3D shapes, they have primarily been applied to either volumetric [132] or spherical (e.g. by projecting shapes onto an enclosing sphere via ray casting) [28] representations. Instead, we apply these constructions directly in an efficient and powerful *point*-based architecture.

Perhaps most closely related to ours are two very recent unpublished methods, [140, 124], that also aim to introduce invariance into point-based networks. Our approach is different from both, since unlike the PRIN method in [140] our convolution operates directly on the point clouds, thus avoiding the construction of spherical voxel space. As we show below, this gives our method greater invariance to rotations and higher accuracy. At the same time while the authors of [124] explore similar general ideas and describe related constructions, including the use of spherical harmonics kernels, they do not describe a detailed architecture, and only show results with dozens of points (the released implementation is also limited to toy examples), rendering both the method and its exact practical utility unclear. Nevertheless, we stress that both [140] and [124] are very recent unpublished methods and thus concurrent to our approach.

Contribution: Our key contributions are as follows:

1. We develop an effective rotation invariant point-based network. To the best of our knowledge, ours is the first such method achieving higher accuracy than PointNet++ [100] *with data augmentation* on a range of tasks.

2. We significantly improve the efficiency of PCNN by Extension Operators [6] using space partitioning.

3. We demonstrate the efficiency and accuracy of our method on tasks such as shape classification, segmentation and matching on standard benchmarks and introduce a novel dataset for RNA molecule segmentation.

5.3 Background

In this section we first introduce the main notation and give a brief overview of the PCNN approach [6] that we use as a basis for our architecture.

5.3.1 Notation

We use the notation from [6]. In particular, we use tensor notation: $a \in \mathbb{R}^{I \times I \times J \times L \times M}$ and the sum of tensors $c = \sum_{ijl} a_{ii'jlm} b_{ijl}$ is defined by the free indices: $c = c_{i'm}$. $C(\mathbb{R}^3, \mathbb{R}^K)$ represent the collections of volumetric functions $\psi : \mathbb{R}^3 \to \mathbb{R}^K$.

5.3.2 The PCNN framework

The PCNN framework consists of three simple steps. First a signal is extended from a point cloud to $\mathbb{R}^3$ using an extension operator $\mathcal{E}_X$. Then standard convolution on volumetric functions O is applied. Finally, the output is restricted to the original point cloud with a restriction operator $\mathcal{R}_X$. The final convolution on point clouds is defined as:

$$O_X = \mathcal{R}_X \circ O \circ \mathcal{E}_X \tag{5.1}$$

In the original work [6], the extension operators and kernel functions are chosen so that the composition of the three operations above, using Euclidean convolution in $\mathbb{R}^3$ can be computed in closed form.

Extension operator. Given an input signal represented as J real-valued functions $f \in \mathbb{R}^{I \times J}$ defined on a point cloud, it can be extended to $\mathbb{R}^3$ via a set of volumetric basis functions $l_i \in C(\mathbb{R}^3, \mathbb{R})$ using the values of f at each point f_i. The extension operator $\mathcal{E}_X : \mathbb{R}^{I \times J} \to C(\mathbb{R}^3, \mathbb{R}^J)$ is then:

$$(\mathcal{E}_X[f])_j(x) = \sum_i f_{ij} l_i(x) \tag{5.2}$$

The authors of [6] use Gaussian basis functions centered at the points of the point cloud so that the number of basis functions equals the number of points.

Convolution operator. Given a kernel $\kappa \in C(\mathbb{R}^3, \mathbb{R}^{J \times M})$, the convolution operator $O : C(\mathbb{R}^3, \mathbb{R}^J) \to C(\mathbb{R}^3, \mathbb{R}^M)$ applied to a volumetric signal $\psi \in C(\mathbb{R}^3, \mathbb{R}^J)$ is defined as:

$$(O[\psi])_m(x) = (\psi * \kappa)_m(x) = \int_{\mathbb{R}^3} \sum_j \psi_j(y) \kappa_{jm}(x - y) dy \tag{5.3}$$

The kernel can be represented in an RBF basis:

$$\kappa_{jm}(z) = \sum_l k_{jml} \Phi(|z - v_l|), \tag{5.4}$$

where k_{jml} are learnable parameters of the network, Φ is the Gaussian kernel and $v_{l=1}^L$ represent translation vectors in $\mathbb{R}^3$. For instance, they can be chosen to cover a standard $3 \times 3 \times 3$ grid.

Restriction operator. The restriction operator $R_X : C(\mathbb{R}^3, \mathbb{R}^J) \to \mathbb{R}^{I \times J}$ is defined as simply as the restriction of the volumetric signal to the point cloud:

$$(R_X[\psi])_{i,j} = \psi_j(x_i) \tag{5.5}$$

Architecture With these definitions in hand, the authors of [6] propose to stack a series of convolutions with non-linear activation and pooling steps into a robust and flexible deep neural network architecture showing high accuracy on a range of tasks.

5.4 Our approach

Overview Our main goal is to extend the PCNN approach to develop a rotation invariant convolutional neural network on point clouds. We call our network SPHNet due to the key role that the spherical harmonics kernels play in it. Figure 5.1 gives an overview. Following PCNN we first extend a function on point clouds to a volumetric function by the operator $\mathcal{E}_X$. Secondly, we apply the convolution operator SPHConv to the volumetric signal. Finally, the signal is restricted by R_X to the original point cloud.

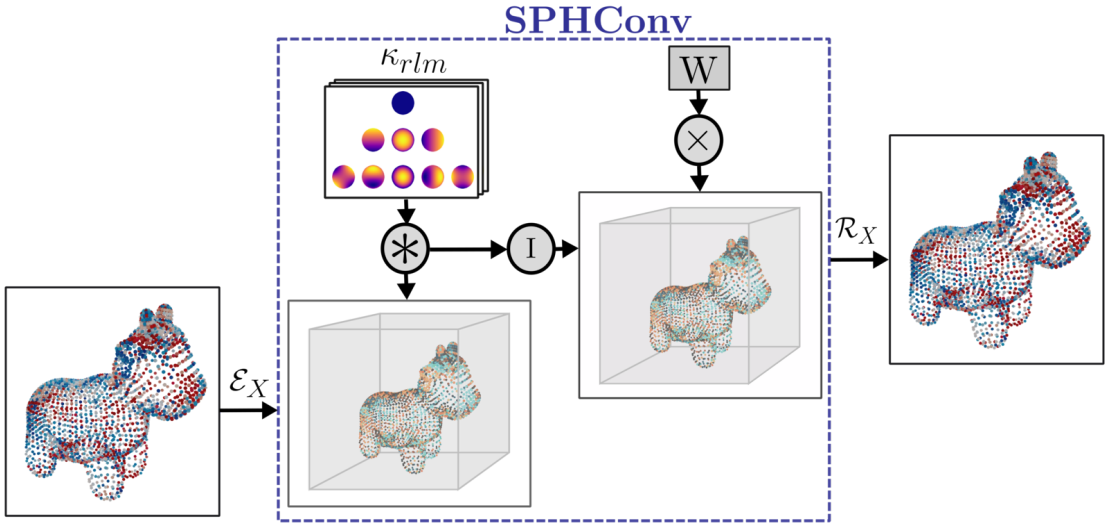


Figure 5.1 – SPHNet framework. A signal on point cloud is first extended to $\mathbb{R}^3$ (left). Our convolution operator is applied to the extended function (center). The signal is restrained to the original point cloud (right).

5.4.1 Spherical harmonics kernel

In this work, we propose to use spherical harmonics-based kernels to design a point-based rotation-invariant network. In [132], the authors define spherical harmonics kernels with emphasis on rotation equivariance, applied to volumetric data. We adopt these constructions to our setting to define rotation-invariant convolutions.

Spherical harmonics is a family of real-valued functions on the unit sphere which can be defined, in particular, as the eigenfunctions of the spherical Laplacian. Namely, the ℓ^{th} spherical harmonic space has dimension $2\ell + 1$ and is spanned by spherical harmonics $(Y_{\ell,m})_{\ell>=0, m\in -\ell \dots \ell}$, where ℓ is the degree. Thus, each $Y_{\ell,m}$ is a real-valued function on the sphere $Y_{\ell,m} : \mathcal{S}_2 \to \mathbb{R}$.

Spherical harmonics are rotation equivariant. For any rotation $R \in \mathrm{SO}(3)$ we have:

$$Y_{\ell,m}(Rx) = \sum_{n=-\ell}^{\ell} D^{\ell}_{mn}(R) Y_{\ell,n}(x). \tag{5.6}$$

Where $D^{\ell}(R)$ is the so-called Wigner matrix of size $2\ell+1 \times 2\ell+1$, [134]. Importantly, Wigner matrices are *orthonormal* for all ℓ making the norm of every spherical harmonic space invariant to rotation. This classical fact has been exploited, for example in [55] to define global rotation-invariant shape descriptors. More generally the idea of using bases of functions having a predictable behaviour under rotation to define invariant or equivariant filters or descriptors is closely related to the classical concept of steerable filters [42].

The spherical harmonic kernel basis introduced in [132] is defined as:

$$\kappa_{r\ell m}(x) = \exp\left(-\frac{\left|\|x\|_2 - \rho\frac{r}{n_R-1}\right|^2}{2\sigma^2}\right) Y_{\ell,m}\left(\frac{x}{\|x\|_2}\right), \tag{5.7}$$

where, ρ is a positive scale parameter, n_R is the number of radial samples and $\sigma = \frac{\rho}{n_R-1}$. Note that the kernel depends on a radial component, indexed by $r \in 0...n_R-1$, and defined by Gaussian shells of radius $r\frac{\rho}{n_R-1}$, and an angular component indexed by ℓ, m with $\ell \in 0...n_L-1$ and $m \in -\ell...\ell$, defined by values of the spherical harmonics.

This kernel inherits the behaviour of spherical harmonics under rotation, that is:

$$\kappa_{r\ell m}(Rx) = \sum_{n=-\ell}^{\ell} D^{\ell}_{mn}(R)\kappa_{r\ell n}(x) \tag{5.8}$$

where $R \in SO(3)$.

5.4.2 Convolution layer

Below, we describe our SPHNet method. To define it we need to adapt the convolution and extension operators used in PCNN [6], while the restriction operators are kept exactly the same.

Extension. PCNN uses Gaussian basis functions to extend functions to $\mathbb{R}^3$. However, convolution of the spherical harmonics kernel with Gaussians does not admit a closed-form expression. Therefore, we "extend" a signal f defined on a point cloud via a weighted combination of Dirac measures. Namely $\mathcal{E}_X : \mathbb{R}^{I\times J} \to C(\mathbb{R}^3, \mathbb{R}^J)$ is:

$$(\mathcal{E}_X[f])_j = \sum_i f_{ij}\omega_i \delta_{x_i}, \tag{5.9}$$

where we use the weights: $\omega_i = 1/(\sum_j \exp(-\frac{||x_j - x_i||^2}{2\sigma^2}))$. The Dirac measure has the following property:

$$\int_X f(y)\delta_x(y)\,\mathrm{d}y = f(x) \tag{5.10}$$

Convolution. We first introduce a non-linear rotation-invariant convolution operator: SPHConv. Using Eq. (5.10), the convolution between an extended signal and the spherical harmonic kernel $S[f] : C(\mathbb{R}^3, \mathbb{R}^J) \to C(\mathbb{R}^3, \mathbb{R}^J)$ is given by:

$$(S[f])_j(x) = (\mathcal{E}_X[f] * \kappa_{r\ell m})_j(x) = \sum_i f_{ij}\omega_i \kappa_{r\ell m}(x_i - x) \tag{5.11}$$

Using Eq. (5.8), we can express the convolution operator when a rotation R is applied to the point cloud X as a function of the kernel functions:

$$\begin{aligned}(R(\mathcal{E}_X[f] * \kappa_{r\ell m}))_j(x) &= \sum_i f_{ij}\omega_i \kappa_{r\ell m}(R(x_i - x)) \\ &= \sum_{n=-\ell}^{\ell} D^{\ell}_{mn}(R) \sum_i f_{ij}\omega_i \kappa_{r\ell n}(x_i - x) \\ &= \sum_{n=-\ell}^{\ell} D^{\ell}_{mn}(R)(\mathcal{E}_X[f] * \kappa_{r\ell n})_j(x)\end{aligned} \tag{5.12}$$

We observe that a rotation of the point cloud induces a rotation in feature space. In order to ensure rotation invariance we recall that the Wigner matrices D^l are all orthonormal. Thus, by taking the norm of the convolution with respect to the degree of the spherical harmonic kernels, we can gain independence from R. To see this, note that thanks to Eq. (5.12) only the m-indexed dimension of $(\mathcal{E}_X[f] * \kappa_{r\ell m})_j(x)$ is affected by the rotation. Therefore, we define our rotation-invariant convolution operator $I_{rl}[X, f] : C(\mathbb{R}^3, \mathbb{R}^J) \to C(\mathbb{R}^3, \mathbb{R}^J)$ as:

$$(I_{rl}[X, f])_j(x) = \|(\mathcal{E}_X[f] * \kappa_{r\ell m})_j(x)\|_2^m, \tag{5.13}$$

where for a tensor T_{rlmj} we use the notation $\|T\|_2^m$ to denote a tensor T^* obtained by taking the L_2 norm along the m dimension: $T^*_{rlj} = \sqrt{\sum_m T^2_{rlmj}}$.

