Thesis for the degree
Doctor of Philosophy

Submitted to the Scientific Council of the
Weizmann Institute of Science
Rehovot, Israel

עבודת גמר (תזה) לתואר
דוקטור לפילוסופיה

מוגשת למועצה המדעית של
מכון ויצמן למדע
רחובות, ישראל

By
Haggai Maron

מאת
חגי מרון

אנליזה עמוקה וקמורה של צורות
Deep and Convex Shape Analysis

Advisor: professor
Yaron Lipman

מנחה: פרופסור
ירון ליפמן

August 2019

אב תשע״ט

1 Acknowledgments

First, I would like to thank my advisor Professor Yaron Lipman for his guidance, constant encouragement, and advice. I was fortunate to have him as an advisor. I would also like to thank the members of my Ph.D. committee, Prof. Ronen Basri and Prof. Yosef Yomdin.

Most of the work presented in this dissertation was executed as part of collaborations with my wonderful labmates. I want to thank them for their help, advice, and for creating a productive working environment.

Last but not least, I want to thank my family: my amazing wife Maayan, for her patience, encouragement and help during deadlines, my beloved children, Nadav and Noa, which will always be my most important contributions, My parents Ilana and Ami Maron who raised me to be a curious person I am, and to my grandparents Pnina Merling-Reiss and Zeev Reiss who introduced me to the world of science at a very young age.

Haggai Maron,
Rehovot, August 2019

2 Declaration

I hereby declare that this thesis summarizes my original research, performed under the guidance of my advisor Prof. Yaron Lipman. Wherever the contributions of others are involved, every effort was made to indicate this clearly by citing the relevant literature. In particular, [72, 14, 161] involved equal contributors, where all authors contributed to the idea, theoretic analysis and implementation.

Haggai Maron

Contents

3 Abstract

This dissertation summarizes the main results obtained during my Ph.D. studies at the Weizmann Institute of Science under the guidance of Professor Yaron Lipman. Two fundamental problems in shape analysis were considered: (1) how to apply deep learning techniques to irregular data and (2) how to compute meaningful maps between shapes. My work has resulted in several novel methods for applying deep learning to surfaces, point clouds (*i.e.*, finite subsets of the Euclidean space), graphs and hyper-graphs as well as new efficient techniques to solve relaxations of well-known matching problems. The report discusses these two problems, surveys the suggested solutions and points out several directions for future work, including a promising direction that combines both problems.

4 Introduction

Shape analysis is concerned with studying and quantifying properties of geometric objects such as curves, surfaces, and higher dimensional manifolds. Among other fields, shape analysis techniques are widely used in computer vision [272], computer graphics [85], computational anatomy [34] and medical imaging [164]. During the last few years we tried to tackle two key questions: (1) deep learning of irregular data: *i.e.*, How can we apply deep learning to common shape representations such as point clouds, surfaces, and graphs? (2) shape matching: Given two shapes, how can we compute meaningful maps between them.

Figure 1 illustrates these two problems. The top part illustrates the geometric deep learning problem: applying deep learning to irregular domains. The bottom part illustrates two instances of the shape matching problem: matching 3D models (left, the map is represented using color coding) and matching an image collection to a grid structure according to color features (right). The rest of the introduction puts these two problems in the proper context.

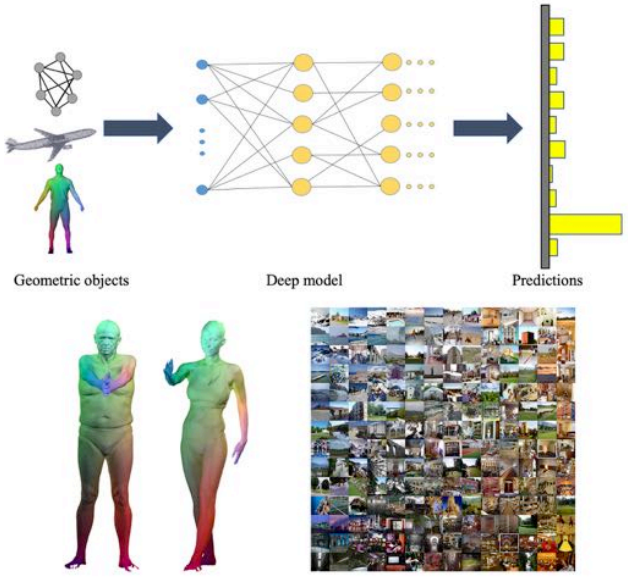


Figure 1: The main problems considered in my Ph.D. thesis: applying deep learning to geometric objects such as point clouds, surfaces, and graphs. Bottom: Two instances of shape matching problems.

Applying deep learning to irregular data is a relatively new problem. The overwhelming success of deep learning in advancing the state of the art in various learning challenges and domains [140] inspires research efforts attempting to achieve similar success for geometric objects such as point-clouds, graphs and discretized surfaces [39]. Adapting deep learning methods to the irregular setting is a particularly interesting and challenging problem since each of these data types admits different representations, and consequently, different symmetries: for example, surfaces and point clouds are invariant to rigid Euclidean transformations, while graphs are invariant to node relabeling. Trying to directly apply commonly used neural networks (*e.g.*, convolutional neural networks or fully connected networks) to irregular data is not well-defined in some cases, or performs poorly in other cases. During the last years, we developed network architectures and layers for all of the data types mentioned above, as well as analyzed widely-used models.

In contrast to the first problem, shape matching problems have been studied for decades [234]. These problems are among the most fundamental problems in geometric data analysis, where the task is finding a (semantically) meaningful map between a pair of shapes. A popular way of handling these hard problems is

by first posing them as quadratic integer optimization problems, relaxing them to a continuous domain and solving the relaxed problem, which is often more tractable. In most cases, there is an inherent tradeoff between the tightness of the relaxation (*i.e.*, how well its solution approximates the original problem's solution), and the computational resources needed to optimize it. During the last few years, we devised several efficient methods for solving widely known relaxations of prevalent matching problems.

4.1 Deep learning of irregular data

The problem setting is as follows: the input consists of n objects O_i, possibly with corresponding descriptor functions $f_i : O_i \to \mathbb{R}^d$, and targets t_i that describe some semantic property of the object (*e.g.*, does a surface represent a dog or a cat). Our task is to find a function $\mathcal{F}$ that approximates this functional relation, *i.e.*, $\mathcal{F}(O_i, f_i) \approx t_i$ and generalizes well to unseen data.

Discretized surfaces and point clouds. We first studied deep learning of discretized surfaces [157] and point clouds [14]. In both cases, our methods are based on the observation that finding a well-behaved mapping from the given object (surface or point cloud) to a domain with a well-defined convolution, allows us to *pullback* this convolution to the object. The pullback operation is done by first mapping a function from the object to this domain, applying the convolution and mapping the result back to the object. In [157] we devised a way to map sphere-type surfaces (*i.e.*, surfaces that are topologically equivalent to a sphere) to a periodic planar domain (a torus), for which we have the well-known 2D convolution. Using this mapping mechanism, we convert input surface descriptors to images, thus reducing the problem of learning surfaces to the problem of learning images. This reduction has the advantage of being able to utilize powerful deep learning machinery and successful architectures developed for images, as opposed to methods that are specifically tailored for surfaces (*e.g.*, [162]). In a follow-up work [23], we used a similar surface-image representation for the task of generating novel shapes using generative adversarial networks. In particular, we have successfully applied our method to the task of human model generation (for another approach for generating realistic 3D surfaces, see [15]). In another recent follow-up work [101], we devise a broad family of mappings from sphere-type surfaces to the torus, which represents surfaces more faithfully due to reduced area distortion. This method achieves state of the art results on 3D shape analysis tasks such as model segmentation and retrieval.

The mapping methods mentioned above rely heavily on the surface's connectivity information and cannot be adapted to the point cloud scenario [14]. In this case, which is of particular interest for applications, we opt for mapping point cloud functions to functions defined on $\mathbb{R}^3$. This is done by defining an *extension* operator that generates a volumetric function from a point cloud function via a Radial Basis Function (RBF) approximation. Similarly, the kernel is defined to be a sum of weighted RBFs, where the weights are the learnable parameters. The convolution of a point cloud function can now be defined using the pullback mechanism mentioned above: first mapping the point cloud function to a volumetric function, applying the standard convolution in $\mathbb{R}^3$ and sampling the result on the point cloud in order to get new point cloud function. This process is illustrated in Figure 2.

Graphs and hyper-graphs In a related line of work, we study a popular model for constructing networks that are invariant to natural transformations of the input object [264, 203, 105, 158]. Given a group $\mathcal{G}$ acting on an input object, this model is composed of a concatenation of equivariant/invariant linear layers with nonlinearities. A fundamental problem when constructing such models is finding the maximal set of linear invariant or equivariant operators with respect to the relevant group. In [158], we addressed this problem for the natural symmetry groups of graphs and hyper-graphs. In this case, the input is an affinity tensor $A \in \mathbb{R}^{n^k}$

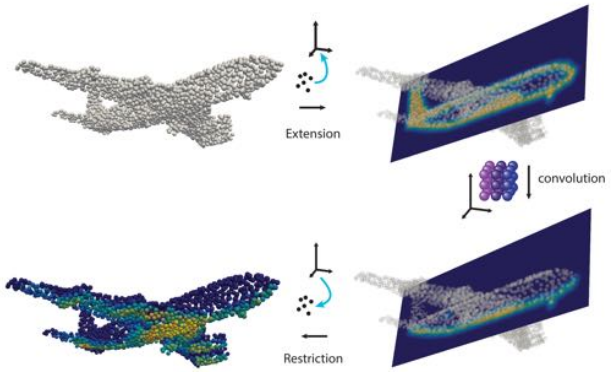


Figure 2: The point cloud convolution suggested in [15]: First, a function over the point cloud (in this case the constant one) is *extended* to a continuous volumetric function over the ambient space; second, a continuous volumetric *convolution* is applied to this function (without any discretization or approximation); and lastly, the result is *restricted* back to the point cloud.

that describes relations between ordered subsets of k elements in a set (*e.g.*, in the case $k = 2$, A is a graph affinity matrix). Note that these tensors adhere to a specific reordering symmetry: reordering the nodes of the hyper-graph results in a different affinity tensor that represents the same hyper-graph. Figure 3 illustrates this symmetry. In this paper, we provided a full characterization of affine functions that are equivariant to this reordering operation. One surprising fact is that the dimension of the space of equivariant functions does not depend on n, the number of nodes. This fact allowed us to construct deep invariant networks that can process graphs of any size. Theoretically, we show that this construction gives rise to a deep invariant model that can approximate any *message passing neural network* [90], the current state of the art in graph neural networks.

In [159], we study the approximation power of the invariant models mentioned above. We consider the rather general case of permutation groups acting on $\mathbb{R}^n$ by permuting the coordinates of vectors and show three main results. The first result, states that this model can approximate any continuous invariant function to arbitrary precision. The proof is constructive and makes use of high order tensors, that is, tensors of the form $A \in \mathbb{R}^{n^k}$ for $k > 1$, which might be computationally prohibitive. Our second result shows that this problem cannot be alleviated since there exist an infinite family of permutation groups for which using high order tensors is necessary for obtaining the universal approximation property. Our last result considers the most important case for applications, *i.e.*, networks that use only first order tensors (*e.g.*, $k = 1$), and provides a necessary condition for a permutation group to have the universal approximation property in this case. In

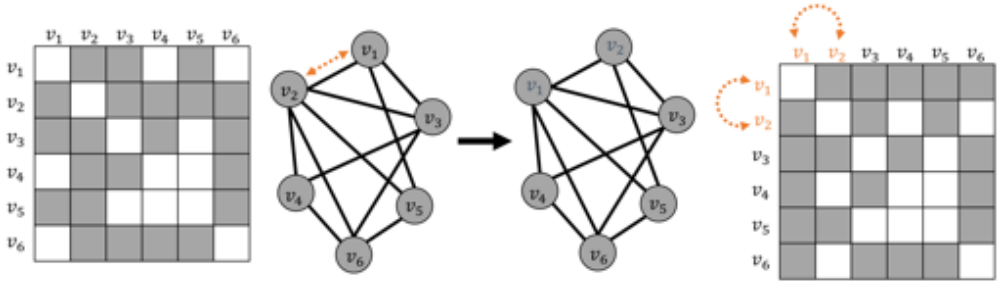


Figure 3: An illustration of symmetries of graph representations. When representing a graph using an adjacency matrix, each order of the nodes gives rise to a new, possibly different adjacency matrix of the same graph. In [158, 161] we suggest neural networks that are invariant to such symmetries.

our latest paper [161] we give more evidence to the fact that higher-order networks are more expressive, and suggest to deviate from the linear equivariant model that was suggested above for achieving better complexity vs. expressivity tradeoff. More precisely, we show that a k-order networks can approximate the k-Weisfeiler Lehman (k-WL) graph isomorphism test [255], which gives rise to graph models that are more powerful than message passing neural networks. We also suggest a new simple architecture, composed of blocks that apply Multi-Layer Perceptrons (MLP) to the edge and node features and then matrix multiplication. This model is shown to have 3-WL expressivity, strictly more powerful than message passing networks. See Figure 4 for an illustration of the suggested block.

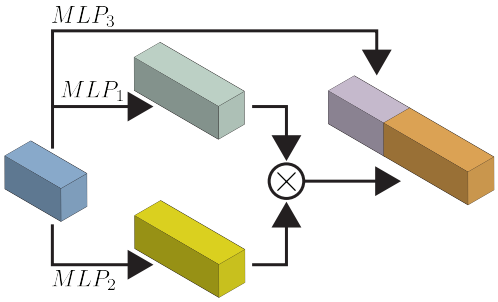


Figure 4: The permutation equivariant block suggested in [161]. Three different Multi-Layer Perceptrons are applied to the feature dimension of the input tensor. Matrix multiplication is applied to the output of two of them while the third output is concatenated.

4.2 Relaxations of matching problems

Three of our works devise scalable approaches for solving well-known tight relaxations of popular matching problems. In all cases, we started from a classic semidefinite relaxation [107] in which the quadratic terms are linearized at the cost of adding a new large optimization variable. In general, this is a tight relaxation that can be solved efficiently for only small-sized problems.

In [160] the problem of jointly aligning and matching two point clouds is considered. More precisely, Given two d-dimensional point clouds, $P, Q \in \mathbb{R}^{d\times n}$, which are neither aligned nor consistently labeled, the task is to find an orthogonal transformation $R \in \mathcal{O}(d)$ and a permutation $X \in \Pi_n$ minimizing the distance between the point clouds:

$$d(P,Q) = \min_{X,R} \|RP - QX\|_F^2 \tag{1a}$$

$$\text{s.t.} \quad X \in \Pi_n \tag{1b}$$

$$R \in \mathcal{O}(d) \tag{1c}$$

This is a central problem in shape analysis with many applications in computer vision and computer graphics. Applying the standard semidefinite relaxation results in a relaxation that can be solved for up to 15 points. Our key insight is that the large semidefinite constraint can be shown to be equivalent to several smaller semidefinite constraints. This observation allowed us to solve problems with significantly larger number of points.

Our motivation in [72] was to find an efficient way of optimizing a tight relaxation to the graph matching problem (GM). Given two graphs represented by affinity matrices $A, B \in \mathbb{R}^{n\times n}$ the GM problem is the problem of finding a permutation matrix X that minimizes a quadratic objective that measures the discrepancy between edge affinities of the graph, *e.g.*,

$$E(X) := -\text{tr}(AXBX^T) \tag{2}$$

As in [160], applying the standard semidefinite relaxation is impractical for real-sized problems. The key contribution of this paper is showing that this semidefinite relaxation is equivalent to a convex quadratic program, which can be solved more efficiently. Using this observation, we optimize this relaxation for graphs of much larger size. Figure 1 (bottom) shows results that were obtained by this method. In [138], we show how to approximately minimize another relaxation that originates from the lifting method described above, by using a Sinkhorn-type method [58].

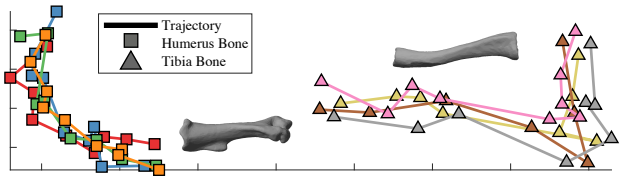


Figure 5: Anatomical dataset embedding in the plane obtained using the method suggested in [156]. Squares and triangles represent different bone types; lines represent temporal trajectories.

In [156], we analyze the common doubly-stochastic relaxation for the GM problem. In this case, the domain of the relaxation is the convex hull of all permutation matrices, and the objective is again $E(X)$. Our first result shows that many instances of this relaxation, *e.g.*, when matching graphs represented by Euclidean distance affinities, are concave relaxations. This is a significant result since concave relaxations have two important advantages: (i) every local minimum is a permutation matrix and (ii) the set of global minima of the original problem and the relaxation is the same. Differently put, the relaxation process does not yield new solutions as well as alleviates the need to project the solution of the relaxed problem onto the permutation matrices (a step which is often not optimal). Our second result shows that many other popular use cases, *e.g.*, when matching graphs represented by geodesic distance affinity matrices, are concave with high probability, meaning it is rare to find a direction on which the restriction of the objective is convex. we also show that in these cases the relaxation enjoys the advantages mentioned above with high probability. Figure 5 illustrates an application of this concave relaxation to anatomical shape space analysis: We match a dataset of 67 mice bone surfaces acquired using micro-CT. The dataset consists of eight time series. Each time series captures the development of one type of bone over time. We used Multi-Dimensional Scaling (MDS) [57] to assign 2D coordinates to each surface using a dissimilarity matrix we obtained from matching all pairs of bones.

Part I
Deep Learning of Irregular Data

5 Learning sphere-type surfaces via toric covers

This section is based on [157].

5.1 Introduction

A recent research effort in the geometry processing and vision communities is to translate the incredible success of deep convolutional neural networks (CNN) to geometric settings. One particularly interesting scenario is applying CNNs for supervised learning of functions or labels over curved two dimensional sphere-like surfaces. This is a common problem in analyzing human bodies, anatomical, and medical data.

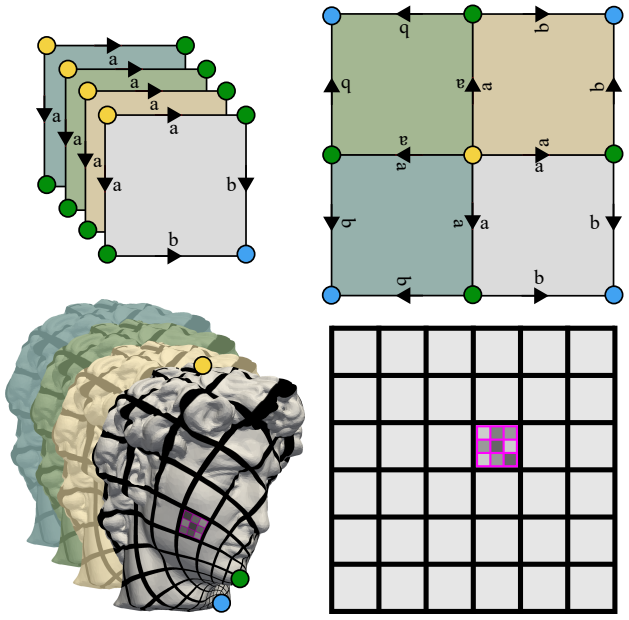


Figure 6: Defining convolutional neural networks on sphere-like surfaces: we construct a torus (top-right) from four exact copies of the surface (top-left) and map it to the flat-torus (bottom-right) to define a local, translation invariant convolution (bottom-left). This construction is unique up to a choice of three points on the surface (colored disks).

Applying CNNs to extrinsic surface representation such as volumetric grids [194] or depth map projections on extrinsic 2D cameras [245] requires working with 3D grids, or dealing with many camera and lighting parameters, and is very sensitive to deformations (e.g., human pose changes). While it might be possible to learn a representation that is invariant to these deformations, this requires substantial amount of training data. In contrast, the goal of this section is to provide an intrinsic representation that would enable applying CNNs directly to the surfaces.

One of the main challenges in applying CNN to curved surfaces is that there is no clear generalization of the convolution operator. In particular, two properties of the convolution operator that are considered pivotal in the success of the CNN architectures are *locality* and *translation invariance*. It is possible to parameterize a surface locally on a geodesic patch around a point [162], however, this representation lacks global context. Sinha et al. [223] proposed geometry images to globally parameterize sphere-like surface into an image,

however, although continuous, their representation is not *seamless* (the parameterization is dependent on the cuts made for its computation), their space of parameterizations, namely area-preserving maps has a very high number of degrees of freedom (*i.e.*, it requires an infinite number of point constraints to uniquely define a mapping) and therefore can represent the same shape in many different arbitrary ways in the image (see Figure 7 for an example). Lastly, the convolution on the geometry image is not translation invariant.

Defining a local translation invariant convolution operator on surfaces is not trivial. The first hurdle is topological: the only surface type for which a translation invariant convolution is well defined is the torus (this will be explained in Section 5.4). However, clearly it is not possible to map sphere-like surfaces to a torus without introducing discontinuities, that is, mapping some neighboring points on the surface to distant ones on the torus. Information will propagate differently through discontinuities and locality of the convolution would be lost. The second difficulty is geometric: We want mappings to be consistent enough across different surfaces, so that mapped test surfaces look similar to training surfaces. This is related to the space of mappings, or in other words, the number of degrees of freedom that are needed to prescribe a unique map, up to translation of the torus. The more parameters one needs the harder it is to learn from these mappings (*e.g.*, in the worst case, target position of every source point could be a degree of freedom).

We tackle these challenges with a topological construction: instead of dealing with the original sphere-like surface we construct a *cover* of the surface that is made out of four exact copies of the surface and has the topology of the torus, see Figure 6 - top row. Furthermore, we show that this torus can be mapped conformally (preserving orthogonal directions) to the flat torus using a very efficient algorithm. This defines a local translation invariant convolution on the 4-cover, see example of a single convolution stencil in Figure 6 - bottom row. This construction is unique up to a choice of three points on the surface; the convolution depicted in Figure 6 is created by the three points (shown as colored disks) in the bottom-left inset.

This construction provides a six dimensional space of seamless convolutions on a sphere-like surface: Every choice of a triplet of points corresponds to a unique conformal map which in turn defines a convolution operator, or equivalently, a conformal flat-torus structure on the 4-cover of the surface. Since isometries are in particular conformal maps this construction is also invariant to isometric deformations of the shapes. The relatively low dimension of the convolution space allows efficient sampling of this space in the context of data augmentation. The conformality preserves the directionality of the translation directions on the flat-torus but introduces scale changes; in that sense the triplet acts as a magnifying glass - zooming into different parts of the surface.

We employ the above constructions for supervised learning over surfaces. Our goal is to learn a non-linear relation between "easy" vector valued functions over surfaces (*e.g.*, coordinate functions, normals, curvature or other commonly used geometric features) and target "hard" vector valued functions (*e.g.*, semantic segmentation or landmark labeling). The conformal map from the 4-cover to the flat-torus will be used to seamlessly transfer these functions to the flat-torus which will be used as our domain for training. To leverage existing image-based CNN architecture and optimization algorithms on the flat-torus domain, we provide three novel technical components: (i) A cyclic-padding layer replaces zero padding to achieve fully-translational invariance over the flat-torus; (ii) a projection layer on the function space of the surface to properly map functions between the original surface and its 4-cover, and (iii) an aggregation procedure to infer prediction from multiple triplets.

Experiments show that our method is able to learn and generalize semantic functions better than state of the art geometric learning approaches in segmentation tasks. Furthermore, it can use only basic local data (Euclidean coordinates, curvature, normals) to achieve high success rate, demonstrating ability to learn high-level features from a low-level signal. This is the key advantage of defining a local translation invariant convolution operator. Finally, it is easy to implement and is fully compatible with current standard CNN implementations for images.

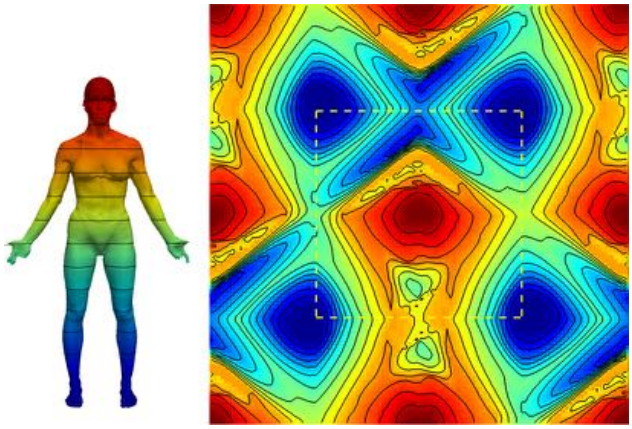


Figure 7: Parameterization produced by the geometry image method of [223]; the parameterization is not seamless as the isolines break at the dashed image boundary (right); although the parameterization preserves area it produces large variability in shape.

5.2 Previous work

Recent advances in convolutional neural networks (CNNs) motivated many researchers to apply these methods to geometric data. Extrinsic representations, such as 3D volumetric grid [194, 252], 2D projections [229, 245, 120], or point coordinates [196], have many shortcomings when analyzing non-rigid 2D manifolds: they are sensitive to articulations, they do not leverage the intrinsic structure of the geometry, and only allow very coarse representations. While these limitations can be addressed by analyzing the manifolds directly, applying CNNs to surfaces is challenging because they do not come with a planar parameterization, and thus are not directly suitable for deep learning. One possible way to address this limitation is to represent a surface as a graph of triangles and use spectral convolutions [43, 109, 63]. However, this representation does not take advantage of the fact that we are analyzing 2-manifold that can be parameterized in 2D domain.Another disadvantage of spectral methods (which is targeted by [261]) is their difficulty with cross-shape learning which stems from the fact that the spectral decomposition of each shape can be inconsistent. We next discuss existing techniques which preform deep learning on 2-manifolds and parameterization methods they use.

For segmentation task, Guo et al. [99] proposed to lay out per-triangle features to a single 2D grid, and used CNN to classify each triangle. This approach cannot use contextual information on relationships between different triangles on the same surface unless this relationships are encoded in the input features.

The first paper adapting neural networks to surfaces, Masci et al. [162], use local geodesic polar coordinates to parameterize a surface patch around a point; and map features to this patch. This parameterization requires modifying the traditional convolution filter to account for angular coordinate ambiguity, essentially ignoring patch orientation. In a follow up work, [32] use anisotropic heat kernels in order to generate a local description of the input function and incorporate orientation.

For classification tasks, Sinha et al. [223] parameterize the entire surface using geometry images [193] combined with spherical area-preserving parameterizations. As mentioned briefly above, geometry images are not seamless and introduce a direction jump at the cut, see Figure 7. Additionally, the convolution over the geometry image is not translation invariant since it represents a sphere-like topology. Finally, since geometry images are computed using area-preserving mappings, which have an infinite number of degrees of freedom, they can produce a wide variability of different planar realizations which will make the learning process more challenging. See, for example, how one of the hands (green) and one of the legs (blue) are strongly sheared in Figure 7 (one copy of the surface is marked with dashed square; all four corners correspond to one of the legs). Lastly, their parameterization algorithm is not guaranteed to produce

a bijective embedding and can cause different parts of the surface to overlap in the geometry image, *e.g.*, only one of the hands is visible in the geometry image in Figure 7.

In contrast to the methods described above, we propose a seamless parameterization technique that maps the surface to a flat torus, thus providing a well-defined convolution everywhere. Our map is uniquely defined by selection of 3 points on the surface, providing a relatively small space of possible parameterizations which makes learning easier. Our map computation routine is very effective, as we only need to solve a sparse system of linear equations per triplet of points.

5.3 Method

Convolutional neural networks (CNN) is a specific family of neural networks that leverages the spatial relationships of local regions using mostly convolutional layers. Deep CNN's with multiple consecutive convolutions followed by nonlinear functions have shown to be immensely successful in image understanding [136]. Our goal is to adapt CNN architectures to geometric surfaces.

5.3.1 Overview

Problem definition. Our training data consists of a set of triplets $\{(\mathcal{S}_i, x_i, y_i)\}_{i\in I}$ of sphere-like surface meshes $\mathcal{S}_i \subset \mathbb{R}^3$, "easy" d-vector valued functions over the surface $\mathcal{S}_i$, denoted $x_i \in \mathcal{F}(\mathcal{S}_i, \mathbb{R}^d)$, and ground-truth "hard" functions $y_i \in \mathcal{F}(\mathcal{S}_i, \mathcal{L})$, where $\mathcal{L} = \{1, \ldots, L\}$ is a label set. By "easy" functions we mean functions that can be computed efficiently over the surfaces, such as coordinate functions, curvature, normals or shape features; by "hard" we mean functions for which no simple algorithm exists, such as a semantic label (e.g., object part) that has to be prescribed manually.

Our goal is to find a non-linear mapping relating the "easy" and "hard" functions on surfaces. Mathematically we are looking for a function F,

$$F : \mathcal{F}(\mathcal{S}_i, \mathbb{R}^d) \to \mathcal{F}(\mathcal{S}_i, \mathbb{R}^L) \tag{3}$$

that takes as input a d-vector valued ("easy") function over a surface $\mathcal{S}_i$ and produces a confidence L-vector valued ("hard") function over the same surface. That is, it produces a vector of confidences $F(x_i)[p] \in \mathbb{R}^L_+$ per point $p \in \mathcal{S}_i$ that correctly predicts its ground-truth label $y_i[p] \in \mathcal{L}$ (*i.e.*, the maximal value in $F(x_i)[p]$ is achieved at its $y_i[p]$-th coordinate).

CNN on the flat-torus $\mathcal{T}$. While CNN is a powerful tool for mapping "easy" to "hard" functions, existing architectures cannot run directly over $\mathcal{S}$. Therefore we propose to transfer functions to a flat torus[1], denoted $\mathcal{T}$, and train CNN over that domain. The flat torus space is favorable since we can use a traditional CNN with 2D convolutions directly to solve the problem over $\mathcal{T}$, by discretizing the flat torus space as an $m \times n$ grid (we used $m = n = 512$).

Mapping $\mathcal{S}$ to $\mathcal{T}$ is not trivial because these two domains have different topologies. We address this issue by considering a new topological construction $\mathcal{S}^4$ (Section 5.3.2). The domain $\mathcal{S}^4$ consists of four copies of the surface cut in the same way to a disk topology and stitched together to one (boundaryless) torus-like surface. We map $\mathcal{S}^4$ conformally to the plane, where these 4 surface copies seamlessly tile the flat-torus. Note that this mapping is not unique, and is defined by a triplet of points on $\mathcal{S}$. Each triplet provides a different image over $\mathcal{T}$ where resolution (surface area scaling) is non-uniform, and varies over $\mathcal{S}$.

[1]The flat-torus is the planar square $[0, 1]^2$ with its opposite sides identified.

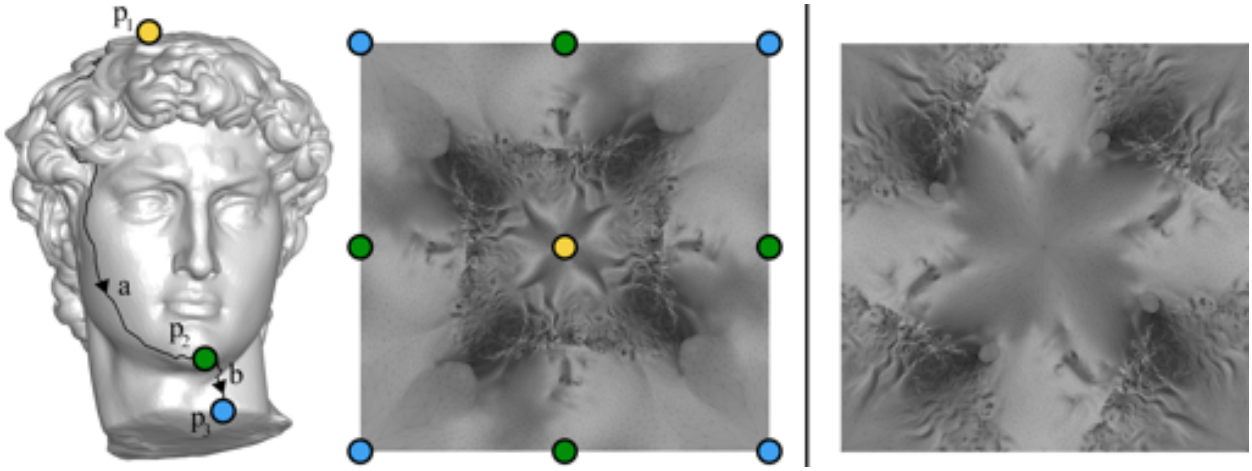


Figure 8: Computing the flat-torus structure (middle) on a 4-cover of a sphere-type surface (left) defined by prescribing three points (colored disks). The right inset shows the flat-torus resulted from a different triplet choice.

We address the mapping ambiguity by modifying network architecture, training data, and the way we interpret the network output (Section 5.3.3). First, we add a new cyclic-padding layer enabling translation-invariant convolutions (i.e., invariance to torus symmetry). Second, we incorporate a projection operator layer that ensures that our network's output is invariant to symmetries of $\mathcal{S}^4$ (i.e., multiple copies of the input surface). Both of these layers are differentiable and support end-to-end training. Third, we sample multiple triplets to produce multiple training images over $\mathcal{T}$, substantially augmenting our training data. And finally, as we analyze a surface at test time, we aggregate several predictions over the surface (produced with different triplets).

5.3.2 Transferring functions between $\mathcal{S}$ and $\mathcal{T}$

A key component of our approach is transferring functions between the surface $\mathcal{S}$ and flat torus $\mathcal{T}$. That is, given a function x_i over the surface $\mathcal{S}_i$ we want to transfer it to the flat-torus in a seamless way that preserves locality and topological (*i.e.*, neighboring) relations. We also want this transfer to be as-unique-as-possible and invariant to isometric deformations of $\mathcal{S}_i$ to avoid the need for unnecessary data augmentation. We next show that there is a unique transfer operator for a given triplet of points $\mathcal{P} = \{p_1, p_2, p_3\} \subset \mathcal{S}_i$.

Since $\mathcal{S}$ and $\mathcal{T}$ have different topologies, to create a desired seamless map between these two domains we define an intermediate surface, $\mathcal{S}^4$, a torus-type 4-cover of $\mathcal{S}$ (branched cover, similarly to [118]). To create $\mathcal{S}^4$ we first make four copies of the surface and cut each one along the path

$$p_1 \xrightarrow{a} p_2 \xrightarrow{b} p_3$$

to obtain disk-like surfaces (Figure 8, left). Next, we stitch the four surfaces according to the instructions shown in Figure 6, top-right, to get a torus-like surface, $\mathcal{S}^4$. Note that this construction is indifferent to the actual cuts used (*e.g.*, a, b in Figure 8) and different cuts would produce the same surface $\mathcal{S}^4$. Lastly, we compute a map $\Phi_{\mathcal{P}} : \mathcal{S}^4 \to \mathcal{T}$ taking S^4 conformally to $\mathcal{T}$ (see Figure 8, middle). In practice, we compute this map by first mapping a single disk-like copy of $\mathcal{S}$ in $\mathcal{S}^4$ (*e.g.*, Figure 8, left) to the plane using the method in [7] (we used the $\{\pi/2, \pi, \pi/2\}$ orbifold) followed by duplicating the resulting planar mesh until we cover the representative square tile of the flat torus, namely $[0, 1]^2$. For weights we used cotan weights with negative values clamped to 10^{-2} to ensure positivity and hence bijective mapping. Per triplet $\mathcal{P}$, this approximately-conformal map can be computed very efficiently by solving a sparse system of linear equations, where the resulting map defines a 2D position for each vertex of the disk-like $\mathcal{S}$.

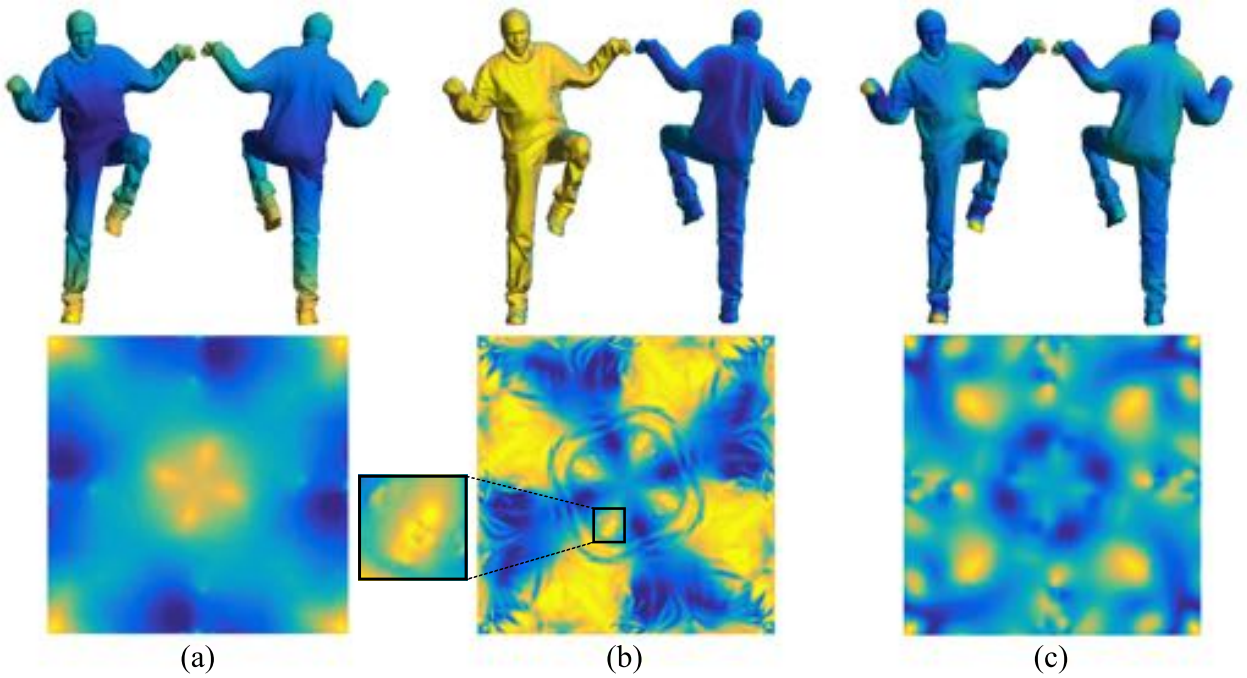


Figure 9: Visualization of "easy" functions on the surface (top-row) and their pushed version on the flat-torus (bottom-row). We show three examples of functions we use as input to the network: (a) average geodesic distance (left), (b) the x component of the surface normal (middle), and (c) Wave Kernel Signature [16]. The blowup shows the face area, illustrating that the input functions capture relevant information in the shape.

We use $\Phi_{\mathcal{P}}$ to transfer functions between the surface $\mathcal{S}$ and the flat-torus $\mathcal{T}$. Given a function $x \in \mathcal{F}(\mathcal{S}, \mathbb{R}^d)$ we define its *push-forward* to the flat-torus, $\text{push}_{\mathcal{P}}(x) \in \mathbb{R}^{m \times n \times d}$, by

$$\text{push}_{\mathcal{P}}(x) = x \circ \Psi \circ \Phi_{\mathcal{P}}^{-1}, \tag{4}$$

where $\Psi : \mathcal{S}^4 \to \mathcal{S}$ is the projection map taking each point in $\mathcal{S}^4$ to its corresponding point in $\mathcal{S}$. That is, given a cell (*i.e.*, pixel) on the discretized torus, we map its centroid to $\mathcal{S}^4$ via the map $\Phi_{\mathcal{P}}^{-1}$, and then to $\mathcal{S}$ via the map Ψ. We evaluate x at that point and assign the corresponding d-dimensional vector value to the cell. In practice, we use Matlab's "patch" command to generate each channel of $\text{push}_{\mathcal{P}}(x)$. Figure 9 visualizes "easy" functions x and their push-forward to $\mathcal{T}$.

An interesting alternative to the above construction of $\mathcal{S}^4$ and the mapping to the flat torus $\mathcal{S}^4 \mapsto \mathcal{T}$ would be to use a single copy of the surface, $\mathcal{S}$, and a mapping to a sphere-type flat representation (Euclidean orbifold) $\mathcal{S} \mapsto \mathcal{O}$. This corresponds to taking one quarter of the flat torus (*e.g.*, upper left quarter of the squares in Figure 9). Although this representation is more compact it will not be topologically correct as the convolution kernels will be applied in different orientations to points on different sides of the cut.

5.3.3 Neural Networks on the flat-torus $\mathcal{T}$

Now that we are able to map functions between $\mathcal{S}$ and $\mathcal{T}$ we explain how we train CNN over the flat torus. A CNN over the flat-torus is defined as a non-linear function taking d-vector valued functions over the torus to L-vector valued function over the torus. Therefore we denote:

$$f(\cdot, w) : \mathbb{R}^{m \times n \times d} \to \mathbb{R}^{m \times n \times L}, \tag{5}$$

where w denotes the network parameters.

We first describe the appropriate architecture for f on $\mathcal{T}$ that takes into account translational symmetries of $\mathcal{T}$ and the fact it is covered by four copies of the surface (i.e., $\mathcal{S}^4$). To train f, we use multiple triplets $\mathcal{P}_k$ to push training examples on the surface $(\mathcal{S}_i, x_i, y_i)$ to the flat-torus, augmenting our training data by mapping the same surface in different ways. We use these training examples to optimize for w, parameters of the CNN. Lastly, we explain how our trained network can be used to analyze an input test shape.

Network architecture for CNN on $\mathcal{T}$. The input and output of the network $f(\cdot, w)$ is represented in the form of discrete two dimensional images, and there are many state-of-the-art network architectures that have proven highly successful for this type of data. In this work, we used the FCN32 CNN architecture of [151] with two main differences: First, since we need f to be fully-translation invariant and well-defined on the flat-torus we employ a cyclic padding instead of the standard zero padding used for images. This is crucial for seamless learning (as demonstrated by experiments in Section 5.5). Second, since there are 4 copies of $\mathcal{S}$ in $\mathcal{S}^4$, several predictions over the flat-torus might correspond to the same point on $\mathcal{S}$. Thus, for the final output of the network $f(\cdot, w) \in \mathbb{R}^{m \times n \times L}$ to be well-defined on $\mathcal{S}$ (so that we can use push^{-1}) we incorporate a projection operator that considers values in the $m \times n$ grid that correspond to the same point in $\mathcal{S}$ and replaces them with their max. Similar to standard pooling layers, averaging corresponding values resulted in inferior results. We implemented two differentiable layers that correspond to these modifications, enabling end-to-end learning for $f(\cdot, w)$.Figure 10 shows the new layers and their incorporation in the network's architecture.

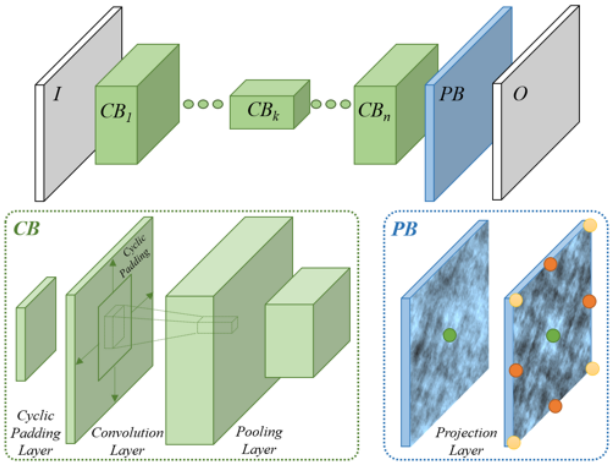


Figure 10: Top: Segmentation network architecture where CB denotes a convolutional block, and PB denotes a projection block. Bottom: Breakdown of the convolutional and projection blocks. Our network has two new layer types: the cyclic padding layer in each convolutional block, and the final projection layer.

We note that the max-projection layer mentioned above has certain resemblance to the TI-pooling operator introduced in [139] which pools over corresponding pixels of multiple transformed versions of the input image and aim at learning transformation invariant features. In contrast, our layer pools over corresponding points on the surface in order to get a single prediction at every point on the surface.

Data generation. To train the network, we first need to push the training data to images defined over the flat-torus $\mathcal{T}$. Given training data $\{(\mathcal{S}_i, x_i, y_i)\}_{i \in I}$, for each i we sample ρ triplets $\mathcal{P} = (p_1, p_2, p_3) \subset \mathcal{S}_i$ from $\mathcal{S}_i \times \mathcal{S}_i \times \mathcal{S}_i$. Then for each $\mathcal{P}$ we create a pair

$$(X_k, Y_k) = (\text{push}_{\mathcal{P}}(x_i), \text{push}_{\mathcal{P}}(y_i)), \tag{6}$$

where each pair corresponds to training input $X_k \in \mathbb{R}^{m\times n\times d}$ and output $Y_k \in \mathbb{R}^{m\times n\times L}$ directly compatible with $f(\cdot, w)$, and k is an index for $|I| \times \rho$ such pairs. The choice of triplets follow the rationale of well-covering the surface to allow each point to be represented with a reasonable (=not too low) scale factor at-least in one map. Hence, we sampled a small number (5-20) of uniformly spread points (including the AGD local maxima [126]) on the surface and randomly sampled triplets from this set.

Training CNN on $\mathcal{T}$. We use this data to train CNN by finding locally optimal parameters w with respect to the following loss function:

$$E(w) = \sum_k \sigma\Big(f(X_k, w), Y_k\Big), \tag{7}$$

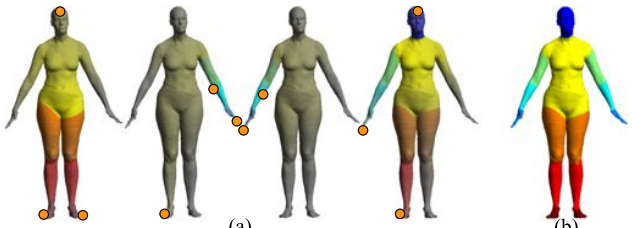
where σ is the standard softmax loss per pixel, weighted by $1/(\delta + c)$; c is the size of the pixel's ground-truth class, and $\delta = 4000$ is a regularizer. We used Matconvnet [235] for training using its SGD (Stochastic gradient descent) with learning rate of 0.0001 as in the original implementation of [151]. We initialized the network with the parameters of FCN32, removing and/or duplicating channels to fit our data dimension.

Aggregating CNN output on $\mathcal{S}$. Given a new surface $\mathcal{S}$ and corresponding vector valued function x, we use the trained CNN to define the final predictor F via:

$$F(x) = \sum_{\mathcal{P}} S(\mathcal{P}) \odot \mathrm{push}_{\mathcal{P}}^{-1}\Big(f(\mathrm{push}_{\mathcal{P}}(x), w)\Big), \tag{8}$$

where $\mathcal{P}$ is a triplet from a set of ρ random triplets, $S(\mathcal{P})$ is a weight function chosen to compensate for the scale changes induced by the mapping $\Phi_{\mathcal{P}}$, and $\odot$ is pointwise multiplication of functions. The weight function $S(\mathcal{P})$ is taken to be the scale factor of the mapping $\Phi_{\mathcal{P}}$. It is defined at every vertex of the surface using a triangle-area weighted average of the adjacent triangles' scale. Our aggregation method is motivated by the observation that the scale factor can serve as a confidence measure for the CNN prediction at each point of the surface.

Figure 11 depicts an aggregation example where the four left models show the contribution of four different triplets $\mathcal{P}$ visualized using orange disks (gray color corresponds to points with low scale factor), and the model on the right is the final result.



(a) (b)

Figure 11: Aggregating predictions from different triplets (four models on the left; triplets shown as orange disks) to produce final prediction (right). Each triplet serves as a magnifying glass for some local or global part of the surface. Note that on the third shape the third point is on the back side of the model.

5.4 Properties

In this section we justify the choice of the flat torus as the target domain and explain the translation invariance of the new convolution operators. Specifically, we show that the convolution operator on $\mathcal{S}^4$ is invariant to a two dimensional group of conformal translations.

Convolution on the flat torus. We start by considering the Euclidean case, namely the flat torus $\mathcal{T}$. A translation is an isometry $\tau_v : \mathcal{T} \to \mathcal{T}$ defined by $\tau_v(x) = x - v$. Translations act on functions over the torus $\tau_v : \mathcal{F}(\mathcal{T}, \mathbb{R}) \to \mathcal{F}(\mathcal{T}, \mathbb{R})$ via $\tau_v(f)(x) = f(\tau_v(x)) = f(x - v)$. *Translation invariance* of the convolution operator means it commutes with the translation operator as was just defined:

$$\tau_v(f * g) = \tau_v(f) * g$$

Conversely, under certain conditions, one can show that a linear and translation invariant operator is a convolution with some fixed function g [61]. Therefore, linearity and translation invariance are defining properties of convolution.

Translations on surfaces. To define a convolution operator on a surface $\mathcal{S}$, a natural requirement is that it would be linear and translation invariant. But what is a translation $\tau : \mathcal{S} \to \mathcal{S}$? In tune with the definition of a surface, a translation on a surface should locally be a Euclidean translation. That is, a flow along a non-vanishing vector field. According to the Poincaré-Hopf Theorem [168] the only surfaces with non-vanishing vector fields are of Euler characteristic zero - in case of closed orientable surfaces, the torus. This implies that the only surfaces on which we can define translations in the above mentioned way are toric.

The pullback convolution. Given two toric (not necessarily flat) surfaces $\mathcal{T}_1, \mathcal{T}_2$ and a homeomorphism $\Phi : \mathcal{T}_1 \to \mathcal{T}_2$ one can define a convolution operator $*_1$ on $\mathcal{T}_1$ from a convolution operator $*_2$ defined on $\mathcal{T}_2$ via

$$f *_1 g = \left(\left(f \circ \Phi^{-1}\right) *_2 \left(g \circ \Phi^{-1}\right)\right) \circ \Phi,$$

The pullback convolution $*_1$ is linear and translation invariant w.r.t. the *pullback translations* $\Phi^{-1} \circ \tau \circ \Phi$, where τ represents translations in $\mathcal{T}_2$ for which $*_2$ is invariant to. This is proved in the Section 5.8.

In our case $\mathcal{T}_1 = \mathcal{S}^4$, $\mathcal{T}_2 = \mathcal{T}$ the flat-torus with the Euclidean convolution and translations (modulo 1), and the mapping $\Phi : \mathcal{S}^4 \to \mathcal{T}$ is a conformal homeomorphism. As the convolution on the flat-torus is invariant to Euclidean translations τ of the type $x \mapsto x - v (\mathrm{mod}\ 1)$, the pullback convolution on $\mathcal{S}^4$ is invariant to the pullback translations $\Phi^{-1} \circ \tau \circ \Phi$. Since Φ, Φ^{-1}, τ are all conformal maps, these pullback translations are conformal maps as well. To visualize an example of these translations consider Figure 6 (bottom left) and imagine that each square on the David surface moves to a well-defined "right" neighbor.

5.5 Evaluation

In this section, we compare our method to alternative approaches for surface parameterization and network architecture. We compare to three other methods: The first two methods focus on the parameterization component of our algorithm, the last one on the architecture:

1. GIM - where we used the geometry images parameterization method of [223] followed by application of the FCN32 network [151]. To the best of our knowledge this is the only parameterization method used with CNNs.

2. Tutte - we consider parameterization using Tutte's embedding [232] to the unit square followed by application of the FCN32 network [151]. To generate the embedding, we use the same cut as in our method and map the boundary in a length-preserving manner (up to global scale) to the boundary of the unit square. This is a natural selection for comparison, since Tutte's embedding can be computed as-efficiently as our method by solving a sparse system of linear equations.

3. Seamless+FCN - where we use our parameterization technique but without the two additional layers of cyclic padding and projection. This is equivalent to considering the flat-torus as a disk with its opposite sides disconnected.

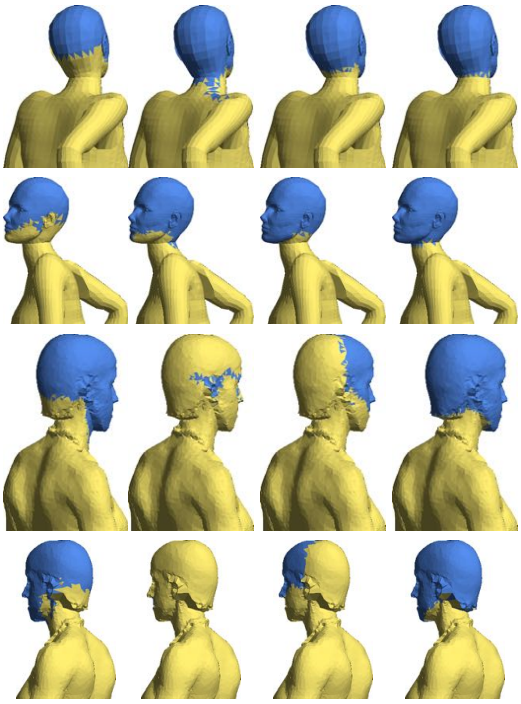


Figure 12: Head segmentation on two test surfaces by the four algorithms in Table 1. Our algorithm produces accurate segmentation, while competing methods provide suboptimal results.

We perform the evaluation on the task of segmenting the head in human models. Our training set for this task is composed of 370 models from the SCAPE [10], FAUST [30] and MIT animation datasets [240]. The ground truth labels are head indicator functions that are manually labeled. Our test data are the 18 sphere-like human models from the SHREC dataset [91]. We use *intersection-over-union* as our evaluation metric: Denoting by GT the set of faces labeled "head" and by ALG the faces labeled "head" by the algorithm, this metric as defined as

$$\frac{|GT \cap ALG|}{|GT \cup ALG|}$$

weighted by triangle areas. In all experiments, as input features ("easy" functions over the surfaces), we use a set of 26 basic shape features: 21 WKS features (equally spaced), 4 curvature features (max, min, arithmetic mean and geometric mean of the principal curvatures) and average geodesic distance (AGD) as input. We initialize training with the parameters obtained by the FCN32 net [151]. We trained all networks with the same parameters and same number of epochs.

For all methods we considered $\rho = 10$ different parameterizations per mesh, resulting in a dataset with 3700 segmented images of size 512×512. For all methods excluding GIM, the different parameterizations correspond to $\rho = 10$ choices of triplets. For GIM the different parameterizations correspond to 10 uniform rotations around the polar axis of the sphere as suggested in [223].All networks converged after 20 epochs. For GIM we tried applying aggregation like in Eq. (8) with two choices of parameterization weighting: scale dependent (like in our method), and uniform (all predictions get the same weight). The rationale in the

second version is that GIM uses area preserving parameterizations so theoretically no scaling is introduced. Both aggregation methods produced the same results.

	intersection-over-union
1. Geometry images + FCN	0.625
2. Tutte + FCN	0.567
3. Seamless + FCN	0.503
4. Ours	**0.710**

Table 1: Evaluation of CNN on surfaces with different parameterization methods and network architectures.

Table 1 summarizes the results. Our method achieves superior results with respect to all alternatives. The results indicate that the layers added to the network to account for the flat-torus topology indeed play an important role.

Figure 12 shows results of the head segmentation task of the four algorithms on two different models from the test set. The first two rows show the first model from two viewing directions, and the third and fourth rows show the second model. For both models our segmentation was satisfactory while the segmentation of the remaining methods was suboptimal. The small variability in the head shape in our parametric space, as well invariance to cuts that could pass through the head, allow our method to learn the head function to a greater accuracy.

In an additional experiment we replaced the weighted aggregation method described in (8) with maximal aggregation method and found that the performance of all methods degraded, with our algorithm still providing superior results to the alternative methods.

5.6 Applications

In this section we demonstrate the usefulness of our method for two applications: semantic segmentation and automatic landmark detection on anatomical surfaces.

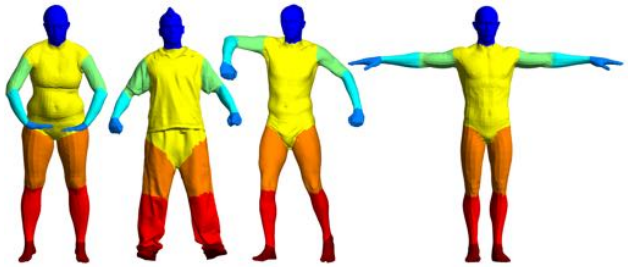


Figure 13: Examples from the semantic segmentation training set.