Importantly, unlike the original PCNN approach [6], we cannot apply learnable weights directly on $(\mathcal{E}_X[f] * \kappa_{r\ell m})_j(x)$ as the result would not be rotation invariant. Instead, we take a linear combination of $(I_{rl}[X, f])$, obtained after taking the reduction along the m dimension above, using learnable weights $W \in \mathbb{R}^{G \times I \times n_R \times r_L}$, where G is the number of output channels. This leads to:

$$(O[f])_g(x) = \xi\left(\sum_{jr\ell} W_{gjr\ell}(I_{rl}[X, f])_j(x) + b_g\right) \tag{5.14}$$

Finally, we define the convolution operator $O_X : C(\mathbb{R}^3, \mathbb{R}^J) \to C(\mathbb{R}^3, \mathbb{R}^G)$ by restricting the result to the point cloud as in Eq. (5.5):

$$((O_X)_g[f])_i = \xi\left(\sum_{jr\ell} W_{gjr\ell}(I_{rl}[X, f](x_i) + b_g\right) \tag{5.15}$$

5.4.3 Pooling and upsampling

In addition to introducing a rotation-invariant convolution into the PCNN framework, we also propose several improvements, primarily for computational efficiency.

The original PCNN work [6] used Euclidean farthest point sampling and pooling in Voronoi cells. Both of these steps can be computationally inefficient, especially when handling large datasets and with data augmentation. Instead, we propose to use a space-partitioning data-structure for both steps using constructions similar to those in [58]. For this, we start by building a kd-tree for each point cloud, which we then use as follows.

Pooling. For a given point cloud P, our pooling layer of depth k reduces P from size 2^n to 2^{n-k} by applying max pooling to the features of each subtree. The coordinates of the points of the subtree are averaged. The resulting reduced point cloud kd-tree structure and indexing can be immediately computed from the one computed for P. This gives us a family $T_{k\in 1\dots n}$ of kd-trees of varying depths.

Upsampling. The upsampling layer is computed simply by repeating the features of each point of a point cloud at layer k using the kd-tree of structure T_k. The upsampled point cloud follows the structure of T_{k+1}.

Comparison to PCNN. In PCNN pooling is performed through farthest point sampling. The maximum over the corresponding Voronoi cell is then assigned to each point of the sampled set. This method has a complexity of $O(|P|^2)$ while ours has a complexity of $O(|P| \log^2 |P|)$, leading to noticeable improvement in practice.

We remark that kd-tree based pooling breaks full rotation invariance of our architecture, due to the construction of axis-aligned separating hyperplanes. However, as we show in the results below, this has a very mild effect in practice and our approach has very similar performance regardless of the rotation of the data-set. A possible way to circumvent this issue would be to modify the kd-tree construction by splitting the space along local PCA directions.

5.5 Architecture and implementation details

We adapted the classification and segmentation architectures from [6]. Using these as generic models we derive three general networks: our main rotation-invariant architecture **SPHnet**, which stands for Spherical Harmonic Net and uses the rotation invariant convolution we described. We also compare it to two baselines: **SPHBase** is identical to **SPHnet**, except that we do not take the norm of each spherical harmonic component and apply the weights directly instead. We use this as the closest non-invariant baseline. We also compare to PCNN (mod), which is also non rotation-invariant. It uses the Gaussian kernels from the original PCNN, but employs the same architecture and pooling as we use in **SPHnet** and **SPHBase**.

The original architectures in [6] consist of arrangements of convolution blocks, pooling and up-sampling layers that we replaced by our own. We kept the basic structure described below. In all cases, we construct the convolutional block using one convolutional layer followed by a batch normalization layer and a ReLU non linearity. Our convolution layer depends on the following parameters:

- Number of input and output channels
- Number n_L of spherical harmonics spaces in our kernel basis
- Number n_R of spherical shells in our kernel basis
- The kernel scale $\rho > 0$

For the sake of efficiency we also restrict the computation of convolution to a fixed number of points using k-nearest neighbor patches. Note that unlike other works, e.g. [129] we do not learn an embedding for the intermediate features, as they are always defined on the original point cloud. Thus, to ensure full invariance we apply our rotation invariant convolution at every layer.

The number of input channels is deduced from the number of channels of the preceding layer. We used 64 points per patch in the classification case and 48 in the segmentation case. We fixed $n_L = 4$ and $n_R = 2$ throughout all of our experiments. The scale factor ρ can be defined only for the first layer and deduced for the other ones as will be explained below.

5.5.1 Classification

The classification architecture is made of 3 convolutional blocks with 64, 256, 1024 output channels respectively: the first two are followed by a max pooling layer of ratio 4, and the last one if followed by a global max pooling layer. Finally we have a fully connected block over channels composed of two layers with 512 and 256 units, each followed by a dropout layer of rate 0.5 and a final softmax layer for the classification as shown in Figure 5.2.

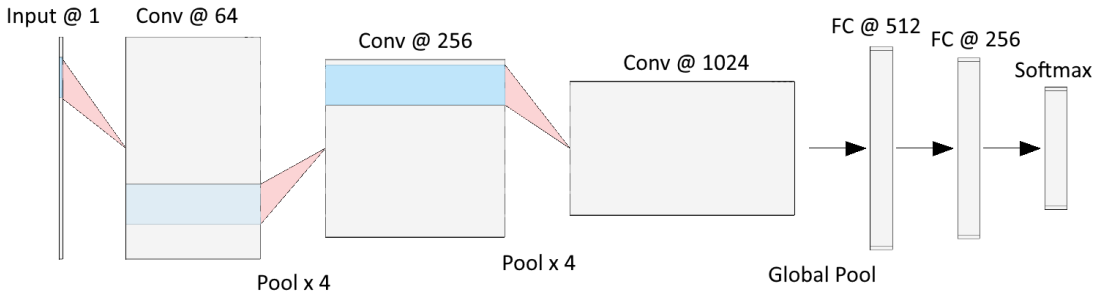


Figure 5.2 – Our classification architecture. Conv @ k indicates a conv layer with k output channels, Pool $\times k$ is a pooling of factor k and FC stands for fully connected layer.

We use $\rho = 0.1$ for the first layer and 2ρ, 4ρ for the second and third layers. We illustrate the importance of the scale parameter ρ by showing its impact on classification

accuracy (see Appendix 5.4). The classification architecture we use expects a point cloud of 1024 points as input and defines the convolution layers on it according to our method. We use the constant function equal to 1 as the input feature to the first layer. Note that, since our goal is to ensure rotation invariance, we cannot use coordinate functions as input.

5.5.2 Segmentation

Our segmentation network takes as input a point cloud with 2048 points and, similarly to the classification case, we use the constant function equal to 1 as the input feature to the first layer. Our segmentation architecture is shown in Figure 5.3.

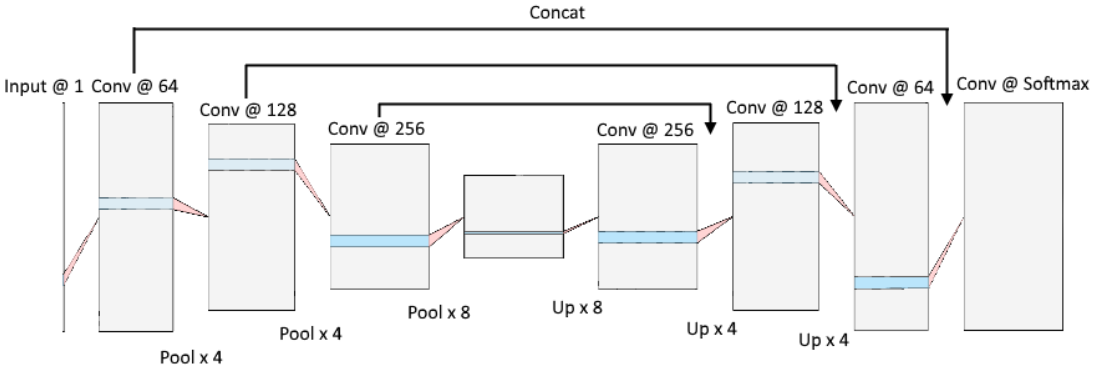


Figure 5.3 – Segmentation architecture. Conv @ k indicates a conv layer with k output channels, Pool $\times k$ is a pooling of factor k and Up $\times k$ is an upsampling by a factor k.

The architecture is U-shaped consisting of an encoding and a decoding block, each having 3 convolutional blocks. The encoding convolutional blocks have 64, 128, 256 output channels respectively. The first two are followed by max pooling layers of ratio 4 and the third by a pooling layer of ratio 8. These are followed by a decoding block where each conv block is preceded by an up-sampling layer to match the corresponding encoding block, which is then concatenated with it. The final conv-layer with softmax activation is then applied to predict pointwise labels. The two last conv-blocks of the decode part are followed by dropout of rate 0.5. We chose a scale factor of $\rho = 0.08$ for the first convolutional layer, the two next layers in the encoding block having respective scale factors 2ρ and 4ρ. The scale factors for the decode block are the same in reverse order. The scale factor for the final conv layer is ρ.

5.6 Results

5.6.1 Classification

We tested our method on the standard ModelNet40 benchmark [137]. This dataset consists of 9843 training and 2468 test shapes in 40 different classes, such as guitar, cone,

laptop etc. We use the same version as in [6] with point clouds consisting of 2048 points centered and normalized so that the maximal distance of any point to the origin is 1. We randomly sub-sample the point clouds to 1024 points before sending them to the network. The dataset is aligned, so that all point clouds are in canonical position.

We compare the classification accuracy of our approach SPHNet with different methods for learning on point clouds in Table 5.1. In addition to the original PCNN architecture and our modified versions of it, we compare to PointNet++ [100]. We also include the results of the recent rotation-invariant framework PRIN [140].

We train all different models in two settings: we first train with the original (denoted by 'O') dataset and also with the dataset augmented by random rotations (denoted by 'A'). We then test with either the original testset or the testset augmented by rotations, again denoted by 'O' and 'A' respectively.

We observe that while other methods have a significant drop in accuracy when trained with rotation augmentation, our method remains stable, and in particular outperforms all methods trained and tested with data augmentation (A/A column), implying that the orientation of different shapes is not an important factor in the learning process. Moreover, our method achieves the best accuracy in this augmented training set setting. For PRIN evaluation, we used the architecture for classification described in [140] trained for 40 epochs as suggested in the paper. In all our experiments we train the PRIN model with the default parameters except for the bandwidth, which we set to 16 in order to fit within the 24GB of memory available in Titan RTX. As demonstrated in [140] this parameter choice produces slightly lower results but they are still comparable. In our experiments, we observed that PRIN achieves poor performance when trained with rotation augmented data.

We also remark that even when training with data augmentation, our method converges significantly faster and to higher accuracies since it does not need to learn invariance to rotations. We show the evolution of the validation accuracy curves for different methods in Figure 5.4.

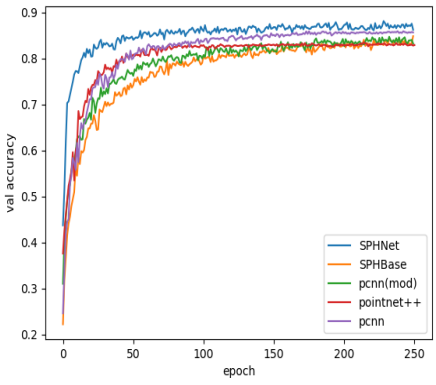


Figure 5.4 – Validation accuracy for ModelNet40 classification with rotation augmentation at training.

Method	O / O	A / O	O / A	A / A	time
PCNN	**92.3**	85.9	11.9	85.1	264 s
PointNet++	91.4	84.0	10.7	83.4	78 s
PCNN (mod)	91.1	83.4	9.4	84.5	22.6 s
SPHBaseNet	90.7	82.8	10.1	84.8	25.5 s
SPHNet (ours)	87.7	**87.1**	**86.6**	**87.6**	25.5 s
PRIN	71.5	0.78	43.1	0.78	811 s

Table 5.1 – Classification accuracy on the modelnet40 dataset. A stands for data augmented by random rotations and O for original data. E.g., the model A/O was trained with augmented and tested on the original data. Timings per epoch are given when training on a NVIDIA RTX 2080 Ti card.

5.6.2 Segmentation

We also applied our approach to tackle the challenging task of segmenting molecular surfaces into functionally-homologous regions within RNA molecules. We considered a family of 640 structures of 5s ribosomal RNAs (5s rRNAs), downloaded from the PDB database [10]. A surface, or molecular envelope, was computed for each RNA model by sweeping the volume around the atomic positions with a small ball of fixed radius. This task was performed using the scripting interface of the Chimera structure-modelling environment [93]. Individual RNA sequences were then globally aligned, using an implementation of the Needleman-Wunsch algorithm [84], onto the RFAM [54] reference alignment RF00001 (5s rRNAs). Since columns in multiple sequence alignments are meant to capture functional homology, we treated each column index as a label, which we assigned to individual nucleotides, and their corresponding atoms, within each RNA. Labels were finally projected onto individual vertices of the surface by assigning to a vertex the label of its closest atom. This results in each shape represented as a triangle mesh consisting of approximately 10k vertices, and its segmentation into approximately 120 regions, typically represented as connected components. Shapes in this dataset are not pre-aligned and can have fairly significant geometric variability arising both from different conformations as well as from the molecule acquisition and reconstruction process (see Fig. 5.5 for an example).

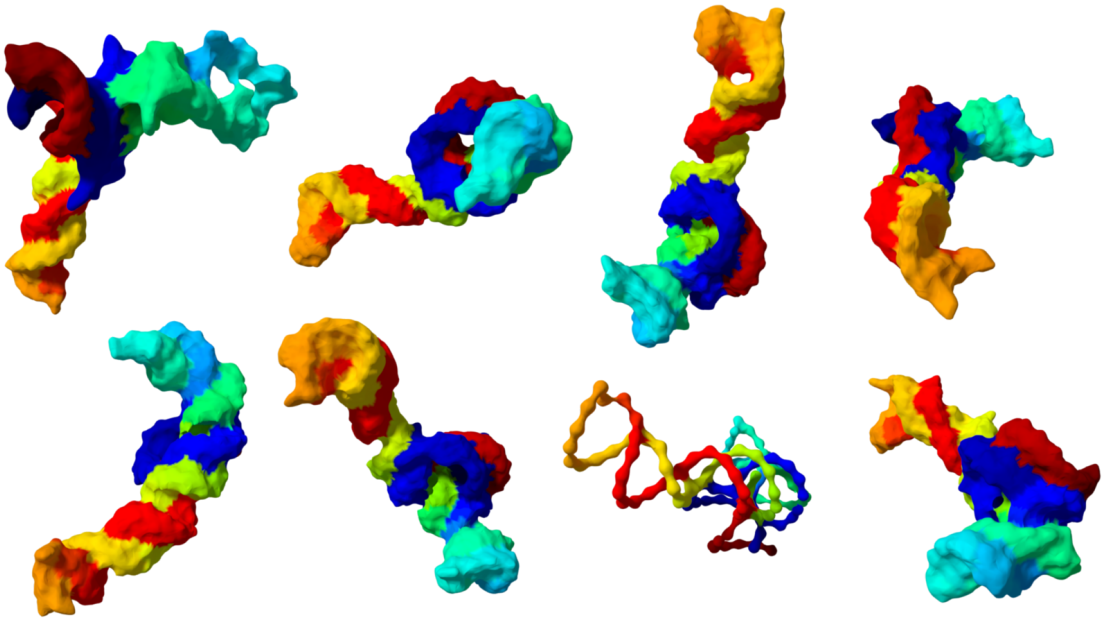


Figure 5.5 – Labeled RNA molecules.

Given a surface mesh, we first downsample it to 4096 points with farthest point sampling and then randomly sample to 2048 points before sending the data to the network. We report the segmentation accuracy in Table 5.2. As in the classification case we observed that PRIN [140] degrades severely when augmenting the training set by

random rotations. Overall, we observe that our method achieves the best accuracy in all settings. Furthermore, similarly to the classification task, the accuracy of our method is stable in different settings (with and without data augmentation) for this dataset.

Method	O/O	A/O	O/A	A/A	time
PCNN	76.7	78.0	35.1	77.8	65 s
PointNet++	72.3	74.4	46.1	74.2	18 s
PCNN (mod)	74.2	74.3	30.9	73.7	9.7 s
SPHBaseNet	74.8	74.7	28.3	74.8	17 s
SPHNet (ours)	**80.8**	**80.1**	**79.5**	**80.4**	18 s
PRIN	66.9	6.84	53.7	6.57	10 s

Table 5.2 – Segmentation accuracy on the RNA molecules dataset. Timings per epoch are given for an NVIDIA RTX 2080 Ti card.

We also show a qualitative comparison to PCNN in Figure 5.6. We note that when trained on the original dataset and tested on an augmented dataset, we achieve significantly better performance than PCNN. This demonstrates that unlike PCNN, the performance of our method does not depend on the orientation of the shapes.

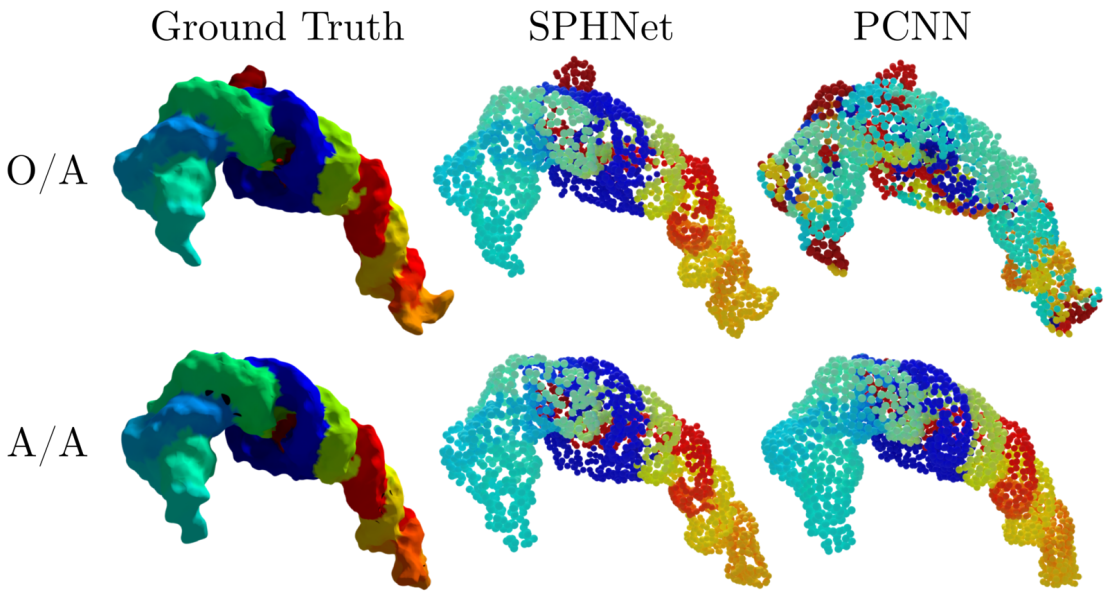


Figure 5.6 – RNA segmentation results.

5.6.3 Matching

We also apply our method on the problem of finding correspondences between non-rigid shapes. This is an especially difficult problem since in addition to *global* rigid motion, the shapes can undergo non-rigid deformations, such as articulated motion of humans.

In this setting, we trained and tested different methods on point clouds sampled from the D-FAUST dataset [13]. This dataset contains scans of 10 different subjects completing various sequences of motions given as meshes with the same structure and indexing. We prepared a test set consisting of 10 subject, motion sequence pairs and the complementary pairs defining our training set. Furthermore we sampled the motion sequences every 10 time-steps we selected 4068 shapes in total with a 3787/281 train/test split. We sampled 10k points uniformly on the first selected mesh and then subsampled 2048 points from them using farthest point sampling. We then transferred these points to all other meshes using their barycentric coordinates in the triangles of the first mesh to have a consistent point indexing on all point clouds. We produce labels by partitioning the first shape in 256 Voronoi cells associated to 256 farthest point samples. We then associate a label to each cell. Our goal then is to predict these labels and thus infer correspondences between shape pairs. Since this experiment is a particular instance of segmentation, we evaluate it with two metrics, first we measure the standard segmentation accuracy. In addition, to each cell a we associate a cell $f(a)$ by taking the most represented cell among the predictions over a and measure the average Euclidean distance between the respective centroids of $f(a)$ and the ground truth image of a. Table 5.3 shows quantitative performance of different methods. Note that the accuracy of PointNet++ and PCNN decreases drastically when trained on the original dataset and tested on rotated (augmented) data sets. Our SPHNet performs well in all training and test settings. Moreover, SPHNet strongly outperforms existing methods with data augmentation applied both during training and testing, which more closely reflects a scenario of non pre-aligned training/test data. In Figure 5.7, we show that the correspondences computed by SHPNet when trained on both original and augmented data are highly accurate. For qualitative evaluation we associate a color to each Voronoi cell using the x coordinate of its barycenter and transfer this color using computed correspondences. Figure 5.8 shows a qualitative comparison of different methods. We note that PCNN and PointNet++ correspondences present visually more artefacts including symmetry issues, while our SPHNet results in more smooth and accurate maps across all training and test settings.

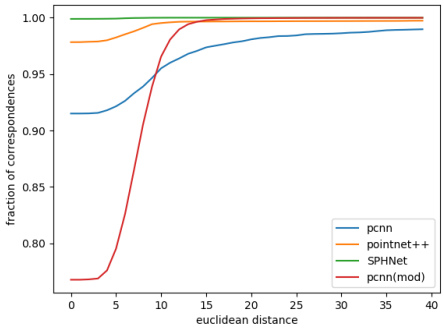


Figure 5.7 – Fraction of correspondences on the D-FAUST dataset within a certain Euclidean error in the A/A case.

Method	O/O	A/O	O/A	A/A	disterr
PCNN	**99.6**	79.9	5.7	77.1	$7.7e{-3}$
PointNet++	97.1	85.0	10.4	84.5	$1.8e{-3}$
PCNN (mod)	60.4	55.2	2.5	54.7	0.01
SPHNet (ours)	98.0	**97.2**	**91.5**	**97.1**	$\mathbf{3.5e{-5}}$
PRIN	86.7	3.24	11.3	3.63	0.59

Table 5.3 – Part label prediction accuracy on our D-FAUST dataset and average Euclidean distance error of the inferred correspondences between Voronoi cell centroids in the A/A case.