5.6.1 Semantic segmentation of surfaces

We applied our algorithm to the task of semantic segmentation of human models. As training data we used 370 models from SCAPE, FAUST, MIT (excluding two classes which are not suitable for full-body segmentation), and Adobe Fuse [3]. All models are manually segmented into eight labels according to the labels in [119]. Our test set is again the 18 models from the SHREC07 dataset in human category (all sphere-like models). Note that, in contrast with previous works, the training set does not include any models from the SHREC07 dataset which we use solely for testing. Figure 13 shows examples from the training set.

method	# feat	features used	accuracy
Ours	10	normals, Euclidean, curv.	81.6%
Guo	10	normals, Euclidean, curv.	43.6%
Ours	26	WKS, AGD, curv.	**88.0**%
Guo	26	WKS, AGD, curv.	76.0%
Guo	600	HKS, WKS, AGD, curv.	87.8%

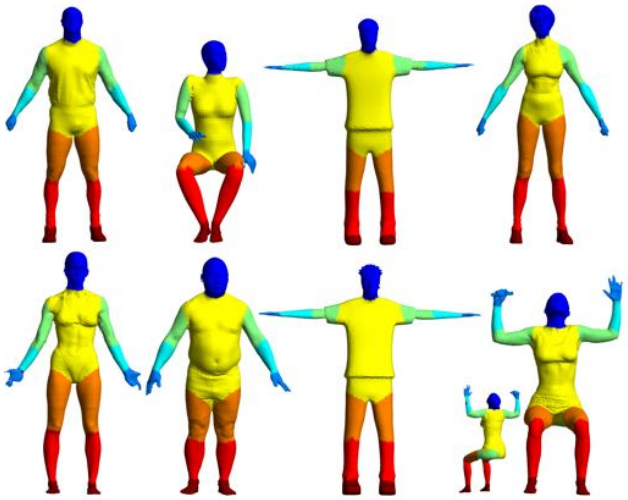


Figure 14: Results of our method for human body segmentation. Bottom right: Failure case where part of the thigh was incorrectly identified.

We generated data from 300 triplets per model in the training set which resulted in approximately 110K segmented images. The networks converged after 30-50 epoches. We compared our method to the state-of-the-art CNN method for mesh segmentation [99]. Both methods were trained on the same training data. Gou et al. also use convolutional nets, but on per-triangle features reshaped to a square grid. These convolutions do not enable aggregating information from nearby regions on the surface, so to get global context Guo et al. need to leverage global features. The training set for Guo et al. was created by randomly sampling $\sim 360k$ triangles from the set of training models. We applied their method with different sets of training features (ranging from 10 to 600) and as shown in Table 2 their performance drops considerably as the number of training features decreases; accuracy is measured as ratio of correctly labeled triangles, weighted by triangle areas. In contrast, our method can learn features by leveraging convolution defined over the surface, demonstrating consistently better performance with the same number of features. In fact, our method, using only 26 features, modestly outperforms the method of Guo et al. with 600 features, see Table 2 (for evaluation we used $\rho = 260/480$ triplets for $26/10$ features, respectively). Figure 14 shows several results obtained with our algorithm.

Figure 15 shows the segmentation accuracy for the above 26 feature experiment as a function of the number of triplets ρ used for aggregation. Using 5 triplets already provides meaningful results, while 10 triplets are almost identical to the final result with ~ 300 triplets.

Figure 16 shows successful application of the human segmentation method to shapes from other classes: robot [263], armadillo and four-legged [91]. Note that although the training set contains only standard human segmentation data (see Figure 13), the network still produces plausible segmentations on this very different set of models. This demonstrates the robustness and generalization power of our method.

The robot example also demonstrates a possible way to bypass the sphere topology restriction of our method:

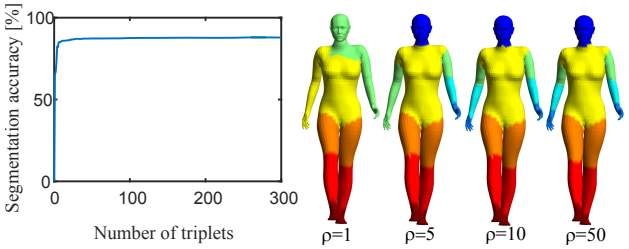


Figure 15: Average result of human body segmentation as a function of the number of triplets used in the prediction (using 26 features on the SHREC07 dataset).

In this case the input surface contains multiple connected components and non-manifold edges. We approximated the input with a genus-0 surface using a simple reconstruction algorithm [269] and applied our method to it. The result (shown on the approximated surface) is plausible.

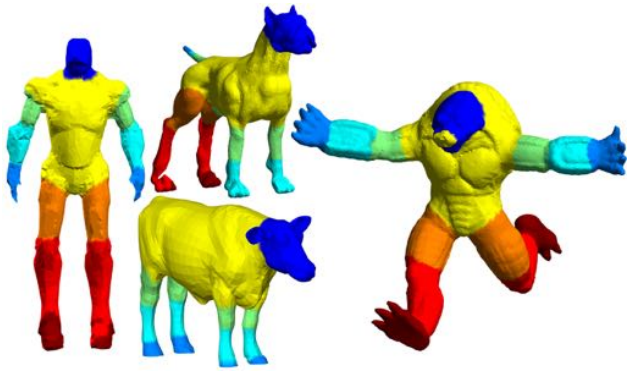


Figure 16: Results of our algorithm trained solely on human body semantic segmentation applied to surfaces from other classes. Note that the method still produces semantic plausible results, which demonstrates the strong generalization power of the method.

5.6.2 Landmark detection on anatomical surfaces

Our algorithm can be applied to automatic landmark detection on 3D shapes in general and on biological data in particular. Here we show results of an experiment we conducted on models of animal teeth from the [34] dataset. On each tooth 6 biologically significant landmarks where manually marked by experts. Our task here is to detect these landmarks on an unseen tooth. For this purpose we took 81 teeth from this dataset, converted them to sphere topology using [117] and marked each landmark area using a geodesic disk of constant radius. We trained on a random subset of 73 teeth and tested on 8 models. For each tooth we generated 125 triplets, resulting in a training set of approximately 9K images. For the input "easy" function we only used curvature and (the logarithm of) the conformal scale factor. The network converged after 50 epoches.

Figure 17 shows the results we obtained on all eight test models. In a dashed rectangle we show the ground-truth labeling for the tooth in the middle of the first row. Although we used only a five-dimensional vector of basic features our method was successful in identifying most of the landmarks despite the local as-well as global variability in the data.

This dataset contains both right-side as-well as left-side teeth. The landmarks are therefore reflected when comparing right- and left-side teeth. Remarkably, our algorithm is able to correctly label landmarks on both right- and left-side teeth according to their biological meaning and is not "fooled" by their orientation (see the first tooth in the second row).

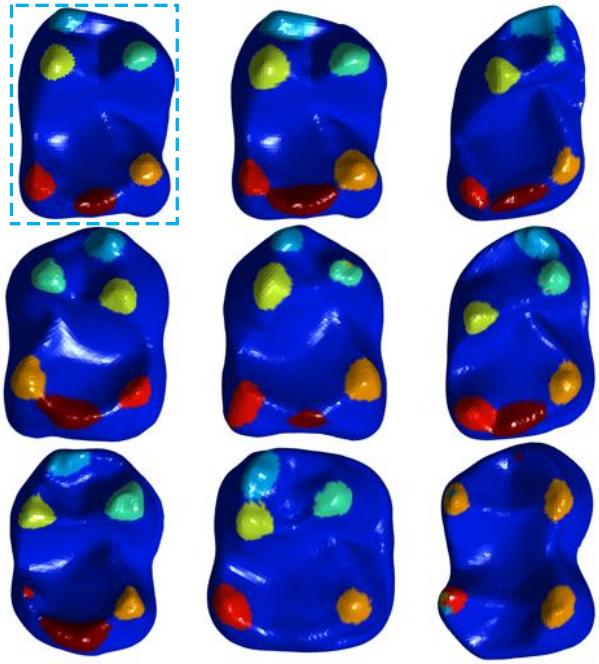


Figure 17: Landmark detection in anatomical surfaces. Results on the test set are shown; in dashed rectangle we show ground-truth for the tooth in the middle of the first row.

In the bottom row of Figure 17, one of the landmark areas in the first tooth from the left is small, but is still detected. In the middle tooth 5/6 landmarks were detected successfully and one is missing (possibly due to the lack of the ridge on which the missing landmark is usually marked). In the last tooth on the right only one landmark was detected successfully. The failure in this case may be related to the fact that both the genus of the animal and the specific peak structure are not represented in the training set [155]. This experiment further demonstrates the ability of our method to learn high-level semantic data from low-level information on the surface.

Using the output of our algorithm, we extract the point landmarks by computing the geodesic centroid of each label. Figure 18 shows a quantitative evaluation of the keypoints we extracted using the above method compared to ground truth. We plot the fraction of the predicted points (y-axis) that are within a certain geodesic error threshold of their true position (x-axis). We compared our algorithm to a baseline random forest classifier [35] (using maltab's implementation) which was recently shown to be a successful classifier for shape analysis problems [207]. The input per-face feature vectors consisted of the 600 WKS, HKS, curvature and AGD as before, with additional (logarithm of) the conformal scale factor generate with 125 triplet (generated as above). We tried two versions - the first with 50 trees and 73K sampled faces and the second with 100 trees and 292K sampled faces. Our algorithm was able to extract more accurate landmarks despite the large number and expressive power of the features fed to the baseline.

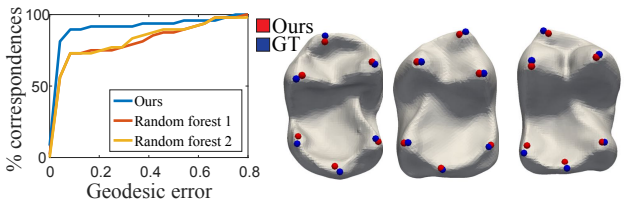


Figure 18: Quantitative evaluation of the landmarks extracted by our method.

In Figure 19 we show a smooth bijective map between a pair of teeth obtained by interpolating the 6 landmarks identified by our method. The map was obtained using the method of [6]. This provides a fully automatic pipeline for producing semantically correct mappings between biological surfaces.

5.6.3 Timing

We present average running times. Computing the parameterization on a mesh with 12.5K vertices takes 0.862 seconds for a given triplet of cones. Computing the scale factor of the parametrization (used for aggregation) takes 0.534 seconds. The training can process about one image with 5-26 channels per second for a single GPU in Nvidia K80. We used three such dual GPUs which made the training 6 times faster. For a dataset containing 110k images (as in the full segmentation experiment) a single epoch takes about 5 hours. Feed-forward calculation in the network (using a single GPU of Nvidia K80) takes 0.35 seconds on average for a single image with 5-26 channels. Full prediction for a single triplet (feed-forward and pull-back of functions to the surface) takes about 2.94 seconds, and this process can be parallelized for multiple triplets. Consequently, in case of sequential runs, a prediction on a single model with 1/10/50 triples takes 3/30/150 seconds. Using 50 triples, it takes 45 minutes to calculate predictions on the human class of SHREC07 and 20 minutes on the teeth dataset. These experiments were done on an Intel Xeon E5 CPU with 64GB of RAM.

Figure 19: A visualization of a map between teeth obtained by interpolating the correspondence between the 6 landmarks found automatically by our method.

5.7 Conclusion

We presented a methodology and an algorithm for applying deep convolutional neural networks to geometric surfaces. The algorithm is based on seamless, conformal mapping of surfaces to the flat-torus on which convolution is well defined. Standard CNN architecture can then be used with minor modifications to perform supervised learning on the flat-torus. We demonstrated the usefulness of our approach for semantic segmentation and automatic landmark detection on anatomical surfaces, and showed it compares favorably to competing methods.

A limitation of our technique is that it assumes the input shape is a mesh with a sphere-like topology. An interesting direction for future work is extending our method to meshes with arbitrary topologies. This problem is especially interesting since in certain cases shapes from the same semantic class may have different genus. Another limitation is that currently aggregation is done as a separate post-process step and not as a part of the CNN optimization. An interesting future work in this regard is to incorporate the aggregation in the learning stage and produce end-to-end learning framework.

5.8 Proofs

We prove that given a bijection $\Phi : \mathcal{T}_1 \to \mathcal{T}_2$ and assuming $\tau(f *_2 g) = \tau(f) *_2 g$ we have that $\overline{\tau}(f *_1 g) = \overline{\tau}(f) *_1 g$ where $\overline{\tau} = \Phi^{-1} \circ \tau \circ \Phi$. As before, we use the notation $\tau(f) = f \circ \tau$. First,

$$\begin{aligned}
\overline{\tau}(f *_1 g) &= \left[(f \circ \Phi^{-1}) *_2 (g \circ \Phi^{-1})\right] \circ \Phi \circ \left[\Phi^{-1} \circ \tau \circ \Phi\right] \\
&= \left[(f \circ \Phi^{-1}) *_2 (g \circ \Phi^{-1})\right] \circ \tau \circ \Phi \\
&= \tau \left[(f \circ \Phi^{-1}) *_2 (g \circ \Phi^{-1})\right] \circ \Phi
\end{aligned}$$

On the other hand we have

$$\begin{aligned}
\overline{\tau}(f) *_1 g &= \left[(\overline{\tau}(f) \circ \Phi^{-1}) *_2 (g \circ \Phi^{-1})\right] \circ \Phi \\
&= \left[(f \circ \Phi^{-1} \circ \tau \circ \Phi \circ \Phi^{-1}) *_2 (g \circ \Phi^{-1})\right] \circ \Phi \\
&= \left[(f \circ \Phi^{-1} \circ \tau) *_2 (g \circ \Phi^{-1})\right] \circ \Phi \\
&= \left[\tau(f \circ \Phi^{-1}) *_2 (g \circ \Phi^{-1})\right] \circ \Phi \\
&= \tau \left[(f \circ \Phi^{-1}) *_2 (g \circ \Phi^{-1})\right] \circ \Phi,
\end{aligned}$$

where in the last equality we used the invariance of $*_2$ to τ. We proved invariance of $*_1$ to $\overline{\tau}$.

6 Convolutional neural networks on point clouds by extension operators

This section is based on [14].

6.1 Introduction

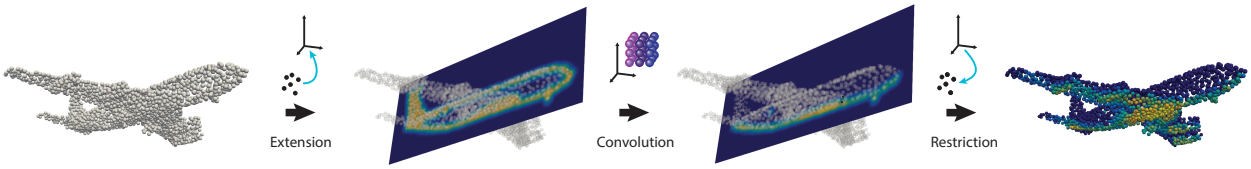


Figure 20: A new framework for applying convolution to functions defined over point clouds: First, a function over the point cloud (in this case the constant one) is *extended* to a continuous volumetric function over the ambient space; second, a continuous volumetric *convolution* is applied to this function (without any discretization or approximation); and lastly, the result is *restricted* back to the point cloud.

The huge success of deep learning in image analysis motivates researchers to generalize deep learning techniques to work on 3D shapes. Differently from images, 3D data has several popular representation, most notably surface meshes and points clouds. Surface-based methods exploit connectivity information for 3D deep learning based on rendering [229], local and global parameterization [162, 223, 157], or spectral properties [261]. Point cloud methods rely mostly on points' locations in three-dimensional space and need to implicitly infer how the points are connected to form the underlying shape.

The goal of this section is to introduce Point Cloud Convolutional Neural Networks (PCNN) generalizing deep learning techniques, and in particular Convolutional Neural Networks (CNN) [136], to point clouds. As a point cloud $X \subset \mathbb{R}^3$ is merely an approximation to some underlying shape S, the main challenges in building point cloud networks are to achieve: (i) Invariance to the order of points supplied in X; (ii) Robustness to sampling density and distribution of X in S; and (iii) Translation invariance of the convolution operator (*i.e.*, same convolution kernel is used at all points) .

Invariance to point order in X was previously tackled in [196, 202, 195, 264] by designing networks that are composition of euuquivariant layers (*i.e.*, commute with permutations) and a final symmetric layer (*i.e.*, invariant to permutations). As shown in [202], any linear equivariant layer is a combination of scaled identity and constant linear operator and therefore missing many of the degrees of freedom existing in standard linear layers such as fully connected and even convolutional.

Volumetric grid methods [252, 163, 194, 204] use 3D occupancy grid to deal with the point order in X and provide translation invariance of the convolution operator. However, they quantize the point cloud to a 3D grid, usually producing a crude approximation to the underlying shape (*i.e.*, piecewise constant on voxels) and are confined to a fixed 3D grid structure.

Our approach toward these challenges is to define CNN on a point cloud X using a pair of operators we call *extension* $\mathcal{E}_X$ and *restriction* $\mathcal{R}_X$. The extension operator maps functions defined over the point cloud X to volumetric functions (*i.e.*, functions defined over the entire ambient space $\mathbb{R}^3$), where the restriction operator does the inverse action. Using $\mathcal{E}_X, \mathcal{R}_X$ we can translate operators such as Euclidean volumetric convolution to point clouds, see Figure 20. In a nutshell, if O is an operator on volumetric functions then its restriction to the point cloud X would be

$$O_X = \mathcal{R}_X \circ O \circ \mathcal{E}_X. \tag{9}$$

We take $\mathcal{E}_X$ to be a Radial Basis Function (RBF) approximation operator, and $\mathcal{R}_X$ to be a sampling operator, *i.e.*, sample a volumetric function at the points in X. As O we take continuous volumetric convolution operators with general kernels κ represented in the RBF basis as-well. In turn (9) is calculated using a sparse linear tensor combining the learnable kernel weights k, function values over the point cloud X, and a tensor connecting the two, defined directly from the point cloud X.

Since our choice of $\mathcal{E}_X$ is invariant to point order in X, and $\mathcal{R}_X$ is an equivariant operator (w.r.t. X) we get that O_X in (9) is equivariant. This construction leads to new equivariant layers, in particular convolutions, with more degrees of freedom compared to [202, 196, 264]. The second challenge of robustness to sampling density and distribution is addressed by the *approximation power* of the extension operator $\mathcal{E}_X$. Given a continuous function defined over a smooth surface, $f : S \to \mathbb{R}$, we show that the extension of its restriction to X approximates the restriction of f to S, namely

$$\mathcal{E}_X \circ \mathcal{R}_X[f] \approx f\Big|_S.$$

This means that two different samplings $X, X' \subset S$ of the same surface function are extended to the *same volumetric function*, up to an approximation error. In particular, we show that extending the simplest, constant one function over the point cloud, $\mathcal{E}_X[\mathbf{1}]$, approximates the indicator function of the surface S, while the gradient, $\nabla\mathcal{E}_X[\mathbf{1}]$, approximates the mean curvature normal field over the surface. Then, the translation invariance and robustness of our convolution operator naturally follows from the fact that the volumetric convolution is translation invariant and the extension operator is robust.

PCNN provides a flexible framework for adapting standard image-based CNNs to the point cloud setting, while maintaining data only over the point cloud on the one hand, and learning convolution kernels robust to sampling on the other. We have tested our PCNN framework on standard classification, segmentation and normal estimation datasets where PCNN outperformed all other point cloud methods and the vast majority of other methods that use more informative shape representations such as surface connectivity.

6.2 Previous Work

We review different aspects of geometric deep learning with a focus on the point cloud setting. For a more comprehensive survey on geometric deep learning we refer the reader to [39].

Deep learning on point clouds. PointNet [196] pioneered deep learning for point clouds with a Siamese, per-point network composed with a symmetric max operator that guarantees invariance to the points' order. PointNet was proven to be a universal approximator (*i.e.*, can approximate arbitrary continuous functions over point clouds). A follow up work [195] suggests a hierarchical application of the PointNet model to different subsets of the point cloud; this allows capturing structure at different resolutions when applied with a suitable aggregation mechanism. In [98] the PointNet model is used to predict local shape properties from point clouds. In a related work [202, 264] suggest to approximate set function, with equivariant layers composed with a symmetric function such as max. Most related to our work is the recent work of [129] that suggested to generalize convolutional networks to point clouds by defining convolutions directly on kd-trees built out of the point clouds [24], and [216] that suggested a convolutional architecture for modeling quantum interactions in molecules represented as point clouds, where convolutions are defined by multiplication with continuous filters. The main difference to our work is that we define the convolution of a point cloud function using an exact volumetric convolution with an extended version of the function. The approximation properties of the extended function facilitate a robust convolution on point clouds.

Volumetric methods. Another strategy is to generate a tensor volumetric representation of the shape restricted to a regular grid (*e.g.*, by using occupancy indicators, or a distance function) [252, 163, 194]. The main limitation of these methods is the approximation quality of the underlying shape due to the low resolution enforced by the three dimensional grid structure. To overcome this limitation a few methods suggested to use sparse three dimensional data structures such as octrees [244, 204]. Our work can be seen as a generalization of these volumetric methods in that it allows replacing the grid cell's indicator functions as the basis for the network's functions and convolution kernels with more general basis functions (*e.g.*, radial basis functions).

Deep learning on Graphs. Shapes can be represented as graphs, namely points with neighboring relations. In spectral deep learning the convolution is being replaced by a diagonal operator in the graph-Laplacian eigenbasis [43, 63, 109]. The main limitation of these methods in the context of geometric deep learning is that different graphs have different spectral bases and finding correspondences between the bases or common bases is challenging. This problem was recently targeted by [261] using the functional map framework.

Deep learning on surfaces. Other approaches to geometric deep learning work with triangular meshes that posses also connectivity and normal information, in addition to the point locations. One class of methods use rendering and 2D projections to reduce the problem to the image setting [229, 120]. Another line of works uses local surface representations [162, 32, 171] or global parameterizations of surfaces [223, 157] for reducing functions on surfaces to the planar domain or for defining convolution operators directly over the surfaces.

RBF networks. RBF networks are a type of neural networks that use RBF functions as an activation layer, see [183, 184]. This model was first introduced in [40], and was used, among other things, for function approximation and time series prediction. Usually, these networks have three layers and their output is a linear combination of radial basis functions. Under mild conditions this model can be shown to be a universal approximator of functions defined on compact subsets of $\mathbb{R}^d$ [188]. Our use of RBFs is quite different: RBFs are used in our extension operator solely for the purpose of defining point cloud operators, whereas the ReLU is used as an activation.

6.3 Method

Notations. We will use tensor (*i.e.*, multidimensional arrays) notation, *e.g.*, $a \in \mathbb{R}^{I\times I\times J\times L\times M}$. Indexing a particular entry is done using corresponding lower-case letters, $a_{ii'jlm}$, where $1 \leq i, i' \leq I$, $1 \leq j \leq J$, etc. When summing tensors $c = \sum_{ijl} a_{ii'jlm} b_{ijl}$, where $b \in \mathbb{R}^{I\times J\times L}$ the dimensions of the result tensor c are defined by the free indices, in this case $c = c_{i'm} \in \mathbb{R}^{I\times M}$.

Goal. Our goal is to define convolutional neural networks on point clouds $X = \{x_i\}_{i=1}^{I} \in \mathbb{R}^{I\times 3}$. Our approach to defining point cloud convolution is to extend functions on point clouds to volumetric functions, perform standard Euclidean convolution on these functions and sample them back on the point cloud.

We define an extension operator

$$\mathcal{E}_X : \mathbb{R}^{I\times J} \to C(\mathbb{R}^3, \mathbb{R}^J), \tag{10}$$

where $\mathbb{R}^{I\times J}$ represents the collection of functions $f : X \to \mathbb{R}^J$, and $C(\mathbb{R}^3, \mathbb{R}^J)$ volumetric functions $\psi : \mathbb{R}^3 \to \mathbb{R}^J$. Together with the extension operator we define the restriction operator

$$\mathcal{R}_x : C(\mathbb{R}^3, \mathbb{R}^M) \to \mathbb{R}^{I\times M}. \tag{11}$$

Given a convolution operator $O : C(\mathbb{R}^3, \mathbb{R}^J) \to C(\mathbb{R}^3, \mathbb{R}^M)$ we adapt O to the point cloud X via (9). We will show that a proper selection of such point cloud convolution operators possess the following desirable properties:

1. *Efficiency*: O_X is computationally efficient.
2. *Invariance*: O_X is indifferent to the order of points in X, that is, O_X is equivariant.
3. *Robustness*: Assuming $X \subset S$ is a sampling of an underlying surface S, and $f \in C(S, \mathbb{R})$, then $\mathcal{E}_X \circ \mathcal{R}_X[f] \in C(\mathbb{R}^3, \mathbb{R})$ approximates f when sampled over S and decays to zero away from S. In particular $\mathcal{E}_X[\mathbf{1}]$ approximates the volumetric indicator function of S, where $\mathbf{1} \in \mathbb{R}^{I\times 1}$ is the vector of all ones; $\nabla\mathcal{E}_X[\mathbf{1}]$ approximates the mean curvature normal field over S. The approximation property in particular implies that if $X, X^* \subset S$ are different samples of S then $O_X \approx O_{X*}$.
4. *Translation invariance*: O_X is translation invariant, defined by a stationary (*i.e.*, location independent) kernel.

In the next paragraphs we define these operators and show how they are used in defining the main building blocks of PCNN, namely: convolution, pooling and upsampling. We discuss the above theoretical properties in Section 6.4.

6.3.1 Extension operator

The extension operator $\mathcal{E}_X : \mathbb{R}^{I\times J} \to C(\mathbb{R}^3, \mathbb{R}^J)$ is defined as an operator of the form,

$$\mathcal{E}_X[f](x) = \sum_i f_{ij}\ell_i(x), \tag{12}$$

where $f \in \mathbb{R}^{I\times J}$, and $\ell_i \in C(\mathbb{R}^3, \mathbb{R})$ can be thought of as basis functions defined per evaluation point x. One important family of bases are the Radial Basis Functions (RBF), that were proven to be useful for surface representation[26, 47]. For example, one can consider interpolating bases (*i.e.*, satisfying $\ell_i(x_{i'}) = \delta_{ii'}$) made out of an RBF $\Phi : \mathbb{R}_+ \to \mathbb{R}$. Unfortunately, computing (12) in this case amounts to solving a dense linear system of size $I \times I$. Furthermore, it suffers from bad condition number as the number of points is increased [248]. In this section, we will advocate a novel approximation scheme of the form

$$\ell_i(x) = c\omega_i\Phi(|x - x_i|), \tag{13}$$

where c is a constant depending on the RBF Φ and ω_i can be thought of the amount of shape area corresponding to point x_i. A practical choice of ω_i is

$$\omega_i = \frac{1}{c\sum_{i'}\Phi(|x_{i'} - x_i|)}. \tag{14}$$

Note that although this choice resembles the Nadaraya-Watson kernel estimator [178], it is in fact different as the denominator is independent of the approximation point x; this property will be useful for the closed-form calculation of the convolution operator.

Figure 21: Applying the extension operator to the constant **1** function over three airplane point clouds in different sampling densities: 2048, 1024 and 256 points. Note how the extended functions resemble the airplane indicator function, and hence similar to each other.

As we prove in Section 6.4, the point cloud convolution operator, O_X, defined using the extension operator, (12)-(13), satisfies the properties (1)-(4) listed above, making it suitable for deep learning on point clouds. In fact, as we show in Section 6.4, robustness is the result of the extension operator $\mathcal{E}_X$ approximating a continuous, sampling independent operator over the underlying surface S denoted $\mathcal{E}_S$. This continuous operator applied to a function f, $\mathcal{E}_S[f]$, is proved to approximate the restriction of f to the surface S.

Figure 21 demonstrates the robustness of our extension operator $\mathcal{E}_X$; applying it to the constant one function, evaluated on three different sampling densities of the same shape, results in approximately the same shape.

6.3.2 Kernel model

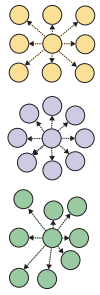
We consider a continuous convolution operator $O : C(\mathbb{R}^3, \mathbb{R}^J) \to C(\mathbb{R}^3, \mathbb{R}^M)$ applied to vector valued function $\psi \in C(\mathbb{R}^3, \mathbb{R}^J)$,

$$O[\psi](x) = \psi * \kappa\,(x) = \int_{\mathbb{R}^3} \sum_j \psi_j(y)\,\kappa_{jm}(x-y)\,dy, \tag{15}$$

where $\kappa \in C(\mathbb{R}^3, \mathbb{R}^{J\times M})$ is the convolution kernel that is also represented in the same RBF basis:

$$\kappa_{jm}(z) = \sum_l k_{ljm}\Phi(|z - y_l|), \tag{16}$$

where with a slight abuse of notation we denote by $k \in \mathbb{R}^{L\times J\times M}$ the tensor representing the continuous kernel in the RBF basis. Note, that k represents the network's learnable parameters, and has similar dimensions to the convolution parameters in the image case (*i.e.*, spatial dimensions $\times$ input channels $\times$ output channels).



The translations $\{y_l\}_{l=1}^L \subset \mathbb{R}^3$ are also a degree of freedom and can be chosen to generate a regular $3\times3\times3$ grid or any other point distribution such as spherical equispaced points. The translations can be predefined by the user or learned (with some similarly to [60]). See inset for illustration of some possible translations.

6.3.3 Restriction operator

Our restriction operator $\mathcal{R}_X : C(\mathbb{R}^3, \mathbb{R}^J) \to \mathbb{R}^{I\times J}$ is the sampling operator over the point cloud X,

$$\mathcal{R}_X[\psi] = \psi_j(x_i), \tag{17}$$

where $\psi \in C(\mathbb{R}^3, \mathbb{R}^J)$. Note that $\mathcal{R}_X[\psi] \in \mathbb{R}^{I\times J}$.

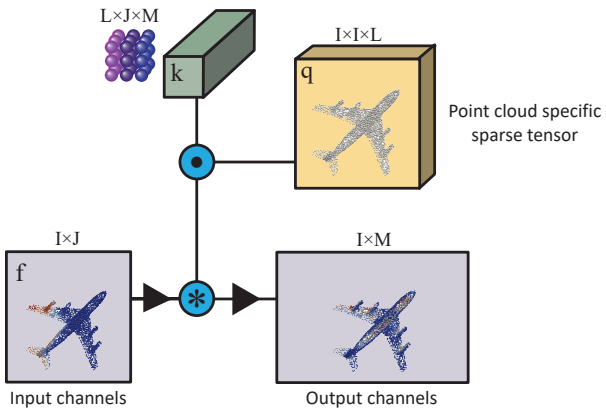


Figure 22: Point cloud convolution operator, computational flow.

6.3.4 Sparse extrinsic convolution

We want to compute the convolution operator $O_X : \mathbb{R}^{I\times J} \to \mathbb{R}^{I\times M}$ restricted to the point cloud X as defined in (9) with the convolution operator O from (15). First, we compute $\mathcal{E}_X[f] * k$

$$\mathcal{E}_X[f] * k\,(x) = c\sum_{ijl} f_{ij}k_{ljm}w_i \int_{\mathbb{R}^3} \Phi(|y - x_i|)\Phi(|x - y - y_l|)\,dy$$

Applying our restriction operator finally gives our point cloud convolution operator:

$$\boxed{O_X[f] = c\sum_{ijl} f_{ij}k_{ljm}w_i q_{ii'l},} \tag{18}$$

where $q = q(X) \in \mathbb{R}^{I\times I\times L}$ is the tensor defined by

$$q_{ii'l} = \int_{\mathbb{R}^3} \Phi(|y - x_i|)\Phi(|x_{i'} - y - y_l|)\,dy. \tag{19}$$

Note that $O_X[f] \in \mathbb{R}^{I\times M}$, as desired. Equation (18) shows that the convolution's weights k_{ljm} are applied to the data f_{ij} using point cloud-dependent weights w, q that can be seen as "translators" of k to the point cloud geometry X. Figure 22 illustrates the computational flow of the convolution operator.

6.3.5 Choice of RBF

Our choice of radial basis function Φ stems from two desired properties: First, we want the extension operator (12) to have approximation properties; second, we want the computation of the convolution of a pair of RBFs in (19) to have an efficient closed-form solution. A natural choice satisfying these requirements is the Gaussian:

$$\Phi_\sigma(r) = \exp\left(-\frac{r^2}{2\sigma^2}\right) \tag{20}$$

To compute the tensor $q \in \mathbb{R}^{I\times I\times L}$ in (19) we make use of the following convolution rule for Gaussians, proved in Section 6.7 for completeness:

Lemma 1. *Let Φ denote the Gaussian as in* (20)*. Then,*

$$\Phi_\alpha(|\cdot - a|) * \Phi_\beta(|\cdot - b|) \propto \Phi_\gamma(|\cdot - a - b|), \tag{21}$$

where $\gamma = \sqrt{\alpha^2 + \beta^2}$.

6.3.6 Up-sampling and pooling

Aside from convolutions, there are two operators that perform spatially and need to be defined for point clouds: up-sampling $U_{X,X^*} : \mathbb{R}^{I\times J} \to \mathbb{R}^{I^*\times J}$, and pooling $P_{X,X^*} : \mathbb{R}^{I\times J} \to \mathbb{R}^{I^*\times J}$, where $X^* \subset \mathbb{R}^3$ is superset of X (*i.e.*, $I^* > I$) in the upsampling case and subset of X (*i.e.*, $I^* < I$) in the pooling case. The upsample operator is defined by

$$U_{X,X^*}[f] = \mathcal{R}_{X^*} \circ \mathcal{E}_X[f]. \tag{22}$$

Pooling does not require the extension/restriction operators and (similarly to [195]) is defined by

$$P_{X,X^*}[f](x_i^*) = \max_{i\in\mathcal{V}_{i^*}} f_{ij}, \tag{23}$$

where $\mathcal{V}_{i^*} \subset \{1, 2, \ldots, I\}$ denotes the set of indices of points in X that are closer in Euclidean distance to x_i^* than any other point in X^*. The point cloud $X^* \subset X$ in the next layer is calculated using farthest point sampling of the input point cloud X.

Lastly, similarly to [121] we implement *deconvolution* layers by an upsampling layer followed by a regular convolution layer.

6.4 Properties

In this section we discuss the properties of the point cloud operators we have defined above.

6.4.1 Invariance and equivariance

Given a function $f \in \mathbb{R}^{I\times J}$ on a point cloud $X \in \mathbb{R}^{I\times 3}$, an equivariant layer $L : \mathbb{R}^{I\times J} \to \mathbb{R}^{I\times M}$ satisfies

$$L(\pi f) = \pi L(f),$$

where $\pi \in \Pi_I \subset \mathbb{R}^{I\times I}$ is an arbitrary permutation matrix. Equivariant layers have been suggested in [196, 202, 264] to learn data in the form of point clouds (or sets in general). The key idea is that equivariant layers can be used to represent a set function $F : 2^{\mathbb{R}^3} \to \mathbb{R}$. Indeed, a set function restricted to sets of fixed size (say, I) can be represented as a symmetric function (*i.e.*, invariant to the order of its arguments). A rich class of symmetric functions can be built by composing equivariant layers and a final symmetric layer.

The equivariance of our point cloud operators O_X stems from the invariance property of the extension operator and equivariance property of the restriction operator. We will next show these properties.

Lemma 2. *The extension operators defined in* (12) *is invariant to permutations, i.e., $\mathcal{E}_{\pi X}[f] = \mathcal{E}_X[f]$, for all permutations $\pi \in \Pi_I$. The restriction operator* (17) *is equivariant to permutations, $\mathcal{R}_{\pi X}[\psi] = \pi\mathcal{R}_X[\psi]$, for all $\pi \in \Pi_I$.*

Proof. The properties follow from the definitions of the operators.

$$\mathcal{E}_{\pi(X)}[f] = \sum_i f_{\pi(i)j}\ell_{\pi(i)} = \sum_i f_{ij}\ell_i = \mathcal{E}_X[f],$$

and

$$\mathcal{R}_X[\psi] = \psi_j(x_{\pi(i)}) = \Pi\mathcal{R}_X[\psi].$$

□

A consequence of this lemma is that any convolution O acting on volumetric functions in $\mathbb{R}^3$ translates to an equivariant operator O_X,

Theorem 1. *Let $O : C(\mathbb{R}^3, \mathbb{R}^J) \to C(\mathbb{R}^3, \mathbb{R}^M)$ be a volumetric function operator. Then $O_X : \mathbb{R}^{I\times J} \to \mathbb{R}^{I\times M}$ defined by* (9) *is equivariant. Namely,*

$$O_{\pi X}[f] = \pi O_X[f].$$

Theorem 1 applies in particular to convolutions (15), and therefore our point cloud convolutions are all equivariant by construction. Note that this model provides "data-dependent" equivariant operator that are more general than those suggest in [202, 264].

6.4.2 Robustness

Overview. Robustness is the key property that allows applying the same convolution kernel to functions over different irregular point clouds. The key idea is to make the extension operator produce approximately the same volumetric function when applied to different samplings of the same underlying shape function. To make things concrete, let $X \in \mathbb{R}^{I\times 3}$, $X^* \in \mathbb{R}^{I^*\times 3}$ be two different point clouds samples of a compact smooth surface $S \subset \mathbb{R}^3$. Let $f \in C(S, \mathbb{R}^J)$ be some function over S and $\mathcal{R}_X[f], \mathcal{R}_{X^*}[f]$ its sampling on the points clouds X, X^*, respectively.

We will show the following:

1. We introduce a continuous extension operator E_S from surface functions to volumetric functions. We show that E_S has several favorable properties.
2. We show that (under mild assumptions) our extension operator $\mathcal{E}_X$, defined in (12)-(13) converges to $\mathcal{E}_S$,
$$\mathcal{E}_X \circ \mathcal{R}_X[f] \approx \mathcal{E}_S[f]. \tag{24}$$
3. We deduce that (under mild assumptions) the properties of $\mathcal{E}_S$ are inherited by $\mathcal{E}_X$ and in particular we have:
$$\mathcal{E}_X \circ \mathcal{R}_X[f] \approx \mathcal{E}_{X^*} \circ \mathcal{R}_{X^*}[f]. \tag{25}$$

Continuous extension operator. We define $\mathcal{E}_S : C(S, \mathbb{R}^J) \to C(\mathbb{R}^3, \mathbb{R}^J)$ which is an extension operator from surface functions to volumetric functions so that $\mathcal{E}_S[f]|_S \approx f$ and $\mathcal{E}_S[f] \to 0$ away from S:

$$\mathcal{E}_S[f](x) = \frac{1}{2\pi\sigma^2} \int_S f(y)\Phi_\sigma(|x-y|)\, da(y), \tag{26}$$

where da is the area element of the surface S.

The operator $\mathcal{E}_S$ enjoys several favorable approximation properties: First,

$$\mathcal{E}_S[f](x) \xrightarrow{\sigma\to 0} \begin{cases} f(x) & x \in S \\ 0 & \text{otherwise} \end{cases}. \tag{27}$$

That is, $\mathcal{E}_S[f]$ approximates f over S and decays to zero away from S. In particular, this implies that the constant one function, $\mathbf{1} : S \to \mathbb{R}$, satisfies

$$\mathcal{E}_S[\mathbf{1}](x) \to \chi_S(x), \tag{28}$$

where $\chi_S(x)$ is the volumetric indicator function of $S \subset \mathbb{R}^3$. Interestingly, $\mathcal{E}_S[\mathbf{1}]$ provides also higher-order geometric information of the surface S,

$$\nabla \mathcal{E}_S[\mathbf{1}]\Big|_S \to -H \cdot n, \tag{29}$$

where $H : S \to \mathbb{R}$ is the mean curvature function of S and $n : S \to S^2$ ($S^2 \subset \mathbb{R}^3$ is the unit sphere) is the normal field to S.

We prove that the approximation quality in (24) improves as the point cloud sample $X \subset S$ densifies S, and the operator $\mathcal{E}_X$ becomes more and more consistent. In that case $\mathcal{E}_X[\mathbf{1}]$ furnishes an approximation to the indicator function of the surface S and its gradient, $\nabla \mathcal{E}_X[\mathbf{1}]$, to the mean curvature vectors of S. This demonstrates that given the simplest, all ones input data $\mathbf{1} \in \mathbb{R}^{I \times 1}$, the network can already reveal the indicator function and the mean curvature vectors of the underlying surface by simple linear operators corresponding to specific choices of the kernel k in (16).

These results are summarized in the following theorem which is proved in Section 6.7.

Theorem 2. *Let $f \in C(S, \mathbb{R}^J)$ be a continuous function defined on a compact smooth surface $S \subset \mathbb{R}^3$. The extension operator* (12)*,*(13) *with*

$$c = \frac{1}{2\pi\sigma^2}, \tag{30}$$

and

$$\omega_i = area(\Omega_i), \tag{31a}$$
$$\Omega_i = \left\{ y \in S \mid d_S(y - x_i) \leq d_S(y - x_{i'}),\ \forall i' \right\}, \tag{31b}$$

the Voronoi cell of $x_i \in S$, where d_S denotes the distance function of points on S, satisfies

$$\mathcal{E}_X \circ \mathcal{R}_X[f](x) \to \mathcal{E}_S[f](x), \tag{32}$$

where $X \subset S$ is a δ-net and $\delta \to 0$. Furthermore, $\mathcal{E}_S$ satisfies the approximation and mean curvature properties as defined in (27)*,* (28)*,* (29)*.*

6.4.3 Revisiting image CNNs

Our model is a generalization of image CNNs. Images can be viewed as point clouds in regular grid configuration, $X = \{x_i\} \subset \mathbb{R}^3$, with image intensities $\mathcal{I}_i$ as functions over this point cloud,

$$\mathcal{E}_X(\mathcal{I}) = \sum_i \mathcal{I}_i \Phi(x - x_i),$$

where Φ is the indicator function over one square grid cell (*i.e.*, pixel). In this case the extension operator reproduces the image as a volumetric function over $\mathbb{R}^2$. Writing the convolution kernel also in the basis Φ with regular grid translations leads to (9) reproducing the standard image discrete convolution.

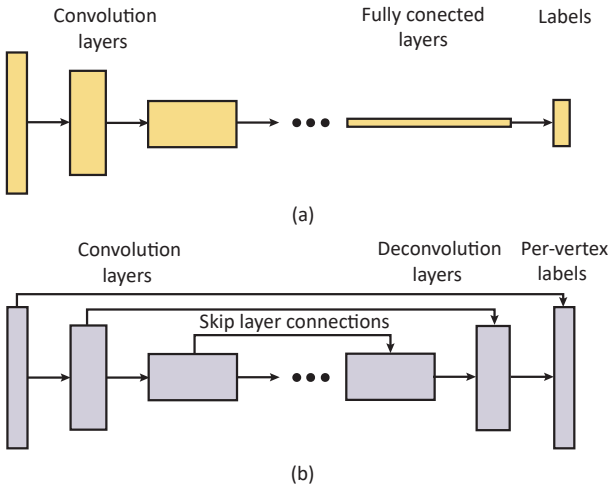


Figure 23: Different architectures used in the section: (a) classification network; and (b) segmentation network.

6.5 Experiments

We have tested our PCNN framework on the problems of point cloud classification, point cloud segmentation, and point cloud normal estimation. We also evaluated the different design choices and network variations.

6.5.1 Point cloud classification

We tested our method on the standard ModelNet40 and ModelNet10 benchmarks [252]. ModelNet40 is composed of 9843 train models and 2468 test models in 40 different classes, such as guitar, cone, laptop etc. ModelNet 10 consists 3991 train and 908 test models from ten different classes. The models are originally given as triangular meshes. The upper part of Table 3 compares our classification results versus state of the art learning algorithms that use only the point clouds (*i.e.*, coordinates of points in $\mathbb{R}^3$) as input: PointNet [196], PointNet++ [195], deep sets [264], ECC[220] and kd-network[129]. For completeness we also provide results of state of the art algorithms taking as input additional data such as meshes and normals. Our method outperforms all point cloud methods and all other non-ensemble methods.

We use the point cloud data of [196, 195] that sampled a point cloud from each model using farthest point sampling. In the training we randomly picked 1024 farthest point sample out of a fixed set of 1200 farthest point sample for each model. As in [129] we also augment the data with random anisotropic scaling in the range $[-0.66, 1.5]$ and uniform translations in the range $[-0.2, 0.2]$. As input to the network we provide the constant one tensor, together with the coordinate functions of the points, namely $(1, x) \in \mathbb{R}^{I\times 4}$. The σ parameter controls the variance of the RBFs (both in the convolution kernels and the extension operator) and is chosen to be $\sigma = I^{-1/2}$. The translations of the convolution are chosen to be regular $3 \times 3 \times 3$ grid with size 2σ.

At test time, similarly to [129] we use voting: we sample ten different samples of size 1024 from 1200 points on each point cloud, apply anisotropic scaling, propagate it through the net and sum the label probability vectors before taking the label with the maximal probability.

We used standard convolution architecture, see Figure 23:

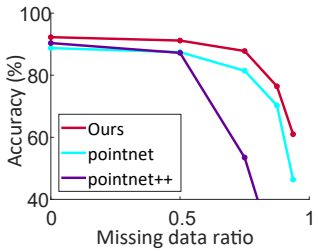
$$\text{conv\_block}(1024, 256, 64) \rightarrow \text{conv\_block}(256, 64, 256)$$
$$\rightarrow \text{conv\_block}(64, 1, 1024) \rightarrow \text{fully\_connected\_block},$$

where conv_block(#points in, #points out ,#channels) consists of a convolution layer, batch normalization, Relu activation and pooling. The fully connected block is a concatenation of a two fully connected layers with dropout after each one.

algorithm	# points	10 models	40 models
Point cloud methods			
pointnet [196]	1024	-	89.2
pointnet++ [195]	1024	-	90.7
deep sets [264]	1000	-	87.1
ECC [220]	1000	90.8	87.4
kd-network [129]	1024	93.3	90.6
kd-network [129]	32k	94.0	91.8
ours	1024	**94.9**	**92.3**
Additional input features			
FusionNet (uses mesh structure) [106]	-	93.1	90.8
VRN single(uses mesh structure) [37]	-	-	92.0
OctNet [204]	-	90.9	86.5
VRN ensemble(uses mesh structure) [37]	-	97.1	95.5
MVCNN (uses mesh structure)[194]	-	-	92.0
MVCNN-MultiRes(uses mesh structure)[194]	-	-	93.8
pointnet++ (uses normals) [195]	5K	-	91.9
OCNN (uses normals) [244]	-	-	90.6

Table 3: Shape classification results on the ModelNet40 and ModelNet10 datasets.

Robustness to sampling. The inset compares our method with [196, 195] when feeding a trained 1024 point model on sparser test point clouds of size $k = 1024, 512, 256, 128, 64$. The favorable robustness of our method to sub-sampling can be possibly explained by the fact that our extension operator possess approximation power, even with sparse samples, *e.g.*, for smooth shapes, see Figure 21.



Method variants. We evaluate the performance of our algorithm subject to the variation in: the number of points I, the kernel translations $\{y_l\}$, the input tensor f, different bases ℓ_i in (12), the choice of σ, and number of learnable parameters. Points were randomly sampled by the same ratio as in the above (*e.g.* 512 out of 600).

Table 4 presents the results. Using the constant one as input $f = \mathbf{1} \in \mathbb{R}^{I\times 1}$ provides almost comparable results to using the full coordinates of the points $f = (\mathbf{1}, x) \in \mathbb{R}^{I\times 4}$. This observation is partially supported by the theoretical analysis shown in section 6.4 which states that our extension operator applied to the constant one tensor already provides good approximation to the underlying surface and its normal as

Method variations on ModelNet40		
Variation	# points	accuracy
Less points	256	90.8
Less points	512	91.2
Less points	870	92.2
More points	1800	92.3
Interpolation	1024	92.0
Spherical kernel	1024	91.5
Learned translations	1024	91.3
Indicator input	1024	91.2
xyz only input	1024	91.5
Less parameters	1024	92.2
Small sigma	1024	85.5

Table 4: Classification with variations to the PCNN model.

well as curvature. Using interpolation basis $\{\ell_i\}$ in the extension operator (12), although heavier computationally, does not provide better results. Applying too small σ provides worse classification result. This can be explained by the observation that small σ results in separated Gaussians centered at the points which deteriorates the approximation (X and σ should be related). Interestingly, using a relatively small network size of 1.4M parameters provides comparable classification result.

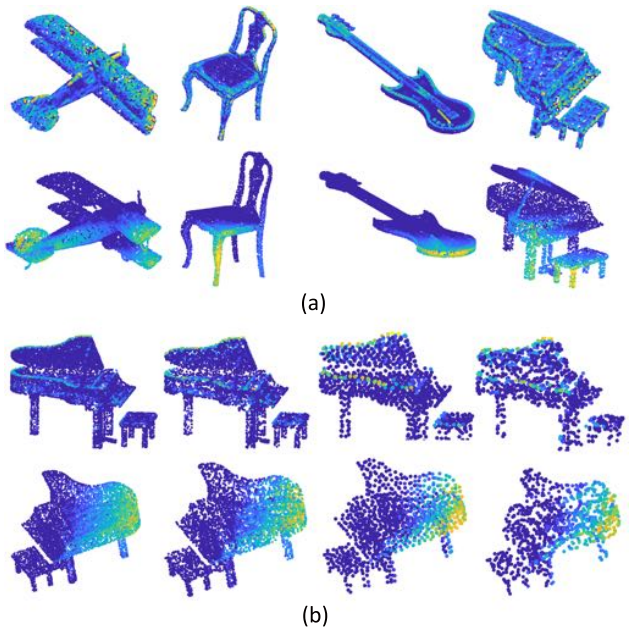


Figure 24: Our point cloud convolution is translation invariant and robust to sample size and density: (a) shows feature activations of two kernels (rows) learned by our network's first convolution layer on different shapes (columns). The features seems consistent across the different models; (b) shows another pair of kernels (rows) on a single model with varying sampling density (from left to right): $10K$ points, $5K$ points (random sampling), 1K points (farthest point sampling) and $1K$ (random sampling). Note that the convolution captures the same geometric properties on all models regardless of the sampling.

Feature visualizations. Figure 24 visualizes the features learned in the first layer of PCNN on a few shapes from the ModelNet40 dataset. As in the case of images, the features learned on the first layer are mostly edge detectors and directional derivatives. Note that the features are consistent through different sampling and shapes. Figure 25 shows 9 different features learned in the third layer of PCNN. In this layer the features capture more semantically meaningful parts.

Figure 25: High level features learned by PCNN's third convolution layer and visualized on the input point cloud. As expected, the features are less geometrical than the first layer's features (see Figure 24) and seem to capture more semantically meaningful shape parts.

6.5.2 Point cloud segmentation

Our method can also be used for part segmentation: given a point cloud that represents a shape the task is to label each point with a correct part label. We evaluate PCNN performance on ShapeNet part dataset [260]. ShapeNet contains 16,881 shapes from 16 different categories, and total of 50 part labels.

Table 5 compares per-category and mean IoU(%) scores of PCNN with state of the art point cloud methods: PointNet [196], kd-network [129], and 3DCNN (results taken from [196]). Our method outperforms all of these methods. For completeness we also provide results of other methods that use additional shape features or mesh normals as input. Figure 26 depicts several of our segmentation results.

For this task we used standard convolution segmentation architecture, see Figure 23:

$$\begin{aligned}&\text{conv\_block}(2048, 512, 64) \rightarrow \text{conv\_block}(512, 128, 128)\\ &\rightarrow \text{conv\_block}(128, 16, 256) \rightarrow \text{deconv\_block}(16, 128, 512)\\ &\rightarrow \text{deconv\_block}(128, 512, 256) \rightarrow \text{deconv\_block}(512, 2048, 256)\\ &\rightarrow \text{deconv\_block}(2048, 2048, 256) \rightarrow \text{conv\_block}(2048, 2048, 50),\end{aligned}$$

where deconv_block(#points in,#points out,#features) consists of an upsampling layer followed by a convolution block. In order to provide the last layers with raw features we also add skip-layers connections, see Figure 23(b). This is a common practice in such architectures where fine details are needed at the output layer (*e.g.*, [51]).

We use the data from [196] (2048 uniformly sampled points on each model). As done in [196] we use a single network to predict segmentations for each of the object classes and concatenate a hot-one encoding of the object's label to the bottleneck feature layer. At test time, we use only the part labels that correspond to the input shape (as in [196, 129]).

	input	mean	aero	bag	cap	car	chair	ear-p	guitar	knife	lamp	laptop	motor	mug	pistol	rocket	skate	table
Point clouds																		
Ours	2K pnts	**85.1**	82.4	**80.1**	**85.5**	**79.5**	**90.8**	73.2	91.3	86.0	**85.0**	**95.7**	**73.2**	**94.8**	**83.3**	51.0	**75.0**	**81.8**
PointNet	2K pnts	83.7	**83.4**	78.7	82.5	74.9	89.6	73	**91.5**	85.9	80.8	95.3	65.2	93	81.2	57.9	72.8	80.6
kd-network	4K pnts	82.3	80.1	74.6	74.3	70.3	88.6	**73.5**	90.2	**87.2**	81	94.9	57.4	86.7	78.1	51.8	69.9	80.3
3DCNN		79.4	75.1	72.8	73.3	70	87.2	63.5	88.4	79.6	74.4	93.9	58.7	91.8	76.4	51.2	65.3	77.1
Additional input																		
SyncSpecCNN	sf	84.7																
Yi	sf	81.4	81	78.4	77.7	75.7	87.6	61.9	92.0	85.4	82.5	95.7	70.6	91.9	85.9	53.1	69.8	75.3
PointNet++	pnts, nors	85.1	82.4	79.0	87.7	77.3	90.8	71.8	91.0	85.9	83.7	95.3	71.6	94.1	81.3	58.7	76.4	82.6
OCNN (+CRF)	nors	85.9	85.5	87.1	84.7	77.0	91.1	85.1	91.9	87.4	83.3	95.4	56.9	96.2	81.6	53.5	74.1	84.4

Table 5: ShapeNet segmentation results by point cloud methods (top) and methods using additional input data (sf - shape features; nors - normals). The methods compared to are: PointNet [196]; kd-network [129]; 3DCNN [196]; SyncSpecCNN [261]; Yi [260]; PointNet++ [195]; OCNN (+CRF refinement) [244].

Figure 26: Results of PCNN on the part segmentation benchmark from [260]

6.5.3 Normal estimation

Estimating normals of a point cloud is a central sub-problem of the 3D reconstruction problem. We cast this problem as supervised regression problem and employ segmentation network with the following changes: the output layer is composed of 3 channels instead of 50 which are then normalized and fed into cosine-loss with the ground truth normals.

We have trained and tested our network on the standard train/test splits of the ModelNet40 dataset (we used the data generator code by [195]). Table 6 compares the mean cosine loss (distance) of PCNN and the normal estimation of [196] and [195]. Figure 27 depicts normal estimation examples from this challenge.

Data	algorithm	# points	error
ModelNet40	PointNet	1024	0.47
	PointNet++	1024	0.29
	ours	**1024**	**0.19**

Table 6: Normal estimation in ModelNet40.

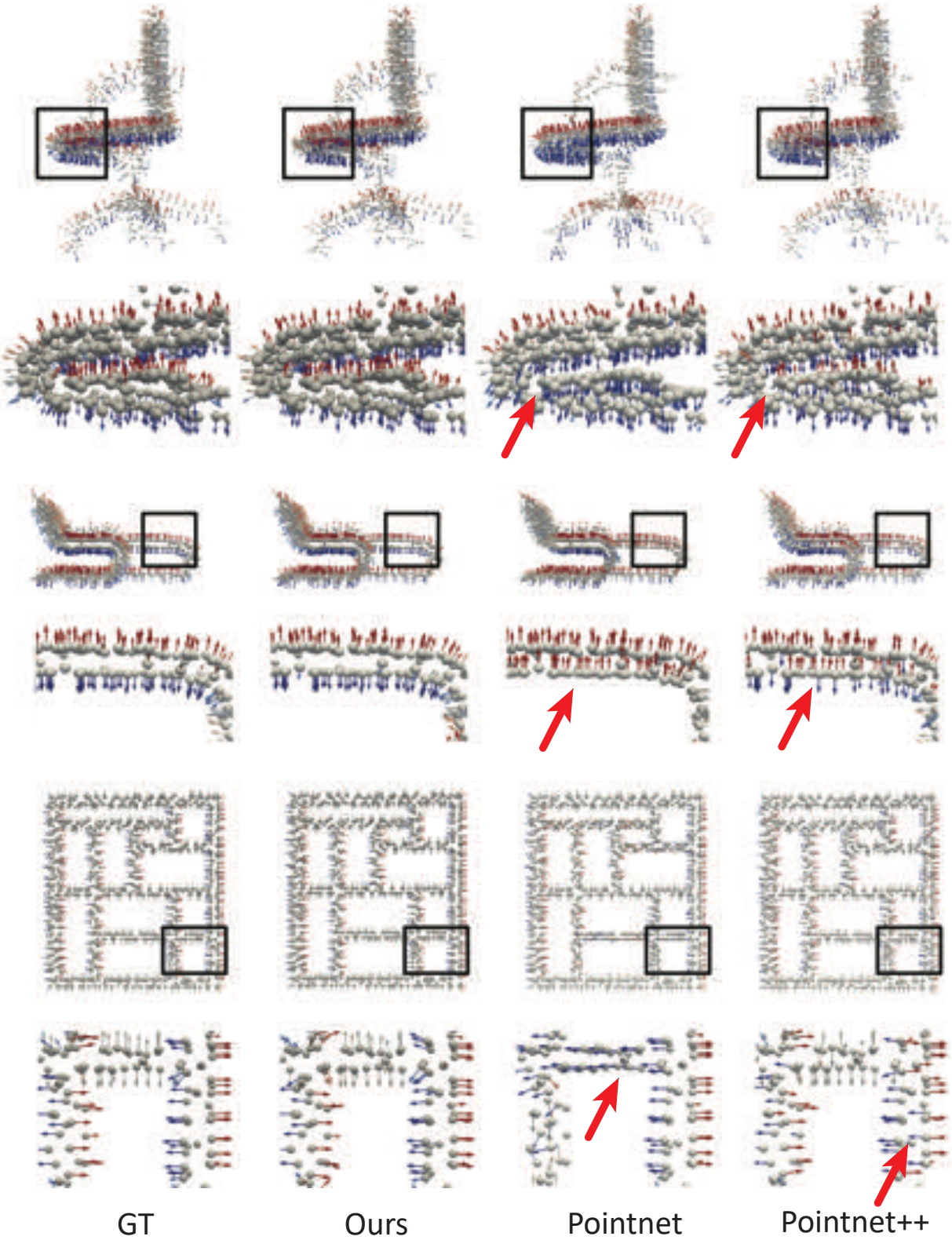


Figure 27: Normal estimation in ModelNet40. We show normal estimation of four models (rows) with blow-ups. Normals are colored by one of their coordinates for better visualization. Note that competing methods sometimes fail to recognize the outward normal direction (examples indicated by red arrows).

6.5.4 Training details, timings and network size

We implemented our method using the TensorFlow library [1] in Python. We used the Adam optimization method with learning rate 0.001 and decay rate 0.7. The models were trained on Nvidia p100 GPUs. Table 7 summarizes running times and network sizes. Our smaller classification network achieves state of the art result (see Table 4, previous to last row) and has only 1.4M parameters with a total model size of 17 MB.

	#Param.	Size (mb)	Converge (epochs)	Training (min)	Forward (msec)
Classification					
Large	8.1M	98	250	6	80
Small	1.4M	17	250	5	70
Segmentation					
	5.4M	62	85	37	200

Table 7: Timing and network size. Training time is measured in minuets per epoch.

6.6 Conclusions

This section describes PCNN: a methodology for defining convolution of functions over point clouds that is efficient, invariant to point cloud order, robust to point sampling and density, and posses translation invariance. The key idea is to translate volumetric convolution to arbitrary point clouds using extension and restriction operators.

Testing PCNN on standard point cloud benchmarks show state of the art results using compact networks. The main limitation of our framework compared to image CNN is the extra computational burden due to the computation of farthest point samples in the network and the need to compute the "translating" tensor $q \in \mathbb{R}^{I \times I \times L}$ which is a function of the point cloud X. Still, we believe that a sparse efficient implementation can alleviate this limitation and mark it as a future work. Other venues for future work is to learn kernel translations per channel similarly to [60], and apply the method to data of higher dimension than $d = 3$ which seems to be readily possible. Lastly, we would like to test this framework on different problems and architectures.

6.7 Proofs

6.7.1 Multiplication law for Gaussians

Proposition 1. *Let*

$$\Phi_{\mu,\sigma}(x) = \exp(-\frac{\|x-\mu\|^2}{2\sigma^2})$$

and

$$B(\sigma) = \frac{1}{(2\pi\sigma^2)^{\frac{3}{2}}}$$

then

$$\Phi_{\mu_1,\sigma_1} * \Phi_{\mu_2,\sigma_2} = C(\sigma_1,\sigma_2) \cdot \Phi_{\mu,\sigma}$$

where $\mu = \mu_1 + \mu_2$, $\sigma = \sqrt{\sigma_1^2 + \sigma_2^2}$ *and* $C(\sigma_1,\sigma_2) = \frac{B(\sigma_1)B(\sigma_2)}{B(\sigma)}$

Proof. It is well known [247] that the convolution of two normal distributions is again a normal distribution:

$$B(\sigma_1)\Phi_{\mu_1,\sigma_1} * B(\sigma_2)\Phi_{\mu_2,\sigma_2} = B(\sqrt{\sigma_1^2+\sigma_2^2}) \cdot \Phi_{\mu,\sigma}$$

The result above follows from the linearity of the convolution. □

6.7.2 Theoretical properties of the extension operator

Proof of theorem 2. Let us first show (32). Denote $g_x(y) = f(y)\Phi_\sigma(|x-y|)$. By Lemma 4 for arbitrary $\epsilon > 0$ there exists $\delta > 0$ so that $d_S(y,x) < \delta$ implies $|g_x(y) - g_x(x)| < \epsilon$ for all $x \in \mathbb{R}^3$. Taking X to be δ-net of S we get that

$$\left|\mathcal{E}_S[f](x) - \mathcal{E}_X \circ \mathcal{R}_X[f](x)\right| \leq \epsilon \frac{\sum_i area(\Omega_i)}{2\pi\sigma^2} \leq \epsilon \frac{area(S)}{2\pi\sigma^2}.$$

To show (27) first let $x \notin S$. Then as $\sigma \to 0$ we have $\max_{y\in S} \Phi_\sigma(|y-x|) \to 0$ and therefore $\mathcal{E}_S[f](x) \to 0$. Next consider $x \in S$. It is enough to show that

$$\frac{1}{2\pi\sigma^2}\int_S \Phi_\sigma(|x-y|)da(y) \xrightarrow{\sigma\to 0} 1.$$

Indeed, let $\epsilon > 0$. Since f is uniformly continuous, take $\delta > 0$ sufficiently small so that $|f(x) - f(x')| < \epsilon$ if $d_S(x,x') < \delta$. Take $\sigma > 0$ sufficiently small so that

$$\frac{1}{2\pi\sigma^2}\int_{S\setminus B(x,\delta)} \Phi_\sigma(|x-y|)da(y) \leq \epsilon,$$

where $B(x,\delta) = \{y \mid |y-x| < \delta\}$ and

$$\left|\frac{1}{2\pi\sigma^2}\int_S \Phi_\sigma(|x-y|)da(y) - 1\right| \leq \epsilon.$$

Hence

$$\left|\frac{1}{2\pi\sigma^2}\int_{S\cap B(x,\delta)} \Phi_\sigma(|x-y|)da(y) - 1\right| \leq 2\epsilon.$$

Therefore,

$$\left|\frac{1}{2\pi\sigma^2}\int_S f(y)\Phi_\sigma(|x-y|)da(y) - f(x)\right| \leq 3\epsilon(1+|f|_\infty).$$

Lastly, to show (29) note that

$$\nabla_x \mathcal{E}_S[\mathbf{1}](x) = -\frac{1}{2\pi\sigma^4}\int_S (x-y)\Phi(|x-y|)da(y).$$

Using an argument from [21] (see Section 4.2) where we take $\sigma^2 = 2t$ in their notation and get convergence to $-\frac{1}{2}\Delta_S x$, where Δ_S is the Laplace-Beltrami operator on surfaces S. To finish the proof remember that [44]

$$-\frac{1}{2}\Delta_S x = -H \cdot n.$$

□

Lemma 3. *Let $S \subset \mathbb{R}^3$ be a compact smooth surface. Then,*

$$\lim_{\sigma \to 0} \frac{1}{2\pi\sigma^2} \int_S \Phi_\sigma(|x-y|) da(y) = 1 \tag{33}$$

Proof. Denote $T_x S$ the tangent plane to S centered at $x \in S$. Let $y = y(u) : T_x S \to S$ be the local parameterization to S over $T_x S$, where u is the local coordinate at $T_x S$. Since S is smooth and compact we have that $\forall u \in \Upsilon_\delta = T_x S \cap B(x, \delta)$,

$$|y(u) - u| = \mathcal{O}(\delta^2) \tag{34}$$
$$||dy(u)| - 1| = \mathcal{O}(\delta), \tag{35}$$

where $|dy(u)|$ is the pulled-back area element of S [67]. We break the error to $\left|\frac{1}{2\pi\sigma^2} \int_S \Phi_\sigma(|x-y|) da(y) - 1\right|$

$$\leq \underbrace{\frac{1}{2\pi\sigma^2} \int_{S \setminus y(\Upsilon_\delta)} \Phi_\sigma da}_{(i)} + \underbrace{\frac{1}{2\pi\sigma^2} \left| \int_{y(\Upsilon_\delta)} \Phi_\sigma da - \int_{\Upsilon_\delta} \Phi_\sigma du \right|}_{(ii)}$$
$$\underbrace{\left| \frac{1}{2\pi\sigma^2} \int_{T_x S} \Phi_\sigma du - 1 \right|}_{(iii)} + \underbrace{\frac{1}{2\pi\sigma^2} \int_{T_x S \setminus \Upsilon_\delta} \Phi_\sigma du}_{(iv)}.$$

First, we note that $(iii) = 0$. Now, take $\delta = \sigma^{1-\tau}$, for some fixed $0 < \tau < 1$, where $\sigma > 0$. Then, $(i) = \mathcal{O}(\sigma)$, $(iv) = \mathcal{O}(\sigma)$. Lastly (ii)

$$\begin{aligned}
&\leq \frac{1}{2\pi\sigma^2} \int_{\Upsilon_\delta} \Big| \Phi_\sigma(|y(u)|) - \Phi_\sigma(|u|) \Big| |dy(u)| du \\
&+ \frac{1}{2\pi\sigma^2} \int_{\Upsilon_\delta} \Phi_\sigma(|u|) \Big| \, |dy(u)| - 1 \Big| du \\
&\leq \frac{1}{2\pi\sigma^2} \frac{\max_{u \in \Upsilon_\delta} \Big| |y(u)| - |u| \Big|}{\sigma e^{1/2}} (1 + \mathcal{O}(\delta)) \mathcal{O}(\delta^2) + \mathcal{O}(\delta) \\
&= \mathcal{O}(\sigma^{1-4\tau}),
\end{aligned}$$

where we used Lemma 5 in the last inequality. Taking any $\tau < 1/4$ proves the result. □

Lemma 4. *Let $f \in C(S, \mathbb{R})$, with $S \subset \mathbb{R}^3$ a compact surface. The family of functions $\{g_x\}_{x \in \mathbb{R}^3}$ defined by $g_x(y) = f(y)\Phi_\sigma(|x-y|)$, $y \in S$ is uniformly equicontinuous.*

Proof. $|g_x(y) - g_x(y')|$

$$\begin{aligned}
&\leq \big| f(y) - f(y') \big| \, \Phi_\sigma(|x-y|) + \\
&\big| f(y') \big| \, \big| \Phi_\sigma(|x-y|) - \Phi_\sigma(|x-y'|) \big| \\
&\leq \big| f(y) - f(y') \big| + |f|_\infty \frac{|y-y'|}{\sigma e^{1/2}}
\end{aligned}$$

where in the last inequality we used $||x-y| - |x-y'|| \leq |y-y'|$ and Lemma 5. Since $f, |\cdot|$ are both uniformly continuous over S (as continuous functions over a compact surface), f is bounded, *i.e.*, $|f|_\infty < \infty$, equicontinuity of $\{g_x\}$ is proved. □

Lemma 5. *The gaussian satisfies $|\Phi_\sigma(r') - \Phi_\sigma(r)| \leq \frac{(r'-r)}{\sigma e^{1/2}}$ for $0 \leq r < r'$.*

Proof.

$$\left|\Phi_\sigma(r') - \Phi_\sigma(r)\right| \leq \int_r^{r'} \frac{t}{\sigma^2} e^{-\frac{t^2}{2\sigma^2}} dt \leq \frac{(r'-r)}{\sigma e^{1/2}},$$

where in the last inequality we used the fact that $te^{-\frac{t^2}{2\sigma^2}} \leq \sigma e^{-1/2}$. □

Lastly, to justify (14) let us use Theorem 2 and consider $f(x) \equiv 1$,

$$1 \approx \mathcal{E}_S[f](x) \approx c \sum_i \omega_i \Phi(|x - x_i|).$$

Plugging $x = x_{i'}$ we get

$$1 \approx c \sum_i \omega_i \Phi(|x_{i'} - x_i|) = c\tilde{\omega}_{i'} \sum_i \Phi(|x_{i'} - x_i|),$$

where $\tilde{\omega}_{i'}$ is an average of values of ω_i (note that $\Phi(|x_{i'} - x_i|)$ are fast decaying weights away from $x_{i'}$). Hence,

$$c\tilde{\omega}_{i'} \approx \frac{1}{\sum_i \Phi(|x_{i'} - x_i|)}.$$

7 Invariant graph networks

This section is based on [158].

7.1 Introduction

We consider the problem of graph learning, namely finding a functional relation between input graphs (more generally, hyper-graphs) G^ℓ and corresponding targets T^ℓ, *e.g.*, labels. As graphs are common data representations, this task received quite a bit of recent attention in the machine learning community [42, 108, 172, 262].

More specifically, a (hyper-)graph data point $G = (V, A)$ consists of a set of n nodes V, and values A attached to its *hyper-edges*[2]. These values are encoded in a tensor A. The order of the tensor A, or equivalently, the number of indices used to represent its elements, indicates the type of data it represents, as follows: First order tensor represents *node-values* where A_i is the value of the i-th node; Second order tensor represents *edge-values*, where A_{ij} is the value attached to the (i, j) edge; in general, k-th order tensor encodes *hyper-edge-values*, where $A_{i_1,\ldots,i_k}$ represents the value of the hyper-edge represented by $(i_1, \ldots, i_k)$. For example, it is customary to represent a graph using a binary adjacency matrix A, where A_{ij} equals one if vertex i is connected to vertex j and zero otherwise. We denote the set of order-k tensors by $\mathbb{R}^{n^k}$.

The task at hand is constructing a functional relation $f(A^\ell) \approx T^\ell$, where f is a neural network. If $T^\ell = t^\ell$ is a single output response then it is natural to ask that f is *order invariant*, namely it should produce the same output regardless of the node numbering used to encode A. For example, if we represent a graph using an adjacency matrix $A = A \in \mathbb{R}^{n\times n}$, then for an arbitrary permutation matrix P and an arbitrary adjacency matrix A, the function f is order invariant if it satisfies $f(P^T AP) = f(A)$. If the targets T^ℓ specify output response in a form of a tensor, $T^\ell = T^\ell$, then it is natural to ask that f is *order equivariant*, that is, f commutes with the renumbering of nodes operator acting on tensors. Using the above adjacency matrix example, for every adjacency matrix A and every permutation matrix P, the function f is equivariant if it satisfies $f(P^T AP) = P^T f(A)P$. To define invariance and equivariance for functions acting on general tensors $A \in \mathbb{R}^{n^k}$ we use the *reordering operator*: $P \star A$ is defined to be the tensor that results from renumbering the nodes V according to the permutation defined by P. Invariance now reads as $f(P \star A) = f(A)$; while equivariance means $f(P \star A) = P \star f(A)$. Note that the latter equivariance definition also holds for functions between different order tensors, $f : \mathbb{R}^{n^k} \to \mathbb{R}^{n^l}$.

Following the standard paradigm of neural-networks where a network f is defined by alternating compositions of linear layers and non-linear activations, we set as a goal to characterize all *linear* invariant and equivariant layers. The case of node-value input $A = a \in \mathbb{R}^n$ was treated in the pioneering works of [264, 197]. These works characterize all linear permutation invariant and equivariant operators acting on node-value (*i.e.*, first order) tensors, $\mathbb{R}^n$. In particular it it shown that the linear space of invariant linear operators $L : \mathbb{R}^n \to \mathbb{R}$ is of dimension one, containing essentially only the sum operator, $L(a) = \alpha \mathbf{1}^T a$. The space of equivariant linear operators $L : \mathbb{R}^n \to \mathbb{R}^n$ is of dimension two, $L(a) = \left[\alpha I + \beta(\mathbf{1}\mathbf{1}^T - I)\right] a$.

The general equivariant tensor case was partially treated in [132] where the authors make the observation that the set of standard tensor operators: product, element-wise product, summation, and contraction are all equivariant, and due to linearity the same applies to their linear combinations. However, these do not exhaust nor provide a full and complete basis for *all* possible tensor equivariant linear layers. In this section we provide a full characterization of permutation invariant and equivariant linear layers for general tensor input and output data. We show that the space of invariant linear layers $L : \mathbb{R}^{n^k} \to \mathbb{R}$ is of dimension $\mathrm{b}(k)$, where $\mathrm{b}(k)$

[2] A hyper-edge is an ordered subset of the nodes, V

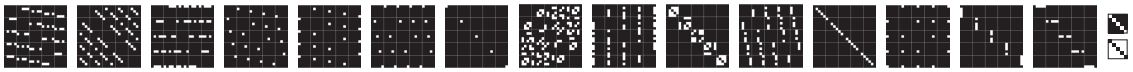
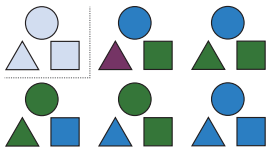


Figure 28: The full basis for equivariant linear layers for edge-value data $A \in \mathbb{R}^{n\times n}$, for $n = 5$. The purely linear 15 basis elements, B^u, are represented by matrices $n^2 \times n^2$, and the 2 bias basis elements (right), C^λ, by matrices $n \times n$, see (44).

is the k-th *Bell number*. The k-th Bell number is the number of possible partitions of a set of size k; see inset for the case $k = 3$. Furthermore, the space of equivariant linear layers $L : \mathbb{R}^{n^k} \to \mathbb{R}^{n^l}$ is of dimension $\mathrm{b}(k+l)$. Remarkably, this dimension is independent of the size n of the node set V. This allows applying the same network on graphs of different sizes. For both types of layers we provide a general formula for an orthogonal basis that can be readily used to build linear invariant or equivariant layers with maximal expressive power.



Going back to the example of a graph represented by an adjacency matrix $A \in \mathbb{R}^{n\times n}$ we have $k = 2$ and the linear invariant layers $L : \mathbb{R}^{n\times n} \to \mathbb{R}$ have dimension $\mathrm{b}(2) = 2$, while linear equivariant layers $L : \mathbb{R}^{n\times n} \to \mathbb{R}^{n\times n}$ have dimension $\mathrm{b}(4) = 15$. Figure 28 shows visualization of the basis to the linear equivariant layers acting on edge-value data such as adjacency matrices.

In [104] the authors provide an impressive generalization of the case of node-value data to several node sets, $V_1, V_2, \ldots, V_m$ of sizes $n_1, n_2, \ldots, n_m$. Their goal is to learn interactions across sets. That is, an input data point is a tensor $A \in \mathbb{R}^{n_1\times n_2\times\cdots\times n_m}$ that assigns a value to each element in the cartesian product $V_1 \times V_2 \times \cdots \times V_m$. Renumbering the nodes in each node set using permutation matrices $P_1, \ldots, P_m$ (resp.) results in a new tensor we denote by $P_{1:m} \star A$. Order invariance means $f(P_{1:m} \star A) = f(A)$ and order equivariance is $f(P_{1:m}\star A) = P_{1:m}\star f(A)$. [104] introduce bases for linear invariant and equivariant layers. Although the layers in [104] satisfy the order invariance and equivariance, they do not exhaust all possible such layers in case some node sets coincide. For example, if $V_1 = V_2$ they have 4 independent learnable parameters where our model has the maximal number of 15 parameters.

Our analysis allows generalizing the multi-node set case to arbitrary tensor data over $V_1 \times V_2 \times \cdots \times V_m$. Namely, for data points in the form of a tensor $A \in \mathbb{R}^{n_1^{k_1}\times n_2^{k_2}\times\cdots\times n_m^{k_m}}$. The tensor A attaches a value to every element of the Cartesian product $V_1^{k_1} \times \cdots \times V_2^{k_2}$, that is, k_1-tuple from V_1, k_2-tuple from V_2 and so forth. We show that the linear space of invariant linear layers $L : \mathbb{R}^{n_1^{k_1}\times n_2^{k_2}\times\cdots\times n_m^{k_m}} \to \mathbb{R}$ is of dimension $\prod_{i=1}^m \mathrm{b}(k_i)$, while the equivariant linear layers $L : \mathbb{R}^{n_1^{k_1}\times n_2^{k_2}\times\cdots\times n_m^{k_m}} \to \mathbb{R}^{n_1^{l_1}\times n_2^{l_2}\times\cdots\times n_m^{l_m}}$ has dimension $\prod_{i=1}^m \mathrm{b}(k_i + l_i)$. We also provide orthogonal bases for these spaces. Note that, for clarity, the discussion above disregards biases and features; we detail these later on.

In Section 7.8 we show that our model is capable of approximating any message-passing neural network as defined in [89] which encapsulate several popular graph learning models. One immediate corollary is that the universal approximation power of our model is not lower than message passing neural nets.

In the experimental part of the section we concentrated on possibly the most popular instantiation of graph learning, namely that of a single node set and edge-value data, *e.g.*, with adjacency matrices. We created simple networks by composing our invariant or equivariant linear layers in standard ways and tested the networks in learning invariant and equivariant graph functions: (i) We compared identical networks with our basis and the basis of [104] and showed we can learn graph functions like trace, diagonal, and maximal singular vector. The basis in [104], tailored to the multi-set setting, cannot learn these functions demonstrating it is not maximal in the graph-learning (*i.e.*, multi-set with repetitions) scenario. We also demonstrate our representation allows extrapolation: learning on one size graphs and testing on another size; (ii) We also tested our networks on a collection of graph learning datasets, achieving results that are comparable to the

state-of-the-art in 3 social network datasets.

7.2 Previous work

Our work builds on two main sub-fields of deep learning: group invariant or equivariant networks, and deep learning on graphs. Here we briefly review the relevant works.

Invariance and equivariance in deep learning. In many learning tasks the functions that we want to learn are invariant or equivariant to certain symmetries of the input object description. Maybe the first example is the celebrated *translation invariance* of Convolutional Neural Networks (CNNs) [141, 136]; in this case, the image label is invariant to a translation of the input image. In recent years this idea was generalized to other types of symmetries such as rotational symmetries [54, 52, 246, 53]. [54] introduced Group Equivariant Neural Networks that use a generalization of the convolution operator to groups of rotations and reflections; [246, 53] also considered rotational symmetries but in the case of 3D shapes and spherical functions. [203] showed that any equivariant layer is equivalent to a certain parameter sharing scheme. If we adopt this point of view, our work reveals the structure of the parameter sharing in the case of graphs and hypergraphs. In another work, [131] show that a neural network layer is equivariant to the action of some compact group iff it implements a generalized form of the convolution operator. [259] suggested certain group invariant/equivariant models and proved their universality. To the best of our knowledge these models were not implemented.

Learning of graphs. Learning of graphs is of huge interest in machine learning and we restrict our attention to recent advancements in deep learning on graphs. [93, 213] introduced Graph Neural Networks (GNN): GNNs hold a state (a real valued vector) for each node in the graph, and propagate these states according to the graph structure and learned parametric functions. This idea was further developed in [146] that use gated recurrent units. Following the success of CNNs, numerous works suggested ways to define convolution operator on graphs. One promising approach is to define convolution by imitating its spectral properties using the Laplacian operator to define generalized Fourier basis on graphs [42]. Multiple follow-up works [108, 64, 128, 144] suggest more efficient and spatially localized filters. The main drawback of spectral approaches is that the generalized Fourier basis is graph-dependent and applying the same network to different graphs can be challenging. Another popular way to generalize the convolution operator to graphs is learning stationary functions that operate on neighbors of each node and update its current state [13, 69, 102, 181, 236, 172, 221]. This idea generalizes the *locality* and *weight sharing* properties of the standard convolution operators on regular grids. As shown in the important work of [89], most of the the above mentioned methods (including the spectral methods) can be seen as instances of the general class of *Message Passing Neural Networks*.

7.3 Linear invariant and equivariant layers

In this section we characterize the collection of linear invariant and equivariant layers. We start with the case of a single node set V of size n and edge-value data, that is order 2 tensors $A = A \in \mathbb{R}^{n\times n}$. As a typical example imagine, as above, an adjacency matrix of a graph. We set a bit of notation. Given a matrix $X \in \mathbb{R}^{a\times b}$ we denote $\mathrm{vec}(X) \in \mathbb{R}^{ab\times 1}$ its column stack, and by brackets the inverse action of reshaping to a square matrix, namely $at\mathrm{vec}(X) = X$. Let p denote an arbitrary permutation and P its corresponding permutation matrix.

Let $L \in \mathbb{R}^{1\times n^2}$ denote the matrix representing a general linear operator $L : \mathbb{R}^{n\times n} \to \mathbb{R}$ in the standard basis, then L is order invariant iff $L\mathrm{vec}(P^T AP) = L\mathrm{vec}(A)$. Using the property of the Kronecker product that $\mathrm{vec}(XAY) = Y^T \otimes X\mathrm{vec}(A)$, we get the equivalent equality $LP^T \otimes P^T\mathrm{vec}(A) = L\mathrm{vec}(A)$. Since

the latter equality should hold for every A we get (after transposing both sides of the equation) that order invariant L is equivalent to the equation

$$P \otimes P \operatorname{vec}(L) = \operatorname{vec}(L) \tag{36}$$

for every permutation matrix P. Note that we used $L^T = \operatorname{vec}(L)$.

For equivariant layers we consider a general linear operator $L : \mathbb{R}^{n \times n} \to \mathbb{R}^{n \times n}$ and its corresponding matrix $L \in \mathbb{R}^{n^2 \times n^2}$. Equivariance of L is now equivalent to $atL\operatorname{vec}(P^T A P) = P^T atL\operatorname{vec}(A)P$. Using the above property of the Kronecker product again we get $LP^T \otimes P^T \operatorname{vec}(A) = P^T \otimes P^T L\operatorname{vec}(A)$. Noting that $P^T \otimes P^T$ is an $n^2 \times n^2$ permutation matrix and its inverse is $P \otimes P$ we get to the equivalent equality $P \otimes PLP^T \otimes P^T \operatorname{vec}(A) = L\operatorname{vec}(A)$. As before, since this holds for every A and using the properties of the Kronecker product we get that L is order equivariant iff for all permutation matrices P

$$P \otimes P \otimes P \otimes P \operatorname{vec}(L) = \operatorname{vec}(L). \tag{37}$$

From equations 36 and 37 we see that finding invariant and equivariant linear layers for the order-2 tensor data over one node set requires finding fixed points of the permutation matrix group represented by Kronecker powers $P \otimes P \otimes \cdots \otimes P$ of permutation matrices P. As we show next, this is also the general case for order-k tensor data $A \in \mathbb{R}^{n^k}$ over one node set, V. That is,

$$\text{invariant } L: \qquad P^{\otimes k} \operatorname{vec}(L) = \operatorname{vec}(L) \tag{38}$$

$$\text{equivariant } L: \qquad P^{\otimes 2k} \operatorname{vec}(L) = \operatorname{vec}(L) \tag{39}$$

for every permutation matrix P, where $P^{\otimes k} = \overbrace{P \otimes \cdots \otimes P}^{k}$. In (38), $L \in \mathbb{R}^{1 \times n^k}$ is the matrix of an invariant operator; and in (39), $L \in \mathbb{R}^{n^k \times n^k}$ is the matrix of an equivariant operator. We call equations 38,39 the *fixed-point equations*.

To see this, let us add a bit of notation first. Let p denote the permutation corresponding to the permutation matrix P. We let $P \star A$ denote the tensor that results from expressing the tensor A after renumbering the nodes in V according to permutation P. Explicitly, the $(p(i_1), p(i_2), \ldots, p(i_k))$-th entry of $P \star A$ equals the $(i_1, i_2, \ldots, i_k)$-th entry of A. The matrix that corresponds to the operator $P\star$ in the standard tensor basis $e^{(i_1)} \otimes \cdots \otimes e^{(i_k)}$ is the Kronecker power $P^{T \otimes k} = (P^T)^{\otimes k}$. Note that $\operatorname{vec}(A)$ is exactly the coordinate vector of the tensor A in this standard basis and therefore we have $\operatorname{vec}(P \star A) = P^{T \otimes k} \operatorname{vec}(A)$. We now show:

Proposition 2. *A linear layer is invariant (equivariant) if and only if its coefficient matrix satisfies the fixed-point equations, namely* (38) *(*(39)*).*

Proof. Similarly to the argument from the order-2 case, let $L \in \mathbb{R}^{1 \times n^k}$ denote the matrix corresponding to a general linear operator $L : \mathbb{R}^{n^k} \to \mathbb{R}$. Order invariance means

$$L\operatorname{vec}(P \star A) = L\operatorname{vec}(A). \tag{40}$$

Using the matrix $P^{T \otimes k}$ we have equivalently $LP^{T \otimes k} \operatorname{vec}(A) = L\operatorname{vec}(A)$ which is in turn equivalent to $P^{\otimes k} \operatorname{vec}(L) = \operatorname{vec}(L)$ for all permutation matrices P. For order equivariance, let $L \in \mathbb{R}^{n^k \times n^k}$ denote the matrix of a general linear operator $L : \mathbb{R}^{n^k} \to \mathbb{R}^{n^k}$. Now equivariance of L is equivalent to

$$[L\operatorname{vec}(P \star A)] = P \star [L\operatorname{vec}(A)]. \tag{41}$$

Similarly to above this is equivalent to $LP^{T \otimes k} \operatorname{vec}(A) = P^{T \otimes k} L\operatorname{vec}(A)$ which in turn leads to $P^{\otimes k} LP^{T \otimes k} = L$, and using the Kronecker product properties we get $P^{\otimes 2k} \operatorname{vec}(L) = \operatorname{vec}(L)$. □

7.3.1 Solving the fixed-point equations

We have reduced the problem of finding all invariant and equivariant linear operators L to finding all solutions L of equations 38 and 39. Although the fixed point equations consist of an exponential number of equations with only a polynomial number of unknowns they actually possess a solution space of constant dimension (*i.e.*, independent of n).

To find the solution of $P^{\otimes \ell}\mathrm{vec}(X) = \mathrm{vec}(X)$, where $X \in \mathbb{R}^{n^\ell}$, note that $P^{\otimes \ell}\mathrm{vec}(X) = \mathrm{vec}(Q \star X)$, where $Q = P^T$. As above, the tensor $Q \star X$ is the tensor resulted from renumbering the nodes in V using permutation Q. Equivalently, the fixed-point equations we need to solve can be formulated as

$$Q \star X = X, \quad \forall Q \text{ permutation matrices} \tag{42}$$

The permutation group is acting on tensors $X \in \mathbb{R}^{n^\ell}$ with the action $X \mapsto Q \star X$. We are looking for fixed points under this action. To that end, let us define an equivalence relation in the index space of tensors $\mathbb{R}^{n^\ell}$, namely in $[n]^\ell$, where with a slight abuse of notation (we use light brackets) we set $[n] = \{1, 2, \ldots, n\}$. For multi-indices $a, b \in [n]^\ell$ we set $a \sim b$ iff a, b have the same *equality pattern*, that is $a_i = a_j \Leftrightarrow b_i = b_j$ for all $i, j \in [\ell]$.

The equality pattern equivalence relation partitions the index set $[n]^\ell$ into equivalence classes, the collection of which is denoted $[n]^\ell/_\sim$. Each equivalence class can be represented by a unique partition of the set $[\ell]$ where each set in the partition indicates maximal set of identical values. Let us exemplify. For $\ell = 2$ we have two equivalence classes $\gamma_1 = \{\{1\}, \{2\}\}$ and $\gamma_2 = \{\{1, 2\}\}$; γ_1 represents all multi-indices (i, j) where $i \neq j$, while γ_2 represents all multi-indices (i, j) where $i = j$. For $\ell = 4$, there are 15 equivalence classes $\gamma_1 = \{\{1\}, \{2\}, \{3\}, \{4\}\}$, $\gamma_2 = \{\{1\}, \{2\}, \{3, 4\}\}$, $\gamma_3 = \{\{1, 2\}, \{3\}, \{4\}\}$, $\ldots$; γ_3 represents multi-indices (i_1, i_2, i_3, i_4) so that $i_1 = i_2$, $i_2 \neq i_3$, $i_3 \neq i_4$, $i_2 \neq i_4$.

For each equivalence class $\gamma \in [n]^\ell/_\sim$ we define an order-ℓ tensor $B^\gamma \in \mathbb{R}^{n^\ell}$ by setting

$$B^\gamma_a = \begin{cases} 1 & a \in \gamma \\ 0 & \text{otherwise} \end{cases} \tag{43}$$

Since we have a tensor B^γ for every equivalence class γ, and the equivalence classes are in one-to-one correspondence with partitions of the set $[\ell]$ we have $\mathrm{b}(\ell)$ tensors B^γ. (Remember that $\mathrm{b}(\ell)$ denotes the ℓ-th Bell number.) We next prove:

Proposition 3. *The tensors B^γ in* (43) *form an orthogonal basis (in the standard inner-product) to the solution set of equations 42. The dimension of the solution set is therefore* $\mathrm{b}(\ell)$.

Proof. Let us first show that: X is a solution to (42) iff X is constant on equivalence classes of the equality pattern relation, $\sim$. Since permutation $q : [n] \to [n]$ is a bijection the equality patterns of $a = (i_1, i_2, \ldots, i_\ell) \in [n]^\ell$ and $q(a) = (q(i_1), q(i_2), \ldots, q(i_\ell)) \in [n]^\ell$ are identical, *i.e.*, $a \sim q(a)$. Taking the $a \in [n]^\ell$ entry of both sides of (42) gives $X_{q(a)} = X_a$. Now, if X is constant on equivalence classes then in particular it will have the same value at a and $q(a)$ for all $a \in [n]^\ell$ and permutations q. Therefore X is a solution to (42). For the only if part, consider a tensor X for which there exist multi-indices $a \sim b$ (with identical equality patterns) and $X_a \neq X_b$ then X is not a solution to (42). Indeed, since $a \sim b$ one can find a permutation q so that $b = q(a)$ and using the equation above, $X_b = X_{q(a)} = X_a$ which leads to a contradiction.

To finish the proof note that any tensor X, constant on equivalence classes, can be written as a linear combination of B^γ, which are merely indicators of the equivalence class. Furthermore, the collection B^γ have pairwise disjoint supports and therefore are an orthogonal basis. □

Combining propositions 2 and 3 we get the characterization of invariant and equivariant linear layers acting on general k-order tensor data over a single node set V:

Theorem 3. *The space of invariant (equivariant) linear layers $\mathbb{R}^{n^k} \to \mathbb{R}$ ($\mathbb{R}^{n^k} \to \mathbb{R}^{n^k}$) is of dimension $\mathrm{b}(k)$ ($\mathrm{b}(2k)$) with basis elements B^γ defined in* (43)*, where γ are equivalence classes in $[n]^k/_\sim$ ($[n]^{2k}/_\sim$).*

Biases Theorem 3 deals with purely linear layers, that is without *bias*, *i.e.*, without constant part. Nevertheless extending the previous analysis to constant layers is straight-forward. First, any constant layer $\mathbb{R}^{n^k} \to \mathbb{R}$ is also invariant so all constant invariant layers are represented by constants $c \in \mathbb{R}$. For equivariant layers $L : \mathbb{R}^{n^k} \to \mathbb{R}^{n^k}$ we note that equivariance means $C = L(P \star A) = P \star L(A) = P \star C$. Representing this equation in matrix form we get $P^{T\otimes k}\mathrm{vec}(C) = \mathrm{vec}(C)$. This shows that constant equivariant layers on one node set acting on general k-order tensors are also characterized by the fixed-point equations, and in fact have the same form and dimensionality as invariant layers on k-order tensors, see (38). Specifically, their basis is B^λ, $\lambda \in [n]^k/_\sim$. For example, for $k = 2$, the biases are shown on the right in figure 28.

Features. It is pretty common that input tensors have vector values (*i.e.*, features) attached to each hyperedge (k-tuple of nodes) in V, that is $A \in \mathbb{R}^{n^k \times d}$. Now linear invariant $\mathbb{R}^{n^k \times d} \to \mathbb{R}^{1\times d'}$ or equivariant $\mathbb{R}^{n^k \times d} \to \mathbb{R}^{n^k \times d'}$ layers can be formulated using a slight generalization of the previous analysis. The operator $P \star A$ is defined to act only on the nodal indices, *i.e.*, $i_1, \ldots, i_k$ (the first k indices). Explicitly, the $(p(i_1), p(i_2), \ldots, p(i_k), i_{k+1})$-th entry of $P \star A$ equals the $(i_1, i_2, \ldots, i_k, i_{k+1})$-th entry of A.

Invariance is now formulated exactly as before, (40), namely $L\mathrm{vec}(P \star A) = L\mathrm{vec}(A)$. The matrix that corresponds to $P\star$ acting on $\mathbb{R}^{n^k \times d}$ in the standard basis is $P^{T\otimes k} \otimes I_d$ and therefore $L(P^{T\otimes k} \otimes I_d)\mathrm{vec}(A) = L\mathrm{vec}(A)$. Since this is true for all A we have $(P^{\otimes k} \otimes I_d \otimes I_{d'})\,\mathrm{vec}(L) = \mathrm{vec}(L)$, using the properties of the Kronecker product. Equivariance is written as in (41), $[L\mathrm{vec}(P \star A)] = P \star [L\mathrm{vec}(A)]$. In matrix form, the equivariance equation becomes $L(P^{T\otimes k} \otimes I_d)\mathrm{vec}(A) = (P^{T\otimes k} \otimes I_{d'})L\mathrm{vec}(A)$, since this is true for all A and using the properties of the Kronecker product again we get to $P^{\otimes k} \otimes I_d \otimes P^{\otimes k} \otimes I_{d'}\ \mathrm{vec}(L) = \mathrm{vec}(L)$. The basis (with biases) to the solution space of these fixed-point equations is defined as follows. We use $a, b \in [n]^k,\ i, j \in [d],\ i', j' \in [d'], \lambda \in [n]^k/_\sim, \mu \in [n]^{2k}/_\sim$.

$$\text{invariant:} \qquad B^{\lambda,j,j'}_{a,i,i'} = \begin{cases} 1 & a\in\lambda,\ i=j,\ i'=j' \\ 0 & \text{otherwise} \end{cases} \quad ; \quad C^{j'}_{i'} = \begin{cases} 1 & i'=j' \\ 0 & \text{otherwise} \end{cases} \tag{44a}$$

$$\text{equivariant:} \qquad B^{\mu,j,j'}_{a,i,b,i'} = \begin{cases} 1 & (a,b)\in\mu,\ i=j,\ i'=j' \\ 0 & \text{otherwise} \end{cases} ; \quad C^{\lambda,j'}_{b,i'} = \begin{cases} 1 & b\in\lambda,\ i'=j' \\ 0 & \text{otherwise} \end{cases} \tag{44b}$$

Note that these basis elements are similar to the ones in (43) with the difference that we have different basis tensor for each pair of input j and output j' feature channels.

An invariant ((45a))/ equivariant ((45b)) linear layer L including the biases can be written as follows for input $A \in \mathbb{R}^{n^k \times d}$:

$$L(A)_{i'} = \sum_{a,i} T_{a,i,i'} A_{a,i} + Y_{i'}; \quad T = \sum_{\lambda,j,j'} w_{\lambda,j,j'} B^{\lambda,j,j'}; Y = \sum_{j'} b_{j'} C^{j'} \tag{45a}$$

$$L(A)_{b,i'} = \sum_{a,i} T_{a,i,b,i'} A_{a,i} + Y_{b,i'}; \quad T = \sum_{\mu,j,j'} w_{\mu,j,j'} B^{\mu,j,j'}; Y = \sum_{\lambda,j'} b_{\lambda,j'} C^{\lambda,j'} \tag{45b}$$

where the learnable parameters are $w \in \mathbb{R}^{\mathrm{b}(k)\times d\times d'}$ and $b \in \mathbb{R}^{d'}$ for a single linear *invariant* layer $\mathbb{R}^{n^k \times d} \to \mathbb{R}^{d'}$; and it is $w \in \mathbb{R}^{\mathrm{b}(2k)\times d\times d'}$ and $b \in \mathbb{R}^{\mathrm{b}(k)\times d'}$ for a single linear *equivariant* layer $\mathbb{R}^{n^k \times d} \to \mathbb{R}^{n^k \times d'}$. The natural generalization of theorem 3 to include bias and features is therefore:

Theorem 4. *The space of invariant (equivariant) linear layers* $\mathbb{R}^{n^k,d} \to \mathbb{R}^{d'}$ ($\mathbb{R}^{n^k \times d} \to \mathbb{R}^{n^k \times d'}$) *is of dimension* $dd'\mathrm{b}(k) + d'$ *(for equivariant:* $dd'\mathrm{b}(2k) + d'\mathrm{b}(k)$*) with basis elements defined in* (44)*; equation 45a (45b) show the general form of such layers.*

Since, by similar arguments to proposition 3, the purely linear parts B and biases C in (44) are independent solutions to the relevant fixed-point equations, theorem 4 will be proved if their number equals the dimension of the solution space of these fixed-point equations, namely $dd'\mathrm{b}(k)$ for purely linear part and d' for bias in the invariant case, and $dd'\mathrm{b}(2k)$ for purely linear and $d'\mathrm{b}(k)$ for bias in the equivariant case. This can be shown by repeating the arguments of the proof of proposition 3 slightly adapted to this case, or by a combinatorial identity we show in Section 7.7 .

For example, figure 28 depicts the 15 basis elements for linear equivariant layers $\mathbb{R}^{n\times n} \to \mathbb{R}^{n\times n}$ taking as input edge-value (order-2) tensor data $A \in \mathbb{R}^{n\times n}$ and outputting the same dimension tensor. The basis for the purely linear part are shown as $n^2 \times n^2$ matrices while the bias part as $n \times n$ matrices (far right); the size of the node set is $|V| = n = 5$.

Mixed order equivariant layers. Another useful generalization of order equivariant linear layers is to linear layers between *different order* tensor layers, that is, $L : \mathbb{R}^{n^k} \to \mathbb{R}^{n^l}$, where $l \neq k$. For example, one can think of a layer mapping an adjacency matrix to per-node features. For simplicity we will discuss the purely linear scalar-valued case, however generalization to include bias and/or general feature vectors can be done as discussed above. Consider the matrix $L \in \mathbb{R}^{n^l \times n^k}$ representing the linear layer L, using the renumbering operator, $P\star$, order equivariance is equivalent to $[L\mathrm{vec}(P \star A)] = P \star [L\mathrm{vec}(A)]$. Note that while this equation looks identical to (41) it is nevertheless different in the sense that the $P\star$ operator in the l.h.s. of this equation acts on k-order tensors while the one on the r.h.s. acts on l-order tensor. Still, we can transform this equation to a matrix equation as before by remembering that $P^{T\otimes k}$ is the matrix representation of the renumbering operator $P\star$ acting on k-tensors in the standard basis. Therefore, repeating the arguments in proof of proposition 2, equivariance is equivalent to $P^{\otimes(k+l)}\mathrm{vec}(L) = \mathrm{vec}(L)$, for all permutation matrices P. This equation is solved as in subseection 7.3.1. The corresponding bases to such equivariant layers are computed as in (44b), with the only difference that now $a \in [n]^k$, $b \in [n]^l$, and $\mu \in [n]^{k+l}/\sim$.

7.4 Experiments

Implementation details. We implemented our method in Tensorflow [1]. The equivariant linear basis was implemented efficiently using basic row/column/diagonal summation operators, see Section 7.6 for details. The networks we used are composition of $1-4$ equivariant linear layers with ReLU activation between them for the equivariant function setting. For invariant function setting we further added a max over the invariant basis and $1-3$ fully-connected layers with ReLU activations.

Table 8: Comparison to baseline methods on synthetic experiments.

	Symmetric projection			Diagonal extraction			Max singular vector				Trace		
# Layers	1	2	3	1	2	3	1	2	3	4	1	2	3
Trivial predictor	4.17	4.17	4.17	0.21	0.21	0.21	0.025	0.025	0.025	0.025	333.33	333.33	333.33
Hartford et al.	2.09	2.09	2.09	0.81	0.81	0.81	0.043	0.044	0.043	0.043	316.22	311.55	307.97
Ours	**1E-05**	**7E-06**	**2E-05**	**8E-06**	**7E-06**	**1E-04**	**0.015**	**0.0084**	**0.0054**	**0.0016**	**0.005**	**0.001**	**0.003**

Synthetic datasets. We tested our method on several synthetic equivariant and invariant graph functions that highlight the differences in expressivity between our linear basis and the basis of [104]. Given an input matrix data $A \in \mathbb{R}^{n\times n}$ we considered: (i) projection onto the symmetric matrices $\frac{1}{2}(A + A^T)$; (ii) diagonal

extraction $\mathrm{diag}(\mathrm{diag}(A))$ (keeps only the diagonal and plugs zeros elsewhere); (iii) computing the maximal right singular vector $\arg\max_{\|v\|_2=1} \|Av\|_2$; and (iv) computing the trace $\mathrm{tr}(A)$. Tasks (i)-(iii) are equivariant while task (iv) is invariant. We created accordingly 4 datasets with $10K$ train and $1K$ test examples of 40×40 matrices; for tasks (i), (ii), (iv) we used i.i.d. random matrices with uniform distribution in $[0, 10]$; we used mean-squared error (MSE) as loss; for task (iii) we random matrices with uniform distribution of singular values in $[0, 0.5]$ and spectral gap ≥ 0.5; due to sign ambiguity in this task we used cosine loss of the form $l(x, y) = 1 - \langle x/\|x\|, y/\|y\| \rangle^2$.

We trained networks with 1, 2, and 3 hidden layers with 8 feature channels each and a single fully-connected layer. Both our models as well as [104] use the same architecture but with different bases for the linear layers. Table 8 logs the *best* mean-square error of each method over a set of hyper-parameters. We add the MSE for the trivial mean predictor.

Table 9: Generalization.

	30	40	50
sym	0.0053	3.8E-05	0.0013
svd	0.0108	0.0084	0.0096
diag	0.0150	1.5E-05	0.0055

This experiment emphasizes simple cases in which the additional parameters in our model, with respect to [104], are needed. We note that [104] target a different scenario where the permutations acting on the rows and columns of the input matrix are not necessarily the same. The assumption taken in section, namely, that the same permutation acts on both rows and columns, gives rise to additional parameters that are associated with the diagonal and with the transpose of the matrix (for a complete list of layers for the $k = 2$ case see section 7.6). In case of an input matrix that represents graphs, these parameters can be understood as parameters that control self-edges or node features, and incoming/outgoing edges in a different way. Table 9 shows the result of applying the learned equivariant networks from the above experiment to graphs (matrices) of unseen sizes of $n = 30$ and $n = 50$. Note, that although the network was trained on a fixed size, the network provides plausible generalization to different size graphs. We note that the generalization of the invariant task of computing the trace did not generalize well to unseen sizes and probably requires training on different sizes as was done in the datasets below.

Table 10: Graph Classification Results.

dataset	MUTAG	PTC	PROTEINS	NCI1	NCI109	COLLAB	IMDB-B	IMDB-M
size	188	344	1113	4110	4127	5000	1000	1500
classes	2	2	2	2	2	3	2	3
avg node #	17.9	25.5	39.1	29.8	29.6	74.4	19.7	13
				Results				
DGCNN	85.83±1.7	58.59±2.5	75.54±0.9	74.44±0.5	NA	73.76±0.5	70.03±0.9	47.83±0.9
PSCN (k=10)	88.95±4.4	62.29±5.7	75±2.5	76.34±1.7	NA	72.6±2.2	71±2.3	45.23±2.8
DCNN	NA	NA	61.29±1.6	56.61± 1.0	NA	52.11±0.7	49.06±1.4	33.49±1.4
ECC	76.11	NA	NA	76.82	75.03	NA	NA	NA
DGK	87.44±2.7	60.08±2.6	75.68±0.5	80.31±0.5	80.32±0.3	73.09±0.3	66.96±0.6	44.55±0.5
DiffPool	NA	NA	78.1	NA	NA	75.5	NA	NA
CCN	91.64±7.2	70.62±7.0	NA	76.27±4.1	75.54±3.4	NA	NA	NA
GK	81.39±1.7	55.65±0.5	71.39±0.3	62.49±0.3	62.35±0.3	NA	NA	NA
RW	79.17±2.1	55.91±0.3	59.57±0.1	$>$ 3 days	NA	NA	NA	NA
PK	76±2.7	59.5±2.4	73.68±0.7	82.54±0.5	NA	NA	NA	NA
WL	84.11±1.9	57.97±2.5	74.68±0.5	84.46±0.5	85.12±0.3	NA	NA	NA
FGSD	92.12	62.80	73.42	79.80	78.84	80.02	73.62	52.41
AWE-DD	NA	NA	NA	NA	NA	73.93±1.9	74.45 ±5.8	51.54 ±3.6
AWE-FB	87.87±9.7	NA	NA	NA	NA	70.99 ± 1.4	73.13 ±3.2	51.58 ± 4.6
ours	84.61±10	59.47±7.3	75.19±4.3	73.71±2.6	72.48±2.5	77.92±1.7	71.27±4.5	48.55±3.9

Graph classification. We tested our method on standard benchmarks of graph classification. We use 8 different real world datasets from the benchmark of [257]: five of these datasets originate from bioinformat-

ics while the other three come from social networks. In all datasets the adjacency matrix of each graph is used as input and a categorial label is assigned as output. In the bioinformatics datasets node labels are also provided as inputs. These node labels can be used in our framework by placing their 1-hot representations on the diagonal of the input.

Table 10 specifies the results for our method compared to state-of-the-art deep and non-deep graph learning methods. We follow the evaluation protocol including the 10-fold splits of [268]. For each dataset we selected learning and decay rates on one random fold. In all experiments we used a fixed simple architecture of 3 layers with $(16, 32, 256)$ features accordingly. The last equivariant layer is followed by an invariant max layer according to the invariant basis. We then add two fully-connected hidden layers with $(512, 256)$ features.

We compared our results to seven deep learning methods: DGCNN [268], PSCN [181], DCNN [13], ECC [221], DGK [257], DiffPool [262] and CCN [132]. We also compare our results to four popular graph kernel methods: Graphlet Kernel (GK) [218],Random Walk Kernel (RW) [239], Propagation Kernel (PK) [179], and Weisfeiler-lehman kernels (WL) [219] and two recent feature-based methods: Family of Graph Spectral Distance (FGSD) [237] and Anonymous Walk Embeddings (AWE) [115]. Our method achieved results comparable to the state-of-the-art on the three social networks datasets, and slightly worse results than state-of-the-art on the biological datasets.

7.5 Generalizations to multi-node sets

Lastly, we provide a generalization of our framework to data that is given on tuples of nodes from a *collection* of node sets $V_1, V_2, \ldots, V_m$ of sizes $n_1, n_2, \ldots, n_m$ (resp.), namely $A \in \mathbb{R}^{n_1^{k_1} \times n_2^{k_2} \times \cdots \times n_m^{k_m}}$. We characterize invariant linear layers $L : \mathbb{R}^{n_1^{k_1} \times \cdots \times n_m^{k_m}} \to \mathbb{R}$ and equivariant linear layer $L : \mathbb{R}^{n_1^{k_1} \times \cdots \times n_m^{k_m}} \to \mathbb{R}^{n_1^{l_1} \times \cdots \times n_m^{l_m}}$, where for simplicity we do not discuss features that can be readily added as discussed in subseection 7.3. Note that the case of $k_i = l_i = 1$ for all $i = 1, \ldots, m$ is treated in [104]. The reordering operator now is built out of permutation matrices $P_i \in \mathbb{R}^{n_i \times n_i}$ (p_i denotes the permutation), $i = 1, \ldots, m$, denoted $P_{1:m}\star$, and defined as follows: the $(p_1(a_1), p_2(a_2), \ldots, p_m(a_m))$-th entry of the tensor $P_{1:m} \star A$, where $a_i \in [n_i]^{k_i}$ is defined to be the $(a_1, a_2, \ldots, a_m)$-th entry of the tensor A. Rewriting the invariant and equivariant equations, *i.e.*, (40), 41, in matrix format, similarly to before, we get the fixed-point equations: $M\mathrm{vec}(L) = \mathrm{vec}(L)$ for invariant, and $M \otimes M\mathrm{vec}(L) = \mathrm{vec}(L)$ for equivariant, where $M = P_1^{\otimes k_1} \otimes \cdots \otimes P_m^{\otimes k_m}$. The solution of these equations would be linear combinations of basis tensor similar to (44) of the form

$$\text{invariant: } B^{\lambda_1,\ldots,\lambda_m}_{a_1,\ldots,a_m} = \begin{cases} 1 & a_i \in \lambda_i,\ \forall i \\ 0 & \text{otherwise} \end{cases}; \quad \text{equivariant: } B^{\mu_1,\ldots,\mu_m}_{a_1,\ldots,a_m,b_1,\ldots,b_m} = \begin{cases} 1 & (a_i,b_i) \in \mu_i,\ \forall i \\ 0 & \text{otherwise} \end{cases} \tag{46}$$

where $\lambda_i \in [n_i]^{k_i}$, $\mu_i \in [n_i]^{k_i+l_i}$, $a \in [n_i]^{k_i}$, $b_i \in [n_i]^{l_i}$. The number of these tensors is $\prod_{i=1}^m \mathrm{b}(i)$ for invariant layers and $\prod_{i=1}^m \mathrm{b}(k_i + l_i)$ for equivariant layers. Since these are all linear independent (pairwise disjoint support of non-zero entries) we need to show that their number equal the dimension of the solution of the relevant fixed-point equations above. This can be done again by similar arguments to the proof of proposition 3 or as shown in section 7.7. To summarize:

Theorem 5. *The linear space of invariant linear layers* $L : \mathbb{R}^{n_1^{k_1} \times n_2^{k_2} \times \cdots \times n_m^{k_m}} \to \mathbb{R}$ *is of dimension* $\prod_{i=1}^m \mathrm{b}(k_i)$. *The equivariant linear layers* $L : \mathbb{R}^{n_1^{k_1} \times n_2^{k_2} \times \cdots \times n_m^{k_m}} \to \mathbb{R}^{n_1^{l_1} \times n_2^{l_2} \times \cdots \times n_m^{l_m}}$ *has dimension* $\prod_{i=1}^m \mathrm{b}(k_i + l_i)$. *Orthogonal bases for these layers are listed in* (46).

7.6 Efficient implementation of layers

For fast execution of order-2 layers we implemented the following 15 operations which can be easily shown to span the basis discussed in the section. We denote by $\mathbf{1} \in \mathbb{R}^n$ the vector of all ones.

1. The identity and transpose operations: $L(A) = A$, $L(A) = A^T$.
2. The diag operation: $L(A) = \text{diag}(\text{diag}(A))$.
3. Sum of rows replicated on rows/ columns/ diagonal: $L(A) = A\mathbf{1}\mathbf{1}^T$, $L(A) = \mathbf{1}(A\mathbf{1})^T$, $L(A) = \text{diag}(A\mathbf{1})$.
4. Sum of columns replicated on rows/ columns/ diagonal: $L(A) = A^T\mathbf{1}\mathbf{1}^T$, $L(A) = \mathbf{1}(A^T\mathbf{1})^T$, $L(A) = \text{diag}(A^T\mathbf{1})$.
5. Sum of all elements replicated on all matrix/ diagonal: $L(A) = (\mathbf{1}^T A\mathbf{1}) \cdot \mathbf{1}\mathbf{1}^T$, $L(A) = (\mathbf{1}^T A\mathbf{1}) \cdot \text{diag}(\mathbf{1})$.
6. Sum of diagonal elements replicated on all matrix/diagonal: $L(A) = (\mathbf{1}^T\text{diag}(A)) \cdot \mathbf{1}\mathbf{1}^T$, $L(A) = (\mathbf{1}^T\text{diag}(A)) \cdot \text{diag}(\mathbf{1})$.
7. Replicate diagonal elements on rows/columns: $L(A) = \text{diag}(A)\mathbf{1}^T$, $L(A) = \mathbf{1}\text{diag}(A)^T$.

We normalize each operation to have unit max operator norm. We note that in case the input matrix is symmetric, our basis reduces to 11 elements in the first layer. If we further assume the matrix has zero diagonal we get a 6 element basis in the first layer. In both cases our model is more expressive than the 4 element basis of [104] and as the output of the first layer (or other inner states) need not be symmetric nor have zero diagonal the deeper layers can potentially make good use of the full 15 element basis.

7.7 Invariant and equivariant subspace dimensions

We prove a useful combinatorial fact as a corollary of proposition 3. This fact will be used later to easily compute the dimensions of more general spaces of invariant and equivariant linear layers. We use the fact that if V is a representation of a finite group G then

$$\phi = \frac{1}{|G|} \sum_{g \in G} g \in \text{End}(V) \tag{47}$$

is a projection onto $V^G = \{v \in V \mid gv = v,\ \forall g \in G\}$, the subspace of fixed points in V under the action of G, and consequently that $\text{tr}(\phi) = \dim(V^G)$ (see [84] for simple proofs).

Proposition 4. *The following formula holds:*

$$\frac{1}{n!} \sum_{P \in \Pi_n} tr(P)^k = \text{b}(k),$$

where Π_n is the matrix permutation group of dimensions $n \times n$.

Proof. In our case, the vector space is the space of order-k tensors and the group acting on it is the matrix group $G = \left\{P^{\otimes k} \mid P \in \Pi_m\right\}$.

$$\dim(V^G) = \text{tr}(\phi) = \frac{1}{|G|} \sum_{g \in G} \text{tr}(g) = \frac{1}{n!} \sum_{P \in \Pi_n} \text{tr}(P^{\otimes k}) = \frac{1}{n!} \sum_{P \in \Pi_n} \text{tr}(P)^k,$$

where we used the multiplicative law of the trace with respect to Kronecker product. Now we use proposition 3 noting that in this case V^G is the solution space of the fixed-point equations. Therefore, $\dim(V^G) = \mathrm{b}(k)$ and the proof is finished. □

Recall that for a permutation matrix P, $\mathrm{tr}(P) = |\{i \in [n] \text{ s.t. } P \text{ fixes } e_i\}|$. Using this, we can interpret the equation in proposition 4 as the k-th moment of a random variable counting the number of fixed points of a permutation, with uniform distribution over the permutation group. Proposition 4 proves that the k-th moment of this random variable is the k-th Bell number.