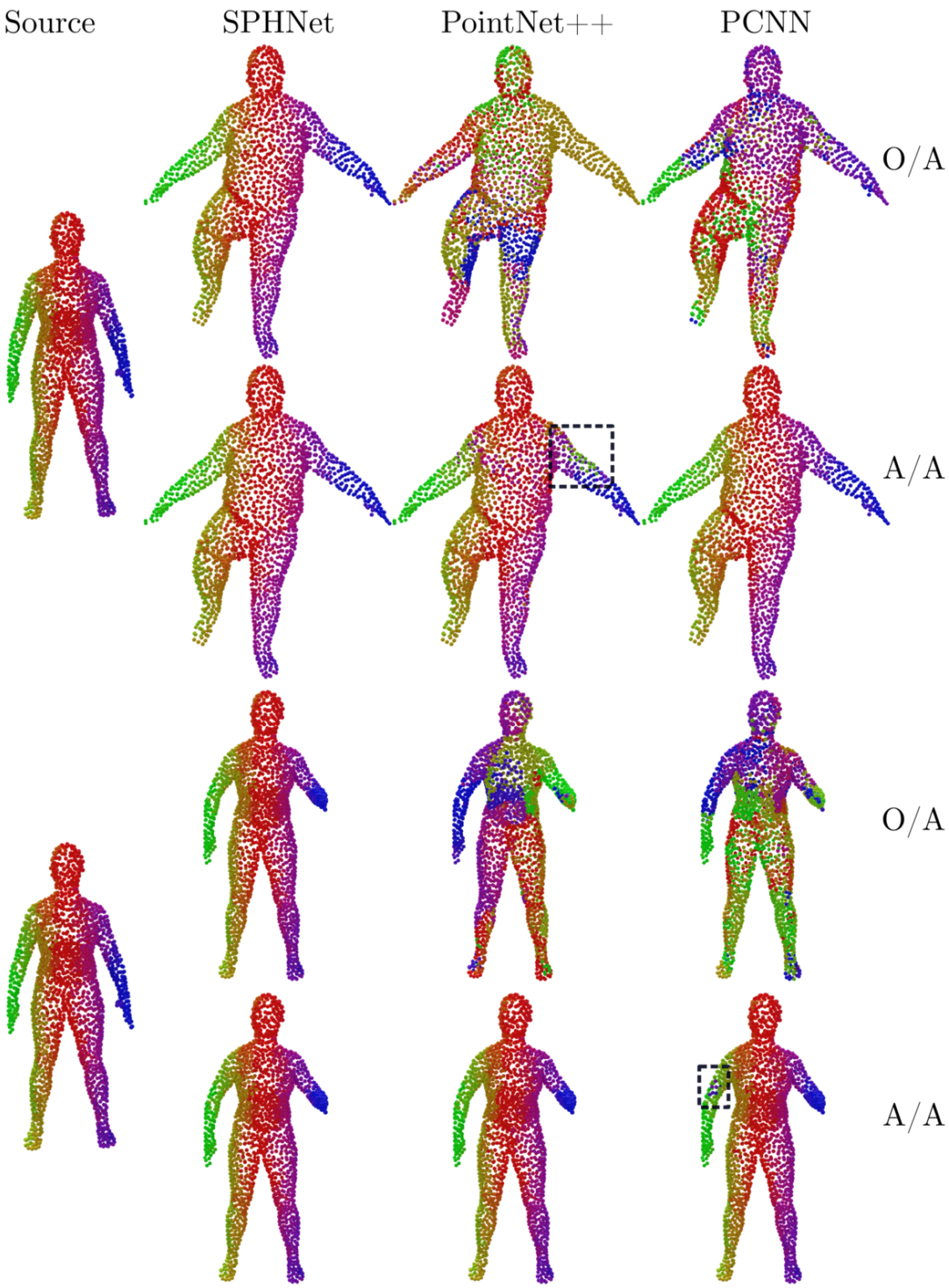


Figure 5.8 – Qualitative comparison of correspondences on the D-FAUST dataset.

5.7 Conclusion

We presented a novel approach for ensuring rotational invariance in a point-based deep neural network, based on the previously-proposed PCNN architecture. Key to our approach is the use of spherical harmonics kernels, which are both efficient, theoretically guaranteed to be rotationally invariant and can be applied at any layer of the network, providing flexibility between local and global invariance. Unlike previous methods in this domain, our resulting network outperforms existing approaches on non pre-aligned datasets even with data augmentation. In the future, we plan to extend our method to a more general framework combining non-invariant, equivariant and fully invariant features at different levels of the network, and to devise ways for *automatically* deciding the optimal layers at which invariance must be ensured. Another direction would be to apply rotationally equivariant features across different shape segments independently. This can be especially relevant for articulated motions where different segments undergo different but all approximately rigid motions.

Appendix

Table 5.4 shows the effect of the scale parameter on the accuracy of the classification task on ModelNet40. Performances decrease when ρ gets too low or too high. We choose the parameter that produces the best accuracy in our experiments.

ρ	O / O	A / O	O / A	A / A
0.2	85.7	84.9	85.0	86.3
0.15	86.8	85.6	86.0	86.7
0.1	**87.7**	**87.1**	**86.6**	**87.6**
0.075	86.1	85.7	85.6	86.5
0.05	85.6	84.5	83.2	85.4

Table 5.4 – Impact of the scale parameter ρ on classification accuracy of SPHNet on ModelNet40.

Chapter 6

Miscellaneous

6.1 Enforcing orientation preservation in shape matching

The functional map pipeline for shape matching [88] summarized in Section 3.6 of Chapter 3 often relies on descriptors such as Heat kernel Signatures (HKS) [91] or Wave Kernel Signatures [8] which are built upon the metric of the shape and are consequently invariant by intrinsic isometries. This is a major problem for computing correspondences between shapes with intrinsic symmetries like animals which often exhibit a left-right symmetry since after conversion of the functional map to a point-to-point map any point on the source shape can be sent to the image of its corresponding point under any symmetry of the target shape. To remedy this problem we proposed a novel technique to enforce orientation preservation of correspondences obtained by the functional map pipeline based on a new regularization energy term which contributed to the following publication:

- Continuous and orientation-preserving correspondences via functional maps, [102], SIGGRAPH asia, 2018.

This work has two main contributions: one is the introduction of the orientation preserving regularization mentioned above. The other, due to its first author, Jing Ren is the introduction of a new refinement technique for improving bijectivity and continuity of correspondences obtained by functional maps called Bijective Continuous Iterative Closest Point method (BCICP). Our regularization term is used during the optimisation phase computing the initial functional map which is then refined by the BCICP method. Results shown in Table 6.1 show that this new pipeline achieves improvements compared to state-of-the-art techniques in shape matching applications and that our orientation preserving energy term contributes to this improvement.

6.1.1 Orientation preserving energy term for functional maps

Below we provide the construction of our energy term for enforcing orientation preservation of functional maps. We start by a brief summary of the basic functional map pipeline (for a more comprehensive description see Section 3.6 or [88]).

Given two shapes $\mathcal{M}$ and $\mathcal{N}$ and corresponding descriptors $B^{\mathcal{N}}$ and $B^{\mathcal{M}}$ on both shapes expressed as matrices in their respective functional bases $\varphi^{\mathcal{M}}$ and $\varphi^{\mathcal{N}}$ the basic functional map pipeline for shape matching consist in finding the functional map $\mathbf{C}_{\text{opt}}$ best preserving the descriptors in least square sense:

$$\mathbf{C}_{\text{opt}} = \underset{\mathbf{C}}{\operatorname{argmin}} \|\mathbf{C}B^{\mathcal{N}} - B^{\mathcal{M}}\|_2^2 \tag{6.1}$$

The matrix $\mathbf{C}_{\text{opt}}$ is interpreted as the matrix of the pull-back map associated to a point-to-point map between $\mathcal{M}$ and $\mathcal{N}$ expressed in the bases $\varphi^{\mathcal{M}}$ and $\varphi^{\mathcal{N}}$. The point-to-point map f_{opt} is typically obtained from C_{opt} via the following optimization problem, which can be solved using a nearest neighbor search:

$$f_{\text{opt}}(i) = \underset{j}{\text{argmin}} \|\mathbf{C}_{\text{opt}} \left(\Phi^{\mathcal{N}}\right)_j^\top - \left(\Phi_i^{\mathcal{M}}\right)^\top \|_2.$$

If the descriptors $B^{\mathcal{M}}$ and $B^{\mathcal{N}}$ are invariant to isometries like [91] and [8] any affine combination of the matrices representing the ground truth map and other isometries is a solution to the convex problem in Eq. (6.1). This makes the conversion of the optimal functional map to point-to-point map ambiguous and often results in a map defined up to intrinsic isometries of both the source and target shapes. We propose to address this problem by adding a regularization term to the functional map pipeline promoting orientation preserving maps. This does not solve the problem in general but is relevant in many practical cases. Human or animal shapes often have only a left-right symmetry which is orientation reversing this motivates the promotion of orientation preserving maps.

We assume our shapes are oriented as it is the case for most shapes represented by triangle meshes in practice. We recall that in the continuous case a map $F : \mathcal{M} \to \mathcal{N}$ is orientation preserving if its differential

$$dF_p : T_p\mathcal{M} \to T_{F(p)}\mathcal{N}$$

is orientation preserving for all $p \in \mathcal{M}$. We denote by $n^{\mathcal{M}}$ and $n^{\mathcal{N}}$ the unit outward pointing normal fields of $\mathcal{M}$ and $\mathcal{N}$ respectively. Denoting by $u \times v$ the cross product between two vectors $u, v \in \mathbb{R}^3$ the map $dF_p : T_p\mathcal{M} \to \mathcal{N}$ is orientation preserving if and only if it commutes with the cross product with the outward normal:

$$dF_p(n_p^{\mathcal{M}} \times v) = n_{F(p)}^{\mathcal{N}} \times dF_p(v), \ \forall v \in T_p\mathcal{M}$$

Let $f : \mathcal{N} \to \mathbb{R}$ and $f : \mathcal{M} \to \mathbb{R}$ be smooth functions such that $g = f \circ F$ then to be orientation preserving an isometry $F : \mathcal{M} \to \mathcal{N}$ must satisfy:

$$\det(\nabla g, \nabla(k \circ F), n^{\mathcal{M}}) = \det(\nabla f, \nabla k, n^{\mathcal{N}}) \circ F$$

for all $k : \mathcal{N} \to \mathbb{R}$.

Our main observation is that since the map $\det(\nabla f, \nabla \bullet, n^{\mathcal{N}})$ is linear it can be represented as a matrix $\Omega_f^{\mathcal{N}}$ in the basis $\Phi^{\mathcal{N}}$ (we build the matrix $\Omega_g^{\mathcal{M}}$ similarly). This leads to the orientation preserving term:

$$||\mathbf{C}\Omega_f^{\mathcal{N}} - \Omega_g^{\mathcal{M}}\mathbf{C}||_2^2 \tag{6.2}$$

and the orientation reversing term:

$$||\mathbf{C}\Omega_f^{\mathcal{N}} + \Omega_g^{\mathcal{M}}\mathbf{C}||_2^2 \tag{6.3}$$

which can be added to the shape matching optimization problem of Eq. (6.1) for promoting orientation preserving or reversing maps respectively. For practical applications we need a descriptor h producing functions $h^{\mathcal{M}}, h^{\mathcal{N}}$ on $\mathcal{M}$ and $\mathcal{N}$ respectively. Ideally we would like h to satisfy the conditions:

$$\begin{cases} h^{\mathcal{M}} \simeq h^{\mathcal{N}} \circ T \\ \|\nabla h\|_2 = 1 \end{cases} \tag{6.4}$$

Where T is the point-to-point map we are looking for so that $(\nabla_p h^{\mathcal{M}}, n_p^{\mathcal{M}})$ and $(\nabla_{T(p)} h^{\mathcal{N}}, n_{T(p)}^{\mathcal{N}})$ are direct orthonormal bases of the tangent planes to $\mathcal{M}$ (resp. $\mathcal{N}$) at p (resp. $T(p)$) in which the matrix of dT is the identity. In general the conditions 6.4 cannot be satisfied but a good choice nearly satisfying them in the context of human or animal shapes is given by:

$$h := \frac{\text{hks}}{\|\nabla \text{hks}\|_2 + \epsilon}$$

where hks is the heat kernel signature for some fixed time. Thus we set $f = h^{\mathcal{N}}$ and $g = h^{\mathcal{M}}$ in the regularising terms in Eq. (6.2) and (6.3).