We can now use proposition 4 to calculate the dimensions of two linear layer spaces: (i) Equivariant layers acting on order-k tensors with features (as in 7.3); and (ii) multi-node sets (as in subseection 7.5).

Theorem 4. *The space of invariant (equivariant) linear layers $\mathbb{R}^{n^k,d} \to \mathbb{R}^{d'}$ ($\mathbb{R}^{n^k \times d} \to \mathbb{R}^{n^k \times d'}$) is of dimension $dd'\mathrm{b}(k) + d'$ (for equivariant: $dd'\mathrm{b}(2k) + d'\mathrm{b}(k)$) with basis elements defined in* (44)*; equations 45a (45b) show the general form of such layers.*

Proof. We prove the dimension formulas for the invariant case. The equivariant case is proved similarly. The solution space for the fixed point equations is the set V^G for the matrix group $G = \left\{P^{\otimes k} \otimes I_d \otimes I_{d'} \mid P \in \Pi_n\right\}$. Using the projection formula 47 we get that the dimension of the solution subspace, which is the space of invariant linear layers, can be computed as follows:

$$\dim(V^G) = \frac{1}{n!}\sum_{P\in\Pi_n} \mathrm{tr}(P)^k \,\mathrm{tr}(I_d)\mathrm{tr}(I_{d'}) = \left(\frac{1}{n!}\sum_{P\in\Pi_n} \mathrm{tr}(P)^k\right) \mathrm{tr}(I_d)\,\mathrm{tr}(I_{d'}) = d \cdot d' \cdot \mathrm{b}(k).$$

□

Theorem 5. *The linear space of invariant linear layers $L : \mathbb{R}^{n_1^{k_1} \times n_2^{k_2} \times \cdots \times n_m^{k_m}} \to \mathbb{R}$ is of dimension $\prod_{i=1}^m \mathrm{b}(k_i)$. The equivariant linear layers $L : \mathbb{R}^{n_1^{k_1} \times n_2^{k_2} \times \cdots \times n_m^{k_m}} \to \mathbb{R}^{n_1^{l_1} \times n_2^{l_2} \times \cdots \times n_m^{l_m}}$ has dimension $\prod_{i=1}^m \mathrm{b}(k_i + l_i)$. Orthogonal bases for these layers are listed in* (46)*.*

Proof. In this case we get the fixed-point equations: $M\mathrm{vec}(L) = \mathrm{vec}(L)$ for invariant, and $M \otimes M\mathrm{vec}(L) = \mathrm{vec}(L)$ for equivariant, where $M = P_1^{\otimes k_1} \otimes \cdots \otimes P_m^{\otimes k_m}$. Similarly to the previous theorem, plugging M into (47), using the trace multiplication rule and proposition 4 we get the above formulas. □

7.8 Implementing message passing with our model

In this section we show that our model can approximate message passing layers as defined in [89] to an arbitrary precision, and consequently that our model is able to approximate any network consisting of several such layers. The key idea is to mimic multiplication of features by the adjacency matrix, which allows summing over local neighborhoods. This can be implemented using our basis.

Theorem 6. *Our model can represent message passing layers to an arbitrary precision on compact sets.*

Proof. Consider input vertex data $H = (h_u) \in \mathcal{R}^{n\times d}$ (n is the number of vertices in the graph, and d is the input feature depth), adjacency matrix $A = (a_{uv}) \in \mathcal{R}^{n\times n}$ of the graph, and additional edge features $E = (e_{uv}) \in \mathcal{R}^{n\times n\times l}$. Recall that a message passing layer of [89] is of the form:

$$m_u^{t+1} = \sum_{v\in N(u)} M_t(h_u^t, h_v^t, e_{uv}) \tag{48a}$$

$$h_u^{t+1} = U_t(h_u^t, m_u^{t+1}) \tag{48b}$$

where u, v are nodes in the graph, h_u^t is the feature vector associated with u in layer t, and e_{uv} are additional edge features. We denote the number of output features of M_t by d'.

In our setting we represent this data using a tensor $Y \in \mathcal{R}^{n \times n \times (1+l+d)}$ where the first channel is the adjacency matrix A, the next l channels are edge features, and the last d channels are diagonal matrices that hold X.

Let us construct a message passing layer using our model:

1. Our first step is constructing an $n \times n \times (1 + l + 2d)$ tensor. In the first channels we put the adjacency matrix A and the edge features E. In the next d channels we replicate the features on the rows, and in the last d channels we replicate features on the columns. The output tensor Z^1 has the form $Z^1_{u,v} = [a_{uv}, e_{uv}, h_u^t, h_v^t]$.

2. Next, we copy the feature channels $[a_{uv}, e_{uv}, h_u^t]$ to the output tensor Z^2. We then apply a multilayer perceptron (MLP) on the last $l + 2d$ feature dimensions of Z^1 that approximates M_t [110]. The output tensor of this stage is $Z^2_{u,v} = [a_{uv}, e_{uv}, h_u^t, M_t(h_u^t, h_v^t, e_{uv}) + \epsilon_1]$.

3. Next, we would like to perform point-wise multiplication $Z^2_{u,v,1} \odot Z^2_{u,v,(l+d+2):end}$. This step would zero out the outputs of M_t for non-adjacent nodes u, v. As this point-wise multiplication is not a part of our framework we can use an MLP on the feature dimension to approximate it and get $Z^3_{u,v} = [a_{uv}, e_{uv}, h_u^t, a_{uv} M_t(h_u^t, h_v^t) + \epsilon_2]$.

4. As before we copy the feature channels $[a_{uv}, e_{uv}, h_u^t]$. We now apply a sum over the rows (v dimension) on the M_t output channels. We put the output of this sum on the diagonal of Z^4 in separate channels. We get $Z^4_{u,v} = [a_{uv}, e_{uv}, h_u^t, \delta_{uv} \sum_{w \in N(u)} M_t(h_u^t, h_w^t) + \epsilon_3]$, where δ_{uv} is the Kronecker delta. We get a tensor $Z^4 \in \mathbb{R}^{n \times n \times (1+l+d+d')}$.

5. The last step is to apply an MLP to the last $d + d'$ feature channels of the diagonal of Z^4. After this last step we have $Z^5_{u,v} = [a_{uv}, e_{uv}, \delta_{uv} U_t(h_u^t, m_u^{t+1}) + \epsilon_4]$.

The errors ϵ_i depend on the approximation error of the MLP to the relevant function, the previous errors ϵ_{i-1} (for $i > 1$), and uniform bounds as-well as uniform continuity of the approximated functions. □

Corollary 1. *Our model can represent any message passing network to an arbitrary precision on compact sets. In other words, in terms of universality our model is at-least as powerful as* any *message passing neural network (MPNN) that falls into the framework of [89].*

8 Universality of invariant networks

This section is based on [159].

8.1 Introduction

The basic paradigm of deep neural networks is repeatedly composing "layers" of linear functions with non-linear, entrywise activation functions to create effective predictive models for learning tasks of interest.

When trying to learn a function (task) f that is known to be invariant to some group of symmetries G (*i.e.*, G-invariant function) it is common to use linear layers that respect this symmetry, namely, invariant and/or equivariant linear layers. Networks with invariant/equivariant linear layers with respect to some group G will be referred here as *G-invariant networks*.

A fundamental question in learning theory is that of *approximation* or *universality* [59, 110]. In the invariant case: Can a G-invariant network approximate an arbitrary continuous G-invariant function?

The goal of this section is to address this question for *all* finite permutation groups $G \leq S_n$, where S_n is the symmetric group acting on $[n] = \{1, 2, \ldots, n\}$. Note that this is a fairly general setting that contains many useful examples (detailed below).

The archetypal example of G-invariant networks is Convolutional Neural Networks (CNNs) [141, 136] that restrict their linear layers to convolutions in order to learn image tasks that are translation invariant or equivariant [3].

In recent years researchers are considering other types of data and/or symmetries and consequently new G-invariant networks have emerged. Tasks involving point clouds or sets are in general invariant to the order of the input and therefore permutation invariance/equivariance was developed [197, 264]. Learning tasks involving interaction between different sets, where the input data is tabular, require dealing with different permutations acting independently on each set [105]. Tasks involving graphs and hyper-graphs lead to symmetries defined by tensor products of permutations [132, 158]. A general treatment of invariance/equivariance to finite subgroups of the symmetric group is discussed in [203]; infinite symmetries are discussed in general in [131] as well as in [54, 52, 53, 246].

Among these examples, universality is known for point-clouds networks and sets networks [197, 264], as well as networks invariant to finite translation groups (*e.g.*, cyclic convolutional neural networks) [259]. However, universality is not known for tabular and multi-set networks [105], graph and hyper-graph networks [132, 158]; and networks invariant to finite translations with rotations and/or reflections. We cover all these cases in this section.

Maybe the most related work to ours is [259] that considered actions of compact groups and suggested provably universal architectures that are based on polynomial layers. In contrast, we study the standard and widely used linear layer model.

The section is organized as follows: First, we prove that an arbitrary continuous function $f : \mathbb{R}^n \to \mathbb{R}$ invariant to an arbitrary permutation group $G \leq S_n$ can be approximated using a G-invariant network. The proof is constructive and makes use of linear equivariant layers between tensors $X \in \mathbb{R}^{n^k}$ of order $k \leq d$, where d depends on the permutation group G.

Second, we prove a lower bound on the order d of tensors used in a G-invariant network so to achieve universality. Specifically, we show that for $G = A_n$ (the alternating group) any G-invariant network that

[3] It is common to use convolutional layers without cyclic padding which implies that these networks are not precisely translation invariant.

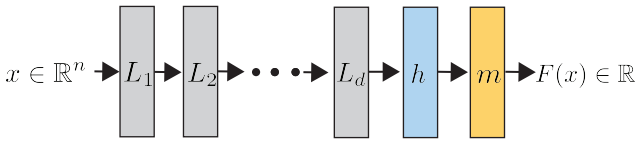


Figure 29: Illustration of invariant network architecture. The function is composed of multiple linear G-equivariant layers (gray), possibly of high order, and ends with a linear G-invariant function (light blue) followed by a Multi Layer Perceptron (yellow).

uses tensors of order at-most $d = (n-2)/2$ cannot approximate arbitrary G-invariant functions.

We conclude the section by considering the question: For which groups $G \leq S_n$, G-invariant networks using only first order tensors are universal? We prove a necessary condition, and describe families of groups for which universality cannot be attained using only first order tensors.

8.2 Preliminaries and main results

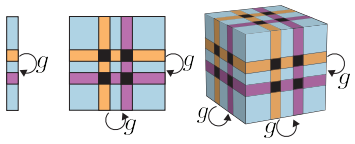
The symmetries we consider in this section are arbitrary subgroups of the symmetric group, *i.e.*, $G \leq S_n$. The action of G on $x \in \mathbb{R}^n$ used in this section is defined as

$$g \cdot x = (x_{g^{-1}(1)}, \dots, x_{g^{-1}(n)}),\ g \in G. \tag{49}$$

The action of G on *tensors* $X \in \mathbb{R}^{n^k \times a}$ (the last index, denoted j represents feature depth) is defined similarly by

$$(g \cdot X)_{i_1 \dots i_k, j} = X_{g^{-1}(i_1) \dots g^{-1}(i_k), j},\ g \in G. \tag{50}$$

The inset illustrates this action on tensors of order $k = 1, 2, 3$: the permutation g is a transposition of two numbers and is applied to each dimension of the tensor.



Definition 1. *A G-invariant function is a function $f : \mathbb{R}^n \to \mathbb{R}$ that satisfies $f(g \cdot x) = f(x)$ for all $x \in \mathbb{R}^n$ and $g \in G$.*

Definition 2. *A linear equivariant layer is an affine map $L : \mathbb{R}^{n^k \times a} \to \mathbb{R}^{n^l \times b}$ satisfying $L(g \cdot X) = g \cdot L(X)$, for all $g \in G$, and $X \in \mathbb{R}^{n^k \times a}$. An invariant linear layer is an affine map $h : \mathbb{R}^{n^k \times a} \to \mathbb{R}^b$ satisfying $h(g \cdot X) = h(X)$, for all $g \in G$, and $X \in \mathbb{R}^{n^k \times a}$.*

A common way to construct G-invariant networks is:

Definition 3. *A G-invariant network is a function $F : \mathbb{R}^{n \times a} \to \mathbb{R}$ defined as*

$$F = m \circ h \circ L_d \circ \sigma \circ \cdots \circ \sigma \circ L_1,$$

where L_i are linear G-equivariant layers, σ is an activation function [4], h is a G-invariant layer, and m is a Multi-Layer Perceptron (MLP).

Figure 29 illustrates the G-invariant network model. By construction, G-invariant networks are G-invariant functions (note that entrywise activation is equivariant as-well). This framework has been used, with appropriate group G, in previous works to build predictive G-invariant models for learning.

[4] We assume any activation function for which the universal approximation theorem for MLP holds, *e.g.*, ReLU and sigmoid.

Our goal is to show the *approximation power* of G-invariant networks. Namely, that G-invariant networks can approximate arbitrary continuous G-invariant functions f. Without loss of generality, we consider only functions of the form $f : \mathbb{R}^n \to \mathbb{R}$. Indeed, in case of multiple features, $\mathbb{R}^{n\times a}$, we rearrange the input as $\mathbb{R}^{n'}$, $n' = na$, and take the appropriate $G' \leq S_{n'}$. We prove:

Theorem 7. *Let $f : \mathcal{R}^n \to \mathcal{R}$ be a continuous G-invariant function for some $G \leq S_n$, and $K \subset \mathcal{R}^n$ a compact set. There exists a G-invariant network that approximates f to an arbitrary precision.*

The proof of Theorem 7 is constructive and builds an f-approximating G-invariant network with hidden tensors $X \in \mathbb{R}^{n^d}$ of order d, where $d = d(G)$ is a natural number depending on the group G. Unfortunately, we show that in the worst case d can be as high as $\frac{n(n-1)}{2}$. Note that $d = 2$ could already be computationally challenging. It is therefore of interest to ask whether there exist more efficient G-invariant networks that use lower order tensors without sacrificing approximation power. Surprisingly, the answer is that in general we can not go lower than order n for general permutation groups G. Specifically, we prove the following for $G = A_n$, the alternating group:

Theorem 8. *If an A_n-invariant network has the universal approximation property then it consists of tensors of order at least $\frac{n-2}{2}$.*

Although in general we cannot expect universal approximation of G-invariant networks with inner tensor order smaller than $\frac{n-2}{2}$, it is still possible that for *specific* groups of interest we can prove approximation power with more efficient (*i.e.*, lower order inner tensors) G-invariant networks. Of specific interest are G-invariant networks that use only first order tensors. In section 8.5 we prove the following necessary condition for universality of first-order G-invariant networks:

Theorem 9. *Let $G \leq S_n$. If first order G-invariant networks are universal, then $\left|[n]^2/H\right| < \left|[n]^2/G\right|$ for any strict super-group $G < H \leq S_n$.*

$|[n]^2/G|$ is the number of equivalence classes of $[n]^2$ defined by the relation: $(i_1, i_2) \sim (j_1, j_2)$ if $j_\ell = g(i_\ell)$, $\ell = 1, 2$ for some $g \in G$. Intuitively, this condition asks that super-groups of G have *strictly* better separation of the double index space $[n]^2$.

8.3 G-invariant networks universality

The key to showing theorem 7, namely that G-invariant networks are universal, is showing they can approximate a set of functions that are: (i) G-invariant; and (ii) can approximate arbitrary G-invariant functions to a desired precision. The G-invariant polynomials are an example of such a set:

Definition 4. *The G-invariant polynomials are all the polynomials in $x_1, \ldots, x_n$ over $\mathbb{R}$ that are also G-invariant functions. They are denoted $\mathbb{R}[x_1, \ldots, x_n]^G$, where $\mathbb{R}[x_1, \ldots, x_n]$ is the set of all polynomials over $\mathbb{R}$.*

To see that G-invariant polynomials can approximate any arbitrary *(*continuous) function $f : K \subset \mathbb{R}^n \to \mathbb{R}$, where K is a compact set, one can use the Stone-Weiestrass (SW) theorem, as done in [259]: First use SW to approximate f over a symmetrized domain $K' = \cup_{g\in G} g \cdot K$ by *some* (not necessarily G-invariant) polynomial $p \in \mathbb{R}[x_1, \ldots, x_n]$. Second, consider

$$q(x) = \frac{1}{|G|} \sum_{g\in G} p(g \cdot x).$$

q is a G-invariant polynomial and hence

$$q \in \mathbb{R}[x_1, \ldots, x_n]^G,$$

furthermore for $x \in K$:

$$|q(x) - f(x)| \leq$$
$$\frac{1}{|G|} \sum_{g \in G} \left| p(g \cdot x) - f(g \cdot x) \right| \leq \max_{x \in K'} |p(x) - f(x)| .$$

Our goal in this section is to prove the following proposition that, together with the comment above, prove theorem 7:

Proposition 5. *For any $\epsilon > 0$, $K \subset \mathbb{R}^n$ compact set, and G-invariant polynomial $p \in \mathbb{R}[x_1, \ldots, x_n]^G$ there exists a G-invariant network F that approximates p to an ϵ-accuracy, namely $\max_{x \in K} |F(x) - p(x)| < \epsilon$.*

The proposition will be proved in several steps:

(i) We represent p as $p(x) = \sum_{k=0}^{d} p_k(x)$, where p_k is a G-invariant *homogeneous* polynomial of degree k, *i.e.*, $p_k \in \mathbb{R}_k[x_1, \ldots, x_n]^G$.

(ii) We characterize all homogeneous G-invariant polynomials of a fixed degree k. In particular we find a basis to all such polynomials, $b_{k1}, b_{k2}, \ldots, b_{kn_k} \in \mathbb{R}_k[x_1, \ldots, x_n]^G$. Using the bases of homogeneous G-invariant polynomials of degrees up-to d we write

$$p(x) = \sum_{k=0}^{d} \sum_{j=1}^{n_k} \alpha_{kj} b_{kj}(x). \tag{51}$$

(iii) We approximate each basis element b_{kj} using a G-invariant network.

(iv) We construct a G-invariant network F approximating p to an ϵ-accuracy using Equation 51 and (iii).

8.3.1 Proof of proposition 5

Part (i): It is a known fact that a G-invariant polynomial can be written as a sum of homogeneous G-invariant polynomials [134]:

Lemma 6. *Let $p : \mathcal{R}^n \to \mathcal{R}$ be a G-invariant polynomial of degree d. Then p can be written as $p(x) = \sum_{k=0}^{d} p_k(x)$ where p_k are homogeneous G-invariant polynomials of degree k.*

Part (ii): We need to find bases for the linear spaces of homogeneous G-invariant polynomials of degree $k = 0, 1, \ldots, d$, *i.e.*, $\mathbb{R}_k[x_1, \ldots, x_n]^G$. Any homogeneous polynomial of degree k can be written as

$$p(x) = \sum_{i_1, \ldots, i_k = 1}^{n} W_{i_1 \ldots i_k} \, x_{i_1} \cdots x_{i_k}, \tag{52}$$

where $W \in \mathcal{R}^{n^k}$ is its coefficient tensor; since $x_{i_1} \cdots x_{i_k} = x_{i_{\sigma(1)}} \cdots x_{i_{\sigma(k)}}$ for all $\sigma \in S_k$, a unique choice of W can be obtained by taking a symmetric W. That is, W that satisfies $W_{i_1 \ldots i_k} = W_{i_{\sigma(1)} \cdots i_{\sigma(k)}}$, for all $\sigma \in S_k$. In short, we ask $W \in \mathrm{Sym}^k(\mathcal{R}^n) \subset \mathbb{R}^{n^k}$. For example, the case $k = 2$ amounts to representing a quadratic form using a symmetric matrix, that is W satisfies in this case $W = W^T$. The next proposition shows that if p is G-invariant, its coefficient tensor is a fixed point of the action of G on symmetric tensors $W \in \mathcal{R}^{n^k}$:

Proposition 6. *Let $p \in \mathbb{R}_k[x_1, \ldots, x_n]^G$. Then its coefficient tensor $W \in \mathcal{R}^{n^k}$ satisfies the fixed point equation:*

$$g \cdot W = W, \; \forall g \in G. \tag{53}$$

Proof. From the fact that p is G-invariant we get the following set of equations $p(x) = p(g \cdot x)$, for all $g \in G$.

$$\begin{aligned} p(x) &= p(g \cdot x) \\ &= \sum_{i_1,\ldots,i_k=1}^{n} W_{i_1 \ldots i_k} \, x_{g^{-1}(i_1)} \cdots x_{g^{-1}(i_k)} \\ &= \sum_{i_1,\ldots,i_k=1}^{n} W_{g(i_1)\ldots g(i_k)} \, x_{i_1} \cdots x_{i_k}. \end{aligned}$$

By equating monomials' coefficients of $p(x)$ and $p(g \cdot x)$ and the symmetry of W we get

$$W_{i_1 \ldots i_k} = W_{g(i_1)\ldots g(i_k)}.$$

This implies that W satisfies $g \cdot W = W$ for all $g \in G$. □

Equation 53 is a linear homogeneous system of equations and therefore the set of solutions W forms a linear space. To define a basis for this linear space we first define the following equivalence relation: $(i_1, \ldots, i_k) \sim (j_1, \ldots, j_k)$ if there exists $g \in G$ and $\sigma \in S_k$ so that $j_\ell = g(i_{\sigma(\ell)})$, $\ell = 1, \ldots, k$. Intuitively, g takes care of the G-invariance while σ factors out the fact that the monomials $x_{i_1} \cdots x_{i_k} = x_{\sigma(i_1)} \cdots x_{\sigma(i_k)}$. For example, let $n = 5$, $k = 3$, $g = (23)(45)$, $\sigma = (23)$ (we use cycle notation), then we have: $(2, 2, 4) \sim (3, 5, 3)$. The equivalence classes are denoted τ and called the *k-classes*. We show:

Proposition 7. *The set of polynomials*

$$p^\tau(x) = \sum_{(i_1,\ldots,i_k)\in\tau} x_{i_1} \cdots x_{i_k}, \tag{54}$$

where τ is a k-class, form a basis to $\mathbb{R}_k[x_1, \ldots, x_n]^G$.

Proof. Denote W^τ the symmetric coefficient tensor of p^τ, Note that

$$W^\tau_{i_1 \ldots i_k} = \begin{cases} 1 & (i_1, \ldots, i_k) \in \tau \\ 0 & \text{otherwise} \end{cases}. \tag{55}$$

Since

$$p^\tau(g \cdot x) = \sum_{(i_1,\ldots,i_k)\in\tau} x_{g^{-1}(i_1)} \cdots x_{g^{-1}(i_k)} = p^\tau(x),$$

$p^\tau \in \mathbb{R}_k[x_1, \ldots, x_n]^G$. The set of polynomials p^τ, with τ a k-classes, is a linearly independent set since each p^τ contains a different collection of monomials. By Proposition 6, the symmetric coefficient tensor W of every $q \in \mathbb{R}_k[x_1, \ldots, x_n]^G$ satisfies the fixed-point equation, (53). This in particular means that W is constant on its k-classes. Hence W can be written as linear combination of W^τ, see also (55). □

As we later show, the fixed point equation, (53), is also used to characterize and compute a basis for the space of *linear* permutation-equivariant and invariant layers [158]. These equations are equivalently formulated using weight sharing scheme in [203]. A slight difference in this case, that deals with polynomials, is the additional constraints that formulate the symmetry of W which are needed since every polynomial of degree > 1 has several representing tensors W.

Part (iii): Our next step is approximating each p^τ with a G-invariant network. The next proposition introduces the building blocks of this construction:

Proposition 8. *Let τ be a k-class and let $L^\tau_\ell : \mathcal{R}^n \to \mathcal{R}^{n^k}$, $\ell = 1, \ldots, k$, be a linear operator defined as follows:*
For $x \in \mathbb{R}^n$

$$L^\tau_\ell(x)_{i_1 \ldots i_k} = \begin{cases} x_{i_\ell} & (i_1, \ldots, i_k) \in \tau \\ 0 & \textit{otherwise} \end{cases}.$$

Then L^τ_ℓ is a linear G-equivariant function, that is

$$L^\tau_\ell(g \cdot x) = g \cdot L^\tau_\ell(x),\ \forall x \in \mathbb{R}^n, g \in G.$$

Proof. We have :

$$g \cdot L^\tau_\ell(x)_{i_1 \ldots i_k} = \begin{cases} x_{g^{-1}(i_\ell)} & (g^{-1}(i_1), \ldots, g^{-1}(i_k)) \in \tau \\ 0 & \text{otherwise} \end{cases}$$

On the other hand,

$$L^\tau_\ell(g \cdot x)_{i_1 \ldots i_k} = \begin{cases} x_{g^{-1}(i_\ell)} & (i_1, \ldots, i_k) \in \tau \\ 0 & \text{otherwise} \end{cases}$$

and both expressions are equal since $(i_1, \ldots, i_k) \in \tau$ if and only if $(g^{-1}(i_1), \ldots, g^{-1}(i_k)) \in \tau$ by definition of τ. □

Next, we construct the approximating G-invariant network:

Proposition 9. *For any $\epsilon > 0$, $K \subset \mathbb{R}^n$ compact set, and τ k-class there exists a G-invariant network F^τ that approximates p^τ from* (54) *to an ϵ-accuracy.*

Proof. Let $c > 0$ be sufficiently large so that $K \subset [-c, c]^n \subset \mathbb{R}^n$. Denote $m^k : \mathbb{R}^k \to \mathbb{R}$ an MLP that approximates the multiplication function, $f(y_1, \ldots, y_k) = \prod_{i=1}^k y_i$, in $[-c, c]^k$ to $n^{-k}\epsilon$-accuracy, *i.e.*, $\max_{-c \le y_i \le c} \left| f(y) - m^k(y) \right| < n^{-k}\epsilon$.

Consider the following G-invariant network: First, given an input $x \in \mathbb{R}^n$ map it to $\mathbb{R}^{n^k \times k}$ (*i.e.*, k is the number of channnels) by

$$L^\tau(x)_{i_1 \ldots i_k, \ell} = L^\tau_\ell(x)_{i_1 \ldots i_k}. \tag{56}$$

$L^\tau : \mathbb{R}^n \to \mathbb{R}^{n^k \times k}$ is a linear equivariant layer (see (50)). Second, apply m^k to the feature dimension in $\mathbb{R}^{n^k \times k}$. That is, given $y \in \mathbb{R}^{n^k \times k}$ define

$$M^k(y)_{i_1, \ldots, i_k} = m^k(y_{i_1 \ldots, i_k, 1}, \ldots, y_{i_1 \ldots, i_k, k}).$$

Note that $M^k : \mathbb{R}^{n^k \times k} \to \mathbb{R}^{n^k}$ can be interpreted as a composition of equivariant linear layers [5].

Lastly, denote $s : \mathbb{R}^{n^k} \to \mathbb{R}$ the summation layer: for $z \in \mathbb{R}^{n^k}$, $s(z) = \sum_{i_1 \ldots i_k = 1}^n z_{i_1 \ldots i_k}$. Note that M^k, s are equivariant, invariant (respectively) for all $G \le S_n$. This construction can be visualized using the following diagram:

$$\mathbb{R}^n \xrightarrow{L^\tau} \mathbb{R}^{n^k \times k} \xrightarrow{M^k} \mathbb{R}^{n^k} \xrightarrow{s} \mathbb{R}$$

[5] In fact, any application of an MLP to the feature dimension is G-equivariant for any $G \le S_n$ since it can be realized by scaling of the identity operator, possibly with a constant and non-linear point-wise activations (see *e.g.*[197, 264]).

This G-invariant network $F^\tau = s \circ M^k \circ L^\tau$ approximates p^τ to an ϵ-accuracy over the compact set $K \subset \mathbb{R}^n$. Indeed, let $x \in K$, then

$$\begin{aligned}
&|F^\tau(x) - p^\tau(x)| \\
&\leq \sum_{i_1 \dots i_k = 1}^{n} \left| M^k(L^\tau(x))_{i_1 \dots i_k} - W^\tau_{i_1 \dots i_k} x_{i_1} \cdots x_{i_k} \right| \\
&\leq \sum_{i_1 \dots i_k = 1}^{n} \begin{cases} \left| m^k(x_{i_1}, \dots, x_{i_k}) - x_{i_1} \cdots x_{i_k} \right| & {\scriptstyle (i_1, \dots, i_k) \in \tau} \\ 0 & \text{otherwise} \end{cases} \\
&\leq \epsilon,
\end{aligned}$$

where in the last inequality we used the $n^{-k}\epsilon$-accuracy of m^k to the product operator in $[-c, c]^k \subset \mathbb{R}^k$.

□

Part (iv): In the final stage, we would like to approximate an arbitrary $p \in \mathbb{R}[x_1, \dots, x_n]^G$ with a G-invariant network to ϵ-accuracy over a compact set $K \subset \mathbb{R}^n$.

Proof. (proposition 5) Let us denote by $b_{k1}, \dots, b_{kn_k}$ the polynomials p^τ, with τ the k-classes. Let F^{kj} denote the G-invariant network approximating b_{kj}, $k = 0, 1, \dots d$, $j \in [n_k]$, to an ϵ-accuracy over the set K, the existence of which is guaranteed by proposition 9. We now utilize the decomposition of p shown in (51) and get

$$\begin{aligned}
&\left| p(x) - \sum_{k=0}^{d} \sum_{j=1}^{n_k} \alpha_{kj} F^{kj}(x) \right| \\
&\leq \sum_{k=0}^{d} \sum_{j=1}^{n_k} |\alpha_{kj}| \left| b_{kj}(x) - F^{kj}(x) \right| \\
&\leq \epsilon \left\| \alpha \right\|_1,
\end{aligned}$$

where $\|\alpha\|_1 = \sum_{k,j} |\alpha_{kj}|$ depends only upon p, where ϵ is arbitrary. To finish the proof we need to show that $F = \sum_{k=0}^{d} \sum_{j=1}^{n_k} \alpha_{kj} F^{kj}$ can indeed be realized as a *single, unified* G-invariant network. This is a simple yet technical construction and we defer the proof of this fact to the supplementary material:

Lemma 7. *There exists a G-invariant network in the sense of definition 3 that realizes the sum of G-invariant networks $F = \sum_{k=0}^{d} \sum_{j=1}^{n_k} \alpha_{kj} F^{kj}$.*

□

8.3.2 Bounded order construction

We have constructed a G-invariant network F that approximates an arbitrary G-invariant polynomial $p \in \mathbb{R}[x_1, \dots, x_n]^G$ of degree d. The network F uses d-dimensional tensors, where d matches the degree of p. In this subsection we construct a G-invariant network F that approximates p with maximal tensor order that depends only on the group $G \leq S_n$. Therefore, the tensor order is independent of the degree of the polynomial p. We use the following theorem by Noether [134]:

Theorem 10. ***(Noether)*** *Let G be a finite group acting linearly on $\mathcal{R}^n$. There exist finitely many G-invariant polynomials $f_1, ..., f_m \in \mathbb{R}[x_1, \ldots, x_n]^G$ such that any invariant polynomial $p \in \mathbb{R}[x_1, \ldots, x_n]^G$ can be expressed as*

$$p(x) = h(f_1(x), ..., f_m(x)),$$

where $h \in \mathbb{R}[x_1, \ldots, x_m]$ is a polynomial and $\deg(f_i) \leq |G|$, $i = 1, \ldots, m$.

The idea of using a set of generating invariant polynomials in the context of universality was introduced in [259].

For the case of interest in this section, namely $G \leq S_n$, there exists a generating set of G-invariant polynomials of degree bounded by $\frac{n(n-1)}{2}$, for $n \geq 3$, see [92]. We can now repeat the construction above, building a G-invariant network F_i approximating f_i to a ϵ_1-accuracy, $i = 1, \ldots, m$. The maximal order of these networks is bounded by $d \leq \frac{n(n-1)}{2}$. These networks can be combined, as above, to a single G-invariant network $F : \mathbb{R}^n \to \mathbb{R}^m$ with the final output approximating $f(x) = (f_1(x), \ldots, f_m(x))$ to a ϵ_1-accuracy. Now we compose the output of F with an MLP $H : \mathbb{R}^m \to \mathbb{R}$ approximating the polynomial h over the compact set $f(K) + B_\epsilon \subset \mathbb{R}^m$ to an ϵ-accuracy, where B_ϵ is a closed ball centered at the origin of radius ϵ and the sum is the Minkowski sum. Since H is continuous and $f(K) + B_\epsilon$ is compact, there exists $\delta > 0$ so that $|H(y) - H(y')| \leq \epsilon$ if $\|y - y'\|_2 \leq \delta$. We use $\epsilon_1 = \min\{\delta, \varepsilon\}$ for the construction of F above. We have:

$$\begin{aligned} |H(F(x)) - h(f(x))| &\leq |H(F(x)) - H(f(x))| \\ &\quad + |H(f(x)) - h(f(x))| \leq 2\epsilon, \end{aligned}$$

for all $x \in K$. We have constructed $H \circ F$ that is a G-invariant network with maximal tensor order bounded by $\frac{n(n-1)}{2}$ approximating p to an arbitrary precision.

8.3.3 Examples

Universality of (hyper-) graph networks. Graph, or hyper-graph data can be described using tensors $X \in \mathbb{R}^{n^k \times a}$, where n is the number of vertices of the graph and $x_{i_1,i_2,\ldots,i_k,:} \in \mathbb{R}^a$ is a feature vector attached to a (generalized-)edge defined by the ordered set of vertices $(i_1, i_2, \ldots, i_k)$. For example, an adjacency matrix of an n-vertex graph is described by $X \in \mathbb{R}^{n^2}$. The graph symmetries are reordering the vertices by a permutation, namely $g \cdot X$, where $g \in S_n$. Typically, any function we would like to learn on graphs would be invariant to this action. Recently, [158] characterized the spaces of equivariant and invariant linear layers with this symmetry, provided a formula for their basis and employed the corresponding G-invariant networks for learning graph-related tasks. A corollary of Theorem 5 is that this construction yields a universal approximator of continuous functions defined on graphs. This is in contrast to the popular *message passing neural network* model [89] that was recently shown to be non-universal [256].

Universality of rotation invariant convolutional networks. For learning tasks involving $m \times m$ images one might require invariance to periodic translations and 90 degree rotations. Note that periodic translations and 90 degree rotations can be seen as permutations in S_n, $n = m^2$, acting on the pixels of the image. Constructing a suitable G-invariant network would lead, according to Theorem 5, to a universal approximator.

8.4 A lower bound on equivariant layer order

In the previous section we showed how an arbitrary G-invariant polynomial can be approximated with a G-invariant network with tensor order $d = d(G) \leq \frac{n(n-1)}{2}$. This upper-bound would be prohibitive in

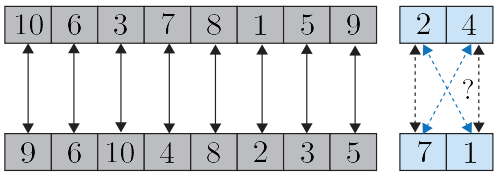


Figure 30: Illustration of the $(n-2)$-transitivity of A_n, the main property we use in this section. Any subset of distinct $n-2$ elements can be mapped to any other subset of distinct $n-2$ elements (gray). If needed, a transposition can be applied to the remaining 2 elements (blue) to assure an even permutation.

practice. In this section we prove a *lower bound*: We show that there exists a group for which the tensor order cannot be less than $\frac{n-2}{2}$ if we wish to maintain the universal approximation property.

We consider the alternating group, $G = A_n \leq S_n$. Remember that $g \in A_n$ if g has an even number of transpositions.

Definition 5. *A group $G \leq S_n$ is k-transitive if for every two sequences $(i_1, i_2, \ldots, i_k)$, $(j_1, j_2, \ldots, j_k)$ of distinct elements in $[n]$ there exists $g \in G$ so that $j_\ell = g(i_\ell)$, for $\ell = 1, \ldots, k$.*

The alternating group is $(n-2)$-transitive (see figure 30 and [66]). Our goal is to prove:

Theorem 8. *If an A_n-invariant network has the universal approximation property, then it consists of tensors of order at least $\frac{n-2}{2}$.*

For the proof we first need a characterization of the linear equivariant layers $L : \mathbb{R}^{n^k \times a} \to \mathbb{R}^{n^l \times b}$, where $l = 0$ represents the invariant case. By definition $L(g \cdot X) = g \cdot L(X)$ for all $X \in \mathbb{R}^{n^k \times a}$. In particular this means that

$$g^{-1} \cdot L(g \cdot X) = L(X)$$

Recall that L is an *affine map* (see definition 2) and therefore can be represented as a sum of a purely linear part and a constant part. Representing the linear part of L as a tensor $L \in \mathbb{R}^{n^{k+l} \times a \times b}$ these equations become the fixed-point equation for linear equivariant layers (see supplementary material for derivation):

$$g \cdot L = L, \; g \in G. \tag{57}$$

The constant part of L can be encoded using a tensor $B \in \mathbb{R}^{n^l \times b}$ that satisfies (57) as-well. Note the that this fixed point equation is similar to the fixed point equation of homogeneous G-invariant polynomials, (53). We denote by $\mathcal{L}^G$ the collection of $L : \mathbb{R}^{n^k \times a} \to \mathbb{R}^{n^l \times b}$ linear G-equivariant ($l > 0$) or G-invariant ($l = 0$) layers.

Proposition 10. *If $k + l \leq n - 2$, then $\mathcal{L}^{A_n} = \mathcal{L}^{S_n}$.*

Proof. In view of the fixed point equation for equivariant/invariant layers (57) we need to show the solution set to this equation is identical for $G = A_n$ and $G = S_n$, as long as $k + l \leq n - 2$. The solution set of the fixed point equation consists of tensors L that are constant on each equivalence class defined by the equivalence relation: $(i_1, \ldots, i_{k+l}) \sim (j_1, \ldots, j_{k+l})$ if $j_\ell = g(i_\ell)$ for $\ell = 1, \ldots, k + l$.

Both A_n and S_n are $(n-2)$-transitive[6]. Therefore, the equivalence relations defined above for A_n and S_n reduce to the *same* equivalence relation $(i_1, \ldots, i_{k+l}) \sim (j_1, \ldots, j_{k+l})$ if $i_\alpha = i_\beta$ if and only if $j_\alpha = j_\beta$, for all $\alpha, \beta \in [k + l]$ (see [158] where these classes are called *equality patterns*). Since this equivalence

[6] S_n is in-fact n-transitive and is therefore also k-transitive for all $k \leq n$.

relation is the same for A_n, S_n, we get that the solution set of the fixed point equation (57) is the same for both groups. Since the constant part tensor B is of smaller order than $k+l \leq n-2$, the same argumentation applies to the constant part, as-well. □

Proposition 10 implies that any A_n-invariant network with tensor order $\leq (n-2)/2$ will be in fact S_n-invariant. Therefore, one approach to show that such networks have limited approximation power is to come up with an A_n-invariant continuous function that is *not* S_n-invariant, as follows:

Proof. (Theorem 8) Consider the Vandermonde polynomial $V(x) = \prod_{1 \leq i < j \leq n}(x_i - x_j)$. It is not hard to check that V is A_n-invariant but not S_n-invariant (consider, *e.g.*, $g = (12) \in S_n$). Pick $x \in \mathcal{R}^n$ with distinct coordinates. Then it holds that $V(x) \neq 0$. Let $\epsilon > 0$ and $K \subset \mathbb{R}^n$ a compact set containing both $x, g \cdot x$ for $g = (12)$. Assume by way of contradiction that there exists an A_n-invariant network F, which is S_n-invariant due to the above, such that $|V(x) - F(x)| \leq \epsilon$ as well as:

$$\begin{aligned} |V(g \cdot x) - F(g \cdot x)| &= |(-1)V(x) - F(x)| \\ &= |V(x) + F(x)| \\ &\leq \epsilon \end{aligned}$$

These last equations imply that $|V(x)| \leq \epsilon$ and since ϵ is arbitrary we get $V(x) = 0$, a contradiction. □

8.5 Universality of first order networks

We have seen that G-invariant networks with tensor order $\frac{n(n-1)}{2}$ are universal. On the other hand for general permutation groups G the tensor order is at least $(n-2)/2$ if universality is required. A particularly important question for applications, where higher order tensors are computationally prohibitive, is which permutation groups G give rise to *first order G-invariant networks* that are universal.

Definition 6. *A first order G-invariant network is a G-invariant network where the maximal tensor order is* 1.

In this section we discuss this (mostly) open question. First, we note that there are a few cases for which first order G-invariant networks are known to be universal: for instance, when $G = \{e\}$ (*i.e.*, the trivial group), G-invariant networks are composed of fully connected layers, a case which is covered by the original universal approximation theorems [59, 110]. First order universality is also known when G is (possibly high dimensional) grid (*e.g.*, $G = Z_{n_1} \times \cdots \times Z_{n_k}$) [259], a case that includes periodic convolutional neural networks. Universality of first order networks is also known when $G = S_n$ [264, 197, 259] in the context of invariant networks that operate on sets or point clouds.

Our goal in this section is to derive a necessary condition on G for the universality of first order G-invariant networks. To this end, we first find a function, playing the role of the Vandermonde polynomial in the previous section, that is G-invariant but not H-invariant, where $G < H \leq S_n$.

Lemma 8. *Let $G < H \leq S_n$. Then there exists a continuous function $f : \mathcal{R}^n \to \mathcal{R}$ which is G-invariant but not H-invariant.*

Proof. Pick a point $x_0 \in \mathcal{R}^n$ with distinct coordinates. Since the stabilizer $(S_n)_{x_0}$ is trivial (*i.e.*, no permutation fixes x_0 excluding the identity), the size of the orbits of x_0 equals the size of the acting group. Namely, $|G \cdot x_0| = |G|$ and $|H \cdot x_0| = |H|$. Furthermore, since $|G| < |H|$ and $G \cdot x_0 \subset H \cdot x_0$, we get that the H orbit strictly includes the G orbit. That is, $G \cdot x \subsetneq H \cdot x$. Since $H \cdot x_0$ is a finite set of points, there exists a

continuous function $\hat{f}$ such that $\hat{f}|_{G\cdot x_0} = 1$, and $\hat{f}|_{H\cdot x_0 \setminus G\cdot x_0} = 0$. Define $f(x) = \frac{1}{|G|}\sum_{g\in G}\hat{f}(g\cdot x)$. Now, f is G-invariant by construction but $f(x_0) = 1$ and $f(h\cdot x_0) = 0$ for $h\cdot x_0 \in H\cdot x_0 \setminus G\cdot x_0$. Therefore, f is not H-invariant. □

In case of first order G-invariant networks the equivariant/invariant layers have the form $L : \mathbb{R}^{n\times a} \to \mathbb{R}^{n\times b}$ and satisfy the fixed point equations (57). The solution set of the purely linear equivariant layers consists of tensors $L \in \mathbb{R}^{n^2\times a\times b}$ that are constant on equivalence classes of indices defined by the equivalence relation $(i_1, i_2) \sim (j_1, j_2)$ if there exists $g \in G$ so that $j_\ell = g(i_\ell)$, $\ell = 1, 2$. We denote the number of equivalence classes by $|[n]^2/G|$. The solution set of constant equivariant operators are tensors $B \in \mathbb{R}^{n\times b}$ that are constant on equivalence classes defined by the equivalence relation $i \sim j$ if there exists $g \in G$ so that $j = g(i)$. We denote the number of these classes by $|[n]/G|$. We prove:

Theorem 9. *Let $G \leq S_n$. If first order G-invariant networks are universal, then $\left|[n]^2/H\right| < \left|[n]^2/G\right|$ for any strict super-group $G < H \leq S_n$.*

Proof. Assume by contradiction that there exists a strict super-group $G < H \leq S_n$ so that $\left|[n]^2/G\right| = \left|[n]^2/H\right|$. This in particular means that $|[n]/G| = |[n]/H|$. Therefore $L^G = L^H$. That is, the spaces of equivariant and invariant linear layers coincide for G and H. This implies, as before, that every first order G-invariant network is also H-invariant.

We proceed similarly to the proof of theorem 8: By lemma 8, there exists a continuous function $f : \mathbb{R}^n \to \mathbb{R}$ that is G-invariant but not H-invariant. Let x_0 be a point with distinct coordinates where $f(x_0) = 1$ (it exists by construction, see proof of theorem 8). Furthermore, by construction $f(h\cdot x_0) = 0$ if $h\cdot x_0 \in H\cdot x_0 \setminus G\cdot x_0$.

Let $\epsilon > 0$ and $K \subset \mathbb{R}^n$ a compact set containing both $x_0, h\cdot x_0$. Assume by way of contradiction that there exists a first order G-invariant network F (which is also H-invariant in view of the above) such that $|f(x_0) - F(x_0)| \leq \epsilon$ as well as:

$$|f(h\cdot x_0) - F(h\cdot x_0)| = |f(h\cdot x_0) - F(x_0)| \leq \epsilon.$$

These last equations imply that $1 = |f(x_0) - f(h\cdot x_0)| \leq |f(x_0) - F(x_0)| + |F(x_0) - f(h\cdot x_0)| \leq 2\epsilon$ and since ϵ is arbitrary we get a contradiction. □

Using theorem 9 we can show that there exist a few infinite families of permutation groups (excluding the alternating group A_n) for which first order invariant networks are not universal. For example, any strict subgroup $G < S_n$ that is 2-transitive is such a group since in this case $\left|[n]^2/G\right| = \left|[n]^2/S_n\right|$ and consequently G-invariant/equivariant layers are also S_n-invariant/equivariant. Examples of 2-transitive permutation groups include projective linear groups over finite fields $PSL_d(F_q)$ (for $q = p^n$ where $p, n \in N$, p is prime) that act on the finite projective space, and can be seen as a subgroup of S_n for $n = (q^d - 1)/(q - 1)$ (the number of elements in this space). Similarly affine subgroups over finite fields $A\Gamma L_d(F_q)$ that act on F_q^d can be shown to be 2-transitive as a subgroup of S_n for $n = q^d$. See [66] for a full classification of 2-transitive subgroups of S_n.

Relation to [203]. Groups for which the condition in theorem 9 holds are called 2-closed and were first introduced by [250] (see [17] for further study). Theorem 9 reveals an interesting connection between our work and the work of [203] that studies parameter sharing schemes. One of the basic notions defined in their paper is the notion of *uniquely G-equivariant functions*, which describes functions that are G-equivariant but not equivariant to any super-group of G. For example, a consequence of proposition 10 is that $A_n \leq S_n$ (with the representation used in this section) has no uniquely equivariant linear functions between tensors

of total order $\leq n-2$. It was shown in [203] that 2-closed groups are exactly the groups for which one can find a uniquely equivariant function. In this section we proved that the existence of a uniquely G-equivariant linear function is a necessary condition for first order universality. As stated in [203] some examples for 2-closed groups are fixed-point free groups (*e.g.*, the cyclic group C_n) and S_n itself.

8.6 Conclusion

In this section we have considered the universal approximation property of a popular invariant neural network model. We have shown that these networks are universal with a construction that uses tensors of order $\leq \frac{n(n-1)}{2}$, which makes this architecture impractical. On the other hand, there exists a permutation group for which we have proved a lower bound of $\frac{n-2}{2}$ on the tensor order required to achieve universality. We then addressed the more practical question of which groups G allow first order G-invariant networks to be universal. We have proved that 2-closedness of G is a necessary condition, and gave examples of infinite permutation group families that do not satisfy this condition.

Our work is a first step in advancing the understanding of approximation power of a large class of invariant neural networks that becomes increasingly popular in applications. Several questions remain open: First, a classification of 2-closed groups will give us a complete answer to which networks are first-order universal. As far as we know this is an open question in group theory. Still, mapping the 2-closed landscape for specific groups G that are interesting for machine learning applications is a worthy challenge. Second, In case one wishes to construct a G-invariant network for a group G that is not 2-closed, developing fast and efficient implementations of higher order layers seems like a potentially useful direction. Lastly, another interesting venue for future work might be to come up with new, possibly non-linear, models for invariant networks.

8.7 Proofs

Lemma 7. *There exists a G-invariant network in the sense of definition 3 that realizes the sum of G-invariant networks $F = \sum_{k=0}^{d} \sum_{j=1}^{n_k} \alpha_{kj} F^{kj}$.*

Proof. We need to show that $F = \sum_{k=0}^{d} \sum_{j=1}^{n_k} \alpha_{kj} F^{kj}$ can indeed be realized as a *single, unified* G-invariant network. As we already saw, each network F^{kj} has the structure

$$\mathbb{R}^n \xrightarrow{L^\tau} \mathbb{R}^{n^k \times k} \xrightarrow{M^k} \mathbb{R}^{n^k} \xrightarrow{s} \mathbb{R},$$

with a suitable k-class τ. To create the unified G-invariant network we first lift each F^{kj} to the maximal dimension d. That is, $\widetilde{F}^{kj}$ with the structure

$$\mathbb{R}^n \xrightarrow{\widetilde{L}^{kj}} \mathbb{R}^{n^d \times k} \xrightarrow{\widetilde{M}^k} \mathbb{R}^{n^d} \xrightarrow{s} \mathbb{R}.$$

This is done by composing each equivariant layer $L : \mathbb{R}^{n^k \times a} \to \mathbb{R}^{n^l \times b}$ with two linear equivariant operators $U^b : \mathbb{R}^{n^k \times b} \to \mathbb{R}^{n^d \times b}$ and $D^a : \mathbb{R}^{n^d \times a} \to \mathbb{R}^{n^k \times a}$,

$$U^b L D^a : \mathbb{R}^{n^d \times a} \to \mathbb{R}^{n^d \times b}, \tag{58}$$

where

$$U^b(x)_{i_1 \dots i_d, j} = x_{i_1 \dots i_k, j}$$

and

$$D^a(y)_{i_1 \dots i_k, j} = n^{k-d} \sum_{i_{k+1} \dots i_d = 1}^{n} y_{i_1 \dots i_k i_{k+1} \dots i_d, j} \ .$$

Since U^b, D^a are equivariant, $U^b L D^a$ in (58) is equivariant. Furthermore $D^a \circ \sigma \circ U^a = \sigma$, where σ is the pointwise activation function. Lastly, given two G-invariant networks with the same tensor order d they can be combined to a single G-invariant network by concatenating their features. That is, if $L_1 : \mathbb{R}^{n^d \times a} \to \mathbb{R}^{n^d \times b}$, and $L_2 : \mathbb{R}^{n^d \times a'} \to \mathbb{R}^{n^d \times b'}$, then their concatenation would yield $L_{1,2} : \mathbb{R}^{n^d \times (a+a')} \to \mathbb{R}^{n^d \times (b+b')}$. Applying this concatenation to all $\widetilde{F}^{kj}$ we get our unified G-invariant network. □

Fixed-point equation for equivariant layers. We have an affine operator $L : \mathbb{R}^{n^k \times a} \to \mathbb{R}^{n^l \times b}$ satisfying

$$g^{-1} \cdot L(g \cdot X) = L(X), \tag{59}$$

for all $g \in G$, $X \in \mathbb{R}^{n^k \times a}$. The purely linear part of L can be written using a tensor $L \in \mathbb{R}^{n^{k+l} \times a \times b}$: Write

$$L(X)_{j_1 \dots j_l, j} = \sum_{i_1 \dots i_k, i} L_{j_1 \dots j_l, i_1 \dots i_k, i, j} X_{i_1 \dots i_k, i}.$$

Writing (59) using this notation gives:

$$\begin{aligned}
&\sum_{i_1 \dots i_k, i} L_{g(j_1) \dots g(j_l), i_1 \dots i_k, i, j} X_{g^{-1}(i_1) \dots g^{-1}(i_k), i} \\
&= \sum_{i_1 \dots i_k, i} L_{g(j_1) \dots g(j_l), g(i_1) \dots g(i_k), i, j} X_{i_1 \dots i_k, i} \\
&= \sum_{i_1 \dots i_k, i} L_{j_1 \dots j_l, i_1 \dots i_k, i, j} X_{i_1 \dots i_k, i},
\end{aligned}$$

for all $g \in G$ and $X \in \mathbb{R}^{n^k \times a}$. This implies (57), namely

$$g \cdot L = L, \ g \in G.$$

The constant part of L is done similarly.

9 Provably powerful graph networks

This section is based on [161].

9.1 Introduction

Graphs are an important data modality which is frequently used in many fields of science and engineering. Among other things, graphs are used to model social networks, chemical compounds, biological structures and high-level image content information. One of the major tasks in graph data analysis is learning from graph data. As classical approaches often use hand-crafted graph features that are not necessarily suitable to all datasets and/or tasks (*e.g.*, [135]), a significant research effort in recent years is to develop deep models that are able to learn new graph representations from raw features (*e.g.*, [93, 69, 181, 128, 236, 172, 102, 175, 256]).

Currently, the most popular methods for deep learning on graphs are *message passing neural networks* in which the node features are propagated through the graph according to its connectivity structure [89]. In a successful attempt to quantify the expressive power of message passing models, [175, 256] suggest to compare the model's ability to *distinguish* between two given graphs to that of the hierarchy of the Weisfeiler-Lehman (WL) graph isomorphism tests [95, 18]. Remarkably, they show that the class of message passing models has limited expressiveness and is not better than the first WL test (1-WL, a.k.a. color refinement). For example, Figure 31 depicts two simple graphs that 1-WL cannot distinguish, hence indistinguishable by any message passing algorithm.

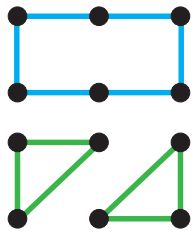


Figure 31: Two graphs not distinguished by 1-WL.

The goal of this work is to explore and develop GNN models that possess higher expressiveness while maintaining scalability, as much as possible. We present two main contributions. First, establishing a baseline for expressive GNNs, we prove that the recent k-order invariant GNNs [158, 159] offer a natural hierarchy of models that are as expressive as the k-WL tests, for $k \geq 2$. Second, as k-order GNNs are not practical for $k > 2$ we develop a simple, novel GNN model, that incorporates standard MLPs of the feature dimension and a matrix multiplication layer. This model, working only with $k = 2$ tensors (the same dimension as the graph input data), possesses the expressiveness of 3-WL. Since, in the WL hierarchy, 1-WL and 2-WL are equivalent, while 3-WL is strictly stronger, this model is provably more powerful than the message passing models. For example, it can distinguish the two graphs in Figure 31. As far as we know, this model is the first to offer both expressiveness (3-WL) and scalability ($k = 2$).

The main challenge in achieving high-order WL expressiveness with GNN models stems from the difficulty to represent the multisets of neighborhoods required for the WL algorithms. We advocate a novel representation of multisets based on Power-sum Multi-symmetric Polynomials (PMP) which are a generalization of the well-known elementary symmetric polynomials. This representation provides a convenient theoretical tool to analyze models' ability to implement the WL tests.

A related work to ours that also tried to build graph learning methods that surpass the 1-WL expressiveness offered by message passing is [175]. They develop powerful deep models generalizing message passing to higher orders that are as expressive as higher order WL tests. Although making progress, their full model is still computationally prohibitive for 3-WL expressiveness and requires a relaxed local version compromising some of the theoretical guarantees.

Experimenting with our model on several real-world datasets that include classification and regression tasks on social networks, molecules, and chemical compounds, we found it to be on par or better than state of the art.

9.2 Previous work

Deep learning on graph data. The pioneering works that applied neural networks to graphs are [93, 213] that learn node representations using recurrent neural networks, which were also used in [146]. Following the success of convolutional neural networks [136], many works have tried to generalize the notion of convolution to graphs and build networks that are based on this operation. [42] defined graph convolutions as operators that are diagonal in the graph laplacian eigenbasis. This paper resulted in multiple follow up works with more efficient and spatially localized convolutions [108, 64, 128, 144]. Other works define graph convolutions as local stationary functions that are applied to each node and its neighbours (*e.g.*, [69, 13, 181, 103, 236, 170]). Many of these works were shown to be instances of the family of message passing neural networks [89]: methods that apply parametric functions to a node and its neighborhood and then apply some pooling operation in order to generate a new feature for each node. In a recent line of work, it was suggested to define graph neural networks using permutation equivariant operators on tensors describing k-order relations between the nodes. [132] identified several such linear and quadratic equivariant operators and showed that the resulting network can achieve excellent results on popular graph learning benchmarks. [158] provided a full characterization of linear equivariant operators between tensors of arbitrary order. In both cases, the resulting networks were shown to be at least as powerful as message passing neural networks. In another line of work, [177] suggest expressive invariant graph models defined using averaging over all permutations of an arbitrary base neural network.

Weisfeiler Lehman graph isomorphism test. The Weisfeiler Lehman tests is a hierarchy of increasingly powerful graph isomorphism tests [95]. The WL tests have found many applications in machine learning: in addition to [256, 175], This idea was used in [219] to construct a graph kernel method, which was further generalized to higher order WL tests in [173]. [142] showed that their suggested GNN has a theoretical connection to the WL test. WL tests were also used in [267] for link prediction tasks. In a concurrent work, [174] suggest constructing graph features based on an equivalent sparse version of high-order WL achieving great speedup and expressiveness guarantees for sparsely connected graphs.