6.2 Experiments and results

The orientation preserving and reversing terms (6.2, 6.3) where used for shape matching in conjunction with the BCICP refinement introduced in [102] for improving bijectivity and continuity of functional map based matching. We report some results presented in [102] below. Figure 6.1 shows that orientation preserving or reversal terms can be used to promote the ground truth map or its composition with the left-right symmetry when using symmetric input descriptors such as WKS [8]. Table 6.1 shows that using the orientation preserving term in conjunction with the BCICP method consistently produces correspondences with lower average geodesic error.

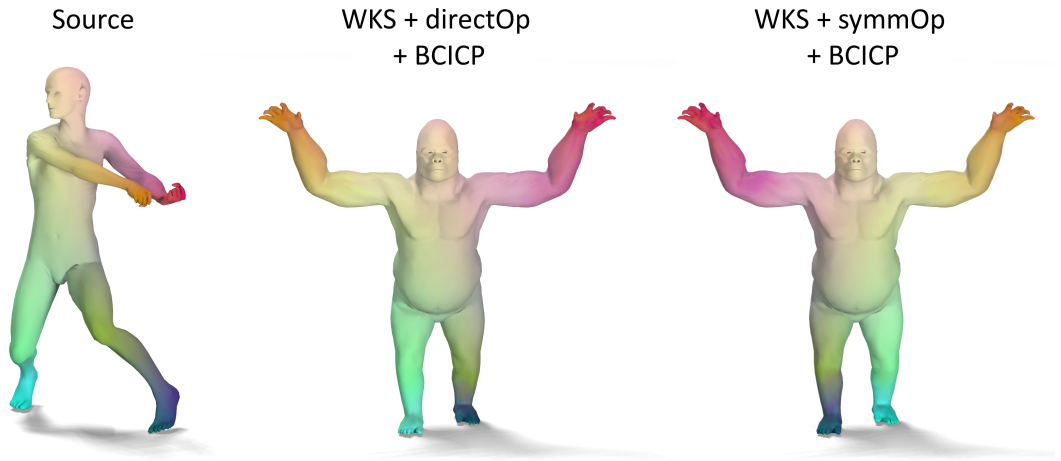


Figure 6.1 – An example of non isometric pair from the TOSCA dataset using the direct/symmetric orientation term with BCICP refinement.

Table 6.1 – Shape matching on 400 FAUST non-Isometric pairs. Combining our orientation preservation term (directOp) produces consistently better matching.

Methods	Ave geodesic error($\times 10^{-3}$)		
	per vertex	per map	direct
BIM	45.49	46.30	79.38
PMF (heat kernel)	55.92	59.95	60.54
WKS + ICP	65.59	166.37	210.11
SEG + ICP	30.94	45.68	45.68
SEG + BCICP	25.21	39.77	39.77
WKS + directOp + BCICP	29.33	46.90	51.31
SEG + directOp + BCICP	**23.50**	**37.29**	**37.29**

Chapter 7

Conclusion, Extensions and Future Work

In this thesis we have made several contributions to 3D shape analysis which we summarize in this chapter. In Section 7.1 we provide a quick overview of the impact of our contributions since their publication and of other closely related works. Finally in Section 7.2 we propose possible future extensions of our work.

In Chapter 3 we have shown that the persistence diagram of a function can be differentiated with respect to the function and we derived a continuous optimization technique for modifying functions to exhibit prescribed topological features or to make functions over potentially different domains without known correspondences more alike. We applied our technique to shape matching within the context of functional maps [88] and proposed a regularizer improving the quality and continuity of correspondences obtained with the functional map pipeline.

In Chapters 4 and 5 we investigated the problem of finding representations of 3D data which are suitable for learning. We have dealt with the problem that 3D data can be represented in different ways and that learning algorithms have to operate over a variable representation which is not known in advance, we mostly investigated the problem of rotational ambiguity. We developed convolutional layers operating on 3D data for learning applications. In Chapter 4 we presented a convolutional layer operating on triangle meshes based on local polar coordinates and resolving the local rotational ambiguity. In Chapter 5 we presented a rotation invariant convolutional layer operating on point clouds to learn independently of the pose of the shape.

7.1 Follow up works

Since its publication our work has been cited in multiple papers. In this section we list some recent papers that either reuse or cite the work presented in this thesis or simply propose new approaches to the problems we considered. For each one we explain how it relates to our work. We dedicate an independent section to each of our papers:

7.1.1 Topological function optimisation for continuous shape matching

In Chapter 3 we introduced a novel pseudo distance between the topology (persistence diagrams) of level sets of functions which is fully differentiable with respect to the functions making it suitable for continuous optimization techniques. We applied it to

derive a regularising term for improving the continuity of correspondences computed within the functional map pipeline. Since its publication our work has inspired several follow-up works listed below:

1. In [43] the authors extend our topological optimization technique (to handle higher dimensional persistence diagrams) and propose a method for surface reconstruction from point clouds. They propose a technique to remove topological noise of the input and in some case recover connectivity between disconnected components from noisy point clouds.

2. Since our method proposed a differentiable distance between topological features of functions, it can be used in deep learning applications. Recently a topology layer for learning based on our approach has been proposed in [18].

3. A general framework for differential calculus on persistence barcodes has been proposed in [65] extending the theory behind our work.

7.1.2 Multi-directional Geodesic Neural Networks via Equivariant Convolution

In Chapter 4 we presented a new CNN architecture on triangle meshes based on a new convolution operator on surfaces using local polar coordinates. However as mentioned in Section 4.6.6 of Chapter 4 our concrete implementation is using simple binning of the angular coordinate this causes stability issues and is computationally inefficient. So far we have not used more advanced tools from the theory steerable filters and group equivariant convolutions which could be beneficial to our approach. Recent works detailing such tools cite our work:

1. In [131] Weiler *et al.* propose a general theoretical framework for SE(2)-equivariant CNNs. Though aimed at images primarily the proposed formalism is applicable to other settings such as MDGCNN.

2. In [120] the authors propose to use the Zernike basis of polynomial functions on the unit disc to define local convolution operators on surfaces in the fashion of [15]. The Zernike basis is equivariant with respect to rotations and is used to define steerable kernels, avoiding to compute local convolution for multiple orientations thus saving computations. This equivariance property can be used to express the MDGCNN convolution operation as a SO(2)-equivariant convolution using the formalism developed in [131].

3. In [29] Cohen *et al.* propose a generalisation of our MDGCNN approach introducing the notion of Gauge Equivariant Convolution on fiber bundles equipped with parallel transport. But they use quite restricted cases in their practical experimentation.

7.1.3 Effective Rotation-invariant Point CNN with Spherical Harmonics Kernels

In Chapter 5 we introduced a new rotation invariant CNN framework based on local rotation invariant filters. Designing rotation robust deep learning algorithms for analysing 3D shapes and point clouds in particular is an important problem which has brought attention from the community. Different approaches to this problem have been proposed since the publication of our paper [96]:

1. In [143] the authors define local rotation invariant features on point clouds based on local directions in the point cloud and aggregate them at different scales in a deep learning pipeline.

2. A bottleneck of our SphNet approach presented in Chapter 5 is that it is based only on local features unlike networks taking global 3D coordinates of points as input. The recent work [144] introduces a new rotation invariant deep learning pipeline for learning on point clouds. The algorithm proposes a canonical pose for the point clouds and learns global features using this pose. In parallel it learns local rotation invariant features of the point cloud and combines it with the global features to make its final prediction.

7.1.4 Continuous and orientation-preserving correspondences via functional maps

In Chapter 6 we presented a method for enforcing orientation preservation of functional maps. Recently a new method for functional maps refinement has been proposed in [79], in this work our orientation preserving method is used to compute initial functional maps.

7.2 Future work

In this section we provide insights on future work extending the contributions presented in this thesis. We dedicate a subsection to each contribution we presented.

7.2.1 Topological function optimisation for continuous shape matching

The functional map induced by a matching should preserve the topology of functions. In Chapter 3 we introduced a technique to optimise a functional map to preserve topological features of functions. A strength of this approach is that it can be unsupervised in that it does not require to know correspondences between the two shapes, furthermore it is based on the minimization of an energy term which is differentiable with respect to the functions. However a practical limitation is how to choose good test functions to optimise the functional map. These two observations make our topological energy a natural candidate for unsupervised adversarial learning. Recently a pipeline for unsupervised learning of matching with functional maps was introduced in [105]. In future work it will be interesting to integrate the topology layer for learning [18] we mentioned in

Subsection 7.1.1 in the pipeline [105] together with an adversarial network producing the test functions. The test functions generator would produce random functions on the target shape which will be transferred to the source shape by the functional map learned by the algorithm of [105]. The generator would be optimised to increase the topological distance between the generated functions and their image by the functional map and the functional map would be optimised to decrease the topological distance between these functions. More generally adversarial training can be considered in conjunction with [105] for any metric between two functions on a shape or on a couple of shapes this includes in particular the orientation preserving energy term we presented in Chapter 6.

7.2.2 Multi-directional Geodesic Neural Networks via Equivariant Convolution

As we mentioned in Section 4.6.6 of Chapter 4 an important limitation of our MDGCNN construction is the disretization of the convolution kernels. First we discretize the space of local directions in a finite number of directions which has two important consequences:

1. It causes numerical instabilities as this discertization is not consistent across points and in particular under parallel transport which we heavily rely on.

2. A dot product with the kernel and the signal is required for each orientation which is computationally inefficient.

Another limitation of our discretization is that we evaluate our kernel sparsely at a discrete set of points which may lie inside triangles and get the values of the input signal at these points by interpolating the signal on the triangles. This sparse sampling might cause numerical instabilities and information loss. In the future it would be useful to improve the discretization to reduce directional instabilities and make better use of finite elements in the domain of the kernel to define more stable convolution. Though not directly aimed at improving MDGCNN the paper [120] we presented in Section 7.1.2 offers a solution to these problems in the form of rotation equivariant kernels applicable locally on triangle meshes.

Another direction for future work is to implement a variant of MDGCNN based on the notion of multi-directional convolution we presented in 4.8.2 as was our original plan when developing MDGCNN. The convolution operator described in 4.8.2 has better properties than the one actually in use in our current implementation as it truly generalise euclidean convolution and captures more information. However its practical implementation using our discretization is highly memory intensive as it requires to pull back the whole directional signal in local patches while the current version of MDGCNN only requires to pull back a single direction per point. We can take advantage of the equivariant basis of kernels introduced in [120] and the theory of equivariant CNNs presented in [131] to implement this version of MDGCNN in an efficient manner.

Finally an important limitation of MDGCNN is that it is currently limited to triangle meshes. The triangle mesh structure is required to compute the exponential map and the parallel transport which are at the core of the MDGCNN definition. A future direction

to explore would be to generalize MDGCNN to unstructured data like point-clouds by learning a tangent space structure. We can take advantage of SO(3)-equivariant constructions like [132]. Given local normals such construction can be reduced to locally SO(2) equivariant ones. Thus it would be interesting to investigate how to reduce the local ambiguity of the data representation as we acquire knowledge about the geometry of the domain and define convolution operators taking advantage of these learned representations.

7.2.3 Effective Rotation-invariant Point CNN with Spherical Harmonics Kernels

Our SphNet architecture presented in Chapter 5 is based on local rotation invariant filters this can result in information loss. A natural question is what level of invariance is required. In future work it would be interesting to learn the appropriate level of invariance at each layer of the network. We can take inspiration from the work [144] mentioned in Section 7.1.3.

An other application of rotation invariant networks is shape encoding and synthesis. In particular in the context of auto encoders it is relevant to learn rotation invariant latent codes to encode only geometric information of the shape and get rid of the extra variability induced by its pose. This idea has been explored in [77] introducing a rotation invariant auto encoder learning representations in quotient space and tackling the difficult problem of rotation invariant shape reconstruction/synthesis. However the approach of [77] relies on quotient losses involving the computation of distances across multiple rotations. Our rotation invariant design can be useful in this context as it might help simplify such constructions and save computations. However the question remains of whether networks based on our rotation invariant filters can universally approximate signals up to rotation and reflection.

Finally our SphNet architecture is also invariant to reflections however some applications might require rotation invariant and reflection equivariant networks. Designing rotation invariant and reflection equivariant filters would be a natural extension of our work.