9.3 Preliminaries

We denote a set by $\{a, b, \ldots, c\}$, an ordered set (tuple) by $(a, b, \ldots, c)$ and a multiset (*i.e.*, a set with possibly repeating elements) by $\{\!\{a, b, \ldots, c\}\!\}$. We denote $[n] = \{1, 2, \ldots, n\}$, and $(a_i \mid i \in [n]) = (a_1, a_2, \ldots, a_n)$. Let S_n denote the permutation group on n elements. We use multi-index $i \in [n]^k$ to denote a k-tuple of indices, $i = (i_1, i_2, \ldots, i_k)$. $g \in S_n$ acts on multi-indices $i \in [n]^k$ entrywise by $g(i) = (g(i_1), g(i_2), \ldots, g(i_k))$. S_n acts on k-tensors $X \in \mathbb{R}^{n^k \times a}$ by $(g \cdot X)_{i,j} = X_{g^{-1}(i),j}$, where $i \in [n]^k$, $j \in [a]$.

9.3.1 k-order graph networks

[158] have suggested a family of permutation-invariant deep neural network models for graphs. Their main idea is to construct networks by concatenating maximally expressive linear equivariant layers. More formally, a k-order invariant graph network is a composition $F = m \circ h \circ L_d \circ \sigma \circ \cdots \circ \sigma \circ L_1$, where $L_i : \mathbb{R}^{n^{k_i} \times a_i} \to \mathbb{R}^{n^{k_{i+1}} \times a_{i+1}}$, $\max_{i \in [d+1]} k_i = k$, are *equivariant linear layers*, namely satisfy

$$L_i(g \cdot X) = g \cdot L_i(X), \qquad \forall g \in S_n, \quad \forall X \in \mathbb{R}^{n^{k_i} \times a_i},$$

σ is an entrywise non-linear activation, $\sigma(X)_{i,j} = \sigma(X_{i,j})$, $h : \mathbb{R}^{n^{k_{d+1}} \times a_{d+1}} \to \mathbb{R}^{a_{d+2}}$ is an *invariant linear layer*, namely satisfies

$$h(g \cdot X) = h(X), \qquad \forall g \in S_n, \quad \forall X \in \mathbb{R}^{n^{k_{d+1}} \times a_{d+1}},$$

and m is a Multilayer Perceptron (MLP). The invariance of F is achieved by construction (by propagating g through the layers using the definitions of equivariance and invariance):

$$F(g \cdot X) = m(\cdots(L_1(g \cdot X))\cdots) = m(\cdots(g \cdot L_1(X))\cdots) = \cdots = m(h(g \cdot L_d(\cdots))) = F(X).$$

When $k = 2$, [158] proved that this construction gives rise to a model that can approximate any message passing neural network [89] to an arbitrary precision; [159] proved that these models are universal for a very high tensor order of $k = \binom{n}{2}$, which is of little practical value (an alternative proof was recently suggested in [123]).

9.3.2 The Weisfeiler-Lehman graph isomorphism test

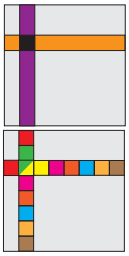
Let $G = (V, E, d)$ be a colored graph where $|V| = n$ and $d : V \to \Sigma$ defines the color attached to each vertex in V, Σ is a set of colors. The Weisfeiler-Lehman (WL) test is a family of algorithms used to test graph isomorphism. Two graphs G, G' are called isomorphic if there exists an edge and color preserving bijection $\phi : V \to V'$.

There are two families of WL algorithms: k-WL and k-FWL (Folklore WL), both parameterized by $k = 1, 2, \ldots, n$. k-WL and k-FWL both construct a coloring of k-tuples of vertices, that is $c : V^k \to \Sigma$. Testing isomorphism of two graphs G, G' is then performed by comparing the histograms of colors produced by the k-WL (or k-FWL) algorithms.

We will represent coloring of K-tuples using a tensor $C \in \Sigma^{n^k}$, where $C_i \in \Sigma$, $i \in [n]^k$ denotes the color of the k-tuple $v_i = (v_{i_1}, \ldots, v_{i_k}) \in V^k$. In both algorithms, the initial coloring C^0 is defined using the *isomorphism type* of each k-tuple. That is, two k-tuples i, i' have the same isomorphism type (*i.e.*, get the same color, $C_i = C_{i'}$) if for all $q, r \in [k]$: (i) $v_{i_q} = v_{i_r} \iff v_{i'_q} = v_{i'_r}$; (ii) $d(v_{i_q}) = d(v_{i'_q})$; and (iii) $(v_{i_r}, v_{i_q}) \in E \iff (v_{i'_r}, v_{i'_q}) \in E$. Clearly, if G, G' are two isomorphic graphs then there exists $g \in S_n$ so that $g \cdot C'^0 = C^0$.

In the next steps, the algorithms refine the colorings C^l, $l = 1, 2, \ldots$ until the coloring does not change further, that is, the subsets of k-tuples with same colors do not get further split to different color groups. It is guaranteed that no more than $l = poly(n)$ iterations are required [68].

The construction of C^l from C^{l-1} differs in the WL and FWL versions. The difference is in how the colors are aggregated from neighboring k-tuples. We define two notions of neighborhoods of a k-tuple $i \in [n]^k$:



$$N_j(i) = \left\{ (i_1, \ldots, i_{j-1}, i', i_{j+1}, \ldots, i_k) \;\middle|\; i' \in [n] \right\} \tag{60}$$

$$N_j^F(i) = \Big((j, i_2, \ldots, i_k), (i_1, j, \ldots, i_k), \ldots, (i_1, \ldots, i_{k-1}, j) \Big) \tag{61}$$

$N_j(i)$, $j \in [k]$ is the j-th neighborhood of the tuple i used by the WL algorithm, while $N_j^F(i)$, $j \in [n]$ is the j-th neighborhood used by the FWL algorithm. Note that $N_j(i)$ is a set of n k-tuples, while $N_j^F(i)$ is an ordered set of k k-tuples. The inset to the right illustrates these notions of neighborhoods for the case $k = 2$: the top figure shows $N_1(3, 2)$ in purple and $N_2(3, 2)$ in orange. The bottom figure shows $N_j^F(3, 2)$ for all $j = 1, \ldots, n$ with different colors for different j.

The coloring update rules are:

$$\text{WL:} \quad C^l_i = \text{enc}\Big(C^{l-1}_i, \Big(\{\!\{ C^{l-1}_j \mid j \in N_j(i) \}\!\} \,\Big|\, j \in [k] \Big) \Big) \tag{62}$$

$$\text{FWL:} \quad C^l_i = \text{enc}\Big(C^{l-1}_i, \Big\{\!\!\Big\{ \big(C^{l-1}_j \mid j \in N^F_j(i)\big) \,\Big|\, j \in [n] \Big\}\!\!\Big\} \Big) \tag{63}$$

where enc is a bijective map from the collection of all possible tuples in the r.h.s. of Equations ((62))-((63)) to Σ.

When $k = 1$ both rules, ((62))-((63)), degenerate to $C^l_i = \text{enc}\Big(C^{l-1}_i, \{\!\{C^{l-1}_j \mid j \in [n]\}\!\}\Big)$, which will not refine any initial color. Traditionally, the first algorithm in the WL hierarchy is called WL, 1-WL, or the *color refinement algorithm*. In color refinement, one starts with the coloring prescribed with d. Then, in each iteration, the color at each vertex is refined by a new color representing its current color and the multiset of its neighbors' colors.

Several known results of WL and FWL algorithms [46, 95, 175, 96] are:

1. 1-WL and 2-WL have equivalent discrimination power.
2. k-FWL is equivalent to $(k+1)$-WL for $k \geq 2$.
3. For each $k \geq 2$ there is a pair of non-isomorphic graphs distinguishable by $(k+1)$-WL but not by k-WL.

9.4 Colors and multisets in networks

Before we get to the two main contributions of this section we address three challenges that arise when analyzing networks' ability to implement WL-like algorithms: (i) Representing the colors Σ in the network; (ii) implementing a multiset representation; and (iii) implementing the encoding function.

Color representation. We will represent colors as vectors. That is, we will use tensors $C \in \mathbb{R}^{n^k \times a}$ to encode a color per k-tuple; that is, the color of the tuple $i \in [n]^k$ is a vector $C_i \in \mathbb{R}^a$. This effectively replaces the color tensors Σ^{n^k} in the WL algorithm with $\mathbb{R}^{n^k \times a}$.

Multiset representation. A key technical part of our method is the way we encode multisets in networks. Since colors are represented as vectors in $\mathbb{R}^a$, an n-tuple of colors is represented by a matrix $X = [x_1, x_2, \ldots, x_n]^T \in \mathbb{R}^{n \times a}$, where $x_j \in \mathbb{R}^a$, $j \in [n]$ are the rows of X. Thinking about X as a multiset forces us to be indifferent to the order of rows. That is, the color representing $g \cdot X$ should be the same as the color representing X, for all $g \in S_n$. One possible approach is to perform some sort (*e.g.*, lexicographic) to the rows of X. Unfortunately, this seems challenging to implement with equivariant layers.

Instead, we suggest to encode a multiset X using a set of S_n-invariant functions called the *Power-sum Multi-symmetric Polynomials* (PMP) [36, 211]. The PMP are the multivariate analog to the more widely known *Power-sum Symmetric Polynomials*, $p_j(y) = \sum_{i=1}^n y_i^j$, $j \in [n]$, where $y \in \mathbb{R}^n$. They are defined next. Let $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_a) \in [n]^a$ be a multi-index and for $y \in \mathbb{R}^a$ we set $y^{\boldsymbol{\alpha}} = y_1^{\alpha_1} \cdot y_2^{\alpha_2} \cdots y_a^{\alpha_a}$. Furthermore, $|\boldsymbol{\alpha}| = \sum_{j=1}^a \alpha_j$. The PMP of degree $\boldsymbol{\alpha} \in [n]^a$ is

$$p_{\boldsymbol{\alpha}}(X) = \sum_{i=1}^n x_i^{\boldsymbol{\alpha}}, \quad X \in \mathbb{R}^{n \times a}.$$

A key property of the PMP is that the finite subset $p_{\boldsymbol{\alpha}}$, for $|\boldsymbol{\alpha}| \leq n$ generates the ring of *Multi-symmetric Polynomials* (MP), the set of polynomials q so that $q(g \cdot X) = q(X)$ for all $g \in S_n$, $X \in \mathbb{R}^{n \times a}$ (see, *e.g.*,

[211] corollary 8.4). The PMP generates the ring of MP in the sense that for an arbitrary MP q, there exists a polynomial r so that $q(X) = r\left(u(X)\right)$, where

$$u(X) := \left(p_{\boldsymbol{\alpha}}(X) \;\middle|\; |\boldsymbol{\alpha}| \leq n\right). \tag{64}$$

As the following proposition shows, a useful consequence of this property is that the vector $u(X)$ is a unique representation of the multi-set $X \in \mathbb{R}^{n\times a}$.

Proposition 11. *For arbitrary $X, X' \in \mathbb{R}^{n\times a}$: $\exists g \in S_n$ so that $X' = g \cdot X$ if and only if $u(X) = u(X')$.*

We note that Proposition 11 is a generalization of lemma 6 in [264] (that considers multisets of scalars) to the case of multisets of vectors. The full proof is provided in section 9.9.

Encoding function. One of the benefits in the vector representation of colors is that the encoding function can be implemented as a simple concatenation: Given two color tensors $C \in \mathbb{R}^{n^k\times a}$, $C' \in \mathbb{R}^{n^k\times b}$, the tensor that represents for each k-tuple i the color pair (C_i, C'_i) is simply $(C, C') \in \mathbb{R}^{n^k\times(a+b)}$.

9.5 k-order graph networks are as powerful as k-WL

Our goal in this section is to show that, for every $2 \leq k \leq n$, k-order graph networks [158] are at least as powerful as the k-WL graph isomorphism test in terms of distinguishing non-isometric graphs. This result is shown by constructing a k-order network model and learnable weight assignment that implements the k-WL test.

To motivate this construction we note that the WL update step, (62), is equivariant (see proof in Section 9.10). Namely, plugging in $g \cdot C^{l-1}$ the WL update step would yield $g \cdot C^l$. Therefore, it is plausible to try to implement the WL update step using linear equivariant layers and non-linear pointwise activations.

Theorem 11. *Given two graphs $G = (V, E, d)$, $G' = (V', E', d')$ that can be distinguished by the k-WL graph isomorphism test, there exists a k-order network F so that $F(G) \neq F(G')$. On the other direction for every two isomorphic graphs $G \cong G'$ and k-order network F, $F(G) = F(G')$.*

The full proof is provided in Section 9.11. Here we outline the basic idea for the proof. First, an input graph $G = (V, E, d)$ is represented using a tensor of the form $B \in \mathbb{R}^{n^2\times(e+1)}$, as follows. The last channel of B, namely $B_{:,:,e+1}$ (':' stands for all possible values $[n]$) encodes the adjacency matrix of G according to E. The first e channels $B_{:,:,1:e}$ are zero outside the diagonal, and $B_{i,i,1:e} = d(v_i) \in \mathbb{R}^e$ is the color of vertex $v_i \in V$.

Now, the second statement in Theorem 11 is clear since two isomorphic graphs G, G' will have tensor representations satisfying $B' = g \cdot B$ and therefore, as explained in Section 9.3.1, $F(B) = F(B')$.

More challenging is showing the other direction, namely that for non-isomorphic graphs G, G' that can be distinguished by the k-WL test, there exists a k-network distinguishing G and G'. The key idea is to show that a k-order network can encode the multisets $\{\!\{B_j \mid j \in N_j(i)\}\!\}$ for a given tensor $B \in \mathbb{R}^{n^k\times a}$. These multisets are the only non-trivial component in the WL update rule, (62). Note that the rows of the matrix $X = B_{i_1,\ldots,i_{j-1},:,i_{j+1},\ldots,i_k,:} \in \mathbb{R}^{n\times a}$ are the colors (*i.e.*, vectors) that define the multiset $\{\!\{B_j \mid j \in N_j(i)\}\!\}$. Following our multiset representation (Section 9.4) we would like the network to compute $u(X)$ and plug the result at the i-th entry of an output tensor C.

This can be done in two steps: First, applying the polynomial function $\tau : \mathbb{R}^a \to \mathbb{R}^b$, $b = \binom{n+a-1}{a-1}$ entrywise to B, where τ is defined by $\tau(x) = (x^{\boldsymbol{\alpha}} \mid |\boldsymbol{\alpha}| \leq n)$ (note that b is the number of multi-indices $\boldsymbol{\alpha}$

such that $|\boldsymbol{\alpha}| \leq n$). Denote the output of this step Y. Second, apply a linear equivariant operator summing over the j-the coordinate of Y to get C, that is

$$C_{i,:} := L_j(Y)_{i,:} = \sum_{i'=1}^{n} Y_{i_1,\cdots,i_{j-1},i',i_{j+1},\ldots,i_k,:} = \sum_{j \in N_j(i)} \tau(B_{j,:}) = u(X), \quad i \in [n]^k,$$

where $X = B_{i_1,\ldots,i_{j-1},:,i_{j+1},\ldots,i_k,:}$ as desired. Lastly, we use the universal approximation theorem [59, 110] to replace the polynomial function τ with an approximating MLP $m : \mathbb{R}^a \to \mathbb{R}^b$ to get a k-order network (details are in Section 9.11). Applying m feature-wise, that is $m(B)_{i,:} = m(B_{i,:})$, is in particular a k-order network in the sense of Section 9.3.1.

9.6 A simple network with 3-WL discrimination power

In this section we describe a simple GNN model that has 3-WL discrimination power. The model has the form

$$F = m \circ h \circ B_d \circ B_{d-1} \cdots \circ B_1, \tag{65}$$

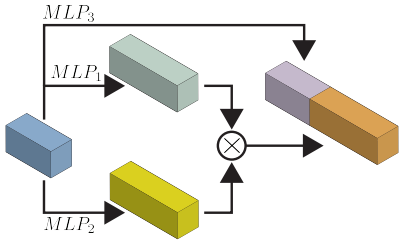


Figure 32: Block structure.

where as in k-order networks (see Section 9.3.1) h is an invariant layer and m is an MLP. $B_1, \ldots, B_d$ are blocks with the following structure (see figure 32 for an illustration). Let $X \in \mathbb{R}^{n \times n \times a}$ denote the input tensor to the block. First, we apply three MLPs $m_1, m_2 : \mathbb{R}^a \to \mathbb{R}^b$, $m_3 : \mathbb{R}^a \to \mathbb{R}^{b'}$ to the input tensor, $m_l(X)$, $l \in [3]$. This means applying the MLP to each feature of the input tensor independently, *i.e.*, $m_l(X)_{i_1,i_2,:} := m_l(X_{i_1,i_2,:})$, $l \in [3]$. Second, matrix multiplication is performed between matching features, *i.e.*, $W_{:,:,j} := m_1(X)_{:,:,j} \cdot m_2(X)_{:,:,j}$, $j \in [b]$. The output of the block is the tensor $(m_3(X), W)$. We start with showing our basic requirement from GNN, namely invariance:

Lemma 9. *The model F described above is invariant, i.e., $F(g \cdot B) = F(B)$, for all $g \in S_n$, and B.*

Proof. Note that matrix multiplication is equivariant: for two matrices $A, B \in \mathbb{R}^{n \times n}$ and $g \in S_n$ one has $(g \cdot A) \cdot (g \cdot B) = g \cdot (A \cdot B)$. This makes the basic building block B_i equivariant, and consequently the model F invariant, *i.e.*, $F(g \cdot B) = F(B)$. □

Before we prove the 3-WL power for this model, let us provide an intuition as to why matrix multiplication improves expressiveness. Let us show matrix multiplication allows this model to distinguish between the two graphs in Figure 31, which are 1-WL indistinguishable. The input tensor B representing a graph G holds the adjacency matrix at the last channel $A := B_{:,:,e+1}$. We can build a network with 2 blocks computing A^3 and then take the trace of this matrix (using the invariant layer h). Remember that the d-th power of the adjacency matrix computes the number of d-paths between vertices; in particular $\mathrm{tr}(A^3)$ computes the number of cycles of length 3. Counting shows the upper graph in Figure 31 has 0 such cycles while the bottom graph has 12. The main result of this section is:

Theorem 12. *Given two graphs $G = (V, E, d)$, $G' = (V', E', d')$ that can be distinguished by the 3-WL graph isomorphism test, there exists a network F ((65)) so that $F(G) \neq F(G')$. On the other direction for every two isomorphic graphs $G \cong G'$ and F ((65)), $F(G) = F(G')$.*

The full proof is provided in the Section 9.12. Here we outline the main idea of the proof. The second part of this theorem is already shown in Lemma 9. To prove the first part, namely that the model in (65) has

3-WL expressiveness, we show it can implement the 2-FWL algorithm, that is known to be equivalent to 3-WL (see Section 9.3.2). As before, the challenge is in implementing the neighborhood multisets as used in the 2-FWL algorithm. That is, given an input tensor $B \in \mathbb{R}^{n^2 \times a}$ we would like to compute an output tensor $C \in \mathbb{R}^{n^2 \times b}$ where $C_{i_1,i_2,:} \in \mathbb{R}^b$ represents a color matching the multiset $\{\!\{(B_{j,i_2,:}, B_{i_1,j,:}) \mid j \in [n]\}\!\}$. As before, we use the multiset representation introduced in section 9.4. Consider the matrix $X \in \mathbb{R}^{n \times 2a}$ defined by

$$X_{j,:} = (B_{j,i_2,:}, B_{i_1,j,:}), \quad j \in [n]. \tag{66}$$

Our goal is to compute an output tensor $W \in \mathbb{R}^{n^2 \times b}$, where $W_{i_1,i_2,:} = u(X)$.

Consider the multi-index set $\left\{\boldsymbol{\alpha} \mid \boldsymbol{\alpha} \in [n]^{2a}, |\boldsymbol{\alpha}| \leq n\right\}$ of cardinality $b = \binom{n+2a-1}{2a-1}$, and write it in the form $\{(\boldsymbol{\beta}_l, \boldsymbol{\gamma}_l) \mid \boldsymbol{\beta}, \boldsymbol{\gamma} \in [n]^a, |\boldsymbol{\beta}_l| + |\boldsymbol{\gamma}_l| \leq n, l \in b\}$.

Now define polynomial maps $\tau_1, \tau_2 : \mathbb{R}^a \to \mathbb{R}^b$ by $\tau_1(x) = (x^{\boldsymbol{\beta}_l} \mid l \in [b])$, and $\tau_2(x) = (x^{\boldsymbol{\gamma}_l} \mid l \in [b])$. We apply τ_1 to the features of B, namely $Y_{i_1,i_2,l} := \tau_1(B)_{i_1,i_2,l} = (B_{i_1,i_2,:})^{\boldsymbol{\beta}_l}$; similarly, $Z_{i_1,i_2,l} := \tau_2(B)_{i_1,i_2,l} = (B_{i_1,i_2,:})^{\boldsymbol{\gamma}_l}$. Now,

$$W_{i_1,i_2,l} := (Z_{:,:,l} \cdot Y_{:,:,l})_{i_1,i_2} = \sum_{j=1}^{n} Z_{i_1,j,l} Y_{j,i_2,l} = \sum_{j=1}^{n} B^{\boldsymbol{\beta}_l}_{j,i_2,:} B^{\boldsymbol{\gamma}_l}_{i_1,j,:} = \sum_{j=1}^{n} (B_{j,i_2,:}, B_{i_1,j,:})^{(\boldsymbol{\beta}_l, \boldsymbol{\gamma}_l)},$$

hence $W_{i_1,i_2,:} = u(X)$, where X is defined in (66). To get an implementation with the model in (65) we need to replace τ_1, τ_2 with MLPs. We use the universal approximation theorem to that end (details are Section 9.12).

To conclude, each update step of the 2-FWL algorithm is implemented in the form of a block B_i applying m_1, m_2 to the input tensor B, followed by matrix multiplication of matching features, $W = m_1(B) \cdot m_2(B)$. Since (63) requires pairing the multiset with the input color of each k-tuple, we take m_3 to be identity and get (B, W) as the block output.

Generalization to k-FWL. One possible extension is to add a generalized matrix multiplication to k-order networks to make them as expressive as k-FWL and hence $(k+1)$-WL. Generalized matrix multiplication is defined as follows. Given $A^1, \ldots, A^k \in \mathbb{R}^{n^k}$, then $(\odot_{i=1}^k A^i)_i = \sum_{j=1}^n A^1_{j,i_2,\ldots,i_k} A^2_{i_1,j,\ldots,i_k} \cdots A^k_{i_1,\ldots,i_{k-1},j}$.

9.7 Experiments

Implementation details. We implemented the GNN model as described in Section 9.6 (see (65)) using the TensorFlow framework [1]. We used three identical blocks B_1, B_2, B_3, where in each block $B_i : \mathbb{R}^{n^2 \times a} \to \mathbb{R}^{n^2 \times b}$ we took $m_3(x) = x$ to be the identity (*i.e.*, m_3 acts as a skip connection, similar to its role in the proof of Theorem 12); $m_1, m_2 : \mathbb{R}^a \to \mathbb{R}^b$ are chosen as d layer MLP with hidden layers of b features. After each block B_i we also added a single layer MLP $m_4 : \mathbb{R}^{b+a} \to \mathbb{R}^b$. Note that although this fourth MLP is not described in the model in Section 9.6 it clearly does not decrease (nor increase) the theoretical expressiveness of the model; we found it efficient for coding as it reduces the parameters of the model. For the first block, B_1, $a = e + 1$, where for the other blocks $b = a$. The MLPs are implemented with 1×1 convolutions. Parameter search was conducted on learning rate and learning rate decay, as detailed below. We have experimented with two network suffixes adopted from previous papers: (i) The suffix used in [158] that consists of an invariant max pooling (diagonal and off-diagonal) followed by a three Fully Connected (FC) with hidden units' sizes of $(512, 256, \#\text{classes})$; (ii) the suffix used in [256] adapted to our network: we apply the invariant max layer from [158] to the output of every block followed by a single fully connected layer to $\#$classes. These outputs are then summed together and used as the network output on which the loss function is defined.

Table 11: Graph Classification Results on the datasets from [257]

dataset	MUTAG	PTC	PROTEINS	NCI1	NCI109	COLLAB	IMDB-B	IMDB-M
size	188	344	1113	4110	4127	5000	1000	1500
classes	2	2	2	2	2	3	2	3
avg node #	17.9	25.5	39.1	29.8	29.6	74.4	19.7	13
				Results				
GK [218]	81.39±1.7	55.65±0.5	71.39±0.3	62.49±0.3	62.35±0.3	NA	NA	NA
RW [239]	79.17±2.1	55.91±0.3	59.57±0.1	$>$ 3 days	NA	NA	NA	NA
PK [179]	76±2.7	59.5±2.4	73.68±0.7	82.54±0.5	NA	NA	NA	NA
WL [219]	84.11±1.9	57.97±2.5	**74.68±0.5**	**84.46±0.5**	**85.12±0.3**	NA	NA	NA
FGSD [237]	**92.12**	**62.80**	73.42	79.80	78.84	**80.02**	73.62	**52.41**
AWE-DD [115]	NA	NA	NA	NA	NA	73.93±1.9	**74.45 ± 5.8**	51.54 ±3.6
AWE-FB [115]	87.87±9.7	NA	NA	NA	NA	70.99 ± 1.4	73.13 ±3.2	51.58 ± 4.6
DGCNN [268]	85.83±1.7	58.59±2.5	75.54±0.9	74.44±0.5	NA	73.76±0.5	70.03±0.9	47.83±0.9
PSCN [181](k=10)	88.95±4.4	62.29±5.7	75±2.5	76.34±1.7	NA	72.6±2.2	71±2.3	45.23±2.8
DCNN [13]	NA	NA	61.29±1.6	56.61± 1.0	NA	52.11±0.7	49.06±1.4	33.49±1.4
ECC [221]	76.11	NA	NA	76.82	75.03	NA	NA	NA
DGK [257]	87.44±2.7	60.08±2.6	75.68±0.5	80.31±0.5	80.32±0.3	73.09±0.3	66.96±0.6	44.55±0.5
DiffPool [262]	NA	NA	**78.1**	NA	NA	75.5	NA	NA
CCN [132]	**91.64±7.2**	**70.62±7.0**	NA	76.27±4.1	75.54±3.4	NA	NA	NA
Invariant Graph Networks [158]	83.89±12.95	58.53±6.86	76.58±5.49	74.33±2.71	72.82±1.45	78.36±2.47	72.0±5.54	48.73±3.41
GIN [256]	89.4±5.6	64.6±7.0	76.2±2.8	82.7±1.7	NA	80.2±1.9	**75.1±5.1**	**52.3±2.8**
1-2-3 GNN [175]	86.1±	60.9±	75.5±	76.2±	NA	NA	74.2±	49.5±
Ours 1	90.55±8.7	66.17±6.54	77.2±4.73	**83.19±1.11**	**81.84±1.85**	80.16±1.11	72.6±4.9	50±3.15
Ours 2	88.88±7.4	64.7±7.46	76.39±5.03	81.21±2.14	**81.77±1.26**	**81.38±1.42**	72.2±4.26	44.73±7.89
Ours 3	89.44±8.05	62.94±6.96	76.66±5.59	80.97±1.91	**82.23±1.42**	**80.68±1.71**	73±5.77	50.46±3.59
Rank	$\mathbf{3^{rd}}$	$\mathbf{2^{nd}}$	$\mathbf{2^{nd}}$	$\mathbf{2^{nd}}$	$\mathbf{2^{nd}}$	$\mathbf{1^{st}}$	$\mathbf{6^{th}}$	$\mathbf{5^{th}}$

Datasets. We evaluated our network on two different tasks: Graph classification and graph regression. For classification, we tested our method on eight real-world graph datasets from [257]: three datasets consist of social network graphs, and the other five datasets come from bioinformatics and represent chemical compounds or protein structures. Each graph is represented by an adjacency matrix and possibly categorical node features (for the bioinformatics datasets). For the regression task, we conducted an experiment on a standard graph learning benchmark called the QM9 dataset [199, 251]. It is composed of 134K small organic molecules (sizes vary from 4 to 29 atoms). Each molecule is represented by an adjacency matrix, a distance matrix (between atoms), categorical data on the edges, and node features; the data was obtained from the pytorch-geometric library [75]. The task is to predict 12 real valued physical quantities for each molecule.

Graph classification results. We follow the standard 10-fold cross validation protocol and splits from [268] and report our results according to the protocol described in [256], namely the best averaged accuracy across the 10-folds. Parameter search was conducted on a fixed random 90%-10% split: learning rate in $\left\{5 \cdot 10^{-5}, 10^{-4}, 5 \cdot 10^{-4}, 10^{-3}\right\}$; learning rate decay in $[0.5, 1]$ every 20 epochs. We have tested three architectures: (1) $b = 400$, $d = 2$, and suffix (ii); (2) $b = 400$, $d = 2$, and suffix (i); and (3) $b = 256$, $d = 3$, and suffix (ii). (See above for definitions of b, d and suffix). Table 11 presents a summary of the results (top part - non deep learning methods). The last row presents our ranking compared to all previous methods; note that we have scored in the top 3 methods in 6 out of 8 datasets.

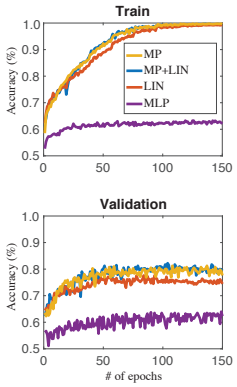
Graph regression results. The data is randomly split into 80% train, 10% validation and 10% test. We have conducted the same parameter search as in the previous experiment on the validation set. We have used the network (2) from classification experiment, *i.e.*, $b = 400$, $d = 2$, and suffix (i), with an absolute error loss adapted to the regression task. Test results are according to the best validation error. We have tried two different settings: (1) training a single network to predict all the output quantities together and (2) training a different network for each quantity. Table 12 compares the mean absolute error of our method with three

Table 12: Regression, the QM9 dataset.

Target	DTNN	MPNN	123-gnn	Ours 1	Ours 2
μ	0.244	0.358	0.476	**0.231**	**0.0934**
α	0.95	0.89	**0.27**	0.382	0.318
ϵ_{homo}	0.00388	0.00541	0.00337	**0.00276**	**0.00174**
ϵ_{lumo}	0.00512	0.00623	0.00351	**0.00287**	**0.0021**
Δ_ϵ	0.0112	0.0066	0.0048	**0.00406**	**0.0029**
$\langle R^2 \rangle$	17	28.5	22.9	**16.07**	**3.78**
$ZPVE$	0.00172	0.00216	**0.00019**	0.00064	0.000399
U_0	2.43	2.05	0.0427	0.234	**0.022**
U	2.43	2	0.111	0.234	**0.0504**
H	2.43	2.02	0.0419	0.229	**0.0294**
G	2.43	2.02	0.0469	0.238	**0.024**
C_v	0.27	0.42	**0.0944**	0.184	0.144

other methods: 123-gnn [175] and [251]; results of all previous work were taken from [175]. Note that our method achieves the lowest error on 5 out of the 12 quantities when using a single network, and the lowest error on 9 out of the 12 quantities in case each quantity is predicted by an independent network.

Equivariant layer evaluation. The model in Section 9.6 does not incorporate all equivariant linear layers as characterized in [158]. It is therefore of interest to compare this model to models richer in linear equivariant layers, as well as a simple MLP baseline (*i.e.*, without matrix multiplication). We performed such an experiment on the NCI1 dataset [257] comparing: (i) our suggested model, denoted Matrix Product (MP); (ii) matrix product + full linear basis from [158] (MP+LIN); (iii) only full linear basis (LIN); and (iv) MLP applied to the feature dimension.



Due to the memory limitation in [158] we used the same feature depths of $b_1 = 32, b_2 = 64, b_3 = 256$, and $d = 2$. The inset shows the performance of all methods on both training and validation sets, where we performed a parameter search on the learning rate (as above) for a fixed decay rate of 0.75 every 20 epochs. Although all methods (excluding MLP) are able to achieve a zero training error, the (MP) and (MP+LIN) enjoy better generalization than the linear basis of [158]. Note that (MP) and (MP+LIN) are comparable, however (MP) is considerably more efficient.

9.8 Conclusions

We explored two models for graph neural networks that possess superior graph distinction abilities compared to existing models. First, we proved that k-order invariant networks offer a hierarchy of neural networks that parallels the distinction power of the k-WL tests. This model has lesser practical interest due to the high dimensional tensors it uses. Second, we suggested a simple GNN model consisting of only MLPs augmented with matrix multiplication and proved it achieves 3-WL expressiveness. This model operates on input tensors of size n^2 and therefore useful for problems with dense edge data. The downside is that its complexity is still quadratic, worse than message passing type methods. An interesting future work is to search for more efficient GNN models with high expressiveness. Another interesting research venue is quantifying the generalization ability of these models.

9.9 Proof of Proposition 11

Proof. First, if $X' = g \cdot X$, then $p_{\boldsymbol{\alpha}}(X) = p_{\boldsymbol{\alpha}}(X')$ for all $\boldsymbol{\alpha}$ and therefore $u(X) = u(X')$. In the other direction assume by way of contradiction that $u(X) = u(X')$ and $g \cdot X \neq X'$, for all $g \in S_n$. That is, X and X' represent different multisets. Let $[X] = \{g \cdot X \mid g \in S_n\}$ denote the orbit of X under the action of S_n; similarly denote $[X']$. Let $K \subset \mathbb{R}^{n \times a}$ be a compact set containing $[X], [X']$, where $[X] \cap [X'] = \emptyset$ by assumption.

By the Stone–Weierstrass Theorem applied to the algebra of continuous functions $C(K, \mathbb{R})$ there exists a polynomial f so that $f|_{[X]} \geq 1$ and $f|_{[X']} \leq 0$. Consider the polynomial

$$q(X) = \frac{1}{n!} \sum_{g \in S_n} f(g \cdot X).$$

By construction $q(g \cdot X) = q(X)$, for all $g \in S_n$. Therefore q is a multi-symmetric polynomial. Therefore, $q(X) = r(u(X))$ for some polynomial r. On the other hand,

$$1 \leq q(X) = r(u(X)) = r(u(X')) = q(X') \leq 0,$$

where we used the assumption that $u(X) = u(X')$. We arrive at a contradiction. □

9.10 Proof of equivairance of WL update step

Consider the formal tensor B^j of dimension n^k with multisets as entries:

$$B^j_i = \{\!\{C^{l-1}_j \mid j \in N_j(i)\}\!\}. \tag{67}$$

Then the k-WL update step ((62)) can be written as

$$C^l_i = \mathrm{enc}\left(C^{l-1}_i, B^1_i, B^2_i, \ldots, B^k_i\right). \tag{68}$$

To show equivariance, it is enough to show that each entry of the r.h.s. tuple is equivariant. For its first entry: $(g \cdot C^{l-1})_i = C^{l-1}_{g^{-1}(i)}$. For the other entries, consider w.l.o.g. B^j_i:

$$\{\!\{(g \cdot C^{l-1})_j \mid j \in N_j(i)\}\!\} = \{\!\{C^{l-1}_{g^{-1}(j)} \mid j \in N_j(i)\}\!\} = \{\!\{C^{l-1}_j \mid j \in N_j(g^{-1}(i))\}\!\} = B^j_{g^{-1}(i)}.$$

We get that feeding k-WL update rule with $g \cdot C^{l-1}$ we get as output $C^l_{g^{-1}(i)} = (g \cdot C^l)_i$.

9.11 Proof of Theorem 11

Proof. We will prove a slightly stronger claim: Assume we are given some finite set of graphs. For example, we can think of all combinatorial graphs (*i.e.*, graphs represented by binary adjacency matrices) of n vertices . Our task is to build a k-order network F that assigns different output $F(G) \neq F(G')$ whenever G, G' are non-isomorphic graphs distinguishable by the k-WL test.

Our construction of F has three main steps. First in Section 9.11.1 we implement the initialization step. Second, Section 9.11.2 we implement the coloring update rules of the k-WL. Lastly, we implement a histogram calculation providing different features to k-WL distinguishable graphs in the collection.

9.11.1 Input and Initialization

Input. The input to the network can be seen as a tensor of the form $B \in \mathbb{R}^{n^2 \times (e+1)}$ encoding an input graph $G = (V, E, d)$, as follows. The last channel of B, namely $B_{:,:,e+1}$ (':' stands for all possible values $[n]$) encodes the adjacency matrix of G according to E. The first e channels $B_{:,:,1:e}$ are zero outside the diagonal, and $B_{i,i,1:e} = d(v_i) \in \mathbb{R}^e$ is the color of vertex $v_i \in V$. Our assumption of finite graph collection means the set $\Omega \subset \mathbb{R}^{n^2 \times (e+1)}$ of possible input tensors B is finite as well. Next we describe the different parts of k-WL implementation with k-order network. For brevity, we will denote by $B \in \mathbb{R}^{n^k \times a}$ the input to each part and by $C \in \mathbb{R}^{n^k \times b}$ the output.

Initialization. We start with implementing the initialization of k-WL, namely computing a coloring representing the isomorphism type of each k-tuple. Our first step is to define a linear equivariant operator that extracts the sub-tensor corresponding to each multi-index i: let $L : \mathbb{R}^{n^2 \times (e+1)} \to \mathbb{R}^{n^k \times k^2 \times (e+2)}$ be the linear operator defined by

$$L(X)_{i,r,s,w} = X_{i_r,i_s,w}, \quad w \in [e+1]$$
$$L(X)_{i,r,s,e+2} = \begin{cases} 1 & i_r = i_s \\ 0 & \text{otherwise} \end{cases}$$

for $i \in [n]^k, r, s \in [k]$.

L is equivariant with respect to the permutation action. Indeed, for $w \in [e+1]$,

$$(g \cdot L(X))_{i,r,s,w} = L(X)_{g^{-1}(i),r,s,w} = X_{g^{-1}(i_r),g^{-1}(i_s),w} = (g \cdot X)_{i_r,i_s,w} = L(g \cdot X)_{i,r,s,w}.$$

For $w = e + 2$ we have

$$(g \cdot L(X))_{i,r,s,w} = L(X)_{g^{-1}(i),r,s,w} = \begin{cases} 1 & g^{-1}(i_r) = g^{-1}(i_s) \\ 0 & \text{otherwise} \end{cases} = \begin{cases} 1 & i_r = i_s \\ 0 & \text{otherwise} \end{cases} = L(g \cdot X)_{i,r,s,w}.$$

Since L is linear and equivariant it can be represented as a single linear layer in a k-order network. Note that $L(B)_{i,:,:,1:(e+1)}$ contains the sub-tensor of B defined by the k-tuple of vertices $(v_{i_1}, \ldots, v_{i_k})$, and $L(B)_{i,:,:,e+2}$ represents the equality pattern of the k-tuple i, which is equivalent to the equality pattern of the k-tuple of vertices $(v_{i_1}, \ldots, v_{i_k})$. Hence, $L(B)_{i,:,:,:}$ represents the isomorphism type of the k-tuple of vertices $(v_{i_1}, \ldots, v_{i_k})$. The first layer of our construction is therefore $C = L(B)$.

9.11.2 k-WL update step

We next implement (62). We achieve that in 3 steps. As before let $B \in \mathbb{R}^{n^k \times a}$ be the input tensor to the the current k-WL step.

First, apply the polynomial function $\tau : \mathbb{R}^a \to \mathbb{R}^b$, $b = \binom{n+a-1}{a-1}$ entrywise to B, where τ is defined by $\tau(x) = (x^{\boldsymbol{\alpha}})_{|\boldsymbol{\alpha}| \leq n}$ (note that b is the number of multi-indices $\boldsymbol{\alpha}$ such that $|\boldsymbol{\alpha}| \leq n$). This gives $Y \in \mathbb{R}^{n^k \times b}$ where $Y_{i,:} = \tau(B_{i,:}) \in \mathbb{R}^b$.

Second, apply the linear operator

$$C^j_{i,r} := L_j(Y)_{i,r} = \sum_{i'=1}^{n} Y_{i_1,\cdots,i_{j-1},i',i_{j+1},\ldots,i_k,r}, \quad i \in [n]^k, r \in [b].$$

L_j is equivariant with respect to the permutation action. Indeed, $L_j(g \cdot Y)_{i,r} =$

$$\sum_{i'=1}^{n}(g \cdot Y)_{i_1,\cdots,i_{j-1},i',i_{j+1},\ldots,r} = \sum_{i'=1}^{n} Y_{g^{-1}(i_1)\cdots,g^{-1}(i_{j-1}),i',g^{-1}(i_{j+1}),\ldots,r} = L_j(Y)_{g^{-1}(i),r} = (g \cdot L_j(Y))_{i,r}.$$

Now, note that

$$C^j_{i,:} = L_j(Y)_{i,:} = \sum_{i'=1}^{n} \tau(B_{i_1,\cdots,i_{j-1},i',i_{j+1},\ldots,i_k,:}) = \sum_{j \in N_j(i)} \tau(B_{j,:}) = u(X),$$

where $X = B_{i_1,\ldots,i_{j-1},:,i_{j+1},\ldots,i_k,:}$ as desired.

Third, the k-WL update step is the concatenation: $(B, C^1, \ldots, C^k)$.

To finish this part we need to replace the polynomial function τ with an MLP $m : \mathbb{R}^a \to \mathbb{R}^b$. Since there is a finite set of input tensors Ω, there could be only a finite set Υ of colors in $\mathbb{R}^a$ in the input tensors to every update step. Using MLP universality [59, 110] , let m be an MLP so that $\|\tau(x) - m(x)\| < \epsilon$ for all possible colors $x \in \Upsilon$. We choose ϵ sufficiently small so that for all possible $X = (B_j \mid j \in N_j(i)) \in \mathbb{R}^{n \times a}$, $i \in [n]^k, j \in [k]$, $v(X) = \sum_{i \in [n]} m(x_i)$ satisfies the same properties as $u(X) = \sum_{i \in [n]} \tau(x_i)$ (see Proposition 11), namely $v(X) = v(X')$ iff $\exists g \in S_n$ so that $X' = g \cdot X$. Note that the 'if' direction is always true by the invariance of the sum operator to permutations of the summands. The 'only if' direction is true for sufficiently small ϵ. Indeed, $\|v(X) - u(X)\| \leq n \max_{i \in [n]} \|m(x_i) - \tau(x_i)\| \leq n\epsilon$, since $x_i \in \Upsilon$. Since this error can be made arbitrary small, u is injective and there is a finite set of possible X then v can be made injective by sufficiently small $\epsilon > 0$.

9.11.3 Histogram computation

So far we have shown we can construct a k-order equivariant network $H = L_d \circ \sigma \circ \cdots \circ \sigma \circ L_1$ implementing d steps of the k-WL algorithm. We take d sufficiently large to discriminate the graphs in our collection as much as k-WL is able to. Now, when feeding an input graph this equivariant network outputs $H(B) \in \mathbb{R}^{n^k \times a}$ which matches a color $H(B)_{i,:}$ (*i.e.*, vector in $\mathbb{R}^a$) to each k-tuple $i \in [n]^k$.

To produce the final network we need to calculate a feature vector per graph that represents the histogram of its k-tuples' colors $H(B)$. As before, since we have a finite set of graphs, the set of colors in $H(B)$ is finite; let b denote this number of colors. Let $m : \mathbb{R}^a \to \mathbb{R}^b$ be an MLP mapping each color $x \in \mathbb{R}^a$ to the one-hot vector in $\mathbb{R}^b$ representing this color. Applying m entrywise after H, namely $m(H(B))$, followed by the summing invariant operator $h : \mathbb{R}^{n^k \times b} \to \mathbb{R}^b$ defined by $h(Y)_j = \sum_{i \in [n]^k} Y_{i,j}$, $j \in [b]$ provides the desired histogram. Our final k-order invariant network is

$$F = h \circ m \circ L_d \circ \sigma \circ \cdots \circ \sigma \circ L_1.$$

□

9.12 Proof of Theorem 12

Proof. The second claim is proved in Lemma 9. Next we construct a network as in (65) distinguishing a pair of graphs that are 3-WL distinguishable. As before, we will construct the network distinguishing any finite set of graphs of size n. That is, we consider a finite set of input tensors $\Omega \subset \mathbb{R}^{n^2 \times (e+2)}$.

Input. We assume our input tensors have the form $B \in \mathbb{R}^{n^2 \times (e+2)}$. The first $e+1$ channels are as before, namely encode vertex colors (features) and adjacency information. The $e+2$ channel is simply taken to be the identity matrix, that is $B_{:,:,e+2} = I_d$.

Initialization. First, we need to implement the 2-FWL initialization (see Section 9.3.2). Namely, given an input tensor $B \in \mathbb{R}^{n^2 \times (e+1)}$ construct a tensor that colors 2-tuples according to their isomorphism type. In this case the isomorphism type is defined by the colors of the two nodes and whether they are connected or not. Let $A := B_{:,:,e+1}$ denote the adjacency matrix, and $Y := B_{:,:,1:e}$ the input vertex colors. Construct the tensor $C \in \mathbb{R}^{n^2 \times (4e+1)}$ defined by the concatenation of the following colors matrices into one tensor:

$$A \cdot Y_{:,:,j}, \quad (\mathbf{1}\mathbf{1}^T - A) \cdot Y_{:,:,j}, \quad Y_{:,:,j} \cdot A, \quad Y_{:,:,j} \cdot (\mathbf{1}\mathbf{1}^T - A), \quad j \in [e],$$

and $B_{:,:,e+2}$. Note that $C_{i_1,i_2,:}$ encodes the isomorphism type of the 2-tuple sub-graph defined by $v_{i_1}, v_{i_2} \in V$, since each entry of C holds a concatenation of the node colors times the adjacency matrix of the graph (A) and the adjacency matrix of the complement graph ($\mathbf{1}\mathbf{1}^T - A$); the last channel also contains an indicator if $v_{i_1} = v_{i_2}$. Note that the transformation $B \mapsto C$ can be implemented with a single block B_1.

2-FWL update step. Next we implement a 2-FWL update step, (63), which for $k = 2$ takes the form $C_i = \mathrm{enc}\Big(B_i, \Big\{\!\!\Big\{ (B_{j,i_2}, B_{i_1,j}) \;\Big|\; j \in [n] \Big\}\!\!\Big\}\Big)$, $i = (i_1, i_2)$, and the input tensor $B \in \mathbb{R}^{n^2 \times a}$. To implement this we will need to compute a tensor Y, where the coloring Y_i encodes the multiset $\Big\{\!\!\Big\{ (B_{j,i_2,:}, B_{i_1,j,:}) \;\Big|\; j \in [n] \Big\}\!\!\Big\}$.

As done before, we use the multiset representation described in section 9.4. Consider the matrix $X \in \mathbb{R}^{n \times 2a}$ defined by

$$X_{j,:} = (B_{j,i_2,:}, B_{i_1,j,:}), \quad j \in [n]. \tag{69}$$

Our goal is to compute an output tensor $W \in \mathbb{R}^{n^2 \times b}$, where $W_{i_1,i_2,:} = u(X)$.

Consider the multi-index set $\{\boldsymbol{\alpha} \mid \boldsymbol{\alpha} \in [n]^{2a}, |\boldsymbol{\alpha}| \leq n\}$ of cardinality $b = \binom{n+2a-1}{2a-1}$, and write it in the form $\{(\boldsymbol{\beta}_l, \boldsymbol{\gamma}_l) \mid \boldsymbol{\beta}, \boldsymbol{\gamma} \in [n]^a, |\boldsymbol{\beta}_l| + |\boldsymbol{\gamma}_l| \leq n, l \in b\}$. Now define polynomial maps $\tau_1, \tau_2 : \mathbb{R}^a \to \mathbb{R}^b$ by $\tau_1(x) = (x^{\boldsymbol{\beta}_l} \mid l \in [b])$, and $\tau_2(x) = (x^{\boldsymbol{\gamma}_l} \mid l \in [b])$. We apply τ_1 to the features of B, namely $Y_{i_1,i_2,l} := \tau_1(B)_{i_1,i_2,l} = (B_{i_1,i_2,:})^{\boldsymbol{\beta}_l}$; similarly, $Z_{i_1,i_2,l} := \tau_2(B)_{i_1,i_2,l} = (B_{i_1,i_2,:})^{\boldsymbol{\gamma}_l}$. Now,

$$\begin{aligned} W_{i_1,i_2,l} := (Z_{:,:,l} \cdot Y_{:,:,l})_{i_1,i_2} &= \sum_{j=1}^{n} Z_{i_1,j,l} Y_{j,i_2,l} = \sum_{j=1}^{n} \tau_1(B)_{j,i_2,l}\, \tau_2(B)_{i_1,j,l} \\ &= \sum_{j=1}^{n} B^{\boldsymbol{\beta}_l}_{j,i_2,:} B^{\boldsymbol{\gamma}_l}_{i_1,j,:} = \sum_{j=1}^{n} (B_{j,i_2,:}, B_{i_1,j,:})^{(\boldsymbol{\beta}_l, \boldsymbol{\gamma}_l)}, \end{aligned}$$

hence $W_{i_1,i_2,:} = u(X)$, where X is defined in (69).

To implement this in the network we need to replace τ_i with MLPs m_i, $i = 1, 2$. That is,

$$W_{i_1,i_2,l} := \sum_{j=1}^{n} m_1(B)_{j,i_2,l}\, m_2(B)_{i_1,j,l} = v(X), \tag{70}$$

where $X \in \mathbb{R}^{n \times 2a}$ is defined in (69).

As before, since input tensors belong to a finite set $\Omega \subset \mathbb{R}^{n^2 \times (e+1)}$, so are all possible multisets X and all colors, Υ, produced by any part of the network. Similarly to the proof of Theorem 11 we can take (using the universal approximation theorem) MLPs m_1, m_2 so that $\max_{x \in \Upsilon, i=1,2} \|\tau_i(x) - m_i(x)\| < \epsilon$. We choose

ϵ to be sufficiently small so that the map $v(X)$ defined in (70) maintains the injective property of u (see Proposition 11): It discriminates between X, X' not representing the same multiset.

Lastly, note that taking m_3 to be the identity transformation and concatenating $(B, m_1(B) \cdot m_2(B))$ concludes the implementation of the 2-FWL update step. The computation of the color histogram can be done as in the proof of Theorem 11.

□

Part II
Relaxations of matching problems

10 An efficient SDP relaxation of the point cloud registration problem

This section is based on [160].

10.1 Introduction

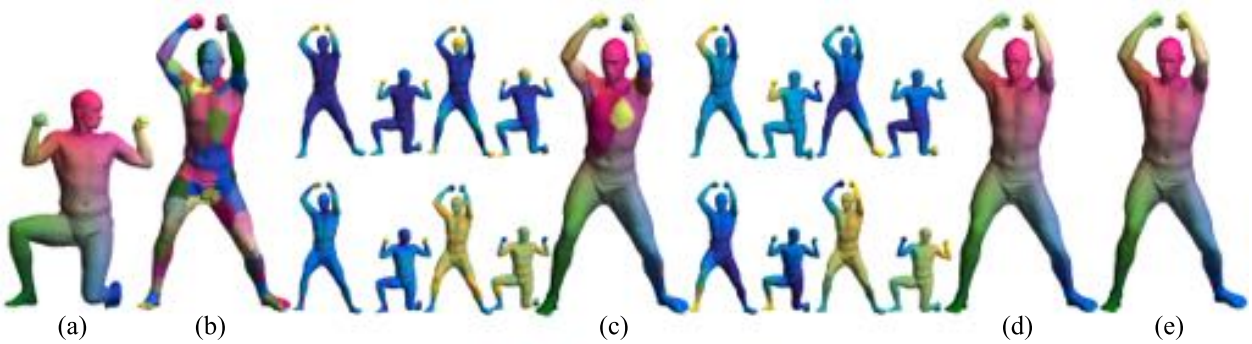


Figure 33: Initializing high-dimensional ICP for non-rigid registration of two human surfaces using different methods: (a) input shape; (b) random initialization; (c) initialization using Wave Kernel Signatures (four examples are shown to its left); (d) initialization with segment correspondence (four examples are shown to its left). The initialization is crucial for good matching; in (e) we show the result of initialization using PM-SDP which provides comparable result to (d).

Registration of point sets is a central problem in computer graphics with many applications including shape analysis, shape retrieval, statistical shape inference, and shape reconstruction.

Among the different formulations of the point set registration problem, the Procrustes matching (PM) formulation is very common: Given two d-dimensional point sets of n points each, $P, Q \in \mathbb{R}^{d\times n}$, which are neither aligned nor consistently labeled, the task is to find a linear isometry (*i.e.*, an orthogonal transformation) $R \in \mathcal{O}(d)$ and a permutation $X \in \Pi_n$ minimizing the distance between the point sets:

$$d(P,Q) = \min_{X,R} \|RP - QX\|_F^2 \tag{71a}$$

$$\text{s.t.} \quad X \in \Pi_n \tag{71b}$$

$$R \in \mathcal{O}(d) \tag{71c}$$

Procrustes matching arises naturally in two and three dimensions ($d = 2, 3$) for rigid matching problems. Non-rigid matching problems are also often formulated in this way, wherein linear isometries in higher dimension ($d \gg 3$) approximate non-rigid isometries of the shapes; this idea is advocated in Functional Maps [186] where the Laplace-Beltrami eigenfunctions are used for the high-dimensional embedding.

The optimization problem (71) is non-convex and globally optimizing it is difficult. In fact, even the subproblem of finding an exact solution for PM when such a solution exists (*i.e.*, when $d(P,Q) = 0$) is difficult. It can be shown that this subproblem can be solved in polynomial time iff there is a polynomial time algorithm for the exact graph matching problem. The latter is a well researched problem for which no polynomial time algorithm is known.

The iterative closest point (ICP) [29] algorithm is a popular algorithm for locally minimizing (71), based on the fact that when either R or X are held constant, (71) can be solved globally. As we will demonstrate, (71) has a vast amount of local minima, so that the success of ICP depends heavily on a good initialization.

Previous methods relied on shape features/signatures and/or prior knowledge to initialize ICP. For example, Figure 33 depicts different initializations for solving (71) in the context of Functional Maps for surface matching; a source model (a) is matched to a target model using different initialization for R: (b) shows results achieved from random initialization; (c) demonstrates an initialization of R using Wave Kernel Signatures [16]; and (d) demonstrates initialization using matched segments. Both (c) and (d) are common initializations used in Functional Maps papers [186, 191]. In this case, all initializations aside from the segmentation correspondence resulted in suboptimal matching (in (c), for instance, the left hand of the model in (a) is incorrectly matched to the chest).

The goal of this work is to approximate the global minimum of (71). To accomplish this, we present a novel convex relaxation of PM using semidefinite programming (SDP), which we name PM-SDP. Standard SDP relaxations are known to give very accurate approximations, at the price of high time complexity. For example, [124] give an extremely accurate relaxation for the quadratic matching problem, but their algorithm can only run on a handful of points. We formulate a similar SDP relaxation for PM, and use results on semidefinite completion problems to significantly reduce the size of the semidefinite constraints while remaining equivalent to the original relaxation. As a result, our relaxation has significantly improved time complexity.

Our relaxation is applicable to point sets consisting of around 100 points of a reasonably high dimension (15-20). Applying our algorithm to initialize the Functional Maps ICP as shown in Figure 33(e) provides results which compare favorably with matchings achieved with random and WKS initializations, and are comparable to results achieved with matched segmentation initialization.

We demonstrate the accuracy of the suggested relaxation both theoretically and experimentally. We prove that for problems without noise, the relaxation returns a correct global minimum of PM. Up to technical details, this analysis is valid for asymmetric shapes, as well as shapes with bilateral symmetries. The latter include many important instances of the shape matching problem, such as matching human bodies. We also show our algorithm achieves the global optimum for perturbed asymmetric shapes. Experimentally we show that at low noise levels our algorithm still returns the global minimum, and at high noise levels it returns a close-to-optimal local minimum.

We demonstrate the strength and applicability of the PM-SDP relaxation by achieving state of the art results on standard non-rigid shape matching benchmarks such as SCAPE [10] and the more recent FAUST [30]. We also demonstrate applications to collective matching and biological shape classification.

10.2 Previous work

Point cloud registration is a basic building block in computer graphics and geometry processing. We mention works that are most relevant to this work. For detailed survey see [230].

For input shapes or point clouds which are close to being aligned, the ICP algorithm and its many variants [209] are a popular choice. For shapes that are not roughly aligned the ICP algorithm can be easily stuck in local minimum. A widely used method for matching shapes in a general position is RANSAC [77], which exhaustively samples the space of possible transformations. Being essentially a brute-force algorithm, its drawback is the high complexity which results in poor scalability to higher dimensions.

Another way to deal with shapes that are not roughly aligned is to try to find a good initialization for ICP. Representative works in this direction are the works of [88, 258] which use combinatorial optimization techniques in order to find the optimal solution in 3D.

Non-rigid shape matching is a central task in geometry processing. We discuss relevant work and refer to relevant surveys for more details [243, 234, 230]. One approach is to formulate the non-rigid problem as a version of the quadratic assignment problem called quadratic matching. [167, 38] suggest to minimize the Gromov-Hausdorff distance where [38] also introduce the generalized multidimensional scaling variant on point metric spaces; [143, 25] relax the quadratic matching problem using linear programming and spectral techniques; [124] suggest a convex SDP relaxation to the quadratic matching problem; and [50] suggest a linear programming relaxation solved using MRF (Markov Random Fields) techniques.

Another approach is to restrict the mapping search space to a smaller, more tractable space such as: low dimensional deformation space [41], conformal mappings [148, 266, 127]; or isometries [187, 231]. Some works try to adapt the 3D ICP algorithm to the non-rigid case [145], whereby typically a deformation model is chosen (*e.g.*, piecewise affine) and the algorithm iterates between finding correspondences and solving for the optimal deformation.