In conclusion, during the past few years a substantial research effort has been made to develop low level representations/structures for learning over 3D data. Though this research effort is still ongoing an will most likely continue we expect it to follow the same path as image analysis and gradually shift toward more high level problems like shape synthesis or finding new, better methods for unsupervised training.

Bibliography

[1] M. Abadi, A. Agarwal, P. Barham, et al., *TensorFlow: Large-scale machine learning on heterogeneous systems*, 2015. Software available from tensorflow.org. (Cited on page 53.)

[2] Adobe, *Adobe fuse 3d characters*, 2016. (Cited on page 55.)

[3] V. Andrearczyk, J. Fageot, V. Oreiller, X. Montet, and A. Depeursinge, *Exploring local rotation invariance in 3d cnns with steerable filters*, (2018). https://openreview.net/forum?id=H1gXZLzxeE. (Cited on page 71.)

[4] D. Anguelov, P. Srinivasan, D. Koller, S. Thrun, J. Rodgers, and J. Davis, *SCAPE: Shape Completion and Animation of People*, in ACM Transactions on Graphics (TOG), vol. 24, ACM, 2005, pp. 408–416. (Cited on page 30.)

[5] D. Attali, M. Glisse, S. Hornus, F. Lazarus, and D. Morozov, *Persistence-sensitive simplification of functions on surfaces in linear time*, Presented at TOPOINVIS, 9 (2009), pp. 23–24. (Cited on page 17.)

[6] M. Atzmon, H. Maron, and Y. Lipman, *Point convolutional neural networks by extension operators*, arXiv preprint arXiv:1803.10091, (2018). (Cited on pages 70, 71, 72, 73, 74, 75, 76, 77 and 79.)

[7] M. Aubry, U. Schlickewei, and D. Cremers, *The Wave Kernel Signature: A Quantum Mechanical Approach to Shape Analysis*, in Proc. ICCV Workshops, IEEE, 2011, pp. 1626–1633. (Cited on page 35.)

[8] ——, *The wave kernel signature: A quantum mechanical approach to shape analysis*, in ICCV Workshops, IEEE, 2011, pp. 1626–1633. (Cited on pages 55, 85, 86 and 87.)

[9] U. Bauer, C. Lange, and M. Wardetzky, *Optimal topological simplification of discrete functions on surfaces*, Discrete & Computational Geometry, 47 (2012), pp. 347–377. (Cited on page 17.)

[10] H. M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. N. Bhat, H. Weissig, I. N. Shindyalov, and P. E. Bourne, *The protein data bank*, Nucleic acids research, 28 (2000), pp. 235–242. (Cited on page 80.)

[11] S. Biasotti, A. Cerri, A. Bronstein, and M. Bronstein, *Recent trends, applications, and perspectives in 3d shape similarity assessment*, in Comp. Graph. Forum, vol. 35, 2016, pp. 87–119. (Cited on page 15.)

[12] F. Bogo, J. Romero, M. Loper, and M. J. Black, *FAUST: Dataset and Evaluation for 3d Mesh Registration*, in Proc. CVPR, 2014, pp. 3794–3801. (Cited on pages 33 and 35.)

[13] F. Bogo, J. Romero, G. Pons-Moll, and M. J. Black, *Dynamic FAUST: Registering human bodies in motion*, in Proc. CVPR, July 2017. (Cited on page 82.)

[14] J.-D. Boissonnat, T. K. Dey, and C. Maria, *The compressed annotation matrix: An efficient data structure for computing persistent cohomology*, Algorithmica, 73 (2015), pp. 607–619. (Cited on page 20.)

[15] D. Boscaini, J. Masci, S. Melzi, M. M. Bronstein, U. Castellani, and P. Vandergheynst, *Learning class-specific descriptors for deformable shapes using localized spectral convolutional networks*, in Computer Graphics Forum, vol. 34, Wiley Online Library, 2015, pp. 13–23. (Cited on pages 39, 40, 41 and 90.)

[16] D. Boscaini, J. Masci, E. Rodola, and M. M. Bronstein, *Learning shape correspondence with anisotropic convolutional neural networks*, in arXiv:1605.06437, 2016. (Cited on pages 40, 41, 42, 54 and 61.)

[17] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst, *Geometric deep learning: going beyond euclidean data*, IEEE Signal Processing Magazine, 34 (2017), pp. 18–42. (Cited on pages 7, 12 and 40.)

[18] R. Brüel-Gabrielsson, B. J. Nelson, A. Dwaraknath, P. Skraba, L. J. Guibas, and G. Carlsson, *A topology layer for machine learning*, arXiv preprint arXiv:1905.12200, (2019). (Cited on pages 90 and 91.)

[19] Z. Cang and G. Wei, *Topologynet: Topology based deep convolutional and multi-task neural networks for biomolecular property predictions*, PLOS Computational Biology, 13 (2017), p. e1005690. (Cited on page 17.)

[20] G. Carlsson, *Topology and data*, Bulletin of the AMS, 46 (2009), pp. 255–308. (Cited on pages 8, 13, 16 and 18.)

[21] G. Carlsson, A. Zomorodian, A. Collins, and L. J. Guibas, *Persistence barcodes for shapes*, International Journal of Shape Modeling, 11 (2005), pp. 149–187. (Cited on pages 8, 13, 16, 18 and 22.)

[22] F. Cazals, F. Chazal, and T. Lewiner, *Molecular shape analysis based upon the morse-smale complex and the connolly function*, in Proceedings of the nineteenth annual symposium on Computational geometry, ACM, 2003, pp. 351–360. (Cited on page 17.)

[23] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, et al., *Shapenet: An information-rich 3d model repository*, arXiv preprint arXiv:1512.03012, (2015). (Cited on pages 7, 8 and 12.)

[24] F. Chazal, D. Cohen-Steiner, M. Glisse, L. J. Guibas, and S. Y. Oudot, *Proximity of persistence modules and their diagrams*, in Proceedings of the twenty-fifth annual symposium on Computational geometry, ACM, 2009, pp. 237–246. (Cited on pages 16, 19 and 28.)

[25] C. Chen and M. Kerber, *An output-sensitive algorithm for persistent homology*, Computational Geometry, 46 (2013), pp. 435–447. (Cited on page 16.)

[26] F. Chollet et al., *Keras*. https://github.com/fchollet/keras, 2015. (Cited on page 53.)

[27] T. Cohen and M. Welling, *Group equivariant convolutional networks*, in ICML, 2016, pp. 2990–2999. (Cited on page 71.)

[28] T. S. Cohen, M. Geiger, J. Köhler, and M. Welling, *Spherical cnns*, arXiv preprint arXiv:1801.10130, (2018). (Cited on page 71.)

[29] T. S. Cohen, M. Weiler, B. Kicanaoglu, and M. Welling, *Gauge equivariant convolutional networks and the icosahedral cnn*, arXiv preprint arXiv:1902.04615, (2019). (Cited on page 90.)

[30] D. Cohen-Steiner, H. Edelsbrunner, and J. Harer, *Stability of persistence diagrams*, Discrete & Computational Geometry, 37 (2007), pp. 103–120. (Cited on pages 16, 19, 20, 21, 24 and 28.)

[31] D. Cohen-Steiner, H. Edelsbrunner, and D. Morozov, *Vines and vineyards by updating persistence in linear time*, in Proceedings of the twenty-second annual symposium on Computational geometry, ACM, 2006, pp. 119–126. (Cited on page 17.)

[32] M. Cuturi, *Sinkhorn distances: Lightspeed computation of optimal transport*, in Advances in neural information processing systems, 2013, pp. 2292–2300. (Cited on page 18.)

[33] M. Defferrard, X. Bresson, and P. Vandergheynst, *Convolutional neural networks on graphs with fast localized spectral filtering*, in Advances in Neural Information Processing Systems, 2016, pp. 3844–3852. (Cited on page 41.)

[34] T. K. Dey, K. Li, C. Luo, P. Ranjan, I. Safa, and Y. Wang, *Persistent heat signature for pose-oblivious matching of incomplete models*, in Computer Graphics Forum, vol. 29, Wiley Online Library, 2010, pp. 1545–1554. (Cited on page 16.)

[35] H. Edelsbrunner and J. Harer, *Persistent homology-a survey*, Contemporary mathematics, 453 (2008), pp. 257–282. (Cited on pages 8 and 13.)

[36] ——, *Computational topology: an introduction*, American Mathematical Soc., 2010. (Cited on pages 16, 18 and 19.)

[37] H. Edelsbrunner, J. Harer, and A. Zomorodian, *Hierarchical morse complexes for piecewise linear 2-manifolds*, in Proceedings of the seventeenth annual symposium on Computational geometry, ACM, 2001, pp. 70–79. (Cited on page 17.)

[38] H. Edelsbrunner, D. Letscher, and A. Zomorodian, *Topological persistence and simplification*, in Foundations of Computer Science, 2000. Proceedings. 41st Annual Symposium on, IEEE, 2000, pp. 454–463. (Cited on pages 16 and 20.)

[39] D. Ezuz and M. Ben-Chen, *Deblurring and denoising of maps between shapes*, in Computer Graphics Forum, vol. 36, Wiley Online Library, 2017, pp. 165–174. (Cited on page 17.)

[40] D. Ezuz, J. Solomon, V. G. Kim, and M. Ben-Chen, *Gwcnn: A metric alignment layer for deep shape analysis*, in Computer Graphics Forum, vol. 36, Wiley Online Library, 2017, pp. 49–57. (Cited on pages 39 and 41.)

[41] C. Fong, *Analytical methods for squaring the disc*, arXiv preprint arXiv:1509.06344, (2015). (Cited on page 54.)

[42] W. T. Freeman and E. H. Adelson, *The design and use of steerable filters*, IEEE Transactions on Pattern Analysis & Machine Intelligence, (1991), pp. 891–906. (Cited on page 74.)

[43] R. B. Gabrielsson, V. Ganapathi-Subramanian, P. Skraba, and L. J. Guibas, *Topology-aware surface reconstruction for point clouds*, arXiv preprint arXiv:1811.12543, (2018). (Cited on page 90.)

[44] M. Gameiro, Y. Hiraoka, and I. Obayashi, *Continuation of point clouds via persistence diagrams*, Physica D: Nonlinear Phenomena, 334 (2016), pp. 118–132. (Cited on page 17.)

[45] M. Garland and P. S. Heckbert, *Surface simplification using quadric error metrics*, in Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '97, 1997, pp. 209–216. (Cited on page 51.)

[46] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*, vol. 1, MIT press Cambridge, 2016. (Cited on page 22.)

[47] P. Guerrero, Y. Kleiman, M. Ovsjanikov, and N. J. Mitra, *Pcpnet learning local shape properties from raw point clouds*, in Computer Graphics Forum, vol. 37, Wiley Online Library, 2018, pp. 75–85. (Cited on pages 41 and 70.)

[48] 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, 2016, pp. 770–778. (Cited on pages 7, 11, 42 and 53.)

[49] P. Hermosilla, T. Ritschel, P.-P. Vázquez, À. Vinacua, and T. Ropinski, *Monte carlo convolution for learning on non-uniformly sampled point clouds*, in SIGGRAPH Asia 2018 Technical Papers, ACM, 2018, p. 235. (Cited on page 70.)

[50] B.-S. Hua, M.-K. Tran, and S.-K. Yeung, *Pointwise convolutional neural networks*, in Proc. CVPR, 2018, pp. 984–993. (Cited on page 70.)

[51] R. Huang and M. Ovsjanikov, *Adjoint map representation for shape analysis and matching*, in Computer Graphics Forum, vol. 36, Wiley Online Library, 2017, pp. 151–163. (Cited on page 17.)

[52] M. Jaderberg, K. Simonyan, A. Zisserman, et al., *Spatial transformer networks*, in NIPS, 2015, pp. 2017–2025. (Cited on page 70.)

[53] E. Kalogerakis, M. Averkiou, S. Maji, and S. Chaudhuri, *3D shape segmentation with projective convolutional networks*, in Proc. CVPR, 2017. (Cited on pages 39 and 40.)

[54] I. Kalvari, E. P. Nawrocki, J. Argasinska, N. Quinones-Olvera, R. D. Finn, A. Bateman, and A. I. Petrov, *Non-coding RNA analysis using the RFAM database.*, Current protocols in bioinformatics, 62 (2018), p. e51. (Cited on page 80.)

[55] M. Kazhdan, T. Funkhouser, and S. Rusinkiewicz, *Rotation invariant spherical harmonic representation of 3 d shape descriptors*, in Symposium on geometry processing, vol. 6, 2003, pp. 156–164. (Cited on page 74.)

[56] V. G. Kim, Y. Lipman, and T. Funkhouser, *Blended Intrinsic Maps*, in ACM Transactions on Graphics (TOG), vol. 30, ACM, 2011, p. 79. (Cited on page 33.)

[57] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, arXiv preprint arXiv:1412.6980, (2014). (Cited on page 53.)

[58] R. Klokov and V. Lempitsky, *Escape from cells: Deep kd-networks for the recognition of 3d point cloud models*, in 2017 IEEE International Conference on Computer Vision (ICCV), IEEE, 2017, pp. 863–872. (Cited on pages 41 and 76.)

[59] S. Koch, A. Matveev, Z. Jiang, F. Williams, A. Artemov, E. Burnaev, M. Alexa, D. Zorin, and D. Panozzo, *Abc: A big cad model dataset for geometric deep learning*, in The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2019. (Cited on pages 7 and 12.)

[60] R. Kondor and S. Trivedi, *On the generalization of equivariance and convolution in neural networks to the action of compact groups*, arXiv preprint arXiv:1802.03690, (2018). (Cited on page 71.)

[61] I. Kostrikov, Z. Jiang, D. Panozzo, D. Zorin, and J. Bruna, *Surface Networks*, ArXiv e-prints, (2017). (Cited on page 41.)

[62] A. Kovnatsky, M. M. Bronstein, X. Bresson, and P. Vandergheynst, *Functional correspondence by matrix completion*, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 905–914. (Cited on pages 17 and 27.)

[63] A. Krizhevsky and G. Hinton, *Learning multiple layers of features from tiny images*, (2009). (Cited on page 54.)

[64] A. Krizhevsky, I. Sutskever, and G. E. Hinton, *Imagenet classification with deep convolutional neural networks*, in Advances in neural information processing systems, 2012, pp. 1097–1105. (Cited on pages 7, 11 and 39.)

[65] J. Leygonie, S. Oudot, and U. Tillmann, *A framework for differential calculus on persistence barcodes*, arXiv preprint arXiv:1910.00960, (2019). (Cited on page 90.)

[66] C. Li, M. Ovsjanikov, and F. Chazal, *Persistence-based structural recognition*, in Proc. CVPR, 2014, pp. 1995–2002. (Cited on pages 16 and 21.)

[67] Y. Li, R. Bu, M. Sun, and B. Chen, *Pointcnn*, arXiv preprint arXiv:1801.07791, (2018). (Cited on pages 41, 53, 55 and 70.)

[68] O. Litany, E. Rodolà, A. M. Bronstein, and M. M. Bronstein, *Fully spectral partial shape matching*, Computer Graphics Forum, 36 (2017), pp. 247–258. (Cited on page 17.)

[69] O. Litany, E. Rodolà, A. M. Bronstein, M. M. Bronstein, and D. Cremers, *Non-rigid puzzles*, in Computer Graphics Forum, vol. 35, 2016, pp. 135–143. (Cited on pages 17 and 27.)

[70] G. Litjens, T. Kooi, B. E. Bejnordi, A. A. A. Setio, F. Ciompi, M. Ghafoorian, J. A. Van Der Laak, B. Van Ginneken, and C. I. Sánchez, *A survey on deep learning in medical image analysis*, Medical image analysis, 42 (2017), pp. 60–88. (Cited on pages 7 and 11.)

[71] J.-Y. Liu, S.-K. Jeng, and Y.-H. Yang, *Applying topological persistence in convolutional neural network for music audio signals*, arXiv preprint arXiv:1608.07373, (2016). (Cited on page 17.)

[72] M. Mandad, D. Cohen-Steiner, L. Kobbelt, P. Alliez, and M. Desbrun, *Variance-minimizing transport plans for inter-surface mapping*, ACM Trans. on Graph., 36 (2017), p. 14. (Cited on pages 18, 36 and 37.)

[73] H. Maron, M. Galun, N. Aigerman, M. Trope, N. Dym, E. Yumer, V. G. Kim, and Y. Lipman, *Convolutional neural networks on surfaces via seamless toric covers*, SIGGRAPH, 2017. (Cited on pages 8, 12, 39, 40, 41, 55 and 56.)

[74] J. Masci, D. Boscaini, M. Bronstein, and P. Vandergheynst, *Geodesic convolutional neural networks on riemannian manifolds*, in Proc. ICCV workshops, 2015, pp. 37–45. (Cited on page 69.)

[75] J. Masci, D. Boscaini, M. M. Bronstein, and P. Vandergheynst, *Geodesic convolutional neural networks on riemannian manifolds*, in Proc. of the IEEE International Conference on Computer Vision (ICCV) Workshops, 2015, pp. 37–45. (Cited on pages 9, 13, 40, 41, 42, 43, 45, 53, 58 and 59.)

[76] D. Maturana and S. Scherer, *Voxnet: A 3d convolutional neural network for real-time object recognition*, in Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on, IEEE, 2015, pp. 922–928. (Cited on pages 8, 12, 39 and 41.)

[77] E. Mehr, A. Lieutier, F. Sanchez Bermudez, V. Guitteny, N. Thome, and M. Cord, *Manifold learning in quotient spaces*, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 9165–9174. (Cited on page 93.)

[78] E. L. Melvær and M. Reimers, *Geodesic polar coordinates on polygonal meshes*, in Computer Graphics Forum, vol. 31, Wiley Online Library, 2012, pp. 2423–2435. (Cited on pages 52 and 66.)

[79] 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 (2019), p. 155. (Cited on page 91.)

[80] M. Meyer, M. Desbrun, P. Schröder, and A. H. Barr, *Discrete Differential-Geometry Operators for Triangulated 2-Manifolds*, in Visualization and mathematics III, Springer, 2003, pp. 35–57. (Cited on page 30.)

[81] N. Milosavljević, D. Morozov, and P. Skraba, *Zigzag persistent homology in matrix multiplication time*, in Proceedings of the twenty-seventh annual symposium on Computational geometry, ACM, 2011, pp. 216–225. (Cited on page 16.)

[82] 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, IEEE Computer Society, 2017, pp. 5425–5434. (Cited on pages 41, 42 and 54.)

[83] 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 Proc. CVPR, 2017, pp. 5115–5124. (Cited on pages 69 and 70.)

[84] S. B. Needleman and C. D. Wunsch, *A general method applicable to the search for similarities in the amino acid sequence of two proteins.*, Journal of molecular biology, 48 (1970), pp. 443–453. (Cited on page 80.)

[85] D. Nogneng, S. Melzi, E. Rodolà, U. Castellani, M. Bronstein, and M. Ovsjanikov, *Improved functional mappings via product preservation*, in Computer Graphics Forum, vol. 37, 2018. (Cited on page 17.)

[86] D. Nogneng and M. Ovsjanikov, *Informative descriptor preservation via commutativity for shape matching*, Computer Graphics Forum, 36 (2017), pp. 259–267. (Cited on pages 17, 27 and 35.)

[87] N. Otter, M. A. Porter, U. Tillmann, P. Grindrod, and H. A. Harrington, *A roadmap for the computation of persistent homology*, EPJ Data Science, 6 (2017), p. 17. (Cited on page 20.)

[88] M. Ovsjanikov, M. Ben-Chen, J. Solomon, A. Butscher, and L. Guibas, *Functional maps: a flexible representation of maps between shapes*, ACM Transactions on Graphics (TOG), 31 (2012), p. 30. (Cited on pages 9, 10, 13, 14, 41, 85 and 89.)

[89] ——, *Functional Maps: A Flexible Representation of Maps Between Shapes*, ACM Transactions on Graphics (TOG), 31 (2012), p. 30. (Cited on pages 15, 17, 27, 33 and 35.)

[90] M. Ovsjanikov, E. Corman, M. Bronstein, E. Rodolà, M. Ben-Chen, L. Guibas, F. Chazal, and A. Bronstein, *Computing and processing correspondences with functional maps*, in ACM SIGGRAPH 2017 Courses, 2017, pp. 5:1–5:62. (Cited on pages 15, 17, 26 and 27.)

[91] M. Ovsjanikov, Q. Mérigot, F. Mémoli, and L. Guibas, *One point isometric matching with the heat kernel*, in Computer Graphics Forum, vol. 29, 2010, pp. 1555–1564. (Cited on pages 85 and 86.)

[92] O. M. Parkhi, A. Vedaldi, A. Zisserman, et al., *Deep face recognition.*, in bmvc, vol. 1, 2015, p. 6. (Cited on pages 7 and 11.)

[93] E. F. Pettersen, T. D. Goddard, C. C. Huang, G. S. Couch, D. M. Greenblatt, E. C. Meng, and T. E. Ferrin, *Ucsf chimera–a visualization system for exploratory research and analysis.*, Journal of computational chemistry, 25 (2004), pp. 1605–1612. (Cited on page 80.)

[94] U. Pinkall and K. Polthier, *Computing Discrete Minimal Surfaces and their Conjugates*, Experimental mathematics, 2 (1993), pp. 15–36. (Cited on page 30.)

[95] A. Poulenard and M. Ovsjanikov, *Multi-directional geodesic neural networks via equivariant convolution*, in Proc. SIGGRAPH Asia, ACM, 2018, p. 236. (Cited on pages 8, 9, 13, 14 and 52.)

[96] A. Poulenard, M.-J. Rakotosaona, Y. Ponty, and M. Ovsjanikov, *Effective rotation-invariant point cnn with spherical harmonics kernels*, in 2019 International Conference on 3D Vision (3DV), IEEE, 2019, pp. 47–56. (Cited on pages 8, 9, 13, 14 and 91.)

[97] A. Poulenard, P. Skraba, and M. Ovsjanikov, *Topological function optimization for continuous shape matching*, in Computer Graphics Forum, vol. 37, Wiley Online Library, 2018, pp. 13–25. (Cited on pages 8, 9, 12 and 13.)

[98] 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 (2017), p. 4. (Cited on pages 41, 69 and 70.)

[99] C. R. Qi, H. Su, M. Niessner, A. Dai, M. Yan, and L. J. Guibas, *Volumetric and multi-view cnns for object classification on 3d data*, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 5648–5656. (Cited on pages 39 and 41.)

[100] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, *Pointnet++: Deep hierarchical feature learning on point sets in a metric space*, in Advances in Neural Information Processing Systems, 2017, pp. 5105–5114. (Cited on pages 8, 12, 41, 53, 55, 58, 69, 70, 71 and 79.)

[101] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, *You only look once: Unified, real-time object detection*, in Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 779–788. (Cited on pages 7 and 11.)

[102] J. Ren, A. Poulenard, P. Wonka, and M. Ovsjanikov, *Continuous and orientation-preserving correspondences via functional maps*, in SIGGRAPH Asia 2018 Technical Papers, ACM, 2018, p. 248. (Cited on pages 10, 14, 85 and 87.)

[103] E. Rodolà, L. Cosmo, M. M. Bronstein, A. Torsello, and D. Cremers, *Partial functional correspondence*, in Computer Graphics Forum, vol. 36, 2017, pp. 222–236. (Cited on page 17.)

[104] O. Ronneberger, P. Fischer, and T. Brox, *U-net: Convolutional networks for biomedical image segmentation*, in International Conference on Medical image computing and computer-assisted intervention, Springer, 2015, pp. 234–241. (Cited on page 53.)

[105] 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, 2019, pp. 1617–1627. (Cited on pages 91 and 92.)

[106] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, *ImageNet Large Scale Visual Recognition Challenge*, International Journal of Computer Vision (IJCV), 115 (2015), pp. 211–252. (Cited on pages 7 and 11.)

[107] 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 (2013), p. 72. (Cited on page 17.)

[108] S. Salti, F. Tombari, and L. Di Stefano, *Shot: Unique signatures of histograms for surface and texture description*, Computer Vision and Image Understanding, 125 (2014), pp. 251–264. (Cited on page 55.)