Non-rigid shape matching can also be tackled using supervised machine learning techniques, where typically the algorithm is focused on a specific type of data. For instance, [273, 245] train a model that matches human bodies.

The most relevant methods to this work pose the non-rigid shape matching problem as a high dimensional rigid shape matching problem [116, 185, 186, 191]. These methods map each point of the input point cloud to a vector containing the values of the eigen-functions of the Laplace-Beltrami operator [210]. The central observation is that if the input shapes are isometric, the intrinsic isometry between them becomes an extrinsic isometry in the high dimensional space [185]. This opens the door for using rigid matching algorithms such as ICP in the context of the more complicated non-rigid matching problem. The problem is that finding a good initialization is difficult for high dimensional problems. Current methods partly employ prior knowledge, such as correspondence between pre-computed segments, to initialize the ICP algorithm [186].

Procrustes analysis is a tool used to perform statistical study of shapes by canceling transformations that are not shape-altering [122]. It has many applications in various fields of science [94] and in particular anatomical shape analysis and morphometrics [169, 34]. There are many variants to the Procrustes problem, maybe the most general is the one addressed in this work - PM, for which no closed-form solution is known [94]. To our knowledge, we provide the first closed-form convex formulation guaranteeing a globally optimal solution for the exact and near-exact cases (under mild assumptions).

Semidefinite relaxation and polynomial optimization. Convex relaxations of quadratic optimization problems are often (*e.g.*, [192, 152]) preformed by replacing quadratic terms with new variables. All quadratic terms then become linear, and a convex semidefinite constraint is added to couple between the original variables and the new variables. A significant drawback of these relaxations is that they are computationally tractable only for very small polynomial problems. [83, 242] show that for problems with certain structure, a positive semidefinite constraint on a large matrix can be replaced by constraining certain principal submatrices to be positive semidefinite, resulting in an equivalent problem with significantly improved time complexity. In this section we devise a quadratic formulation of PM which has this structure, and as a result obtain a relaxation which is tractable for medium sized problems.

10.3 Approach and Formulation

We present our convex relaxation of PM. We first discuss the general quadratic optimization problem and present a novel strategy for replacing its "standard" time consuming SDP relaxations with an equivalent, but

significantly more efficient SDP relaxation. In this context, we then instantiate our relaxation for the PM problem.

Full SDP relaxation of quadratic problems. Quadratic optimization problems are problems of the form

$$\min_{x\in\mathbb{R}^N} \quad f_0(x) \tag{72a}$$

$$\text{s.t.} \quad f_s(x) = 0, \quad s = 1,\dots,S \tag{72b}$$

$$f_t(x) \geq 0, \quad t = S+1\dots T \tag{72c}$$

where f_i are quadratic multivariate polynomials.

A typical relaxation procedure includes two steps (see [152] for a survey on SDP quadratic relaxations): First, the quadratic polynomials f_i are linearized by introducing new variables Y_{ij}, $1 \leq i,j \leq N$, which replace quadratic monomials $x_i x_j$, so that $f_j(x)$ becomes a linear polynomial in the variables x, Y denoted $\mathcal{L}[f_j](x,Y)$. This gives an equivalent formulation of (72):

$$\min_{x,Y} \quad \mathcal{L}\left[f_0\right](x,Y) \tag{73a}$$

$$\text{s.t.} \quad \mathcal{L}\left[f_s\right](x,Y) = 0, \quad s = 1\dots S \tag{73b}$$

$$\mathcal{L}\left[f_t\right](x,Y) \geq 0, \quad t = S+1\dots T \tag{73c}$$

$$Y = xx^T \tag{73d}$$

With the exception of constraint (73d), Problem (73) is a convex problem (in fact a linear program). Therefore, the second step in this relaxation procedure is replacing (73d) with a convex constraint. The convex hull of the set defined by (73d) is the set defined by the convex constraint $Y \succeq xx^T$, which is equivalent to the semidefinite constraint

$$\begin{bmatrix} 1 & x^T \\ x & Y \end{bmatrix} \succeq 0. \tag{74}$$

A natural relaxation of (73) is therefore given by replacing (73d) with (74). [270] and more recently [124] used this approach to relax the quadratic matching and quadratic assignment problems. The obtained relaxation is significantly more accurate than prevalent relaxations for quadratic matching, but its scalability is poor; in fact, it cannot handle more than a handful of points to be matched, completely hindering some applications.

Efficient SDP relaxation. The key to a useful and efficient relaxation of (72) and consequently our problem (71) is to reduce the dimension of the semi-definite constraint (74) which is the main factor determining time efficiency of the semidefinite program.

To obtain a more efficient SDP relaxation, we make the observation that for some problems not all terms in the matrix xx^T appear in the polynomials f_j. This is, for example, the case in the PM problem, as will soon be shown. In such cases, we can find a collection $\mathcal{J}$ of subsets of $\{1,\dots,N\}$ so that all polynomials f_j include only expressions from $x_J x_J^T,\ J \in \mathcal{J}$. An equivalent formulation for (73) can therefore be obtained by replacing (73d) with $Y_J = x_J x_J^T$, for all $J \in \mathcal{J}$. In turn, replacing these with the convex constraints

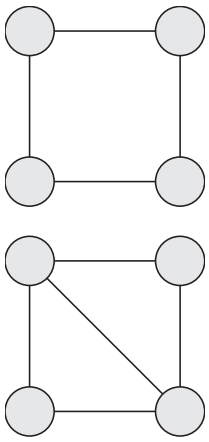
$$\begin{bmatrix} 1 & x_J^T \\ x_J & Y_J \end{bmatrix} \succeq 0, \qquad J \in \mathcal{J} \tag{75}$$

we obtain a convex relaxation for (72). If all subsets $J \in \mathcal{J}$ satisfy $|J| \ll N$, the obtained relaxation is considerably more efficient than the original (full) relaxation.

There is no unique way to apply this more efficient relaxation; a given instance of a quadratic optimization problem may have several different possible decompositions $\mathcal{J}$. Those that use small semidefinite constraints will be more efficient, but not necessarily as accurate as those using larger semidefinite constraints; the latter, however, can quickly become intractable for certain problems. Nevertheless, if $\mathcal{J}$ is chosen so that it satisfies the chordality condition we will soon describe, the obtained relaxation is in fact equivalent to the full relaxation.

In general, any solution for the full relaxation also satisfies (75). For equivalency, we need to ensure that a solution for the efficient relaxation (75) can always be completed to a solution of the full relaxation. For that end we need to show there is a solution for the following matrix completion problem: We are given entries of x_J, Y_J satisfying (75), and we are searching for a completion of Y that satisfies (74). Since the objective and linear constraints depend only on the coordinates which were determined before the completion, the full solution will also fulfill the linear constraints, and the objective will not be affected by the completion.

The condition that allows solving the completion problem is related to the structure of the known coordinates of the matrix. The collection $\mathcal{J}$ defines an undirected graph $G = (V, E)$ whose vertices are $V = \{1, x_1, \ldots, x_N\}$. Two distinct vertices are connected by an edge iff they both appear in one of the matrices (75) defined by some $J \in \mathcal{J}$. A graph G is *chordal* if every (simple) cycle with more than three vertices contains a chord, *i.e.*, an edge between two non-adjacent members of the cycle. For example the graph in the top of the inset has a cycle which does not contain a chord and thus is not a chordal graph. The bottom graph is chordal.



If G is chordal, the following theorem from [97] guarantees that the matrix completion problem has a solution, and therefore that the two relaxations are equivalent:

Theorem 13. *If G is chordal, and $(x_J, Y_J)_{J\in\mathcal{J}}$ satisfy* (75)*, then the missing coordinates of Y can be chosen so that the full semidefinite constraint* (74) *holds.*

PM-SDP Formulation. We now return to the PM problem and instantiate the strategy presented above. First, we note that PM can be formulated as the following quadratic problem:

$$\min_{X,R} \|RP - QX\|_F^2 \tag{76a}$$

$$X\mathbf{1} = \mathbf{1}, \quad \mathbf{1}^T X = \mathbf{1}^T \tag{76b}$$

$$X_j X_j^T = \text{diag}(X_j), \quad j = 1 \ldots n \tag{76c}$$

$$RR^T = R^T R = I \tag{76d}$$

where we denote by $\mathbf{1} \in \mathbb{R}^{n\times 1}$ the all-ones vector, by X_j the j-th column of X, and by $\text{diag}(X_j)$ the diagonal matrix whose diagonal entries are X_j.

To see this is indeed an equivalent formulation of PM, note that if (R, X) is a feasible solution of (76), then R is orthogonal by definition. The constraint (76c) implies that $X_{ij}^2 = X_{ij}$ for all i, j, so that all elements of X are in $\{0, 1\}$. By (76b) the rows and columns of X sum to one, which implies that X is a permutation matrix.

In the other direction, note that if $(R, X) \in \mathcal{O}(d) \times \Pi_n$, then since each column of X has only one non-zero element, $X_j X_j^T$ is diagonal. Since all elements of X are in $\{0, 1\}$, $X_{ij}^2 = X_{ij}$ for all i, j so that (76c) holds. It is straightforward to check that (76b),(76d) hold as well.

All polynomials in (76) are quadratic in the entries of R and X. The full SDP relaxation for quadratic problems described above can then be applied; this will result in a vector x, consisting of the elements of R and X, of dimension $d^2 + n^2$; the SDP constraint will be of size $(d^2 + n^2 + 1) \times (d^2 + n^2 + 1)$.

We obtain an equivalent, more efficient, relaxation by utilizing the efficient SDP relaxation approach; the key observation here is that all the quadratic polynomials participating in the formulation (76) of the PM problem can be expressed using linear polynomials in the entries of $X_j X_j^T$, $X_j \left[R\right]^T$, and $\left[R\right]\left[R\right]^T$, where $\left[R\right] \in \mathbb{R}^{d^2 \times 1}$ is the column stack of the matrix R and $j = 1 \dots n$. We therefore introduce new matrix variables Z_j, constrained to satisfy

$$Z_j = \begin{bmatrix} X_j \\ \left[R\right] \end{bmatrix} \begin{bmatrix} X_j \\ \left[R\right] \end{bmatrix}^T, \quad j = 1 \dots n \tag{77}$$

Let us next see how the objective and constraints of Problem (76) are linear in the variables X, R, Z_j. First, the objective of (76) can be rewritten as

$$\|RP - QX\|_F^2 = \sum_j \|RP_j - QX_j\|_2^2 = \sum_j \mathrm{tr}(W_j Z_j) + const$$

for some constant matrices W_j since $\|RP_j - QX_j\|_2^2$ is linear in the entries of Z_j. Denoting

$$Z_j = \begin{bmatrix} A_j & B_j^T \\ B_j & C \end{bmatrix}, \; A_j \in \mathbb{R}^{n \times n}, C \in \mathbb{R}^{d^2 \times d^2}, B_j \in \mathbb{R}^{d^2 \times n} \tag{78}$$

the constraint (76c) can be rewritten as $A_j = \mathrm{diag}(X_j)$. Finally, the constraints (76d) are affine functions of C and can be rewritten as

$$\mathrm{tr}(H_\ell C) + b_\ell = 0, \quad \ell = 1 \dots 2d^2$$

for some constant matrices H_ℓ. Replacing the non-convex equality constraint of (77) with convex semidefinite constraints of type (74), we obtain our relaxation for the PM problem, PM-SDP:

$$\min_{Z_j, X, R} \quad \sum_j \mathrm{tr}\left(W_j Z_j\right) \tag{79a}$$

$$X\mathbf{1} = \mathbf{1}, \quad \mathbf{1}^T X = \mathbf{1}^T \tag{79b}$$

$$A_j = \mathrm{diag}(X_j), \qquad j = 1 \dots n \tag{79c}$$

$$\mathrm{tr}(H_\ell C) + b_\ell = 0, \qquad \ell = 1 \dots 2d^2 \tag{79d}$$

$$Z_j \succeq \begin{bmatrix} X_j \\ \left[R\right] \end{bmatrix} \begin{bmatrix} X_j \\ \left[R\right] \end{bmatrix}^T, \qquad j = 1 \dots n \tag{79e}$$

where A_j and C are defined as in (78).

Chordality of the relaxation. To show that (79) is equivalent to the full relaxation of the PM problem, we need to show that the graph G induced by $\mathcal{J} = \{J_j\}$ is chordal, where J_j is the set containing the variables $\left[R\right]$ and X_j. The adjacency matrix of G for this case is illustrated in Figure 34 (left) where each gray square represents a full block of ones; on the right, we illustrate the corresponding graph G where each disk represents a clique which corresponds to a diagonal block in the adjacency matrix. To show G is chordal it is enough to show every cycle of length at-least 4 has a chord: Indeed, if a cycle is completely contained in one of the sets J_j (represented as triangles in Figure 34, right) then any two vertices are connected by an edge; otherwise there are two non consecutive visits to vertices in the $1, \left[R\right]$ cliques (top disks), and these vertices are connected by an edge.

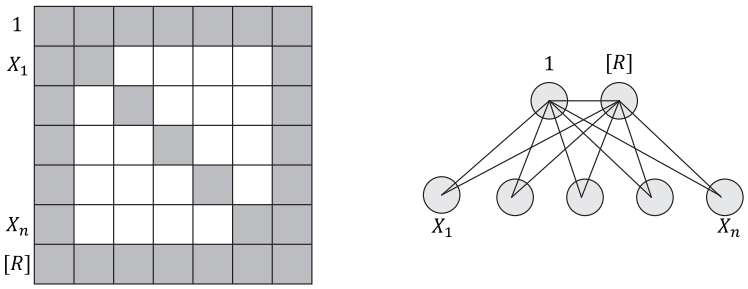


Figure 34: The graph corresponding to the Procrustes Problem (right, each disk represents a clique) is chordal, that is, has no minimal cycles of length at-least four. The adjacency matrix is shown on the left.

Dimension and complexity. We note that (79e) includes n semidefinite constraints involving $N \times N$ matrices, where $N = d^2 + n + 1$, as opposed to the full relaxation which in this case would involve a square matrix with $N = d^2 + n^2 + 1$. When the number of points in the sets n is significantly larger than the dimension d of the space the point reside in (*i.e.*, $n \gg d$), which is often the case in point registration problems, PM-SDP will be significantly more efficient than the full SDP relaxation.

Relaxation properties. The PM-SDP relaxation has the following natural theoretical properties:

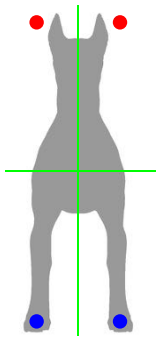
1. *Rotation and relabeling invariance*: Rotating or relabeling the input shapes will not affect the solution provided by the relaxation. More precisely, if P is replaced with a point cloud $\tilde{P}$ obtained from P by relabeling and applying an orthogonal transformation, then the objective of PM-SDP remains unchanged, and X, R transform accordingly.

2. *Lower bound*: Since PM-SDP is a relaxation its optimal objective is less or equal to the PM optimal objective. We denote the objective of PM-SDP by $\underline{d}$.

3. *Positivity*: The objective of PM-SDP is always non-negative. This is natural since this is the case for the objective of PM .

4. *Convex-hull of* R, X: The R, X coordinates of a feasible solution for PM-SDP are in the convex hull of $\mathcal{O}(d) \times \Pi_n$.

The proof of the these properties as well as the theorems presented below are given in [71].



Exact recovery refers to the problem of finding $R, X \in \mathcal{O}(d) \times \Pi_n$ which solve the equation $RP = QX$, when such solutions exists, *i.e.*, when $d(P, Q) = 0$. We call such solutions *exact solutions*. From the computational perspective, exact recovery for PM is equivalent to exact graph matching. The latter is a well-researched problem, not known to be polynomial. Accordingly, proving exact recovery in full generality is not likely. However, under the assumption that the covariance $d \times d$ matrix PP^T of the point cloud P has a simple spectrum, and an additional weak assumption, we are able to prove exact recovery. This too is analogous to the graph matching problem where finding exact solutions for graphs with simple spectrum affinity matrices is solvable in polynomial time [19].

The assumption that PP^T has a simple spectrum implies that the symmetries of P are all reflections along the principle axes of the point set P. In particular, it implies that all symmetries of P are

bilateral. The class of bilateral symmetric shapes includes many important instances of the shape matching problem. Note however, that not all reflections along principle axes are necessarily symmetries of P. The additional weak assumption required for our exact recovery with symmetries result formulated below is that there exists a point P_j in P such that its reflections along principle axes belongs to P only for symmetries of P. The inset figure demonstrates a shape with point having this property (blue); Applying a horizontal flip, which is a symmetry, maps the point to another point on the shape, while applying a (non-symmetry) vertical flip maps it outside of the shape (red). We are not aware of bilateral symmetric shapes of practical interest that do not satisfy this condition.

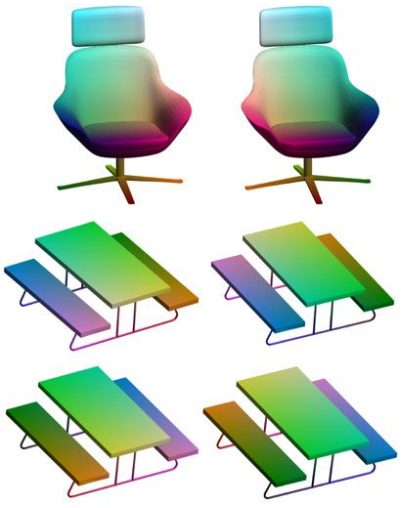
The exactness argument starts with assuming we are given P, Q with $d(P, Q) = 0$ and showing that when $d(P, Q) = 0$ also $\underline{d}(P, Q) = 0$. This follows from relaxation properties 2 and 3 described above: from the lower bound property we know that the objective $\underline{d}$ of PM-SDP is a lower bound of the objective d of PM. From the positivity property, $\underline{d}$ is always non-negative. It follows that when $d(P, Q) = 0$ also $\underline{d}(P, Q) = 0$, and the set of feasible solutions of PM-SDP with zero objective, which we call the *exact convex solution set*, is a superset of the set of exact solutions. When the shapes are asymmetric, the exact convex solution set consists of only one point - the exact solution,

Theorem 14. *Let P, Q be asymmetric shapes with $d(P, Q) = 0$ satisfying the simple spectrum and weak conditions. Then PM-SDP has a unique exact convex solution, which is also the unique exact solution of PM.*

When P, Q are bilateral symmetric, there are several exact solutions. All convex combinations of these solutions will be in the exact convex solution set, so that generally exact convex solutions will not be exact solutions. However, by restricting ourselves to the R coordinate of both the exact solutions and the convex exact solutions, which we refer to as *exact orthogonal solutions* and *exact convex orthogonal solutions*, we are able to show:

Theorem (Full version of theorem 14)**.** *Let P, Q be shapes with $d(P, Q) = 0$ satisfying the simple spectrum and weak conditions. Then the exact orthogonal solutions of PM are the extreme points of the set of exact convex orthogonal solutions.*

The set of exact convex orthogonal solutions is a convex set and therefore its extreme points can be found by simply optimizing linear energies over this set (a convex problem again). One simple algorithm for obtaining *all* exact solutions is repeatedly solving a variation of our convex relaxation: minimize a random linear energy $\mathrm{tr}(WR)$, where $W \in \mathbb{R}^{d \times d}$ is a random matrix drawn from the uniform measure on the unit sphere, under the set of constraints of (79), and adding the linear constraint that the objective (79a) is zero. We prove,



Theorem 15. *The random algorithm returns an extreme point of the set of exact convex orthogonal solutions (i.e., an exact orthogonal solution) with probability one. Moreover, all extreme points are found with the same probability.*

Once an exact orthogonal solution is found it can be shown that the X coordinate of the solution is guaranteed to be a permutation. The inset demonstrates the output of the probabilistic algorithm described in theorem 15 on two point sets with perfect bilateral symmetries sampled from a model of a chair and a picnic table. The random algorithm retrieves the two bilateral symmetries of the chair, and the four bilateral symmetries of the picnic table.

10.4 Implementation details

We discuss implementation details of the PM-SDP algorithm.

Injective matching. We consider a slight variation of PM where we allow the point cloud P to have fewer points than Q, and search for the correspondence between P and a subset of the points of Q. This formulation is useful to account for the inherent noise caused from samplings of different shapes. Furthermore, in case the shapes we wish to compare are not isometric, certain points on one shape might not have a good match on the second one.

We denote the number of points of P by $k \leq n$. The formulation of PM in (71) remains unchanged, except now $X \in \mathbb{R}^{n \times k}$ is constrained to be a matrix with entries in $\{0, 1\}$ such that all columns of X have exactly one non-zero entry. The only necessary modification for PM-SDP is that the constraint $X\mathbf{1} = \mathbf{1}$ in (79) should be replaced with the constraint $X\mathbf{1} \leq \mathbf{1}$.

Utilizing priors for computational efficiency. The PM-SDP framework allows incorporating priors to improve computational complexity by further reducing the size of the SDP constraint. This is done by noting that if we know or assume the points P_j and Q_i should not correspond then setting $X_{ij} = 0$ reduces the size of the SDP constraint by one. Indeed, returning to (77) we see that when $X_{ij} = 0$ we may also assume that the i-th row and column of Z_j are also zero. Similarly, if $R_{st} = 0$ we can eliminate a row and column of Z_j.

Let us demonstrate where this can be used for near-isometric matching. We rule out unlikely correspondences using the average geodesic distance (AGD) descriptor [127]. Corresponding points on nearly isometric shapes should have roughly the same AGD value since this descriptor is isometry-invariant. For close to isometric shapes, we can safely rule out correspondences between points whose AGD value is significantly different. If the possibility of Q_i corresponding to P_j, is ruled out, we set $X_{ij} = 0$.

For isometric shapes, the Laplace-Beltrami operator of both shapes has the same eigenvalues, and the linear isometry R takes eigenfunctions of the first shape with eigenvalue λ to eigenfunctions of the second shape with the same eigenvalue λ. In the most common simple spectrum case, this means that R is a diagonal matrix, if the maximal eigenvalue multiplicity of the shapes is two then R is tridiagonal, etc. For near-isometric shapes, we make the assumption that R is m-diagonal, that is, has m non-zero diagonals symmetrically around the main diagonal.

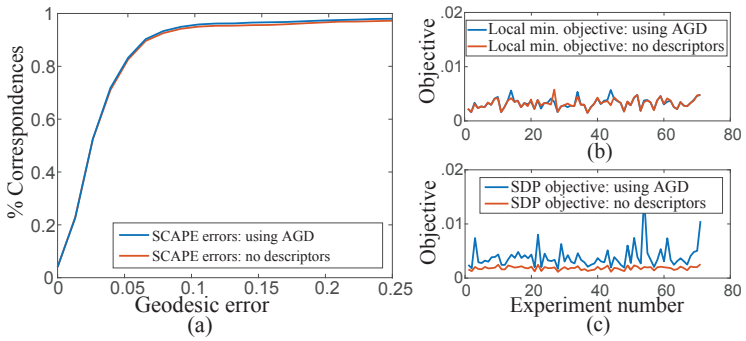


Figure 35: Ruling out matches with the AGD descriptor has a negligible effect on the quality of the relaxation: (a) depicts the results of PM-SDP on SCAPE dataset [10] with and without AGD to rule out unlikely matches; (b) the objective after local minimization; and (c) objective value achieved by PM-SDP. The PM-SDP objective is lower for the unpruned version while the rest of the results are equivalent for both versions.

In practice, we use the AGD descriptor to rule-out 50-70% of the correspondences, and constrain R to

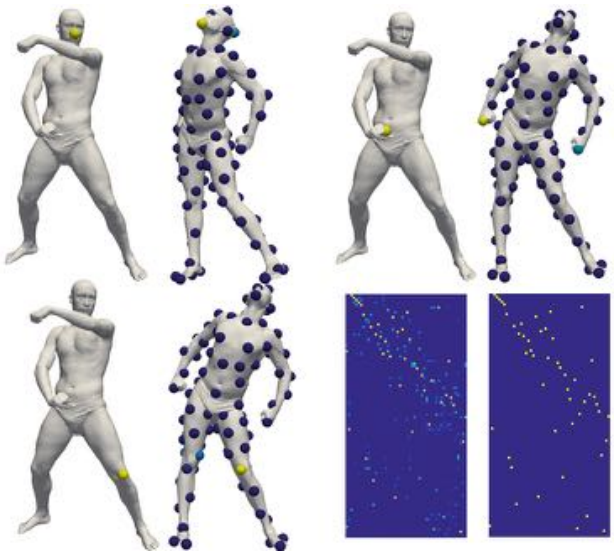


Figure 36: Visualization of the doubly-stochastic map X as generated by the PM-SDP relaxation when comparing two SCAPE models; each pair of surfaces depicts a column of X by coloring the point set Q according to the corresponding value in X; the X matrix before and after projection on the permutations is shown at the bottom-right.

be 1,3, or 5-diagonal. Fortunately, the effect of incorporating these priors on the quality of the relaxation is negligible: Figure 35 demonstrates that using PM-SDP to match shapes from SCAPE dataset (using the protocol that will be described in Section 10.6) produces essentially equivalent results with and without using the AGD descriptor to rule out unlikely matches; the former, however, has the advantage of significantly improved computational efficiency.

Local minimization. Since the feasible set of PM-SDP is larger than the feasible set of PM, the solution of PM-SDP in general may not contain orthogonal and permutation matrices. We therefore project the solution onto the feasible set of PM. We do this by locally minimizing PM using the output of PM-SDP to initialize the algorithm. The local minimization is done using an ICP-like algorithm which interleaves between minimizing over one of the matrices R, X while holding the other constant: fixing R results in a linear program, while for a fixed X there exists a closed-form solution [94]. In Figure 36 we illustrate the doubly-stochastic matrix X as outputted from the PM-SDP relaxation and the permutation achieved after the projection. As shown, the PM-SDP output is already very similar to the projection result demonstrating the tightness of the PM-SDP relaxation. More details are in Section 10.8.

The local minimization following the PM-SDP relaxation allows generalizing Theorem 14 to the inexact case:

Corollary 2. *Let P, Q be point clouds satisfying the conditions of Theorem 14, and let P^ϵ, Q^ϵ be sufficiently small perturbations of P, Q. Then PM-SDP followed by the local minimization returns the unique (global) solution of PM for P^ϵ, Q^ϵ.*

10.5 Evaluation

We test the tightness of the PM-SDP relaxation by comparing it to the ground truth obtained from an exhaustive brute-force sampling algorithm. The latter is only tractable for low dimensional d, and we choose

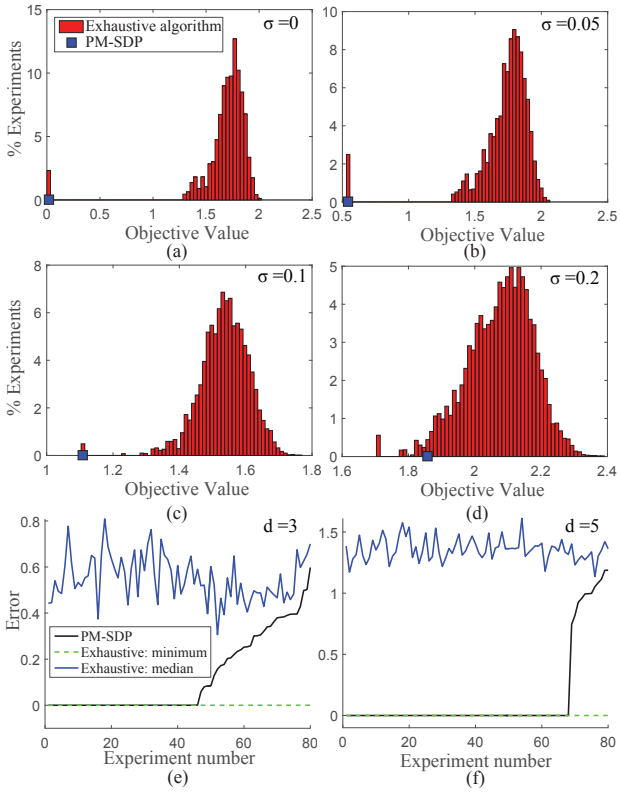


Figure 37: PM-SDP tightness evaluation: (a-d) show the histograms of the objective values achieved by the exhaustive sampling algorithm compared to the optimal PM-SDP objective on a few typical runs. When the noise level is low to medium, our algorithm usually finds the global minimum. On higher noise levels it returns an objective value close to optimal. (e-f) show illustrations of the deviation of the optimal PM-SDP objective value from the global optimum and the median value computed by the exhaustive algorithm when $\sigma = 0.1, d = 3$ and $d = 5$.

$d = 3$: The exhaustive algorithm densely samples $\sim 10k$ points from a uniform distribution over $\mathcal{O}(3)$ and uses each sample R_j as an initialization for the local minimization algorithm described above.

In Figure 37 we compare the histograms of optimal values achieved by the exhaustive sampling algorithm (in red) to the energy achieved by PM-SDP (in blue). The data for this experiment was created by randomizing $Q \in \mathbb{R}^{3\times 50}$ according to a uniform distribution on $[0, 1]$, and setting $P = R^T QX + \varepsilon$, with $X \in \Pi_{50}$, $R \in \mathcal{O}(3)$ and noise $\varepsilon \sim N_{d\times n}(0, \sigma^2)$. (a-d) show the results of a few typical runs with increasing amount of noise $\sigma = 0, 0.05, 0.1, 0.2$. We note that the number of local (sub-optimal) minima for the exhaustive sampling is surprisingly high; for example, for noise level $\sigma = 0.1$ we found more than 1000 local energy minima. Additionally, the experiment in (a) verifies our theoretical exactness result as can be seen by the fact that the blue point achieves the left most value of the red histogram. When the noise level is low to medium ($\sigma = 0.05, 0.1$) the PM-SDP relaxation usually produces optimal result, see (b-c). When noise level is high ($\sigma = 0.2$ in (d)) the relaxation does not provide an optimal solution but nevertheless produces a close to optimal result.

A quantitative evaluation of the optimality of PM-SDP is given in Figure 37, (e-f). We ran 80 random experiments for $d = 3$ and $d = 5$ with noise level $\sigma = 0.1$ and measured the optimal objective value achieved by PM-SDP in comparison to the global minimum and median value of the objective values found by the exhaustive algorithm. For visualization, we subtracted the value of the optimal value from all of the results. PM-SDP (black line) usually returns the optimal value (green) and always returns a better result than the median objective value (blue) of the exhaustive algorithm.

10.6 Applications

10.6.1 Functional maps

The main application of our algorithm is non-rigid shape matching of pairs of surfaces $\mathcal{P}, \mathcal{Q}$. Following [186] we pose this problem as a high dimensional PM problem, replacing non-rigid isometries with linear isometries (orthogonal transformations) in higher dimensional space. More specifically, we sample k points on the first shape and n points on the second shape uniformly using farthest point sampling [73] initialized with extrema of average geodesic distance (AGD) [127], and embed P, Q in $\mathbb{R}^d$. The embedding is done by first computing the first d eigenfunctions of the cot-weight Laplace-Beltrami (LB) operator [190] on each of the surfaces, $\{\Phi_i^{\mathcal{P}}\}$, $\{\Phi_i^{\mathcal{Q}}\}$, $i = 1 \dots d$ and then assigning the d coordinates $(\Phi_1^{\mathcal{P}}(p), .., \Phi_d^{\mathcal{P}}(p))$ to each point $p \in P$, and similarly to every point $q \in Q$.

Current approaches using this formulation, solve the resulting high dimensional PM problem using an ICP-type iterative algorithm; as this problem is shown to have a vast number of local minima even for $d = 3$ (see Figure 37 (a-d)), initialization is crucial.

Using standard shape signatures or features often does not provide a satisfactory initialization (*e.g.*, Figure 47, (b-c)), and previously this ICP procedure was initialized with matched segments for successful results [186, 191]. In the experiments of this subsection we use PM-SDP followed by local minimization to initialize the ICP of [186] that uses the entire embedded models. In comparisons to previous works we used code supplied by the authors with their default parameters.

SCAPE dataset. We evaluated the performance of PM-SDP for non-rigid isometric matching using the SCAPE dataset [10]. We used $n = 100$ sampled points, $d = 17$ eigenfunctions, and injective mapping of $k = 50$ out of $n = 100$. For better efficiency we allowed each point in P to be matched only to the 30% of the points in Q that have the closest AGD, and selected $m = 5$ (5 non-zero diagonals in R). The SDP optimization was performed using Mosek [176]. We extended our results to a full correspondence of all the vertices by solving ICP in dimension $d = 30$. The average running time for a pair with these settings is 30-35 minutes on an Intel Xeon E5 CPU. We also tested a faster version by taking $d = 30$ and using diagonal R, that is $m = 1$; this gave only slightly inferior results with running time of 2.5 minutes per pair.

Figure 38 shows a comparison of our algorithm with several state of the art algorithms. The comparison was done according to the protocol of [127] accepting symmetries. Our method compares favorably to functional maps (FM) when initialized with matched segments [186] and improves upon FM with automatic shape signature initialization. We also show in green the results of the faster, less accurate, variant of our algorithm described in the previous paragraph. Figure 39 shows typical results of our algorithm from this experiment.

FAUST dataset. We evaluated the performance of PM-SDP for non-rigid non-isometric matching on the FAUST dataset [30]. We used a similar setup as in the previous experiment, with the following differences: We generated the LB operator directly on the point cloud sampling generated by [50] using a similar construction to [20] (weights based on geodesic distances instead of Euclidean distances). We also used two versions of the PM-SDP: For the first we chose $d = 17$, $m = 5$ and we allowed each point in P to be matched only to the 40% points in Q that have the closest AGD; the running time of this parameters set is about 40 minutes per pair. For the second, faster parameter set, we used $n = 40, k = 30$, an embedding with $d = 10$, $m = 5$, and kept 80% of Q for each point in P; the running time with these parameters is less than 4 minutes per pair.

Figure 40 compares PM-SDP with the recent method of [50] which demonstrated superb state of the art results on this dataset. However, they rely on the assumption that the shapes are initially aligned in 3D

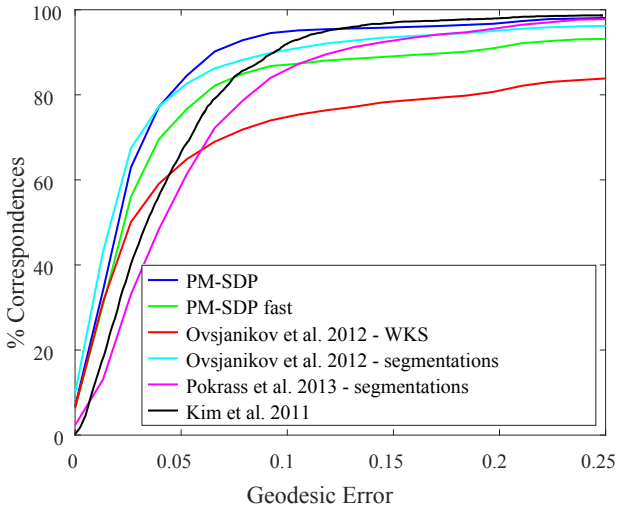


Figure 38: Results of our algorithm and state of the art algorithms on the SCAPE [10] non-rigid shape matching benchmark.

and indeed use this alignment by adding a regularization term. In order to make a fair comparison we disabled this regularization term. When this term is removed, intrinsic symmetries might be found by the algorithm. In order to account for that we sampled a set of 52 ground truth points in each mesh, and added the symmetric flip to the ground truth map. Aside from that, we followed Chen and Koltun's evaluation protocol (including using their point clouds as stated above). As can be read from the graphs, our algorithm (blue) compares favorably in both the inter and intra class matching scenarios in terms of cumulative error distribution and average error. We also show here a faster version of our algorithm (green), which provides good results in a shorter computation time. Figure 41 shows some typical results of our algorithm for both inter and intra class matching.

SCAPE dataset (raw scans). We further tested our algorithm on the SCAPE original raw scans dataset [10] that contain missing data, holes and noise. We used the same preprocessing method of [50] and ran our algorithm with exactly the same parameters as on the FAUST dataset on the 71 pairs as defined in the benchmark of [127]. Figure 42 shows the cumulative error graph and a few typical results. We note that also here we ran [50] without the extrinsic regularization term (in addition to the reasons stated above, for the SCAPE dataset this prior is inappropriate due to its pose diversity).

SHREC07 dataset. We also ran PM-SDP ($n = 100, k = 40$) on the highly non-isometric SHREC07 dataset [91]. On this dataset, PM-SDP achieved good results only on some of the classes; Figure 43 demonstrates typical results from these classes: the Ant, Teddy and Glasses.

10.6.2 Anatomical classification

The Procrustes distance with labeled points (*i.e.*, when X is known) is a well-known measure of shape similarity in fields such as statistical shape analysis [122, 34]. The sampling and labeling of points in a collection of shapes is tedious work that requires the attention of an expert for several months [34]. The possibility of solving Procrustes matching with unlabeled points (*i.e.*, the PM problem in this work) using PM-SDP makes the task of finding meaningful landmarks unnecessary.

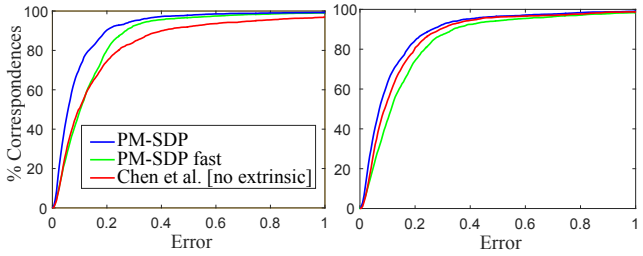
Figure 39: Examples of typical maps obtained with PM-SDP on the SCAPE dataset [10]. In all pairs: left mesh is colored using a predefined color map; right mesh is colored according to the correspondence. Bottom right: a failure case.

We took three anatomical bone datasets containing 116, 61 and 45 models respectively from [34]. We sampled $n = k = 120$ points of each shape using farthest point sampling, ran PM-SDP and used its output to initialize ICP that matches 400 farthest points on the shapes. This computation takes about 7 minutes for each pair.

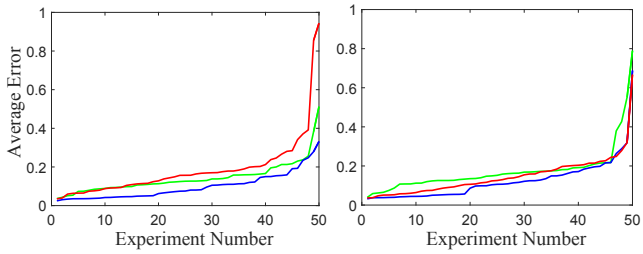
We followed the classification protocol suggested in [34] where each shape is classified according to its nearest (in Procrustes distance) neighbor; each shape in the datasets has three biological tags: Genera, Family and Above Family, and we tested classification of all three categories. Table 13 presents classification success rates (what fraction of shapes were correctly classified in each classification test) and shows PM-SDP compares favorably to Boyer's method [34], and is remarkably comparable to the results achieved using human expert labeled landmarks. Figure 44 shows a few examples of maps that were found by PM-SDP.

10.6.3 Shape collection alignment

We demonstrate another application of PM-SDP to consistent alignment of shapes. The task we would like to solve here is the following: given a set of semantically similar shapes - apply an orthogonal transformation per shape so that the shapes are aligned. We solve this problem by using PM-SDP to solve for pairwise orthogonal transformations and permutations over the entire dataset and then modifying the ICP procedure we mentioned in section 10.4 to project onto the set of *consistent orthogonal transformations*; The details of the projection procedure and definition of consistency are given in Section 10.9. To demonstrate the



(a) Intra: cumulative error (b) Inter: cumulative error



(c) Intra: average error (d) Inter: average error

Figure 40: Cumulative and average errors achieved on the FAUST dataset [30] by PM-SDP compared to [50] without the global extrinsic regularization term.

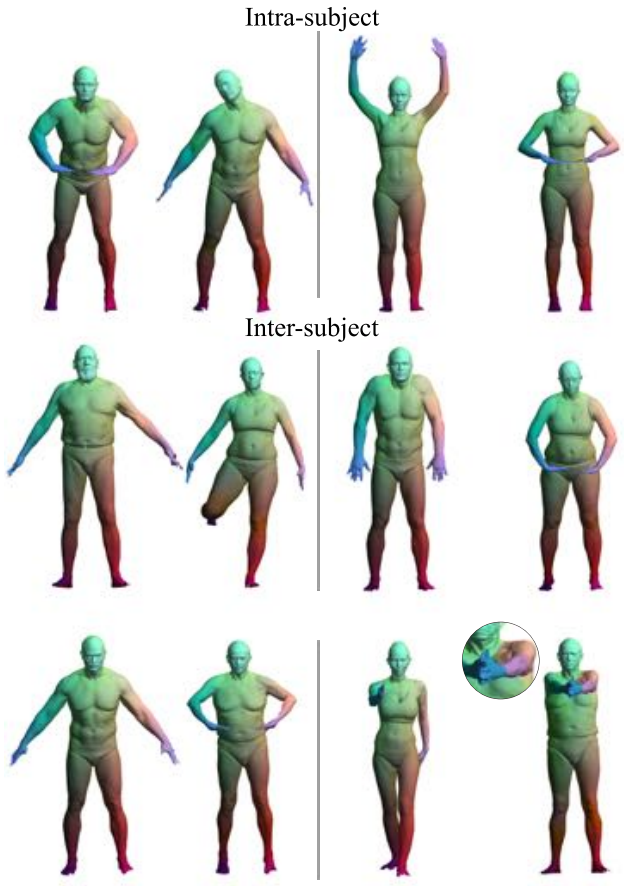


Figure 41: Examples of typical maps obtained with PM-SDP on the FAUST dataset [30]. Top row: intra-subject. Bottom rows: inter-subject. Bottom right: a failure case.

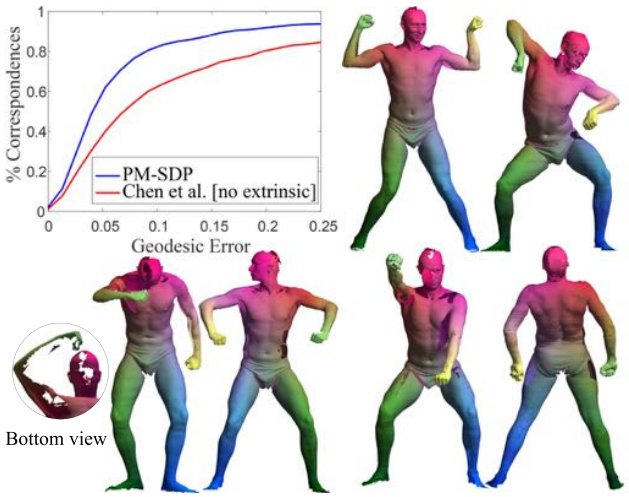


Figure 42: Performance on the SCAPE raw scans dataset [10]. Top left: Cumulative error distribution. Other: Examples of typical maps obtained with PM-SDP . Bottom right: a failure case (forward-backward flip).

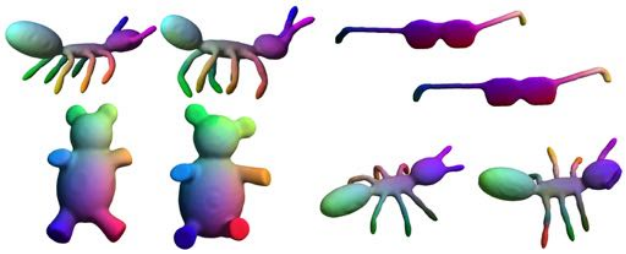


Figure 43: Examples of maps obtained with PM-SDP on the non-isometric SHREC07 dataset [91]. Bottom right: a failure case (incorrect corresponding legs).

flexibility of our approach, we use a variation of the high dimensional embedding used above. We embedded the shapes into a seven-dimensional space, the first three coordinates being the euclidian x, y, z coordinates, and the other 4 were eigenfunctions of the LB operator (as was done for isometric matching above). Since the Euclidian coordinates should not mix with the eigenfunction coordinates we constrain R to be block diagonal.

As demonstrated in Figure 45 PM-SDP with $d = 7$ (second row) yielded a better consistent alignment in comparison with the method for $d = 3$. The shapes for this experiment are taken from three classes of the SHREC07 [91] dataset. We made sure the shapes are arbitrarily rotated, sampled $n = k = 20$ farthest points on each shape and solved for all pairwise matchings; for $d = 3$ each pair is computed in 2-3 seconds and for $d = 7$ each pair takes 15-20 seconds.

Timing. Timing of experiments that appear in the section have already been stated. Here we provide quantitative timing experiments. Figure 46 shows typical run times as a function of dimension or number of points. The experiments were conducted on random and noisy synthetic data. In experiment (a) the dimension d varies from 3 to 20 and we match $k = 50$ points to $n = 100$ points. Experiment (b) compares runtime versus the number of points: in each experiment we match a k point point cloud to a $n = 2k$ point point cloud (up to $k = 50$, $n = 100$) and the dimension is constant $d = 10$. In both cases, R was constrained to be 5-diagonal and we allowed each point to be matched to 30% of the points in the other point cloud based on prior knowledge (in this case these points were selected randomly). (c) Shows comparison of the running time of PM-SDP and the full SDP relaxation discussed in section 10.3. In this case we use $d = 10, k = n = 5 \dots 25$. Notably, the full relaxation becomes intractable for more than 17 point, whereas the equivalent PM-SDP formulation solves these problems in just seconds.

Dataset	Classification	PM-SDP	Boyer et al.	Expert
Teeth	Genera	**91.9**	90.9	**91.9**
	Family	**94.3**	92.5	**94.3**
	Above Family	**98.2**	94.8	95.7
Metatarsal	Genera	79.6	79.6	**88.1**
	Family	**93.4**	91.8	**93.4**
	Above Family	**100**	**100**	**100**
Radius	Genera	82.2	**84.4**	77.8
	Family	NA	NA	NA
	Above Family	NA	NA	NA

Table 13: Classification results (accuracy) achieved by PM-SDP on three anatomical shape data sets compared with [34] and a human expert.

10.7 Conclusions

Summary. We have developed an algorithm that approximates the global minimum of the PM problem with a proven exact recovery property in presence of bilateral symmetries, as well as several other theoretical properties of the algorithm. We demonstrated state of the art results for non-rigid isometric and near-isometric shape matching problems solved using our convex relaxation. We also showed that PM-SDP is useful for anatomical classification of shapes and for aligning shape collections.

Limitations. In contrast to previous SDP relaxations of similar problems, we are able to deal with the registration of around one hundred points. Nonetheless, in comparison with non-SDP based approaches, the main limitation of this algorithm remains its time complexity, which we predict will improve as research on SDP optimization progresses; another limitation of our shape matching framework is the fact that spectral embedding is aimed at near-isometric matching, and is not a good model of the problem for non-isometric shapes.

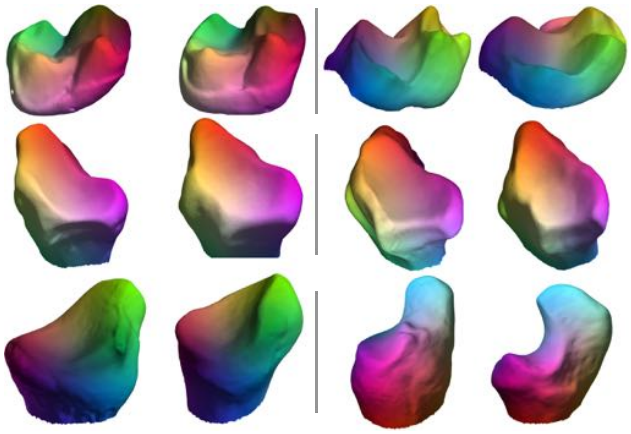


Figure 44: Examples of typical maps that were obtained by PM-SDP on the anatomical datasets of [34]. First row: Teeth; second row: Metatarsal bone; third row: Radius bone.

Future work. One direction we intend to pursue is applying our technique for constructing efficient relaxations for quadratic optimization to different problems other than PM. An interesting theoretical problem which we intend to pursue is proving that PM-SDP (or similar relaxations) give a good approximation of the solution in the general (noisy, far from exact) case, in contrast with our theoretical analysis here which applies only for isometric or near-isometric shapes. Extending to two-way partial matching is also interesting. ggestions.

10.8 Local minimization

The local minimization can be initialized using the output of PM-SDP in four different ways. Two immediate possibilities are given by choosing the R or X coordinates from the optimal solution. We note that R, X may not be in $\mathcal{O}(d) \times \Pi_n$, but as mentioned previously they are in the convex hull of that set. Two other possibilities can be obtained from decomposing the lifted variables B_j as explained next. When (77) holds, we can use (78) to write $B_j = [R]\, X_j^T$. We can then combine the matrices B_j into a larger matrix B so that $B = [R]\,[X]^T$, and X, R can be recovered in this case by factorizing B into an outer product of two vectors. Motivated by this, we do the following:

1. Create the matrix B using the matrices B_j: $B = [B_1, ..., B_k]$.
2. Project B onto the set of rank one matrices using SVD, denote the projection as $\widehat{B}$.
3. Factorize the projected rank-one matrix as an outer product of two vectors $\widehat{B} = [\widehat{R}][\widehat{X}]$, and use these vectors as another two possible initializations of R and X.

10.9 Collection alignment

In this section we present the collection alignment algorithm mentioned in sub-section 10.6.3 . First we solve PM for each pair of shapes in the collection with the additional constraint that R_{ij} is in the convex hull of $SO(3)$ using the formula from [212]. Then, we build a large matrix R containing all the orthogonal transformations obtained by PM-SDP as sub-blocks. More specifically, each $d \times d$ block in the (i, j)-th position is the orthogonal transformation between shape i and shape j. In case the set of transformations

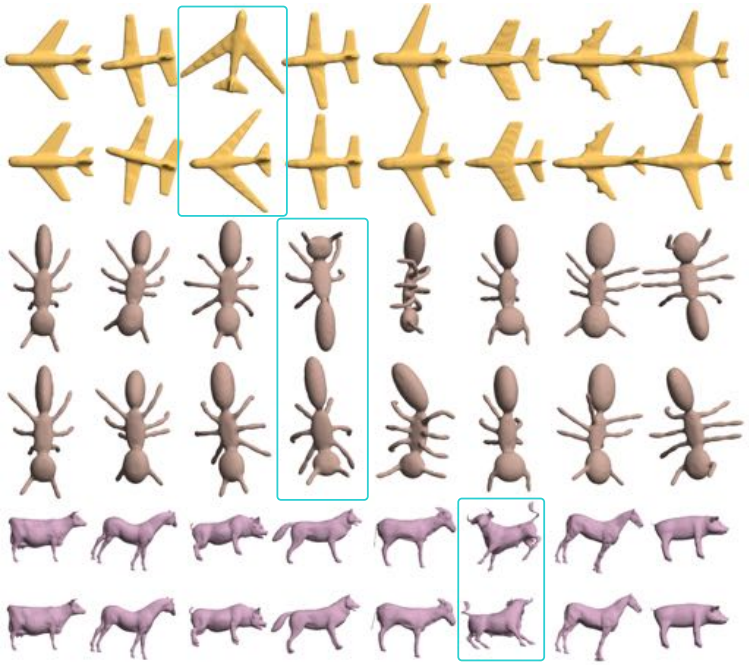


Figure 45: Finding consistent orthogonal transformations between non-isometric shapes. The figure shows three classes from the SHREC07 dataset [91]. The first row of each class shows the alignment obtained by PM-SDP in 3D. The second row shows the alignment achieved using a 7-dimensional embedding with PM-SDP. Rectangles show corrected orthogonal transformations.

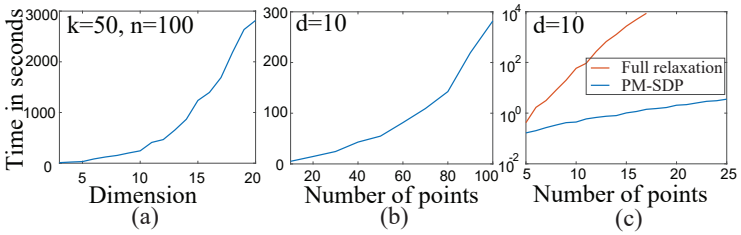


Figure 46: typical run times as functions of the dimension and the number of points. (a) as a function of the dimension d, (b) as a function of the number of points n, (c) Comparison to the full relaxation discussed in section 10.3.

is consistent (i.e., for each i, j, k, $R_{ij} \cdot R_{jk} = R_{ik}$) , R is positive semidefinite and has rank d (for the definition of consistent maps and orthogonal transformations see [222, 112, 180]).

In the spirit of this observation, we feed this matrix into a an iterative ICP-like algorithm that performs the following steps:

1. Project R onto the set of consistent rank d matrices: Let UDU^T be the eigen-decomposition of R, we take the largest d eigenvectors of U scaled by the square root of the corresponding eigenvalue, which we denote U_d and project each of its $d \times d$ block to its closest orthogonal matrix. Denote the matrix with the new blocks as U'_d then the output of this step is $R' = U'_d(U'_d)^T$.
2. For each block of R' solve for the best permutation X_{ij} using linear programming as described in section 10.4 .
3. For each permutation X_{ij} solve for the best orthogonal transformation using the closed form solution mentioned above.
4. Construct R from the matrices R_{ij}.
5. Iterate until convergence.

Our experiments show that this algorithm reaches a steady state after a few iterations that take a few seconds for the problems in Subsection 10.6.3.

11 Graph matching via tight quadratic relaxation

This section is based on [72].

11.1 Introduction

Matching problems, seeking some useful correspondence between two shapes or, more generally, discrete metric spaces, are central in computer graphics and vision. Matching problems are often modeled as optimization of a quadratic energy over permutations. Global optimization and approximation of such problems is known to be NP-hard [150].

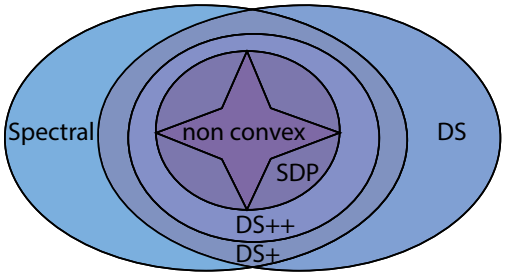
A common strategy for dealing with the computational hardness of matching problems is replacing the original optimization problem with an easier, similar problem that can be solved globally and efficiently. Perhaps the two most common *scalable* relaxations for such problems are the spectral relaxation for energies with non-negative entries [143, 74], and the doubly stochastic (DS) relaxation for convex energies [5, 76].

Our work is motivated by the recent work of [124] who proposed a semi-definite programming (SDP) relaxation which is provably stronger than both spectral and DS relaxations. The obtained relaxation was shown empirically to be extremely tight, achieving the global ground truth in most experiments presented. However, a major limitation was the computational cost of solving a semi-definite program with $O(n^4)$ variables. Accordingly in this section we pursue the following question:

Question: Is it possible to construct a relaxation which is stronger than the spectral and DS relaxations, without compromising efficiency?

Figure 47: Our algorithm offers a flexible and scalable framework for matching metric spaces and is guaranteed to perform better than the classical spectral and doubly-stochastic relaxations. Left: non-rigid matching computed automatically between two raw scans with topological issues from the FAUST dataset [30]; Right, an automatic arrangement of natural images in a 2D grid based on deep features-based pairwise affinity. Note how similar objects are clustered together.

We give an affirmative answer to this question and show that by correctly combining the spectral and DS relaxations in the spirit of [79] we obtain a relaxation which is provably tighter than both, and is in fact in a suitable sense exactly the intersection of both relaxations. We name this relaxation DS+. Moreover, we observe that a refined spectral analysis leads to a significant improvement to this relaxation and a provably tighter quadratic



program we name DS++. This relaxation enjoys the same scalability as DS and DS+ as all three are quadratic programs with n^2 variables and the same number of constraints. Additional time efficiency is provided by specialized solvers for the DS relaxation such as [225]. We note that DS++ is still less tight than the final expensive and accurate relaxation of [124] yet strikes a balance between tightness and computational complexity. The hierarchy between the relaxations is illustrated in the inset and proven in section 11.4.

Since DS++ is a relaxation, it is not guaranteed to output an integer solution (*i.e.*, a permutation). To obtain a feasible permutation we propose a homotopy-type method, in the spirit of [182, 265]. This method continuously deforms the energy functional from convex to concave, is guaranteed to produce an integer-solution and in practice outperforms standard Euclidean projection techniques. Essentially it provides a strategy for finding a local minima for the original non-convex problem using a good initial guess obtained from the convex relaxation.

Our algorithm is very flexible and can be applied to both convex and non-convex energies (in contrast with DS), and to energies combining quadratic and linear terms (in contrast with the spectral relaxation, which also requires energies with non-negative entries). It can also be easily modified to allow for additional linear constraints, injective and partial matching, and solving quadratic optimization problems over the doubly stochastic matrices. We present experiments demonstrating the effectiveness of our method in comparison to random initializations of the non-convex problem, spectral, DS, and DS+ relaxations, as well as lifted linear-programming relaxations.

We have tested our algorithm on three applications: (i) non-rigid matching; (ii) image arrangements; and (iii) coarse-to-fine matching. Comparison to state-of-the-art algorithms for these applications shows that our algorithm produces favorable results in comparable speed.

Our contributions in this work are threefold:

1. We identify the optimal initial convex and concave relaxation.
2. We show, both theoretically and experimentally that the proposed algorithm is more accurate than other popular contemporary methods. We believe that establishing a hierarchy between the various relaxation methods for quadratic matching is crucial both for applications, and for pushing forward the algorithmic state of the art, developing stronger optimization algorithms in the future.
3. Lastly, we build a simple end-to-end algorithm utilizing recent advances in optimization over the doubly-stochastic matrices to provide a scalable yet accurate algorithm for quadratic matching.

11.2 Previous work

Many works in computer vision and graphics model correspondence problems as quadratic optimization problems over permutation matrices. In many cases these problems emerge as discretizations of isometry-invariant distances between shapes [166, 165] . We focus here on the different methods to approximately solve these computationally hard problems.

Spectral relaxation The spectral relaxation for correspondence problems in computer vision has been introduced in [143] and has since become very popular in both computer vision and computer graphics, *e.g.*, [62, 149, 74, 217]. This method replaces the requirement for permutation matrices with a single constraint on the Frobenious norm of the matrices to obtain a maximal eigenvalue problem. It requires energies with positive entries to ensure the obtained solution is positive. This relaxation is scalable but is not a very tight approximation of the original problem. A related relaxation appears in [205], where the variable x is

constrained to be non-negative with $\|x\|_1 = 1$. This optimization problem is generally non-convex, but the authors suggest a method for locally minimizing this energy to obtain a sparse correspondence.

DS relaxation An alternative approach relaxes the set of permutations to its convex hull of doubly stochastic matrices [214]. When the quadratic objective is convex, this results in a convex optimization problem (quadratic program) which can be minimized globally, although the minimum may differ from the global minima of the original problem. [227] argue for the usefulness of the fuzzy maps obtained from the relaxation. For example, for symmetric shapes fuzzy maps can encode all symmetries of the shape.

[5] shows that for the convex graph matching energy the DS relaxation is equivalent to the original problem for generic asymmetric and isomorphic graphs. These results are strengthened in [76]. However when noise is present the relaxations of the convex graph matching energy will generally not be equivalent to the original problem [154] . Additionally, for concave energies the DS relaxation is always equivalent to the original problem [87], since minima of concave energies are obtained at extreme points. The challenge for non-convex energies is that global optimization over DS matrices is not tractable.

To achieve good initialization for local minimization of such problems, [182, 87, 265] suggest to minimize a sequence of energies E_t which gradually vary from a convex energy E_0 to an equivalent concave energy E_1. In this work we adopt this strategy to obtain an integer solution, and improve upon it by identifying the optimal convex and concave energies from within the energies E_t.

The authors of [79, 80] show that the DS relaxation can be made more accurate by adding a concave penalty of the form $-a\|X\|_F^2$ to the objective. To ensure the objective remains convex they suggest to choose a to be the minimial eigenvalue of the quadratic objective. We improve upon this choice by choosing a to be the minimial eigenvalue *over the doubly stochastic subspace*, leading to a provably tighter relaxation. The practical advantage of our choice (DS++) versus Fogel's choice (DS+) is significant in terms of the relaxation accuracy as demonstrated later on. The observation that this choice suffices to ensure convexity has been made in the convergence proof of the softassign algorithm [201].

Optimization of DS relaxation Specialized methods for minimization of linear energies over DS matrices [133, 58, 22, 224] using entropic regularization and the Sinkhorn algorithm are considerably more efficient than standard linear program solvers for this class of problems. Motivated by this, [200] propose an algorithm for globally minimizing quadratic energies over doubly stochastic matrices by iteratively minimizing regularized linear energies using Sinkhorn type algorithms. For the optimization in this work we applied [225] who offer a different algorithm for locally minimizing the Gromov-Wasserstein distance by iteratively solving regularized linear programs. The advantage of the latter algorithm over the former algorithm is its certified convergence to a critical point when applied to non-convex quadratic energies.

Other convex relaxations Stronger relaxations than the DS relaxation can be obtained by lifting methods which add auxiliary variables representing quadratic monomials in the original variables. This enables adding additional convex constraints on the lifted variables. A disadvantage of these methods is the large number of variables which leads to poor scalability. [124] propose in an SDP relaxation in the spirit of [270], which is shown to be stronger than both DS (for convex objective) and spectral relaxations, and in practice often achieves the global minimum of the original problem. However, it is only tractable for up to fifteen points. [50] use a lifted linear program relaxation in the spirit of [249, 2]. To deal with scalability issues they use Markov random field techniques [130] to approximate the solution of their linear programming relaxation.

Quadratic assignment Several works aim at globally solving the quadratic assignment problem using combinatorial methods such as branch and bound. According to a recent survey [150] these methods are not tractable for graphs with more than 30 points. Branch and bound methods are also in need of convex relaxation to achieve lower bounds for the optimization problem. [11] provide a quadratic programming relaxation for the quadratic assignment problem which provably achieves better lower bounds than a competing spectral relaxation using a method which combines spectral, linear, and DS relaxations. Improved lower bounds can be obtained using second order cone programming [253] and semi-definite programming [65] in $O(n^2)$ variables. All the relaxations above use the specific structure of the quadratic assignment problem while our relaxation is applicable to general quadratic objectives which do not carry this structure and are very common in computer graphics. For example, most of the correspondence energies formulated below and considered in this work cannot be formulated using the quadratic assignment energy.

Other approaches for shape matching A similar approach to the quadratic optimization approach is the functional map method (*e.g.*, [186]) which solves a quadratic optimization problem over permutations and rotation matrices, typically using high-dimensional ICP provided with some reasonable initialization. Recently [160] proposed an SDP relaxation for this problem with considerably improved scalability with respect to standard SDP relaxations.

Supervised learning techniques have been successfully applied for matching specific classes of shapes in [207, 162, 273, 245]. A different approach for matching near isometric shapes is searching for a mapping in the low dimensional space of conformal maps which contains the space of isometric maps [148, 266, 126]. More information on shape matching can be found in shape matching surveys such as [234].

11.3 Approach

Motivation Quadratic optimization problems over the set of permutation matrices arise in many contexts. Our main motivating example is the problem of finding correspondences between two metric spaces (*e.g.*, shapes) $(\mathcal{S}, d_{\mathcal{S}})$ and $(\mathcal{T}, d_{\mathcal{T}})$ which are related by a perfect or an approximate isometry. This problem can be modeled by uniformly sampling the spaces to obtain $\{\mathbf{s}_1, \dots \mathbf{s}_n\} \subseteq \mathcal{S}$ and $\{\mathbf{t}_1, \dots, \mathbf{t}_n\} \subseteq \mathcal{T}$, and then finding the permutation $X \in \Pi_n$ which minimizes an energy of the form

$$E(X) = \sum_{ijk\ell} W_{ijk\ell} X_{ij} X_{k\ell} + \sum_{ij} C_{ij} X_{ij}. \tag{80}$$

Here $W_{ijk\ell}$ is some penalty on deviation from isometry: If the points $\mathbf{s}_i, \mathbf{s}_k$ correspond to the points $\mathbf{t}_j, \mathbf{t}_\ell$ (resp.), then the distances between the pair on the source shape and the pair on the target shape should be similar. Therefore we choose

$$W_{ijk\ell} = p(d_{\mathcal{S}}(\mathbf{s}_i, \mathbf{s}_k), d_{\mathcal{T}}(\mathbf{t}_j, \mathbf{t}_\ell)) \tag{81}$$

where $p(u, v)$ is some function penalizing for deviation from the set $\{(u, v) \mid u = v\} \subseteq \mathbb{R}^2$. Several different choices of p exist in the literature.

The linear term C is sometimes used to aid the correspondence task by encouraging correspondences $\mathbf{s}_i \mapsto \mathbf{t}_j$ between points with similar isometric-invariant descriptors.

Problem statement Our goal is to solve quadratic optimization problems over the set of permutations as formulated in (80). Denoting the column stack of permutations $X \in \mathbb{R}^{n \times n}$ by the vector

$$x = [X_{11}, X_{21}, \dots, X_{nn}]^T \in \mathbb{R}^{n^2}$$

leads to a more convenient phrasing of (80):

$$\min_X \quad E(X) = x^T W x + c^T x + d \tag{82a}$$

$$\text{s.t.} \quad X \in \Pi_n \tag{82b}$$

This optimization problem is non-convex for two reasons. The first is the non-convexity of Π_n (as a discrete set of matrices), and the second is that E is often non-convex (if W is not positive-definite). As global minimization of (82) is NP-hard [150] we will be satisfied with obtaining a good approximation to the global solution of (82) using a scalable optimization algorithm. We do this by means of a convex relaxation coupled with a suitable projection algorithm for achieving integer solutions.

11.3.1 Convex relaxation

We formulate our convex relaxation by first considering a one-parameter family of equivalent formulations to (82): observe that for any permutation matrix X we have that $\|X\|_F^2 = n$. It follows that all energies of the form

$$E(X, a) = E(X) - a \|X\|_F^2 + a \cdot n \tag{83}$$

coincide *when restricted to the set of permutations*. Therefore, replacing the energy in (82) with $E(X, a)$ provides a one-parameter family of equivalent formulations. For some choices of a the energy in these formulations is convex, for example, for any $a \leq \lambda_{\min}$, where $\lambda_{\min}$ is the minimal eigenvalue of W.

For each such equivalent formulation we consider its *doubly stochastic* relaxation. That is, replacing the permutation constraint (82b) with its convex-hull, the set of doubly-stochastic matrices:

$$\min_X \quad E(X, a) \tag{84a}$$

$$\text{s.t.} \quad X\mathbf{1} = \mathbf{1}, \quad \mathbf{1}^T X = \mathbf{1}^T \tag{84b}$$

$$X \geq 0 \tag{84c}$$

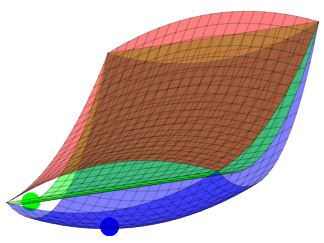
Our goal is to pick a relaxed formulation (*i.e.*, choose an a) that provides the best lower bound to the global minimum of the original problem (82). For that end we need to consider values of a that make $E(X, a)$ convex and consequently turn (84) into a convex program that provide a lower bound to the global minimum of (82).

Among all the convex programs described above we would like to choose the one which provides the tightest lower bound. We will use the following simple lemma proved in Section 11.10:

Lemma 10. *For all doubly stochastic X we have $E(X, a) \leq E(X, b)$ when $a < b$, and $E(X, a) = E(X, b)$ if and only if X is a permutation.*

An immediate conclusion from this lemma is that $\min_{X \in DS} E(X, a) \leq \min_{X \in DS} E(X, b)$ and so the best lower bound will be provided by the largest value of b for which $E(X, b)$ is convex.

See for example the inset illustrating the energy graphs for different a values for a toy-example: in red - the graph of the original (non-convex) energy with $a = 0$; in blue the energy with $a < \lambda_{\min}$; and in green $a = \lambda_{\min}$. Note that the green graph lies above the blue graph and all graphs coincide on the corners (*i.e.*, at the permutations). Since the higher the energy graph the better lower bound we achieve it is desirable to take the maximal a that still provides a convex program in (84). In



the inset the green and blue points indicate the solution of the respective
relaxed problems; in this case the green point is much closer to the sought after solution, *i.e.*, the lower-left corner.

To summarize the above discussion: the desired a is the maximal value for which $E(X, a)$ is a convex function. As noted above choosing $a = \lambda_{\min}$ in the spirit of [79], leads to a convex problem which we denote by DS+. However this is in fact not the maximal value in general. To find the maximal a we can utilize the fact that X is constrained to the affine space defined by the constraints (84b): We parameterize the affine space as $x = x_0 + Fz$, where x_0 is some permutation, F is any parameterization satisfying $F^T F = I$, and $z \in \mathbb{R}^{(n-1)^2}$. Plugging this into $E(X, a)$ provides a quadratic function in z of the form

$$z^T F^T \left(W - aI\right) Fz + \mathrm{aff}(z)$$

where $\mathrm{aff}(z)$ is some affine function of z. It follows that (84) will be convex iff $F^T(W - aI)F$ is positive semi-definite. The largest possible a fulfilling this condition is the minimal eigenvalue of $F^T W F$ which we denote by $\bar{\lambda}_{\text{min}}$. Thus our convex relaxation which we name DS++ boils down to minimizing (84) with $a = \bar{\lambda}_{\text{min}}$.

11.3.2 Projection

We now describe our method for projecting the solution of our relaxation onto the set of permutations. This method is inspired by the "convex to concave" method from [182, 87, 265], but also improves upon these works by identifying the correct interval on which the convex to concave procedure should be applied as we now describe.

Lemma 10 tells us that the global optimum of $E(X, a)$ over the doubly stochastic matrices provides an increasingly better approximation of the global optimum of the original problem (82) as we keep increasing a even beyond the convex regime, that is $a > \bar{\lambda}_{\text{min}}$. In fact, it turns out that if a is chosen large enough so that $E(X, a)$ is strictly *concave*, then the global optima of (84) and the global optima of the original problem over permutations are identical. This is because the (local or global) minima of strictly concave functions on a compact convex set are always obtained at the extreme points of the set. In our case, the permutations are these extreme points.

This leads to a natural approach to approximate the global optimum of (82): Solve the above convex problem with $a = \bar{\lambda}_{\text{min}}$ and then start increasing $a > \bar{\lambda}_{\text{min}}$ until an integer solution is found. We choose a finite sequence $a_0 < a_1 < \ldots < a_N$, where $a_0 = \bar{\lambda}_{\text{min}}$ and $E(X, a_N)$ is strictly concave. We begin by solving (84) with a_0 which is exactly the convex relaxation described above and obtain a minimizer X_0. We then iteratively locally minimize (84) with $a = a_i$ using as an initialization the previous solution X_{i-1}. The reasoning behind this strategy is that when a_i and a_{i-1} are close a good solution for the latter should provide a good initialization for the former, so that at the end of the process we obtain a good initial guess for the minimization of $E(X, a_N)$, which is equivalent to the original integer program. We stress that although the obtained solution may only be a local minimum, it will necessarily be a permutation.

To ensure that $E(X, a_N)$ is strictly concave we can choose any a_N larger than $\bar{\lambda}_{\text{max}}$, which analogously to $\bar{\lambda}_{\text{min}}$ is defined as the largest eigenvalue of $F^T W F$. In practice we select $a_N = \bar{\lambda}_{\text{max}}$ which in the experiments we conducted is sufficient for obtaining integer solutions. We then took a_i by uniformly sampling $[a_0, a_N]$ where unless stated otherwise we used ten samplings ($N = 9$). Throughout the section we will use the term *DS++ algorithm* to refer to our complete method (relaxation+projection) and DS++ or DS++ relaxation to refer only to the relaxation component.

Figure 48 shows the correspondences (encoded in a specific row of X) obtained at different stages of the projection procedure when running our algorithm on the FAUST dataset [30] as described below. The figure

shows the correspondences obtained from optimizing $E(X, a_i)$ for $i = 0, 4, 7, N = 9$.

Our algorithm is summarized in Algorithm 1: In Section 11.5 we discuss efficient methods for implementing this algorithm.

Input: The energy components W, c, d

Compute $\bar{\lambda}_{\text{min}}, \bar{\lambda}_{\text{max}}$ of $F^T W F$;
Choose $N + 1$ uniform samples $a_0 = \bar{\lambda}_{\text{min}}, a_1, \ldots, a_N = \bar{\lambda}_{\text{max}}$;
Solve (84) with $a = a_0$ to obtain X_0 ;
for $i = 1 \ldots N$ **do**
  Solve (84) with $a = a_i$ initialized from X_{i-1} to obtain X_i ;
end

Output: The permutation X_N

Algorithm 1: DS++ algorithm

11.4 Comparison with other relaxations

The purpose of this section is to theoretically compare our relaxation with common competing relaxations. We prove

Theorem 16. *The DS++ relaxation is more accurate than DS+, which in turn is more accurate than the spectral and doubly stochastic relaxation.*
The SDP relaxation of [124] is more accurate than all the relaxations mentioned above.

Our strategy for proving this claim is formulating all relaxations in a unified framework, using the SDP lifting technique in [124], that in turn readily enables comparison of the different relaxations.

The first step in constructing SDP relaxations is transforming the original problem (82) into an equivalent optimization problem in a higher dimension.The higher dimension problem is formulated over the set:

$$\Pi_n^{\uparrow} = \left\{ (X, Y) \;\middle|\; X \in \Pi_n, \quad Y = xx^T \right\}$$

Using the identity

$$\text{tr} WY = \text{tr} Wxx^T = x^T W x$$

we obtain an equivalent formulation to (82):

$$\begin{aligned} \min_{X,Y} \quad & E(X, Y) = \text{tr} WY + c^T x + d \\ \text{s.t.} \quad & (X, Y) \in \Pi_n^{\uparrow} \end{aligned}$$

SDP relaxations are constructed by relaxing the constraint $(X, Y) \in \Pi_n^{\uparrow}$ using linear constraints on X, Y and the semi-definite constraint $Y \succeq xx^T$.

[124] showed that the spectral and doubly stochastic relaxations are equivalent to the following SDP relaxations:

$$(\text{S}^{\uparrow}) \quad \begin{aligned} \max \quad & E(X, Y) \\ \text{s.t.} \quad & \text{tr}\, Y = n \\ & Y \succeq xx^T \end{aligned} \qquad (\text{DS}^{\uparrow}) \quad \begin{aligned} \max \quad & E(X, Y) \\ \text{s.t.} \quad & X \in \text{DS} \\ & Y \succeq xx^T \end{aligned}$$

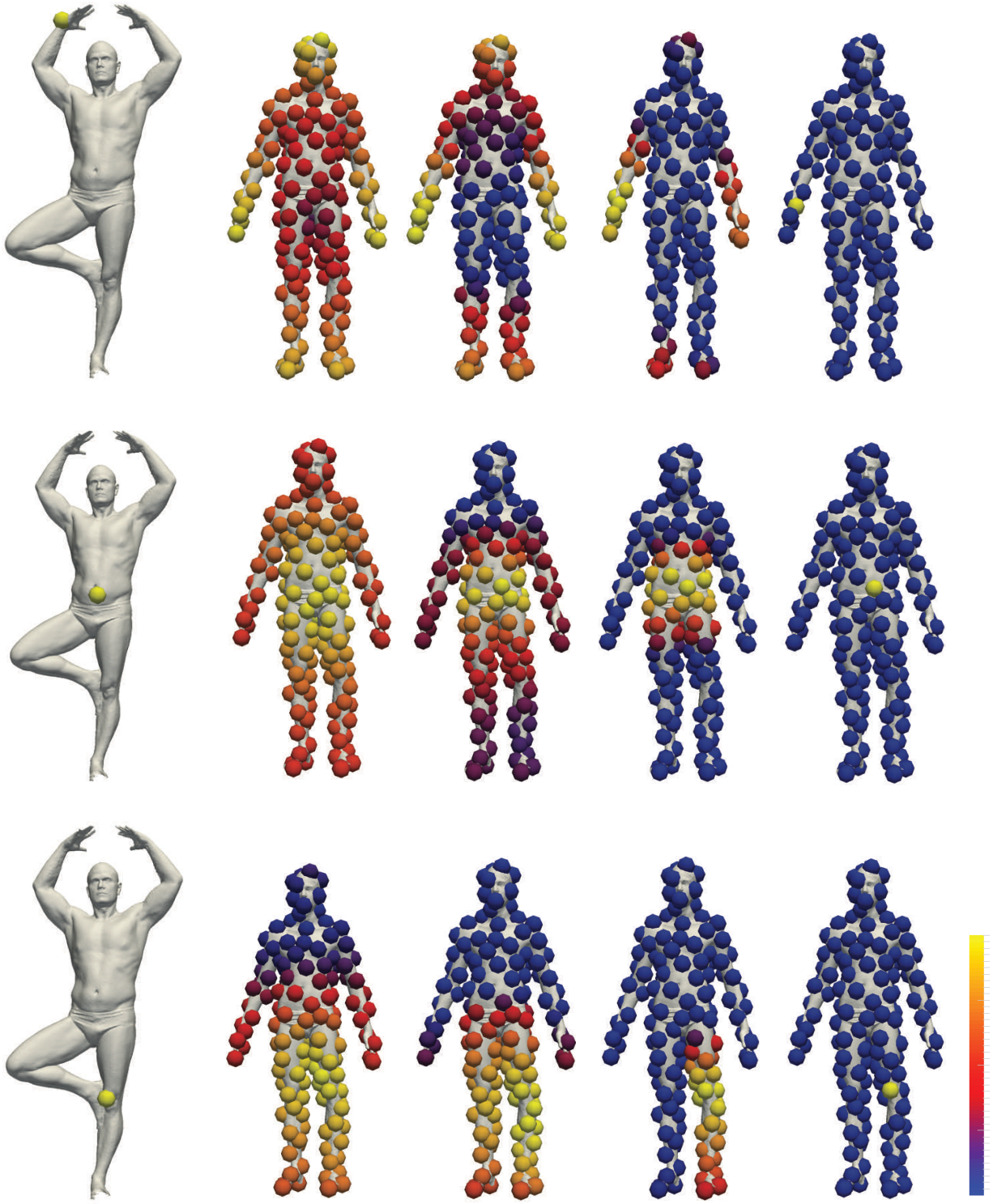


Figure 48: Visualization of the projection procedure. For each point on the source (left) a fuzzy correspondence is obtained by minimizing the convex energy (second from the left). The correspondence gradually becomes sharper as the projection procedure proceeds until the final step of minimizing a concave energy where a well defined map is obtained (right).

We note that the spectral relaxation is applicable only when $c = 0$, and the DS relaxation is tractable only when the objective is convex, *i.e.*, $W \succeq 0$. The equivalence holds under these assumptions.

Given this new formulation of spectral and DS, an immediate method for improving both relaxations is considering the *Intersection-SDP*, obtained by enforcing the constraints from both $(\mathrm{DS}^{\uparrow})$ and $(\mathrm{S}^{\uparrow})$. The relaxation can be further improved by adding additional linear constraints on (X, Y). This is the strategy followed by [124] to achieve their final tight relaxation which is presented in Eq. (88). The main limitation of this approach is its prohibitive computational price resulting from solving SDPs with $O(n^4)$ variables, in strong contrast to the original formulation of spectral and DS that uses only n^2 variables (*i.e.*, the permutation X). This naturally leads to the research question we posed in the introduction, which we can now state in more detail:

Question: Is it possible to construct an SDP relaxation which is stronger than $(\mathrm{DS}^{\uparrow})$ and $(\mathrm{S}^{\uparrow})$, and yet is equivalent to a tractable and scalable optimization problem with n^2 variables?

We answer this question affirmatively by showing that the Intersection-SDP is in fact equivalent to DS+. Additionally DS++ is equivalent to a stronger SDP relaxation which includes all constraints from the *Intersection-SDP*, as well as the following additional $2n^3$ constraints: Let us write the linear equality constraints appearing in the definition of the DS matrices (*i.e.*, (84b)) in the form $Ax = b$. Then any $(X, Y) \in \Pi_n^{\uparrow}$ in particular satisfies $Axx^T = bx^T$ and therefore also:

$$AY = bx^T$$

Adding these constraints to the Intersection-SDP we obtain

$$\begin{aligned} \min_{X,Y} \quad & E(X, Y) && \text{(85a)} \\ \text{s.t.} \quad & \mathrm{tr}Y = n && \text{(85b)} \\ & X \geq 0 && \text{(85c)} \\ & Ax = b && \text{(85d)} \\ & AY = bx^T && \text{(85e)} \\ & Y \succeq xx^T && \text{(85f)} \end{aligned}$$

Theorem 16 now follows from:

Lemma 11. *1. The Intersection-SDP is equivalent to DS+.*

2. The SDP relaxation in (85) *is equivalent to DS++.*

3. The SDP relaxation of [124] can be obtained by adding additional linear constraints to (85)*.*

We prove the lemma in the Section 11.10.

11.5 Implementation details

Entropic regularization Optimization of (84) can be done using general purpose non-convex solvers such as Matlab's *fmincon*, or solvers for convex and non-convex quadratic programs. We opted for the recent method of Solomon et al.[225] that introduced a specialized scalable solver for local minimization of regularized quadratic functionals over the set of doubly stochastic matrices.

The algorithm of [225] is based on an efficient algorithm for optimizing the KL divergence

$$KL(x|y) = \langle x, \log x \rangle - \langle x, \log y \rangle$$

Figure 49: Typical maps obtained using our method on the FAUST dataset [30]. In each pair: left mesh is colored linearly and the computed map is used to transfer the coloring to the target, right mesh.

Figure 50: Image arrangement according to the mean color of images using the DS++ algorithm. Table 14 shows corresponding quantitative results.

where x is the column stack of a doubly stochastic matrix X and y is some fixed positive vector. The solution for the KKT equations of this problem can be obtained analytically for x, up to scaling of the rows and columns, which is performed by the efficient Sinkhorn algorithm. See [58] for more details.

The algorithm of [225] minimizes quadratic functionals $f(x) = x^T Hx + c^T x$ (where in our case $H = W - aI$) over doubly stochastic matrices by iteratively optimizing KL-divergence problems. First the original quadratic functional is regularized by adding a barrier function $\alpha \langle x, \log x \rangle$ keeping the entries of x away from zero to obtain a new functional

$$f_\alpha(x) = f(x) + \alpha \langle x, \log x \rangle$$

The parameter α is chosen to be some small positive number so that its effect on the functional is small. We then define $g_\alpha(x) = \exp\left(-\alpha^{-1}(Hx + c)\right)$ so that

$$f_\alpha(x) = \alpha KL(x|g_\alpha(x))$$

We then optimize f_α iteratively: In iteration $k+1$, g_α is held fixed at its previous value $x = x_k$, and an additional term $KL(x|x_k)$ is added penalizing large deviations of x from x_k. More precisely, x_{k+1} is defined to be the minimizer of

$$\eta KL(x|g_\alpha(x_k)) + (1-\eta)KL(x|x_k) = KL(x|g_\alpha^\eta(x_k) \odot x_k^{1-\eta})$$

where $\odot$ denotes entry-wise multiplication of vectors. For small enough values of η, [225] prove that the algorithm converges to a local minimun of $f_\alpha(x)$.

In our implementation we use $\eta = 0.01$. We choose the smallest possible α so that all entries of the argument of the exponent in the definition of g_α are in $[-100, 100]$. This choice is motivated by the requirement of choosing small α coupled with the breakdown of matlab's exponent function at around e^{700}. Note that this choice requires $\alpha = \alpha_k$ to update at each iteration. We find that with this choice of α the regularization term has little effect on the energy and we obtain final solutions which are close to being permutations. To achieve a perfect permutation we project the final solution using the L_2 projection. The L_2 projection is computed by minimizing a linear program as described, *e.g.*, in [265].

Computing $\bar{\lambda}_{\text{min}}$ and $\bar{\lambda}_{\text{max}}$ We compute $\bar{\lambda}_{\text{min}}$ and $\bar{\lambda}_{\text{max}}$ by solving two maximal magnitude eigenvalue problems: We first solve for the maximal magnitude eigenvalue of $F^T WF$. If this eigenvalue is positive then it is equal to $\bar{\lambda}_{\text{max}}$. We can then find $\bar{\lambda}_{\text{min}}$ by translating our matrix by $\bar{\lambda}_{\text{max}}$ to obtain a positive-definite matrix $\bar{\lambda}_{\text{max}} I - F^T WF$ whose maximal eigenvalue η is related to the minimal eigenvalue of the original matrix via $\bar{\lambda}_{\text{min}} = \bar{\lambda}_{\text{max}} - \eta$.

If the solution of the first maximal magnitude problem is negative then this eigenvalue is $\bar{\lambda}_{\text{min}}$, and we can use a process similar to the one described above to obtain $\bar{\lambda}_{\text{max}}$.

Solving maximal magnitude eigenvalue problems requires repeated multiplication of vectors $v \in \mathbb{R}^{(n-1)^2}$ by the matrix $F^T WF$, where $W \in \mathbb{R}^{n^2 \times n^2}$ and $F \in \mathbb{R}^{n^2 \times (n-1)^2}$. If W is sparse, computing Fv can become a computational bottleneck. To avoid this problem, we note that $F^T WF$ has the same maximal eigenvalue as the matrix $FF^T WFF^T$ and so compute the maximal eigenvalue of the latter matrix. The advantage of this is that multiplication by the matrix $P = FF^T$ can be computed efficiently:

Since P is the orthogonal projection onto Image(F), we can use the identity $Pu = u - P_\perp u$ where $P_\perp$ is the projection onto the orthogonal complement of Image(F). The orthogonal complement is of dimension $2n-1$ and therefore $P_\perp u = F_\perp F_\perp^T u$ where $F_\perp \in \mathbb{R}^{n^2 \times (2n-1)}$.

We solve the maximal magnitude eigenvalue problems using Matlab's function *eigs*.

11.6 Generalizations

Injective matching Our method can be applied with minor changes to injective matching. The input of injective matching is k points sampled from the source shape $\mathcal{S}$ and $n > k$ points sampled from the target shape $\mathcal{T}$, and the goal is to match the k points from $\mathcal{S}$ injectively to a subset of $\mathcal{T}$ of size k.

Matrices $X \in \mathbb{R}^{k \times n}$ representing injective matching have entries in $\{0, 1\}$, and have a unique unit entry in each row, and at most one unit entry in each column. This set can be relaxed using the constraints:

$$X\mathbf{1} = \mathbf{1} \quad , \quad \mathbf{1}^T X \leq \mathbf{1}^T \tag{86a}$$

$$\mathbf{1}^T X \mathbf{1} = k \tag{86b}$$

$$X \geq 0 \tag{86c}$$

We now add a row with positive entries to the variable matrix X to obtain a matrix $\bar{X} \in \mathbb{R}^{(k+1) \times n}$. The original matrix X satisfies the injective constraints described above if $\bar{X}$ satisfies

$$\bar{X}\mathbf{1} = (n_1 - k, 1, \ldots, 1)^T, \quad \mathbf{1}^T \bar{X} = \mathbf{1}^T$$
$$\bar{X} \geq 0$$

These constraints are identical to the constraint defining DS, up to the value of the marginals which have no affect on our algorithm. As a result we can solve injective matching problems without any modification of our framework.

Partial matching The input of partial matching is n_1, n_2 points sampled from the source and target shape, and the goal is to match $k \leq n_1, n_2$ points from $\mathcal{S}$ injectively to a subset of $\mathcal{T}$ of size k. We do not pursue this problem in this work as we did not find partial matching necessary for our applications. However we believe our framework can be applied to such problems by adding a row and column to the matching matrix X.

Adding linear constraints Modeling of different matching problems can suggest adding additional linear constraints on X that can be added directly to our optimization technique. Additional linear equality constraints further decrease the dimension of the affine space X is constrained to and as a result make the interval $[\bar{\lambda}_{\text{min}}, \bar{\lambda}_{\text{max}}]$ smaller, leading to more accurate optimization. We note however that incorporating linear constraints into the optimization method of [225] is not straightforward.

Upsampling Upsampling refers to the task of interpolating correspondences between r points sampled from source and target metric spaces to a match between a finer sampling of $k >> r$ source points and $n \geq k$ target points. We suggest two strategies for this problem: *Limited support interpolation* and *greedy interpolation*.

Limited support interpolation uses the initially matched r points to rule out correspondences between the finely sampled points. The method of ruling out correspondences is discussed in the section 11.11. We enforce the obtained sparsity pattern by writing $X = X_{\text{permissible}} + X_{\text{forbidden}}$, where the first matrix is zero in all forbidden entries and the second is zero in all permissible entries. We then minimize the original energy $E(X)$ only on the permissible entries, and add a quadratic penalty for the forbidden entries. That is, we minimize

$$E(X_{\text{permissible}}) + \rho \left\| X_{\text{forbidden}} \right\|_F^2$$

choosing some large $\rho > 0$. The sparsity of $X_{\text{permissible}}$ enables minimizing this energy for large k, n for which minimizing the original energy is intractable.

When k, n are large we use greedy interpolation. We match each source point s_i separately. We do this by optimizing over correspondences between $r + 1$ source points and n target points, where the $r + 1$ points are the r known points and the point s_i. Since there are only $n - r$ such correspondences optimization can be performed globally by checking all possible correspondences.

Optimization over doubly stochastic matrices Our main focus was on optimization problems over permutations. However in certain cases the requested output from the optimization algorithm may be a doubly stochastic matrix and not a permutation. When the energy E is non convex this still remains a non-convex problem. For such optimization problems our method can be applied by taking samples a_i from the interval $[\bar{\lambda}_{\text{min}}, 0]$, since minimization of (84) with $a = 0$ is the problem to be solved while minimization of (84) with $a = \bar{\lambda}_{\text{max}}$ forces a permutation solution.

11.7 Evaluation

In this section we evaluate our algorithm and compare its performance with relevant state of the art algorithms. We ran all experiments on the 100 mesh pairs from the FAUST dataset [30] which were used in the evaluation protocol of [50].

Comparison with [225] In figure 51(b) we compare our method for minimizing non-convex functionals with the local minimization algorithm of [225]. Since this method is aimed at solving non-convex functionals over doubly-stochastic matrices, we run our algorithm using samples in $[\bar{\lambda}_{\text{min}}, 0]$ as explained in Section 11.6. We sample 200 points from each mesh using farthest point sampling [73], and optimize the Gromov-Wasserstein (GW) functional advocated in [225], which amounts to choosing p from (81) to be $p(u, v) = (u - v)^2$. As local minimization depends on initialization we locally minimize 1000 times per mesh pair, using 1000 different random initializations. The initializations are obtained by randomly generating a positive matrix in $\mathbb{R}^{200\times 200}$ with uniform distribution, and projecting the result onto the doubly stochastic matrices using the Sinkhorn algorithm. As can be seen in the figure our algorithm, using only ten iterations, was more accurate than all the local minima found using random initializations. As a baseline for comparison we note that the difference in energy between randomly drawn permutations and our solution was around 5000, while the difference in energy shown in the graph is around 500. Figure 52 visualizes the advantages of the fuzzy maps obtained by our algorithm in this experiment over the best of the 1000 random maps generated by [225].

Projection evaluation In figure 51(a) we examine how the result obtained from our projection method is influenced by the number of points N sampled from $[\bar{\lambda}_{\text{min}}, \bar{\lambda}_{\text{max}}]$. We compared the behavior of our relaxation with several different choices of N as well as with the standard L_2 projection onto the set of permutations. As expected, our projection is always better that the L_2 projection, and the projection improves as the number of samples is increased.

Comparison with other relaxations We compare our method with other relaxation based techniques. In figure 51 (c)-(d) we compare our relaxation with the spectral relaxation, the DS+ relaxation, and the SDP relaxation of [124]. In this experiment the energy we use is non-convex so DS is not applicable.

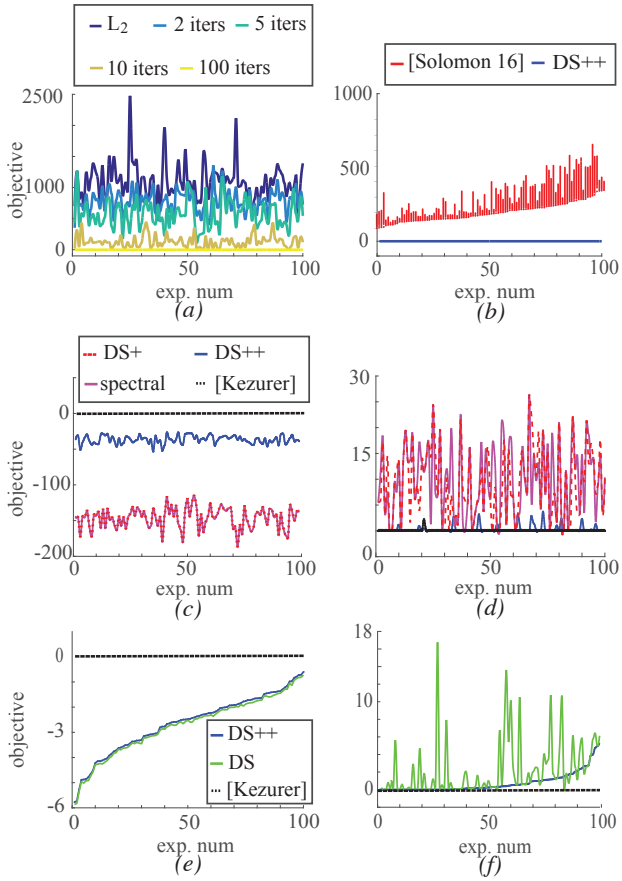


Figure 51: Evaluation of our algorithm. (a) compares the L_2 projection with our projection. Even with only two iterations our projection improves upon the L_2 projection. Additional iterations yield better accuracy at the price of time complexity. (b) compares minimization of the Gromov-Wasserstein distance with our algorithm and [225] with 1000 random initializations. In all cases we attain a lower objective value. The second row compares lower bounds (c) and upper bounds (d) obtained by the DS++ algorithm, DS+, spectral, and [124]. As predicted by Theorem 16 our lower bounds and upper bounds are outperformed by [124] who are able to attain the ground truth in these cases, but improve upon those of the remaining methods. The third row compares lower bounds (e) and upper bounds (f) obtained by the DS++ algorithm, DS and [124] for the convex graph matching functional. The lower bound of the DS++ algorithm modestly improves DS's, while the upper bounds substantially improves the upper bounds of DS's L_2 projection.

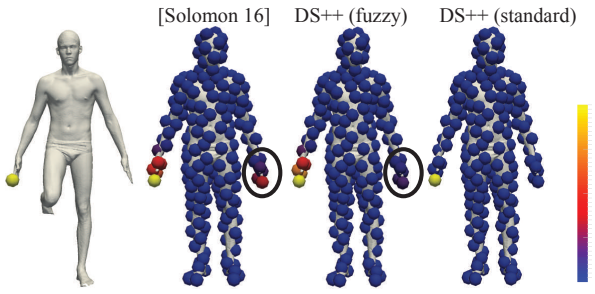


Figure 52: Optimization over fuzzy maps using [225] and the DS++ algorithm as described in Section 11.6. The best fuzzy map obtained by [225] with 1000 random initializations is less accurate than our fuzzy map (middle), as our map gives lower probability to mapping the right hand of the source to the left hand of the target. See also Figure 51 (b). The rightmost image shows the sharp map obtained by the standard DS++ algorithm.

We sampled 10 points from both meshes, and minimized the (non-convex) functional selected by [124], which amounts to choosing p from (81) to be

$$p(u,v) = -\exp\left(\frac{-(u-v)^2}{\sigma^2}\right)$$

We choose the parameter $\sigma = 0.2$. For the minimization we used all four relaxations, obtaining a lower bound for the optimal value, Figure 51 (c). We then projected the solutions obtained onto the set of permutations, thus obtaining an upper bound, Figure 51 (d). For methods other than the DS++ algorithm we used the L_2 projection. In all experiments the upper and lower bounds provided by the SDP relaxation of [124] were identical, thus proving that the SDP relaxation found the globally optimal solution. Additionally, in all experiments the upper bound and lower bound provided by our relaxation were superior to those provided by the spectral method, and our projection attained the global minimum in approximately 80% of the experiments in contrast to 11% obtained by the L_2 projection of the spectral method. The differences between the spectral relaxation and the stronger DS+ relaxation were found to be negligible.

In figure 51(e)-(f) we perform the same experiment, but now we minimize the convex graph matching functional $E(X) = \|AX - XB\|_F^2$ from [5] for which the classical DS relaxation is applicable. Here again the ground truth is achieved by the SDP relaxation. Our relaxation can be seen to modestly improve the lower bound obtained by the classical DS relaxation, while our projection method substantially improves upon the standard projection.

11.8 Applications

We have tested our method for three applications: non-rigid shape matching, image arrangement, and coarse-to-fine matching.

Non-rigid matching We evaluated the performance of our algorithm for non-rigid matching on the FAUST dataset [30]. We compared to [50] which demonstrated superb state of the art results on this dataset (for non learning-based methods). For a fair comparison we used an identical pipeline to [50], including their isometric energy modeling and extrinsic regularization term. We first use the DS++ algorithm to match $n = 160, k = 150$ points, then upsampled to $n = 450, k = 420$ using limited support interpolation and to $n = 5000, k = 1000$ using greedy interpolation, as described in Section 11.6; the final point resolution is as in [50].

Figure 53 depicts the results of the DS++ algorithm and [50]. As can be read from the graphs, our algorithm compares favorably in both the inter and intra class matching scenarios in terms of cumulative error distribution and average error. These results are consistent for both the convex relaxation part (top row) and the upsampled final map (middle row); The graphs show our results both with our upsampling as described above (denoted by DS++(1)) and the results of combining our relaxation with the upsampling of [50] (DS++(2)). We find DS++(1) to be better on the inter class, and DS++(2) is marginally better on the intra class. The error is calculated on a set of 52 ground truth points in each mesh as in [160]. Figure 47 (left), and 49 show typical examples of maps computed using the DS++ algorithm in this experiment.

Image arrangement The task of arranging image collections in a grid has received increasing attention in recent years [198, 228, 82, 48]. Image arrangement is an instance of metric matching: the first metric space is the collection of images and a dissimilarity measure defined between pairs of images; and the second, target metric space is a 2D grid (generally, a graph) with its natural Euclidean metric.

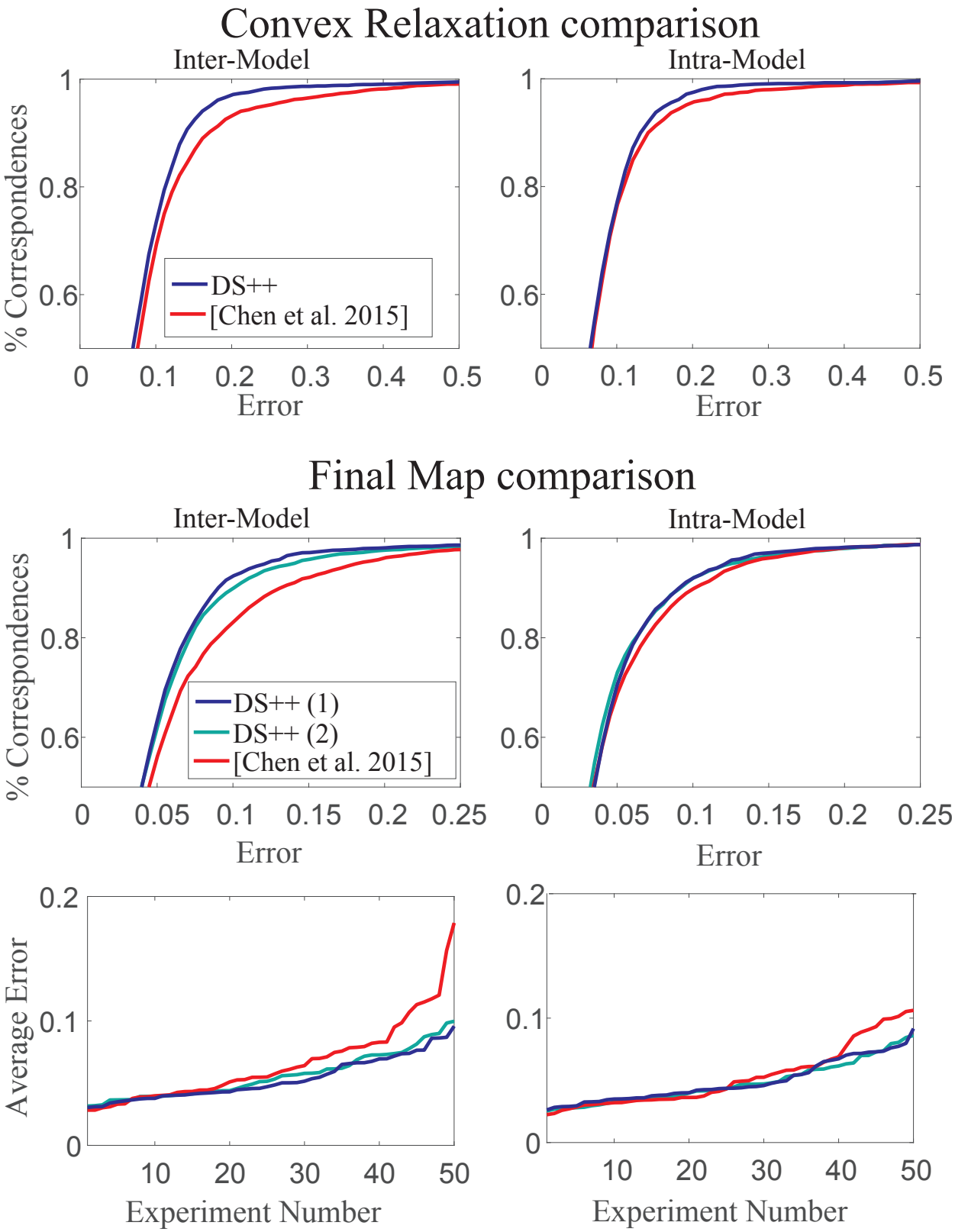


Figure 53: Non-rigid matching. Cumulative and average errors achieved on the FAUST dataset [30] by the DS++ algorithm compared to [50]. Top row compares only the convex relaxation part of both methods; bottom two rows compare final maps after upsampling. DS++(1) uses our upsampling method and DS++(2) uses the upsampling method of [50].

dataset	feature	improvement	rand average	Fried mean	our mean	functional	swaps?	grid size
Random colors	color	**28.33%**	0.478	0.259	**0.198**	Fried	no	12
Random colors	color	**8.86%**	0.478	0.219	**0.197**	Fried	yes	12
Random colors	color	**3.46%**	0.478	0.219	**0.211**	GW	yes	12
SUN dataset	color	**2.05%**	0.581	0.244	**0.237**	Fried	no	10
SUN dataset	color	**0.57%**	0.581	0.225	**0.223**	Fried	yes	10
SUN dataset	deep feature object	**55.97%**	0.433	0.345	**0.295**	Fried	no	14
SUN dataset	deep feature object	**6.31%**	0.433	0.300	**0.292**	Fried	yes	14
LFW	deep feature face	**50.70%**	0.422	0.355	**0.320**	Fried	no	14
LFW	deep feature face	**2.81%**	0.422	0.321	**0.318**	Fried	yes	14
Illumination	Raw L_2 distance	**59.08%**	0.509	0.320	**0.208**	Fried	no	10
Illumination	Raw L_2 distance	**9.94%**	0.509	0.232	**0.204**	Fried	yes	10
Illumination	Raw L_2 distance	**13.70%**	0.527	0.273	**0.238**	Fried	yes	10
Illumination	Raw L_2 distance	**10.65%**	0.518	0.259	**0.231**	Fried	yes	10

Table 14: Image arrangement comparison. We compare DS++ to [82] in arranging different sets of images in a grid with different affinity measures between images; see text for more details.

[82] suggested an energy functional for generating image arrangements, which are represented by a permutation matrix X. Their choice of energy functional was supported by a user study. This energy functional is:

$$E(X) = \min_{c>0} \sum_{ijkl} \left| c \cdot d_{ik} - d'_{jl} \right| X_{ij} X_{kl} \tag{87}$$

where d, d' are the distance measures between images and grid points respectively, and c is the unknown scale factor between the two metric spaces. [82] suggested a two step algorithm to approximate the minimizer of the above energy over the set of permutations: The first step is a dimensionality reduction, and the second is linear assignment to a grid according to Euclidean distances. Fried et al.demonstrated significant quantitative improvement over previous state of the art methods.

We perform image arrangement by using an alternative method for optimizing the energy (87). We fix c so that the mean of d and d' are the same, which leads to a quadratic matching energy which we optimize over permutations using the DS++ algorithm.

Table 14 summarizes quantitative comparison of the DS++ algorithm and [82] on a collection of different image sets and dissimilarity measures. Each row shows the mean energies over 100 experiments of Fried et al.DS++, and random assignments which provide a baseline for comparison; in each experiment we randomized a subset of images from the relevant set of images and generated image arrangements using the two methods. [82] suggested an optional post processing step in which random swaps are generated and applied in case they reduce the energy; our experiment measures the performance of both algorithms with and without random swaps. The first set of experiments tries to arrange random colors in a grid. The second set of experiments uses the mean image color for images form the SUN database [254]. The third set uses the last layer of a deep neural network trained for object recognition [49] as image features, again for images in the SUN dataset. The fourth set of experiments organizes randomly sampled images from the Labeled Faces in the Wild (LFW) dataset [111] according to similar deep features taken from the net trained by [189]. For the last experiment, we rendered three 3D models from the SHREC07 dataset [91] from various illumination directions and ordered them according to

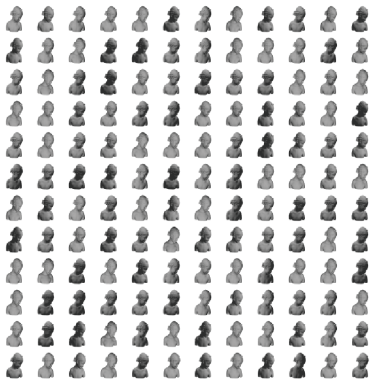


Figure 54: Random lighting.

the raw L_2 distance between pairs of images.

Our algorithm outperformed [82] in all experiments (in some cases our algorithm achieved an improvement of more than 50%). Figures 47, 50 and 57 show some image arrangements from these experiments. Note, for example, how similar faces are clustered together in Figure 57 (a), and similar objects are clustered in Figure 47 (right). Further note how the image arrangement in Figure 57 (b) nicely recovered the two dimensional parameter of the lighting direction, where Figure 54 shows the input random lighting directions renderings of a 3D models.

Coarse-to-fine matching We consider the problem of matching two shapes $\mathcal{S}$ and $\mathcal{T}$ using sparse correspondences specified by the user. User input can be especially helpful for highly non-isometric matching problems where semantic knowledge is often necessary for achieving high quality correspondences. The inset shows such example where three points (indicated by colored circles) are used to infer correspondences between a horse and a giraffe.

We assume the user supplied a sparse set of point correspondences, $\mathbf{s}_i \to \mathbf{t}_i$, $i = 1, \ldots, d$, and the goal is to complete this set to a full correspondence set between the shapes $\mathcal{S} = \{\mathbf{s}_1, \ldots, \mathbf{s}_n\}$ and $\mathcal{T} = \{\mathbf{t}_1, \ldots, \mathbf{t}_k\}$. Our general strategy is to use a linear term to enforce the user supplied constraints, and a quadratic term to encourage maps with low distortion.

For a quadratic term we propose a "log-GW" functional. This functional amounts to choosing $p(u, v) = d_L(u, v)^2$ for the definition of W in (81), where d_L is a metric on $\mathbb{R}_+$ defined by

$$d_L(u, v) = \left| \log \frac{u}{v} \right|$$

This metric punishes for high relative distortion between u, v, and thus is more suitable for our cause than the standard Euclidean metric used for the GW functional.

As a linear term we propose

$$L(X) = w \left[\sum_{i=1}^{d} (-X_{ii}) + \sum_{i=1}^{d} \sum_{k,\ell} p\left(d(\mathbf{s}_i, \mathbf{s}_q), d(\mathbf{t}_i, \mathbf{t}_r)\right) X_{k\ell} \right]$$

The first summand from the left penalizes matchings which violate the known correspondences, while the second summand penalizes matchings which cause high distortion of distances to the user supplied points. The parameter w controls the strength of the linear term. In our experiments we chose $w = 0.01 \left\| F^T W F \right\|$, where $\left\| F^T W F \right\| = \max\{|\bar{\lambda}_{\min}|, |\bar{\lambda}_{\max}|\}$ is the spectral norm of the quadratic form..

We applied the algorithm for coarse-to-fine matching on the SHREC dataset [**Shrec**], using $d = 3, 4, 5, 6$ of the labeled ground truth points and evaluating the error of the obtained correspondence on the remaining $\ell - d$ points. The number of labeled points ℓ is class dependent and varies between 36 and 7.

Representative results are shown in Figure 56. The graph on the left hand side of Figure 55 shows that our algorithm is able to use this minimal user supplied information to obtain significantly better results than those obtained by the unaided BIM algorithm [**BIM**]. The graph compares the algorithms in terms of cumulative error distribution over all 218 SHREC pairs for which BIM results are available.

The graph on the right hand side of Figure 55 shows our results outperform the algorithm presented in [225] for matching aided by user supplied correspondences. Both algorithms were supplied with 6 ground truth

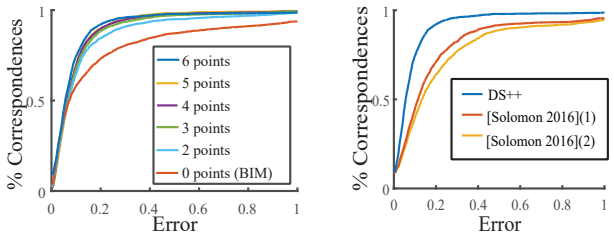


Figure 55: Matching aided by sparse user correspondence. The left graphs shows that our algorithm can exploit user supplied information to outperform state of the art unsupervised methods such as BIM. The right graph shows DS++ outperforms the algorithm of [225] for user aided matching.

points. We ran the algorithm of [225] matching $n = 250, k = 250$ points (we take $n = k$ since [225] does not support injective matching) and using the maximal-coordinate projection they chose to achieve a permutation solution. These results are denoted by (2) in the graph. However we find that better results are achieved when matching only 100 points, and when using the L_2 projection. These results are denoted by (1) in the graph.

Timing Typical running times of our optimization algorithm for the energy of [50] matching $n = k = 50$ points takes 6 seconds; $n = k = 100$ takes 26 seconds; and $n = 160$, $k = 150$ points takes around 2 minutes (130 seconds). The precomputation of $\bar{\lambda}_{\text{min}}$ and $\bar{\lambda}_{\text{max}}$ with these parameters requires around 15 seconds, the L_2 projection requires 5 seconds, and the remaining time is required for our optimization algorithm.

Parameter values of $n = k = 100$ (as well as $n = k = 12^2, 14^2$) were used in the image arrangement task from Section 11.8, and parameter values $n = 160, k = 150$ were used for our results on the FAUST dataset. For the latter application we also upsampled to $n = 450, k = 420$ using limited support interpolation and then upsampled to $k = 1000, n = 5000$ using greedy interpolation as described in Section 11.6.Limited support interpolation required 117 seconds and greedy interpolation required 15 seconds. The total time for this application is around 4.5 minutes.

The efficiency of our algorithm significantly improves if the product of the quadratic form's matrix W with a vector $x \in \mathbb{R}^{n^2}$ can be computed efficiently. This is illustrated by the fact that optimization of the sparse functional we construct for the task of resolution improvement with $n = 450$ takes similar time as optimization of the non-sparse functional of [50] with $n = 160$.

Another case where the product Wx can be computed efficiently is the GW or log-GW energy. In both cases the product can be computed by multiplication of matrices of size $n \times n$ (see [225] for the derivation), thus using $O(n^3)$ operations instead of the $O(n^4)$ operations necessary for general W . Using this energy, matching $n = 160$ points to $k = 150$ points takes only 12 seconds, matching $n = 270$ points to $k = 250$ points takes 22 seconds, matching $n = 500$ to $k = 500$ takes 82 seconds, and for $n = k = 1,000$ we require around six minutes (368 seconds).

The efficiency of our algorithm depends linearly on N. Minimizing the GW energy with $n = 270, k = 250$ using $N + 1 = 5$ sample takes 12 seconds, approximately half of the time needed when using $N + 1 = 10$ samples. These parameters were used for our results on matching with user input described in Section 11.8. For this task we also used greedy interpolation to obtain full maps between the shapes, which required an additional 18 seconds. Overall this application required around half a minute.

Our algorithm was implemented on Matlab. All running times were computed using an Intel i7-3970X CPU 3.50 GHz.

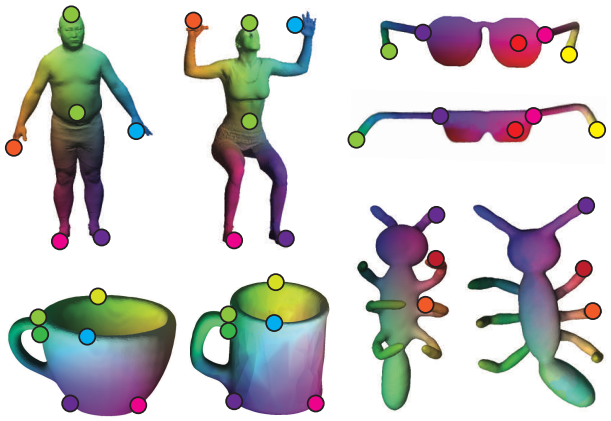


Figure 56: Correspondences obtained using user input. Correspondences were obtained using 6 user input points, with the exception of the correspondence between ants found using only 3 points. Note our method is applicable to surfaces of arbitrary genus such as the genus 1 mugs.

11.9 Conclusions

We have introduced the DS++ algorithm for approximating the minimizer of a general quadratic energy over the set of permutations. Our algorithms contains two components: (i) A quadratic program convex relaxation that is guaranteed to be better than the prevalent doubly stochastic and spectral relaxations; and (ii) A projection procedure that continuously changes the energy to recover a locally optimal permutation, using the convex relaxation as an initialization. We have used recent progress in optimal transport to build an efficient implementation to the algorithm.