[109] J. A. Sethian, *Level set methods and fast marching methods: evolving interfaces in computational geometry, fluid mechanics, computer vision, and materials science*, vol. 3, Cambridge university press, 1999. (Cited on page 52.)

[110] K. Sfikas, T. Theoharis, and I. Pratikakis, *Exploiting the panorama representation for convolutional neural network classification and retrieval*, in Eurographics Workshop on 3D Object Retrieval, 2017. (Cited on page 40.)

[111] B. Shi, S. Bai, Z. Zhou, and X. Bai, *Deeppano: Deep panoramic representation for 3-d shape recognition*, IEEE Signal Processing Letters, 22 (2015), pp. 2339–2343. (Cited on page 40.)

[112] R. K. Singh and J. S. Manhas, *Composition Operators on Function Spaces*, vol. 179, Elsevier, 1993. (Cited on page 28.)

[113] A. Sinha, J. Bai, and K. Ramani, *Deep learning 3d shape surfaces using geometry images*, in European Conference on Computer Vision, Springer, 2016, pp. 223–240. (Cited on pages 39 and 40.)

[114] P. Skraba, M. Ovsjanikov, F. Chazal, and L. Guibas, *Persistence-based segmentation of deformable shapes*, in Computer Vision and Pattern Recognition Workshops (CVPRW), 2010 IEEE Computer Society Conference on, IEEE, 2010, pp. 45–52. (Cited on pages 16 and 21.)

[115] P. Skraba, G. Thoppe, and D. Yogeshwaran, *Randomly weighted d−complexes: Minimal spanning acycles and persistence diagrams*, arXiv preprint arXiv:1701.00239, (2017). (Cited on page 17.)

[116] J. Solomon, F. De Goes, G. Peyré, M. Cuturi, A. Butscher, A. Nguyen, T. Du, and L. Guibas, *Convolutional wasserstein distances: Efficient optimal transportation on geometric domains*, ACM Transactions on Graphics (TOG), 34 (2015), p. 66. (Cited on page 18.)

[117] J. Solomon, G. Peyré, V. G. Kim, and S. Sra, *Entropic metric alignment for correspondence problems*, ACM Transactions on Graphics (TOG), 35 (2016), p. 72. (Cited on pages 18 and 41.)

[118] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller, *Multi-view convolutional neural networks for 3d shape recognition*, in Proceedings of the IEEE international conference on computer vision, 2015, pp. 945–953. (Cited on pages 39, 40 and 69.)

[119] J. Sun, M. Ovsjanikov, and L. Guibas, *A Concise and Provably Informative Multi-Scale Signature Based on Heat Diffusion*, in Computer graphics forum, vol. 28, 2009, pp. 1383–1392. (Cited on page 30.)

[120] Z. Sun, E. Rooke, J. Charton, Y. He, J. Lu, and S. Baek, *Zernet: Convolutional neural networks on arbitrary surfaces via zernike local tangent space estimation*, arXiv preprint arXiv:1812.01082, (2018). (Cited on pages 90 and 92.)

[121] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, *Going deeper with convolutions*, in Proc. CVPR, 2015, pp. 1–9. (Cited on page 60.)

[122] G. K. Tam, Z.-Q. Cheng, Y.-K. Lai, F. C. Langbein, Y. Liu, D. Marshall, R. R. Martin, X.-F. Sun, and P. L. Rosin, *Registration of 3D point clouds and meshes: a survey from rigid to nonrigid*, IEEE TVCG, 19 (2013), pp. 1199–1217. (Cited on page 15.)

[123] T. Tanaka, *On the family of connected subsets and the topology of spaces*, Journal of the Mathematical Society of Japan, 7 (1955), pp. 389–393. (Cited on page 28.)

[124] N. Thomas, T. Smidt, S. Kearnes, L. Yang, L. Li, K. Kohlhoff, and P. Riley, *Tensor field networks: Rotation-and translation-equivariant neural networks for 3d point clouds*, arXiv preprint arXiv:1802.08219, (2018). (Cited on page 71.)

[125] O. Van Kaick, H. Zhang, G. Hamarneh, and D. Cohen-Or, *A survey on shape correspondence*, in Computer Graphics Forum, vol. 30, 2011, pp. 1681–1707. (Cited on page 15.)

[126] M. Vestner, Z. Lähner, A. Boyarski, O. Litany, R. Slossberg, T. Remez, E. Rodola, A. Bronstein, M. Bronstein, R. Kimmel, and D. Cremers, *Efficient deformable shape correspondence via kernel matching*, in Proc. 3DV, 2017. (Cited on page 18.)

[127] 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, 2017, pp. 6681–6690. (Cited on pages 18, 36 and 37.)

[128] P.-S. Wang, Y. Liu, Y.-X. Guo, C.-Y. Sun, and X. Tong, *O-cnn: Octree-based convolutional neural networks for 3d shape analysis*, ACM Transactions on Graphics (TOG), 36 (2017), p. 72. (Cited on pages 41 and 53.)

[129] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, *Dynamic graph cnn for learning on point clouds*, arXiv preprint arXiv:1801.07829, (2018). (Cited on pages 41, 55, 58, 70 and 77.)

[130] L. Wei, Q. Huang, D. Ceylan, E. Vouga, and H. Li, *Dense human body correspondences using convolutional networks*, in Computer Vision and Pattern Recognition (CVPR), 2016 IEEE Conference on, IEEE, 2016, pp. 1544–1553. (Cited on page 40.)

[131] M. Weiler and G. Cesa, *General e (2)-equivariant steerable cnns*, in Advances in Neural Information Processing Systems, 2019, pp. 14334–14345. (Cited on pages 90 and 92.)

[132] M. Weiler, M. Geiger, M. Welling, W. Boomsma, and T. Cohen, *3d steerable cnns: Learning rotationally equivariant features in volumetric data*, in NIPS, 2018, pp. 10381–10392. (Cited on pages 71, 73, 74 and 93.)

[133] M. Weiler, F. A. Hamprecht, and M. Storath, *Learning steerable filters for rotation equivariant cnns*, in Proc. CVPR, 2018, pp. 849–858. (Cited on page 71.)

[134] E. P. Wigner, *Group theory and its application to the quantum mechanics of atomic spectra*, New York: Academic Press, 1959. (Cited on page 74.)

[135] D. Worrall and G. Brostow, *Cubenet: Equivariance to 3d rotation and translation*, in Proc. ECCV, 2018, pp. 567–584. (Cited on page 71.)

[136] K. Wu, Z. Zhao, R. Wang, and G.-W. Wei, *Topp-s: Persistent homology based multi-task deep neural networks for simultaneous predictions of partition coefficient and aqueous solubility*, arXiv preprint arXiv:1801.01558, (2017). (Cited on page 17.)

[137] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, *3d shapenets: A deep representation for volumetric shapes*, in Proceedings of the IEEE conference on computer vision and pattern recognition, 2015, pp. 1912–1920. (Cited on pages 7, 8, 12, 41, 69 and 78.)

[138] K. Xu, V. G. Kim, Q. Huang, N. Mitra, and E. Kalogerakis, *Data-driven shape analysis and processing*, in SIGGRAPH ASIA 2016 Courses, ACM, 2016, p. 4. (Cited on pages 40 and 70.)

[139] L. Yi, H. Su, X. Guo, and L. J. Guibas, *Syncspeccnn: Synchronized spectral CNN for 3d shape segmentation*, in CVPR, IEEE Computer Society, 2017, pp. 6584–6592. (Cited on page 41.)

[140] Y. You, Y. Lou, Q. Liu, L. Ma, W. Wang, Y. Tai, and C. Lu, *Prin: Pointwise rotation-invariant network*, arXiv preprint arXiv:1811.09361, (2018). (Cited on pages 71, 79 and 80.)

[141] L. Yu, X. Li, C.-W. Fu, D. Cohen-Or, and P.-A. Heng, *Pu-net: Point cloud upsampling network*, in Proc. CVPR, 2018. (Cited on page 70.)

[142] W. Yuan, T. Khot, D. Held, C. Mertz, and M. Hebert, *Pcn: Point completion network*, in 3DV, IEEE, 2018, pp. 728–737. (Cited on page 70.)

[143] Z. Zhang, B.-S. Hua, D. W. Rosen, and S.-K. Yeung, *Rotation invariant convolutions for 3d point clouds deep learning*, in 2019 International Conference on 3D Vision (3DV), IEEE, 2019, pp. 204–213. (Cited on page 91.)

[144] C. Zhao, J. Yang, X. Xiong, A. Zhu, Z. Cao, and X. Li, *Rotation invariant point cloud classification: Where local geometry meets global topology*, arXiv preprint arXiv:1911.00195, (2019). (Cited on pages 91 and 93.)

ECOLE
DOCTORALE

Titre : Structures pour l'apprentissage profond et l'optimisation de la topologie de fonctions sur les formes 3D

Mots clés : Analyse de formes, Apprentissage profond, convolution, invariance par rotation, topologie

Résumé : Le domaine du traitement de la géométrie suit un cheminement similaire à celui de l'analyse d'images avec l'explosion des publications consacrées à l'apprentissage profond ces dernières années. Un important effort de recherche est en cours pour reproduire les succès de l'apprentissage profond dans le domaine de la vision par ordinateur dans le contexte de l'analyse de formes 3D. Contrairement aux images, les formes 3D peuvent être représentées de différentes manières comme des maillages ou des nuages de points souvent dépourvus d'une structure canonique. Les algorithmes d'apprentissage profond traditionnels tels que les réseaux neuronaux convolutifs (CNN) ne sont donc pas faciles à appliquer aux formes 3D. Dans cette thèse, nous proposons trois contributions principales : premièrement, nous introduisons une méthode permettant de comparer des fonctions sur des domaines différents sans correspondances et de les déformer afin de rendre la topologie de leur ensemble de niveaux similaires. Nous appliquons notre méthode au problème classique de la correspondance de formes dans le contexte des applications fonctionnelles (functional maps) afin de produire des correspondances plus lisses et plus précises. Par ailleurs notre méthode reposant sur l'optimisation continue d'une énergie différentiable par rapport aux fonctions comparées elle est applicable à l'apprentissage profond. Nous apportons deux contributions directes à l'apprentissage profond des données 3D. Nous introduisons un nouvel opérateur de convolution sur des maillages triangulaires basés sur des coordonnées polaires locales et l'appliquons à l'apprentissage profond sur les maillages. Contrairement aux travaux précédents, notre opérateur prend en compte tous les choix de coordonnées polaires sans perte d'information directionnelle. Enfin, nous introduisons un nouveau module de convolution invariant par rotation sur les nuages de points et montrons que les CNN basés sur ce dernier peuvent surpasser l'état de l'art pour des tâches standard sur des ensembles de données non alignés même avec augmentation des données.

Title : Structures for deep learning and topology optimization of functions on 3D shapes

Keywords : Shape analysis, Deep learning, Convolution, Rotation invariance, Topology

Abstract : The field of geometry processing is following a similar path as image analysis with the explosion of publications dedicated to deep learning in recent years. An important research effort is being made to reproduce the successes of deep learning 2D computer vision in the context of 3D shape analysis. Unlike images shapes come in various representations like meshes or point clouds which often lack canonical structure. This makes traditional deep learning algorithms like Convolutional Neural Networks (CNN) non straightforward to apply to 3D data. In this thesis we propose three main contributions: first, we introduce a method to compare functions on different domains without correspondences and to deform them to make the topology of their level sets more alike. We apply our method to the classical problem of shape matching in the context of functional maps to produce smoother and more accurate correspondences. Furthermore, our method is based on the continuous optimization of a differentiable energy with respect to the compared functions and is applicable to deep learning. We make two direct contributions to deep learning on 3D data. We introduce a new convolution operator over triangles meshes based on local polar coordinates and apply it to deep learning on meshes. Unlike previous works our operator takes all choices of polar coordinates into account without loss of directional information. Lastly we introduce a new rotation invariant convolution layer over point clouds and show that CNNs based on this layer can outperform state of the art methods in standard tasks on unaligned datasets even with data augmentation.

Institut Polytechnique de Paris
91120 Palaiseau, France