The main limitation of our algorithm is that it does not achieve the global minima of the energies optimized. Partially this is unavoidable and due to the computational hardness of our problem. However the experimental results in Figure 51 show that accuracy can be improved by the SDP method of [124] which while computationally demanding, can still be solved in polynomial time. Our future goal is to search for relaxations whose accuracy is close to those of [124] but which are also fairly scalable. One concrete direction of research could be finding the 'best' quadratic programming relaxation in $O(n^2)$ variables.

11.10 Proofs

Proof of Lemma 10 The function $f(X) = \|X\|_F^2$ is strictly convex and satisfies $f(X) = n$ for all extreme points of DS. Therefore

$$E(X, b) - E(X, a) = (a - b)\|X\|_F^2 + (b - a)n \geq 0$$

with equality iff X is a permutation.

Proof of Lemma 11 We omit the proof of the first part of the lemma since it is similar to, and somewhat easier than, the proof of the second part.

To show equivalence of DS++ with (85) we show that every minimizer of DS++ defines a feasible point for (85) with equal energy and vice versa.

Let x be the minimizer of $E(X, \bar{\lambda}_{\text{min}})$ over the doubly stochastic matrices and let v be the eigenvector of $F^T W F$ of unit Euclidean norm corresponding to its minimal eigenvalue $\bar{\lambda}_{\text{min}}$. Denote $u = Fv$. We define

Figure 57: Generating automatic image arrangements with the DS++ algorithm. (a) Using deep features from a face recognition neural network cluster similar faces together, *e.g.*, bald men (faces are picked at random from the LFW [111] dataset). (b) Automatic image arrangement of images of a 3D model with different lighting. (Images were randomly picked from a 30x30 noisy grid of illumination directions.) Note how the two dimensional lighting direction field is recovered by the DS++ algorithm: upper-right illuminated model image landed top-right in the grid, and similarly the other corners; images that are placed lower in the layout present more frontal illumination.

$Y = xx^T + \alpha uu^T$, where we choose $\alpha \geq 0$ so that (85b) holds. This is possible since $\mathrm{tr}(xx^T) = \|X\|_F^2 \leq n$. Further note that Y also satisfies (85f) since $\alpha \geq 0$, and (85e) since

$$AY = Axx^T + \alpha Auu^T = bx^T + \alpha A(Fv)(Fv)^T = bx^T$$

where we used the fact that Fv is a solution to the homogeneous linear equation $Ax = 0$. Finally the energy satisfies (ignoring the constant d)

$$\begin{aligned} E(X,Y) &= \mathrm{tr}WY + c^T x \\ &= x^T W x + \alpha v^T F^T W F v + c^T x \\ &= x^T W x + \alpha \bar{\lambda}_{\min} + c^T x \\ &= x^T W x + (n - \|X\|_F^2)\bar{\lambda}_{\min} + c^T x \\ &= E(X, \bar{\lambda}_{\min}) \end{aligned}$$

Now let (X, Y) be a minimizer of (85), we show that x is a feasible solution of our relaxation with the same energy. In fact due to the previous claim it is sufficient to show that $E(X, \bar{\lambda}_{\min}) \leq E(X, Y)$. The feasibility of X is clear since it is already DS. Next, denote

$$W_\lambda = W - \bar{\lambda}_{\min} I$$

then (ignoring the constant d)

$$\begin{aligned} E(X, \bar{\lambda}_{\min}) &= \mathrm{tr}W_\lambda xx^T + c^T x + \bar{\lambda}_{\min} n \\ &\overset{(*)}{\leq} \mathrm{tr}W_\lambda Y + c^T x + \bar{\lambda}_{\min} n \\ &\overset{(85b)}{=} \mathrm{tr}\ WY + c^T x = E(X,Y) \end{aligned}$$

The inequality $(*)$ follows from the fact that $A(Y - xx^T) = 0$ due to (85d),(85e) and therefore since FF^T is the projection onto the kernel of A:

$$FF^T(Y - xx^T) = Y - xx^T = (Y - xx^T)FF^T$$

and so $(*)$ follows from

$$\begin{aligned}\mathrm{tr}W_\lambda(Y - xx^T) = \mathrm{tr}W_\lambda FF^T(Y - xx^T)FF^T \\ \mathrm{tr}[F^TW_\lambda F][F^T(Y - xx^T)F] \geq 0\end{aligned}$$

where the last inequality follows from the fact that the two matrices in square brackets are positive semi-definite due to the definition of $\bar{\lambda}_{\text{min}}$ and (85f).

We now prove the third part of the lemma:

Comparison with SDP relaxation The SDP relaxation of [124] is

$$\begin{aligned}
\max_Y \quad & \mathrm{tr}WY + c^Tx + d && \text{(88a)}\\
\text{s.t.} \quad & \mathrm{tr}Y = n && \text{(88b)}\\
& x \geq 0 && \text{(88c)}\\
& Ax = b && \text{(88d)}\\
& Y \succeq xx^T && \text{(88e)}\\
& Y \geq 0 && \text{(88f)}\\
& \sum_{qrst} Y_{qrst} = n^2 && \text{(88g)}\\
& Y_{qrst} \leq \begin{cases} 0, & \text{if } \ q = s,\ r \neq t \\ 0, & \text{if } \ r = t,\ q \neq s \\ \min\left\{X_{qr}, X_{st}\right\}, & \text{otherwise} \end{cases} && \text{(88h)}
\end{aligned}$$

where Y_{qrst} is the entry replacing the quadratic monomial $X_{qr}X_{st}$. We note this relaxation contains all constraints from the SDP relaxation (85) with the exception of (85e). It also contains the additional constraints (88f)-(88h) which do not appear in (85). Thus to show that [124] is tighter than our relaxation it is sufficient to show that (85e) is implied by the other constraints of [124]. We recall that (85e) represent all constraints obtained by multiplying linear equality constraints by a linear monomial.

For a quadratic polynomial

$$g(x) = x^TWx + c^Tx + e$$

let us denote by $\bar{g}(x, Y)$ the linearized polynomial

$$\bar{g}(x, Y) = \mathrm{tr}WY + c^Tx + e$$

We will use the following property of SDP relaxations (see [70]): If a quadratic polynomial g is of the form $g = p^2$ then

1. For any feasible x, Y we have $\bar{g}(x, Y) \geq 0$.
2. If $\bar{g}(x, Y) = 0$ is satisfied for all feasible x, Y, then for any quadratic f of the form $f = pq$ we have $\bar{f}(x, Y) = 0$.

Accordingly, it is sufficient to show that the squares $g_q = p_q^2, h_r = m_r^2$ of all the linear equality polynomials

$$p_q(X) = \sum_r X_{qr} - 1 \quad , \quad m_r(X) = \sum_r X_{qr} - 1$$

satisfy $\bar{g}_q = 0, \bar{h}_r = 0$. We obtain $\bar{g}_q = 0$ from

$$\begin{aligned} 0 \leq \bar{g}_q(x, Y) &= \sum_{r,t} Y_{qrqt} - 2\sum_r X_{qr} + 1 \\ &\leq^{(88h)} \sum_r X_{qr} - 2\sum_r X_{qr} + 1 = 0 \end{aligned}$$

the proof that $\bar{h}_r = 0$ is identical.

11.11 Sparsity pattern for improving matching resolution

We construct a sparsity pattern for the task of matching $\mathbf{s}_1, \ldots, \mathbf{s}_k$ to $\mathbf{t}_1, \ldots, \mathbf{t}_n$ using known correspondences $\hat{\mathbf{s}}_\ell \mapsto \hat{\mathbf{t}}_\ell, \ell = 1, \ldots, r$.

For each $\mathbf{s}_i$ we use the following procedure to determine which correspondence will be forbidden: We find the five matched points $\hat{\mathbf{s}}_{\ell_1}, \ldots, \hat{\mathbf{s}}_{\ell_5}$ which are closest to $\mathbf{s}_i$ and compute the geodesic distance of these points from $\mathbf{s}_i$. This gives us a feature vector $v \in \mathbb{R}^5$. We then compute the geodesic distances of each of the points $\mathbf{t}_j, j = 1, \ldots, n$ from the matched points $\hat{\mathbf{t}}_{\ell_1}, \ldots \hat{\mathbf{t}}_{\ell_5}$ corresponding to the five closets points to $\mathbf{s}_i$. This gives us n feature vectors $v_j \in \mathbb{R}^5$. For $\mathbf{t}_j$ to be a viable match we require that $\|v_j - v\|_2$ be small. We therefore allow the top 20% of the correspondences according to this criteria.

To symmetrize this process, we use the same procedure to find permissible matches for each $\mathbf{t}_j$, and then select as permissible all matches $\mathbf{s}_i \mapsto \mathbf{t}_j$ which were found permissible either when starting from $\mathbf{s}_i$ or when starting from $\mathbf{t}_j$.

12 Concave graph matching

This section is based on [156].

12.1 Introduction

Graph matching is a generic and popular modeling tool for problems in computational sciences such as computer vision [25, 271, 206, 27], computer graphics [86, 125], medical imaging [100], and machine learning [233, 113, 56]. In general, graph matching refers to several different optimization problems of the form:

$$\min_{X} \; E(X) \quad \text{s.t.} \quad X \in \mathcal{F} \tag{89}$$

where $\mathcal{F} \subset \mathbb{R}^{n \times n_0}$ is a collection of *matchings* between vertices of two graphs G_A and G_B, and $E(X) = [X]^T M [X] + a^T [X]$ is usually a quadratic function in $X \in \mathbb{R}^{n \times n_0}$ ($[X] \in \mathbb{R}^{n n_0 \times 1}$ is its column stack). Often, M quantifies the discrepancy between edge affinities exerted by the matching X. Edge affinities are represented by symmetric matrices $A \in \mathbb{R}^{n \times n}$, $B \in \mathbb{R}^{n_0 \times n_0}$. Maybe the most common instantiation of (89) is

$$E_1(X) = \|AX - XB\|_F^2 \tag{90}$$

and $\mathcal{F} = \Pi_n$, the matrix group of $n \times n$ permutations. The permutations $X \in \Pi_n$ represent bijections between the set of (n) vertices of G_A and the set of (n) vertices of G_B. We denote this problem as GM. From a computational point of view, this problem is equivalent to the quadratic assignment problem, and as such is an NP-hard problem [45]. A popular way of obtaining approximate solutions is by relaxing its combinatorial constraints [150].

A standard relaxation of this formulation (*e.g.* [9, 4, 76]) is achieved by replacing Π_n with its convex hull, namely the set of doubly-stochastic matrices $\mathrm{DS} = \mathrm{hull}(\mathcal{F}) = \left\{ X \in \mathbb{R}^{n \times n} \mid X\mathbf{1} = \mathbf{1}, X^T\mathbf{1} = \mathbf{1}, X \geq 0 \right\}$. The main advantage of this formulation is the convexity of the energy E_1; the main drawback is that often the minimizer is not a permutation and simply projecting the solution onto Π_n doesn't take the energy into account resulting in a suboptimal solution. The prominent Path Following algorithm [265] suggests a better solution of continuously changing E_1 to a concave energy E' that coincide (up to an additive constant) with E_1 over the permutations. The concave energy E' is called *concave relaxation* and enjoys three key properties: (i) Its solution set is the same as the GM problem. (ii) Its set of local optima are all permutations. This means no projection of the local optima onto the permutations is required. (iii) For every descent direction, a maximal step is always guaranteed to reduce the energy most.

[72, 27] suggest a similar strategy but starting with a tighter convex relaxation. Another set of works [241, 153, 238, 33] have considered the energy

$$E_2(X) = -\mathrm{tr}(BX^T AX) \tag{91}$$

over the doubly-stochastic matrices, DS, as well. Note that both energies E_1, E_2 are identical (up to an additive constant) over the permutations and hence both are considered relaxations. However, in contrast to E_1, E_2 is in general indefinite, resulting in a non-convex relaxation. [241, 153] suggest to locally optimize this relaxation with the Frank-Wolfe algorithm and motivate it by proving that for the class of ρ-correlated Bernoulli adjacency matrices A, B, the optimal solution of the relaxation almost always coincides with the (unique in this case) GM optimal solution. [238, 33] were the first to make the useful observation that E_2 is *itself a concave relaxation* for some important cases of affinities such as heat kernels and Gaussians. This leads to an efficient local optimization using the Frank-Wolfe algorithm and specialized linear assignment solvers (*e.g.*, [28]).

In this section, we analyze and generalize the above works and introduce the concepts of *conditionally concave* and *probably conditionally concave* energies $E(X)$. Conditionally concave energy $E(X)$ means that the restriction of the Hessian M of the energy E to the linear space

$$\mathrm{lin}(\mathrm{DS}) = \left\{X \in \mathbb{R}^{n\times n} \mid X\mathbf{1} = 0, X^T\mathbf{1} = 0\right\} \tag{92}$$

is negative definite. Note that $\mathrm{lin}(\mathrm{DS})$ is the linear part of the affine-hull of the doubly-stochastic matrices, denoted $\mathrm{aff}(\mathrm{DS})$. We will use the notation $M|_{\mathrm{lin}(\mathrm{DS})}$ to refer to this restriction of M, and consequently $M|_{\mathrm{lin}(\mathrm{DS})} \prec 0$ means $v^T M v < 0$, for all $0 \neq v \in \mathrm{lin}(\mathrm{DS})$. Our first result is proving there is a large class of affinity matrices resulting in conditionally concave E_2. In particular, affinity matrices constructed using *positive or negative definite functions*[7] will be conditionally concave.

Theorem 17. *Let $\Phi : \mathbb{R}^d \to \mathbb{R}$, $\Psi : \mathbb{R}^s \to \mathbb{R}$ be both conditionally positive (or negative) definite functions of order 1. For any pair of graphs with affinity matrices $A, B \in \mathbb{R}^{n\times n}$ so that*

$$A_{ij} = \Phi(x_i - x_j), \quad B_{ij} = \Psi(y_i - y_j) \tag{93}$$

for some arbitrary $\{x_i\}_{i\in[n]} \subset \mathbb{R}^d$, $\{y_i\}_{i\in[n]} \subset \mathbb{R}^s$, the energy $E_2(X)$ is conditionally concave, i.e., its Hessian $M|_{\mathrm{lin}(\mathrm{DS})} \prec 0$.

One useful application of this theorem is in matching graphs with Euclidean affinities, since Euclidean distances are conditionally negative definite of order 1 [248]. That is, the affinities are Euclidean distances of points in Euclidean spaces of arbitrary dimensions,

$$A_{ij} = \|x_i - x_j\|_2, \quad B_{ij} = \|y_i - y_j\|_2, \tag{94}$$

where $\{x_i\}_{i\in[n]} \subset \mathbb{R}^d$, $\{y_i\}_{i\in[n]} \subset \mathbb{R}^s$. This class contains, besides Euclidean graphs, also affinities made out of distances that can be isometrically embedded in Euclidean spaces such as diffusion distances [55], distances induced by deep learning embeddings (*e.g.* [215]) and Mahalanobis distances. Furthermore, as shown in [31] the spherical distance, $A_{ij} = d_{S^d}(x_i, x_j)$, is also conditionally negative definite over the sphere and therefore can be used in the context of the theorem as-well.

Second, we generalize the notion of conditionally concave energies to *probably conditionally concave* energies. Intuitively, the energy E is called *probably conditionally concave* if it is rare to find a linear subspace D of $\mathrm{lin}(\mathrm{DS})$ so that the restriction of E to it is convex, that is $M|_D \succeq 0$. The primary motivation in considering probably conditionally concave energies is that they enjoy (with high probability) the same properties as the conditionally concave energies, *i.e.*, (i)-(iii). Therefore, locally minimizing probably conditionally concave energies over $\mathcal{F}$ can be done also with the Frank-Wolfe algorithm, with guarantees (in probability) on the feasibility of both the optimization result and the solution set of this energy.

A surprising fact we show is that probably conditionally concave energies are pretty common and include Hessian matrices M with almost the same ratio of positive to negative eigenvalues. The following theorem bounds the probability of finding uniformly at random a linear subspace D such that the restriction of $M \in \mathbb{R}^{m\times m}$ to D is convex, *i.e.*, $M|_D \succ 0$. The set of d-dimensional linear subspaces of $\mathbb{R}^m$ is called the Grassmannian $G_r(d, m)$ and it has a compact differential manifold structure and a uniform measure P_r.

Theorem 18. *Let $M \in \mathbb{R}^{m\times m}$ be a symmetric matrix with eigenvalues $\lambda_1, \ldots, \lambda_m$. Then, for all $t \in (0, \frac{1}{2\lambda_{\max}})$:*

[7] In a nutshell, positive (negative) definite functions are functions that when applied to differences of vectors produce positive (negative) definite matrices when restricted to certain linear subspaces; this notion will be formally introduced and defined in Section 12.2.

$$P_r(M|_D \succeq 0) \leq \prod_{i=1}^{m}(1-2t\lambda_i)^{-\frac{d}{2}}, \tag{95}$$

where $M|_D$ is the restriction of M to the d-dimensional linear subspace defined by $D \in G_r(d,m)$ and the probability is taken with respect to the Haar probability measure on $G_r(d,m)$.

For the case $d = 1$ the probability of $M|_D \succeq 0$ can be interpreted via distributions of quadratic forms. Previous works aimed at calculating and bounding similar probabilities [114, 208] but in different (more general) settings providing less explicit bounds. As we will see, the case $d > 1$ quantifies the chances of local minima residing at high dimensional faces of $\mathrm{hull}(\mathcal{F})$.

As a simple use-case of theorem 18, consider a matrix where 51% of the eigenvalues are -1 and 49% are $+1$; the probability of finding a convex direction of this matrix, when the direction is uniformly distributed, is exponentially low in the dimension of the matrix. As we (empirically) show, one class of problems that in practice presents probably conditionally concave E_2 are when the affinities A, B describe geodesic distances on surfaces.

Probable concavity can be further used to prove theorems regarding the likelihood of finding a local minimum outside the matching set $\mathcal{F}$ when minimizing E over a relaxed matching polytope $\mathrm{hull}(\mathcal{F})$. We will show the existence of a rather general probability space (in fact, a family) (Ω_m, P_r) of Hessians $M \in \mathbb{R}^{m\times m} \in \Omega_m$ with a natural probability measure, P_r, so that the probability of local minima of $E(X)$ to be outside $\mathcal{F}$ is very small. This result is stated and proved in theorem 19. An immediate conclusion of this result provides a proof of a probabilistic version of properties (i) and (ii) stated above for energies drawn from this distribution. In particular, the global minima of $E(X)$ over DS coincide with those over Π_n with high probability. The following theorem provides a general result in the flavor of [153] for a large class of quadratic energies.

Theorem 20. *Let E be a quadratic energy with Hessian drawn from the probability space (Ω_m, P_r). The chance that a local minimum of $\min_{X\in\mathrm{DS}} E(X)$ is outside Π_n is extremely small, bounded by $exp(-c_1 n^2)$, for some constant $c_1 > 0$.*

Third, when the energy of interest $E(X)$ is not probably conditionally concave over $\mathrm{lin}(\mathcal{F})$ there is no guarantee that the local optimum of E over $\mathrm{hull}(\mathcal{F})$ is in $\mathcal{F}$. We devise a simple variant of the Frank-Wolfe algorithm, replacing the standard line search with a *concave search*. Concave search means subtracting from the energy E convex parts that are constant on $\mathcal{F}$ (*i.e.*, relaxations) until an energy reducing step is found.

12.2 Conditionally concave energies

We are interested in the application of the Frank-Wolfe algorithm [81] for locally optimizing E_2 (potentially with a linear term) from (91) over the doubly-stochastic matrices:

$$\min_{X} \quad E(X) \tag{96a}$$

$$\text{s.t.} \quad X \in \mathrm{DS} \tag{96b}$$

where $E(X) = -[X]^T(B\otimes A)[X] + a^T[X]$. For completeness, we include a simple pseudo-code:

input: $X_0 \in \mathrm{hull}(\mathcal{F})$

while *not converged* **do**

  compute step: $X_1 = \min_{X\in\mathrm{DS}} -2[X_0]^T(B\otimes A)[X] + a^T[X]$;

  line-search: $t_0 = \arg\min_{t\in[0,1]} E((1-t)X_0 + tX_1)$;

  apply step: $X_0 = (1-t_0)X_0 + t_0X_1$;

end

Algorithm 2: Frank-Wolfe algorithm.

Definition 7. *We say that $E(X)$ is* conditionally concave *if it is concave when restricted to the linear space $\mathrm{lin}(\mathcal{F})$, the linear part of the affine-hull $\mathrm{hull}(\mathcal{F})$.*

If $E(X)$ is conditionally concave we have that properties (i)-(iii) of concave relaxations detailed above hold. In particular Algorithm 2 would always accept $t_0 = 1$ as the optimal step, and therefore it will produce a series of feasible matchings $X_0 \in \Pi_n$ and will converge after a finite number of steps to a permutation local minimum $X_* \in \Pi_n$ of (96). Our first result in this section provides sufficient condition for $W = -B \otimes A$ to be concave. It provides a connection between *conditionally positive (or negative) definite* functions [248], and negative definiteness of $-B \otimes A$:

Definition 8. *A function $\Phi : \mathbb{R}^d \to \mathbb{R}$ is called* conditionally positive definite of order m *if for all pairwise distinct points $\{x_i\}_{i\in[n]} \subset \mathbb{R}^d$ and all $0 \neq \eta \in \mathbb{R}^n$ satisfying $\sum_{i\in[n]} \eta_i p(x_i) = 0$ for all d-variate polynomials p of degree less than m, we have $\sum_{ij=1}^n \eta_i \bar{\eta}_j \Phi(x_i - x_j) > 0$.*

Specifically, Φ is conditionally positive definite of order 1 if for all pairwise distinct points $\{x_i\}_{i\in[n]} \subset \mathbb{R}^d$ and zero-sum vectors $0 \neq \eta \in \mathbb{R}^d$ we have $\sum_{ij=1}^n \eta_i \bar{\eta}_j \Phi(x_i - x_j) > 0$. Conditionally negative definiteness is defined analogously. Some well-known functions satisfy the above conditions, for example: $-\|x\|_2$, $-(c^2 + \|x\|_2^2)^\beta$ for $\beta \in (0, 1]$ are conditionally positive definite of order 1, while the functions $\exp(-\tau^2\|x\|_2^2)$ for all τ, and $c_{30} = (1-\|x\|_2^2)_+$ are conditionally positive definite of order 0 (also called just positive definite functions). Note that if Φ is conditionally positive definite of order m, it is also conditionally positive definite of any order $m' > m$. Lastly, as shown in [31], spherical distances $-d(x, x')^\gamma$ are conditionally positive semidefinite for $\gamma \in (0, 1]$, and $\exp(-\tau^2 d(x, x')^\gamma)$ are positive definite for $\gamma \in (0, 1]$ and all τ. We now prove:

Theorem 17. *Let $\Phi : \mathbb{R}^d \to \mathbb{R}$, $\Psi : \mathbb{R}^s \to \mathbb{R}$ be both conditionally positive (or negative) definite functions of order 1. For any pair of graphs with affinity matrices $A, B \in \mathbb{R}^{n\times n}$ so that*

$$A_{ij} = \Phi(x_i - x_j), \quad B_{ij} = \Psi(y_i - y_j) \tag{97}$$

for some arbitrary $\{x_i\}_{i\in[n]} \subset \mathbb{R}^d$, $\{y_i\}_{i\in[n]} \subset \mathbb{R}^s$, the energy $E_2(X)$ is conditionally concave, i.e., its Hessian $M|_{\mathrm{lin(DS)}} \prec 0$.

Lemma 12 (orthonormal basis for $\mathrm{lin(DS)}$)**.** *If the columns of $F \in \mathbb{R}^{n\times(n-1)}$ constitute an orthonormal basis for the linear space $\mathbf{1}^\perp = \left\{x \in \mathbb{R}^n \mid x^T\mathbf{1} = 0\right\}$ then the columns of $F \otimes F$ are an orthonormal basis for $\mathrm{lin(DS)}$.*

Proof. First, $(F \otimes F)^T(F \otimes F) = (F^T \otimes F^T)(F \otimes F) = (F^TF) \otimes (F^TF) = I_{n-1} \otimes I_{n-1} = I_{(n-1)^2}$. Therefore $F \otimes F$ is full rank with $(n-1)^2$ orthonormal columns. Any column of $F \otimes F$ is of the form $F_i \otimes F_j$, where F_i, F_j are the i^{th} and j^{th} columns of F, respectively. Now, reshaping $F_i \otimes F_j$ back into an $n \times n$ matrix using the inverse of the bracket operation we get $X =]F_i \otimes F_j[= F_jF_i^T$ which are clearly in $\mathrm{lin(DS)}$. Lastly, since the dimension of $\mathrm{lin(DS)}$ is $(n-1)^2$ the lemma is proved. □

Proof. (of Theorem 17) Let $A, B \in \mathbb{R}^{n\times n}$ be as in the theorem statement. Checking that $E(X)$ is conditionally concave amounts to restricting the quadratic form $-[X]^T(B \otimes A)[X]$ to $\mathrm{lin(DS)}$: $-(F \otimes F)^T(B \otimes A)(F \otimes F) = -(F^TBF) \otimes (F^TAF) \prec 0$, where we used Lemma 12 and the fact that Φ, Ψ are conditionally positive definite of order 1. □

Corollary 3. *Let A, B be Euclidean distance matrices then the solution set of Problem* (96) *and* GM *coincide.*

12.3 Probably conditionally concave energies

Although Theorem 17 covers a rather wide spectrum of instantiations of Problem (96) it definitely does not cover all interesting scenarios. In this section we would like to consider a more general energy $E(X) = [X]^T M[X] + a^T[X]$, $X \in \mathbb{R}^{n\times n}$, $M \in \mathbb{R}^{n^2\times n^2}$ and the optimization problem:

$$\min_X \quad E(X) \tag{98a}$$

$$\text{s.t.} \quad X \in \text{hull}(\mathcal{F}) \tag{98b}$$

We assume that $\mathcal{F} = \text{ext}(\text{hull}(\mathcal{F}))$, namely, the matchings are extreme points of their convex hull (as happens *e.g.*, for permutations $F = \Pi_n$). When the restricted Hessians $M|_{\text{lin}(\mathcal{F})}$ are $\epsilon-$*negative definite* (to be defined soon) we will call $E(X)$ *probably conditionally concave*.

Probably conditionally concave energies $E(X)$ will possess properties (i)-(iii) of conditionally concave energies with high probability. Hence they allow using Frank-Wolfe algorithms, such as Algorithm 2, with no line search ($t_0 = 1$) and achieve local minima in $\mathcal{F}$ (no post-processing is required). In addition, we prove that certain classes of probably conditionally concave relaxations have no local minima that are outside $\mathcal{F}$, with high probability. In the experiment section we will also demonstrate that in practice this algorithm works well for different choices of probably conditionally concave energies. Popular energies that fall into this category are, for example, (91) with A, B geodesic distance matrices or certain functions thereof.

We first make some preparations. Recall the definition of the *Grassmannian* $G_r(d, m)$: It is the set of d-dimensional linear subspaces in $\mathbb{R}^m$; it is a compact differential manifold defined by the quotient $O(m)/O(d)\times O(m-d)$, where $O(s)$ is the orthogonal group in $\mathbb{R}^s$. The orthogonal group $O(m)$ acts transitively on $G_r(d, m)$ by taking an orthogonal basis of any d-dimensional linear subspace to an orthogonal basis of a possibly different d-dimensional subspace. On $O(m)$ there exists Haar probability measure, that is a probability measure invariant to actions of $O(m)$. The Haar probability measure on $O(m)$ induces an $O(m)$-invariant (which we will also call Haar) probability measure on $G(k, m)$. We now introduce the notion of ϵ-negative definite matrices:

Definition 9. *A symmetric matrix $M \in \mathbb{R}^{m\times m}$ is called* ϵ-negative definite *if the probability of finding a d-dimensional linear subspace $D \in G(d, m)$ so that A is convex over D is smaller than ϵ^d. That is, $P_r(\{M|_D \succeq 0\}) \leq \epsilon^d$ where the probability is taken with respect to a Haar $O(m)$-invariant measure on the Grassmannian $G_r(d, m)$.*

One way to interpret $M|_D$, the restriction of the matrix M to the linear subspace D, is to consider a matrix $F \in \mathbb{R}^{m\times d}$ where the columns of F form a basis to D and consider $M|_D = F^T MF$. Clearly, negative definite matrices are ϵ-negative definite for all $\epsilon > 0$. The following theorem helps to see what else this definition encapsulates:

Theorem 18. *Let $M \in \mathbb{R}^{m\times m}$ be a symmetric matrix with eigenvalues $\lambda_1, \ldots, \lambda_m$. Then, for all $t \in (0, \frac{1}{2\lambda_{\max}})$:*

$$P_r(M|_D \succeq 0) \leq \prod_{i=1}^{m} (1 - 2t\lambda_i)^{-\frac{d}{2}}, \tag{99}$$

where $M|_D$ is the restriction of M to the d-dimensional linear subspace defined by $D \in G_r(d, m)$ and the probability is taken with respect to the Haar probability measure on $G_r(d, m)$.

Proof. Let F be an $m \times d$ matrix of i.i.d. standard normal random variables $\mathcal{N}(0, 1)$. Let F_j, $j \in [d]$, denote the j^{th} column of F. The multivariate distribution of F is $O(m)$-invariant in the sense that for a subset $A \subset \mathbb{R}^{m\times d}$, $P_r(RA) = P_r(A)$ for all $R \in O(m)$. Therefore, $P_r(M|_D \succeq 0) = P_r(F^T MF \succeq 0)$. Next,

$P_r(F^T M F \succeq 0) \leq P_r(\cap_{j=1}^d \left\{F_j^T M F_j \geq 0\right\}) = \prod_{j=1}^d P_r(F_j^T M F_j \geq 0)$, where the inequality is due to the fact that a positive semidefinite matrix necessarily has non-negative diagonal, and the equality is due to the independence of the random variables $F_j^T M F_j$, $j \in [d]$. We now calculate the probability $P_r(F_1^T M F_1)$ which is the same for all columns $j \in [d]$. For brevity let $X = (X_1, X_2, \ldots, X_m)^T = F_1$. Let $M = U \Lambda U^T$, where $U \in O(m)$ and $\Lambda = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_m)$ be the spectral decomposition of M. Since UX has the same distribution as X we have that $P_r(X^T M X \geq 0) = P_r(X^T \Lambda X \geq 0) = P_r(\sum_{i=1}^m \lambda_i X_i^2 \geq 0)$. Since $X_i^2 \sim \chi^2(1)$ we have transformed the problem into a non-negativity test of a linear combination of chi-squared random variables. Using the Chernoff bound we have for all $t > 0$:

$$P_r\left(\sum_{i=1}^m \lambda_i X_i^2 \geq 0\right) \leq \mathbb{E}\left(e^{t\sum_{i=1}^m \lambda_i X_i^2}\right) = \prod_{i=1}^m \mathbb{E}\left[e^{t\lambda_i X_i^2},\right]$$

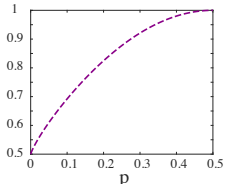
where the last equality follows from the independence of $X_1, ..., X_m$. To finish the proof we note that $\mathbb{E}\left[e^{t\lambda_i X_i^2}\right]$ is the moment generating function of the random variable X_i^2 sampled at $t\lambda_i$ which is known to be $(1-2t\lambda_i)^{-1/2}$ for $t\lambda_i < \frac{1}{2}$ which means that we can take $t < \frac{1}{2\lambda_i}$ when $\lambda_i \neq 0$ and disregard all $\lambda_i = 0$. □

Theorem 18 shows that there is a *concentration of measure* phenomenon when the dimension m of the matrix M increases. For example consider

$$\Lambda_{m,p} = (\overbrace{\lambda_1, \lambda_2, \ldots,}^{(1-p)m}, \overbrace{\mu_1, \mu_2, \ldots}^{pm}), \tag{100}$$

where $\lambda_i \leq -b$, $b > 0$ are the negative eigenvalues; $0 \leq \mu_i \leq a$, $a > 0$ are the positive eigenvalues and the ratio of positive to negative eigenvalues is a constant $p \in (0, 1/2)$. We can bound the r.h.s. of (99) with $(1+2bt)^{-\frac{(1-p)m}{2}}(1-2at)^{-\frac{pm}{2}}$. Elementary calculus shows that the minimum of this function over $t \in (0, 1/2a)$ gives:

$$P_r(v^t M v \geq 0) \leq \left(\frac{a^{1-p}b^p}{\frac{a+b}{2}} \frac{1}{2}(1-p)^{p-1}p^{-p}\right)^{\frac{m}{2}}, \tag{101}$$



where v is uniformly distributed on the unit sphere in $\mathbb{R}^m$. The function $\frac{1}{2}(1-p)^{p-1}p^{-p}$ is shown in the inset and for $p < 1/2$ it is strictly smaller than 1. The term $\frac{a^{1-p}b^p}{(a+b)/2}$ is the ratio of the weighted geometric mean and the arithmetic mean. Using the weighted arithmetic-geometric inequality it can be shown that these terms is at-most 1 if $a \leq b$. To summarize, if $a \leq b$ and $p < 1/2$ the probability to find a convex (positive) direction in M is exponentially decreasing in m, the dimension of the matrix. One simple example is taking $a = b = 1$, $p = 0.49$ which shows that considering the matrices

$$U(\overbrace{-1, -1, \ldots, -1}^{0.51m}, \overbrace{1, 1, \ldots, 1}^{0.49m})U^T$$

it will be extremely hard to get in random a convex direction in dimension $m \approx 300^2$, *i.e.*, the probability will be $\approx 4 \cdot 10^{-5}$ (this is a low dimension for a matching problem where $m = (n-1)^2$).

Another consequence that comes out of this theorem (in fact, its proof) is that the probability of finding a linear subspace $D \in G_r(d, m)$ for which the matrix M is positive semidefinite is bounded by the probability of finding a one-dimensional subspace $D_1 \in G_r(1, m)$ to the power of d. Therefore the d exponent in Definition 9 makes sense. Namely, to show a symmetric matrix M is ϵ-negative definite it is enough to check one-dimensional linear subspaces. An important implication of this fact and one of the motivations

for Definition 9 is that finding local minima at high dimensional faces of the polytope $\mathrm{hull}(\mathcal{F})$ is much less likely than at low dimensional faces.

Next, we would like to prove Theorem 19 that shows that for natural probability space of Hessians $\{M\}$ the local minima of (98) are with high probability in $\mathcal{F}$, *e.g.*, permutations in case that $F = \Pi_n$. We therefore need to devise a natural probability space of Hessians. We opt to consider Hessians of the form discussed above, namely

$$\Omega_m = \left\{ U\Lambda_{m,p}U^T \mid U \in O(m) \right\}, \tag{102}$$

where $\Lambda_{m,p}$ is defined in (100). The probability measure over Ω_m is defined using the Haar probability measure on $O(m)$, that is for a subset $A \subset \Omega_m$ we define $Pr(A) = Pr(\left\{U \in O(m) \mid U\Lambda_{m,p}U^T \in A\right\})$, where the probability measure on the r.h.s. is the probability Haar measure on $O(m)$. Note that (102) is plausible since the input graphs G_A, G_B are usually provided with an arbitrary ordering of the vertices. Writing the quadratic energy E resulted from a different ordering $P, Q \in \Pi_n$ of the vertices of G_A, G_B (resp.) yields the Hessian $H' = (Q \otimes P)(B \otimes A)(Q \otimes P)^T$, where $Q \otimes P \in \Pi_m \subset O(m)$. This motivates defining a Hessian probability space that is invariant to $O(m)$. We prove:

Theorem 19. *If the number of extreme points of the polytope* $\mathrm{hull}(F)$ *is bounded by* $\exp(m^{1-\epsilon})$*, for some fixed arbitrary* $\epsilon > 0$*, and the Hessian of* E *is drawn from the probability space* (Ω_m, P_r)*, the chance that a local minimum of* $\min_{X \in \mathrm{hull}(\mathcal{F})} E(X)$ *is outside* $\mathcal{F}$ *is extremely small, bounded by* $exp(-c_1 m)$*, for some constant* $c_1 > 0$*.*

Proof. Denote all the edges (*i.e.*, one-dimensional faces) of the polytope $P = \mathrm{hull}(\mathcal{F})$ by indices α. Even if every two extreme points of P are connected by an edge there could be at most $\exp(2m^{1-\epsilon})$ edges. A local minimum $X_* \in P$ to (98) that is not in $\mathcal{F}$ necessarily lies in the (relative) interior of some face f of P of dimension at-least one. The restriction of the Hessian M of $E(X)$ to $\mathrm{lin}(f)$ is therefore necessarily positive semidefinite. This implies there is a direction $v_\alpha \in \mathbb{R}^m$, parallel to an edge α of P so that $v_\alpha^T M v_\alpha \geq 0$.

Let us denote by X_α the indicator random variable that equals one if $v_\alpha^T M v_\alpha \geq 0$ and zero otherwise. If $X_\alpha = 1$ we say that the edge α is a *critical edge* for M. Let us denote $X = \sum_\alpha X_\alpha$ the random variable counting critical edges. The expected number of critical edges is $\mathbb{E}(X) = \sum_\alpha P_r(v_\alpha^T M v_\alpha \geq 0)$. We use Theorem 18, in particular (101), to bound the summands.

Since $P_r(v_\alpha^T M v_\alpha \geq 0) = P_r(v_\alpha^T U\Lambda_{m,p}U^T v_\alpha \geq 0)$ and $U^T v_\alpha$ is distributed uniformly on the unit sphere in $\mathbb{R}^m$, we can use (101) to infer that $P_r(v_\alpha^T M v_\alpha \geq 0) \leq \eta^{m/2}$ for some $\eta \in [0, 1)$ and therefore $\mathbb{E}(X) \leq \exp(m \log \eta/2) \sum_\alpha 1$ (note that $\log \eta < 0$). Incorporating the bound on edge number in P discussed above we get $\mathbb{E}(X) \leq \exp(\frac{\log \eta}{2} m + 2m^{1-\epsilon}) \leq \exp(-c_1 m)$ for some constant $c_1 > 0$. Lastly, as explained above, the event of a local minimum not in $\mathcal{F}$ is contained in $X \geq 1$ and by Markov's inequality we finally get $P_r(X \geq 1) \leq \mathbb{E}(X) \leq \exp(-c_1 m)$. □

Let us use this theorem to show that the local optimal solutions to Problem (98) with permutations as matchings, $\mathcal{F} = \Pi_n$, are with high probability permutations:

Theorem 20. *Let* E *be a quadratic energy with Hessian drawn from the probability space* (Ω_m, P_r)*. The chance that a local minimum of* $\min_{X \in \mathrm{DS}} E(X)$ *is outside* Π_n *is extremely small, bounded by* $exp(-c_1 n^2)$*, for some constant* $c_1 > 0$*.*

Proof. In this case the polytope $\mathrm{DS} = \mathrm{hull}(\Pi_n)$ is in the $(n-1)^2$ dimensional linear subspace $\mathrm{lin}(\mathrm{DS})$ of $\mathbb{R}^{n \times n}$. It therefore makes sense to consider the Hessians' probability space restricted to $\mathrm{lin}(\mathrm{DS})$, that is considering $M|_{\mathrm{lin}(\mathrm{DS})}$ and the orthogonal subgroup acting on it, $O((n-1)^2)$. In this case $m = (n-1)^2$. The number of vertices of DS is the number of permutations which by Stirling's bound we have $n! \leq \exp(1 - n + \log n(n + 1/2)) \leq \exp((n-1)^{1.1})$. Hence the number of edges is bounded by $\exp(2(n-1)^{1.1})$, as required. □

Lastly, Theorems 19 and 20, can be generalized by considering d-dimensional faces of the polytope:

Theorem 21. *If the number of extreme points of the polytope* $\mathrm{hull}(F)$ *is bounded by* $\exp(m^{1-\epsilon})$*, for some fixed arbitrary* $\epsilon > 0$*, and the Hessian of* E *is drawn from the probability space* (Ω_m, P_r)*, the chance that a local minimum of* $\min_{X \in \mathrm{hull}(\mathcal{F})} E(X)$ *is in the relative interior of a* d*-dimensional face of* $\mathrm{hull}(\mathcal{F})$ *is extremely small, bounded by* $exp(-c_1 dm)$*, for some constant* $c_1 > 0$*.*

This theorem is proved similarly to Theorem 19 by considering indicator variables X_α for positive semidefinite $M|_{\mathrm{lin}(\alpha)}$ where α stands for a d-dimensional face in $\mathrm{hull}(\mathcal{F})$. This generalized theorem has a practical implication: local minima are likely to be found on lower dimensional faces.

12.4 Graph matching with one sided permutations

In this section we examine an interesting and popular graph matching (89) instance, where the matchings are the one-sided permutations, namely $\mathcal{F} = \left\{X \in \{0,1\}^{n \times n_0} \mid X\mathbf{1} = \mathbf{1}\right\}$. That is $\mathcal{F}$ are well-defined maps from graph G_A with n vertices to G_B with n_0 vertices. This modeling is used in the template and partial matching cases. Unfortunately, in this case, standard graph matching energies $E(X)$ are not probably conditionally concave over $\mathrm{lin}(\mathcal{F})$. Note that $\mathrm{lin}(\mathrm{DS}) \subsetneqq \mathrm{lin}(\mathcal{F})$.

We devise a variation of the Frank-Wolfe algorithm using a *concave search* procedure. That is, in each iteration, instead of standard line search we subtract a convex energy from $E(X)$ that is constant on $\mathcal{F}$ until we find a descent step. This subtraction is a relaxation of the original problem (89) in the sense it does not alter (up to a global constant) the energy values at $\mathcal{F}$.

The algorithm is summarized in Algorithm 3 and is guaranteed to output a feasible solution in $\mathcal{F}$. The linear program in each iteration over $\mathrm{hull}(\mathcal{F})$ has a simple closed form solution. Also, note that in the inner loop only n different λ values should be checked. Details can be found in the supplementary materials.

input: $X_0 \in \mathrm{hull}(\mathcal{F})$
while *not converged* **do**
  while *energy not reduced* **do**
    add concave energy $M_{curr} = M - \lambda\Lambda$;
    compute step: $X_1 = \min_{X \in \mathrm{hull}(\mathcal{F})} [X_0]^T M_{curr}[X]$;
    increase λ;
  end
  Update current solution $X_0 = X_1$ and set $\lambda = 0$;
end

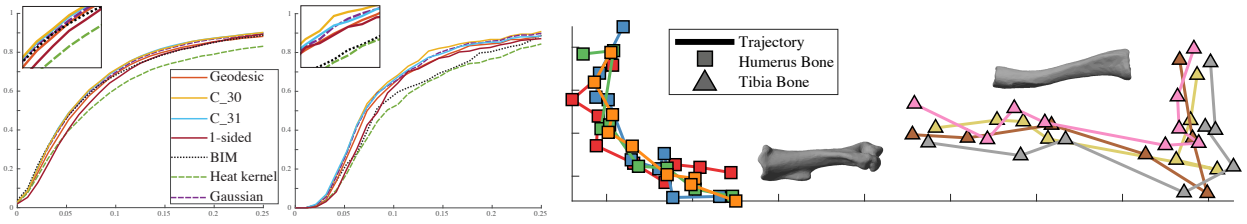
Algorithm 3: Frank-Wolfe with a concave search.

12.5 Experiments

Bound evaluation: Table 15 evaluates the probability bound (99) for Hessians $M \in \mathbb{R}^{100^2 \times 100^2}$ of $E_2(X)$ using affinities A, B defined by functions of geodesic distances on surfaces. Functions that are conditionally negative definite or semi-definite in the Euclidean case: geodesic distances $d(x,y)$, its square $d(x,y)^2$, and multi-quadratic functions $(1 + d(x,y)^2)^{\frac{1}{10}}$. Functions that are positive definite in the Euclidean case: $c_{30}(\|x\|_2) = (1 - \|x\|_2)_+$, $c_{31}(\|x\|_2) = (1 - \|x\|_2)_+^4 (4\|x\|_2 + 1)$ and $\exp(-\tau^2 \|x\|_2^2)$ (note that the last function was used in [238]). We also provide the *empirical* chance of sampling a convex direction. The results in the table are the mean over all the shape pairs (218) in the SHREC07 [91] shape matching benchmark with $n = 100$. The empirical test was conducted using 10^6 random directions sampled from an i.i.d. Gaussian distribution. Note that 0 in the table means numerical zero (below machine precision).

Table 15: Evaluation of probable conditional concavity for different functions of geodesics on $\mathrm{lin}(\mathrm{DS})$.

	Distance	Distance Squared	MultiQuadratic	c_{30}	c_{31}	Gaussian
Bound mean	0	0.024	$7 \cdot 10^{-4}$	0	0	0
Bound std	0	0.021	$1.7 \cdot 10^{-3}$	0	0	0
Empirical mean	0	0.003	$7 \cdot 10^{-5}$	0	0	0
Empirical std	0	0.003	$1.8 \cdot 10^{-4}$	0	0	0



(a) (b)

Figure 58: (a) SHREC07 benchmark: Cumulative distribution functions of all errors (left) and mean error per shape (right). (b) Anatomical dataset embedding in the plane. Squares and triangles represent different bone types, lines represent temporal trajectories.

Initialization: Motivated by [78, 126] and due to the fast running time of the algorithms (*e.g.*, $150msec$ for $n = 200$ with Algorithm 2, and $16sec$ with Algorithm 3, both on a single CPU) we sampled multiple initializations based on randomized l-pairs of vertices of graphs G_A, G_B and choose the result corresponding to the best energy. In Algorithm 2 we used the Auction algorithm [28], as in [238].

Table 16: Comparison to "convex to concave" methods. The table shows the average and the std of the energy differences. Positive averages indicate our algorithm achieves lower energy on average.

	ModelNet10			SHREC07		
# points	30	60	90	30	60	90
DSPP	5.0± 5.3	9.8± 10.8	14.468± 19.8	1.3± 2.3	9.5± 9.5	26.2± 24.3
PATH	101.4±53.9	512.3±198.4	1251.9±426.4	69.263±55.9	307.7±230.6	721.0±549.7
RANDOM	197.9±35.2	865.3±122.1	1986.1±273.0	120.2±83.6	532.7±357.8	1230.7±817.6

Comparison with convex-to-concave methods: Table 16 compares our method to [265, 72] (PATH, DSPP accordingly). As mentioned in the introduction, these methods solve convex relaxations and then project its minimizer while deforming the energy towards concavity. Our method compares favorably in the task of matching point-clouds from the ModelNet10 dataset [252] with Euclidean distances as affinities, and the SHREC07 dataset [91] with geodesic distances. We used $\mathcal{F} = \Pi_n$, and energy (91). The table shows average and standard deviation of energy differences of the listed algorithms and ours; the average is taken over 50 random pairs of shapes. Note that positive averages mean our algorithm achieves lower energy on average; the difference to random energy values is given for scale.

Automatic shape matching: We use our Algorithm 2 for automatic shape matching (*i.e.*, with no user input or input shape features) on a the SHREC07 [91] dataset according to the protocol of [126]. This benchmark consists of matching 218 pairs of (often extremely) non-isometric shapes in 11 different classes such as humans, animals, planes, ants etc. On each shape, we sampled $k = 8$ points using farthest point sampling and randomized $s = 2000$ initializations of subsets of $l = 3$ points. In this stage, we use $n = 300$ points.

We then up-sampled to $n = 1500$ using the exact algorithm with initialization using our $n = 300$ best result. The process takes about $16min$ per pair running on a single CPU. Figure 58 (a) shows the cumulative distribution function of the geodesic matching errors (left - all errors, right - mean error per pair) of Algorithm 2 with geodesic distances and their functions c_{30}, c_{31}. We used (91) and $\mathcal{F} = \Pi$. We also show the result of Algorithm 3 with geodesic distances, see details in the supplementary materials. We compare with Blended Intrinsic Maps (BIM) [126] and the energies suggested by [33] (heat kernel) and [238] (Gaussian of geodesics). For the latter two, we used the same procedure as described above and just replaced the energies with the ones suggested in these works. Note that the Gaussian of geodesics energy of [238] falls into the probably concave framework.

Anatomical shape space analysis: We match a dataset of 67 mice bone surfaces acquired using micro-CT. The dataset consists of eight time series. Each time series captures the development of one type of bone over time. We use Algorithm 2 to match all pairs in the dataset using Euclidean distance affinity matrices A, B, energy (91), and $\mathcal{F} = \Pi_n$. After optimization, we calculated a 67×67 dissimilarity matrix. Dissimilarities are equivalent to our energy over the permutations (up to additive constant) and defined by $\sum_{ijkl} X_{ij} X_{kl} (d_{ik} - d_{jl})^2$. A color-coded matching example can be seen in the inset. In Figure 58 (b) we used Multi-Dimensional Scaling (MDS) [137] to assign a $2D$ coordinate to each surface using the dissimilarity matrix. Each bone is shown as a trajectory. Note how the embedding separated the two types of bones and all bones of the same type are mapped to similar time trajectories. This kind of visualization can help biologists analyze their data and possibly find interesting time periods in which bone growth is changing. Lastly, note that the Tibia bones (on the right) exhibit an interesting change in the midst of its growth. This particular time was also predicted by other means by the biologists.

12.6 Conclusion

In this work, we analyze and generalize the idea of concave relaxations for graph matching problems. We concentrate on *conditionally concave* and *probably conditionally concave* energies and demonstrate that they provide useful relaxations in practice. We prove that all local minima of such relaxations are with high probability in the original feasible set; this allows removing the standard post-process projection step in relaxation-based algorithms. Another conclusion is that the set of optimal solutions of such relaxations coincides with the set of optimal solutions of the original graph matching problem.

There are popular edge affinity matrices, such as $\{0, 1\}$ adjacency matrices, that in general do not lead to conditionally concave relaxations. This raises the general question of characterizing more general classes of affinity matrices that furnish (probably) conditionally-concave relaxations. Another interesting future work could try to obtain information on the quality of local minima for more specific classes of graphs.

12.7 Frank-Wolfe with concave search

An orthogonal basis to $\mathrm{lin}(\mathcal{F})$ is computed similarly to Lemma 12:

Lemma 13 (orthonormal basis for one-sided permutations)**.** *If the columns of $F \in \mathbb{R}^{n_0 \times (n_0 - 1)}$ form an orthonormal basis for $\mathbf{1}^{\perp}$ in $\mathbb{R}^{n_0}$ then the columns of $F \otimes I_n$ are an orthonormal basis for* $\mathrm{lin}(\mathcal{F})$.

The energy $E_2(X)$ in this case does not model the matching problem well since it gives rise to trivial

solutions. Instead, we chose to optimize the similar energy [226]: $E(X) = \sum_{ijkl} X_{ij}X_{kl}(A_{ik} - B_{jl})^2$. This energy can also be written in matrix form: $[X]^T M[X]$ where $M = -2B \otimes A + 11^T \otimes A.^2 + B.^2 \otimes 11^T$ (where $C.^2$ implies entry-wise operation) and after restricting it to $\mathrm{lin}(\mathcal{F})$ its Hessian is of the form $-2FBF \otimes A + FB.^2F \otimes 11^T$. Assuming A, B are Euclidean distance matrices, the right summand is negative semidefinite, but the left summand is not. This is because that A is not conjugated by F: it has a large positive eigenvalue as a result of the Perron-Frobenius Theorem.

The linear program solved in each iteration of the algorithm takes a surprisingly simple form: it amounts to solving $\min_{X \in \mathrm{hull}(\mathcal{F})} \mathrm{tr}(\nabla E(X_0)^T X)$ which can be solved simply by assigning the value 1 to the index of the minimal value in each row of $\nabla E(X_0)$. This procedure always outputs solutions in $\mathcal{F}$.

The convex energies we subtract from the objective during the concave search should be constant on $\mathcal{F}$ so a reduction in the subtracted energy is the same as in the original energy $E(X)$. We use the quadratic form defined by $\lambda * \Lambda$ where Λ is a $nn_0 \times nn_0$ diagonal matrix defined by $D_{ijij} = \max_j \{\sum_{kl} |M_{ijkl}|\}$. D is a positive definite matrix and for $\lambda = 1$, $W - D$ is guaranteed to be negative semidefinite. The values of λ need not be discretized since there are only n different critical values - the ones that change the minimum calculation mentioned in the previous paragraph.

13 Discussion

Although considerable progress was obtained in the last few years, both problems considered in this thesis are far from being solved. As for the problem of deep learning on irregular data, there are currently no methods that can work on all types of meshes, including triangle soups which are abundant in applications. A possible way to tackle this problem is using the hyper-graph learning approach from [158] as was recently suggested by [8]. As for matching problems, there is still no silver bullet solution for matching three-dimensional shapes. One prominent direction is to learn how to efficiently solve these hard optimization problems, as was suggested in *e.g.*[162, 245, 147]. Another interesting venue for future work is trying to leverage the recently discovered connection between the Sherali-Adams relaxations of matching problems [12], WL isomorphism tests and the k-order invariant neural networks suggested in [161].

Publication list (first author)

[14] Matan Atzmon, Haggai Maron, and Yaron Lipman. "Point Convolutional Neural Networks by Extension Operators". In: *ACM Trans. Graph.* 37.4 (July 2018), 71:1–71:12. ISSN: 0730-0301.

[72] Nadav Dym, Haggai Maron, and Yaron Lipman. "DS++: a flexible, scalable and provably tight relaxation for matching problems". In: *ACM Transactions on Graphics (TOG)* 36.6 (2017), p. 184.

[156] Haggai Maron and Yaron Lipman. "(Probably) Concave Graph Matching". In: *Advances in Neural Information Processing Systems*. 2018, pp. 406–416.

[157] Haggai Maron et al. "Convolutional neural networks on surfaces via seamless toric covers". In: *ACM Trans. Graph* 36.4 (2017), p. 71.

[158] Haggai Maron et al. "Invariant and Equivariant Graph Networks". In: *International Conference on Learning Representations*. 2019. URL: https://openreview.net/forum?id=Syx72jC9tm.

[159] Haggai Maron et al. "On the Universality of Invariant Networks". In: *International conference on machine learning*. 2019.

[160] Haggai Maron et al. "Point Registration via Efficient Convex Relaxation". In: *ACM Trans. Graph.* 35.4 (July 2016), 73:1–73:12. ISSN: 0730-0301.

[161] Haggai Maron et al. "Provably Powerful Graph Networks". In: *Advances in Neural Information Processing Systems* (Dec. 2019).

Additional papers (secondary author)

[15] Matan Atzmon et al. "Controlling Neural Level Sets". In: *arXiv preprint arXiv:1905.11911* (2019).

[23] Heli Ben-Hamu et al. "Multi-chart Generative Surface Modeling". In: *ACM Trans. Graph.* 37.6 (Dec. 2018), 215:1–215:15. ISSN: 0730-0301.

[101] Niv Haim et al. "Surface Networks via General Covers". In: *International Conference on Computer Vision* (Oct. 2019).

[138] Yam Kushinsky et al. "Sinkhorn Algorithm for Lifted Assignment Problems". In: *SIAM Journal on Imaging Sciences* 12.2 (2019), pp. 716–735.

References

[1] Martin Abadi et al. "Tensorflow: a system for large-scale machine learning." In: *OSDI*. Vol. 16. 2016, pp. 265–283.

[2] WARREN P Adams and Terri A Johnson. "Improved linear programming-based lower bounds for the quadratic assignment problem". In: *DIMACS series in discrete mathematics and theoretical computer science* 16 (1994), pp. 43–75.

[3] *Adobe Fuse 3D Characters*. `https://www.mixamo.com`. Accessed: 2016-10-15. 2016.

[4] Yonathan Aflalo, Alex Bronstein, and Ron Kimmel. "Graph matching: relax or not?" In: *arXiv preprint arXiv:1401.7623* (2014).

[5] Yonathan Aflalo, Alexander Bronstein, and Ron Kimmel. "On convex relaxation of graph isomorphism". In: *Proceedings of the National Academy of Sciences* 112.10 (Mar. 2015), pp. 2942–2947. ISSN: 1091-6490. DOI: `10.1073/pnas.1401651112`. URL: `http://dx.doi.org/10.1073/pnas.1401651112`.

[6] Noam Aigerman and Yaron Lipman. "Hyperbolic orbifold tutte embeddings". In: *ACM Transactions on Graphics (TOG)* 35.6 (2016), p. 217.

[7] Noam Aigerman and Yaron Lipman. "Orbifold Tutte embeddings." In: *ACM Trans. Graph.* 34.6 (2015), pp. 190–1.

[8] Marjan Albooyeh, Daniele Bertolini, and Siamak Ravanbakhsh. "Incidence Networks for Geometric Deep Learning". In: *arXiv preprint arXiv:1905.11460* (2019).

[9] HA Almohamad and Salih O Duffuaa. "A linear programming approach for the weighted graph matching problem". In: *IEEE Transactions on pattern analysis and machine intelligence* 15.5 (1993), pp. 522–525.

[10] Dragomir Anguelov et al. "SCAPE: shape completion and animation of people". In: *ACM Transactions on Graphics (TOG)*. Vol. 24. 3. ACM. 2005, pp. 408–416.

[11] Kurt M Anstreicher and Nathan W Brixius. "A new bound for the quadratic assignment problem based on convex quadratic programming". In: *Mathematical Programming* 89.3 (2001), pp. 341–357.

[12] Albert Atserias and Elitza N Maneva. "Graph Isomorphism, Sherali-Adams Relaxations and Expressibility in Counting Logics." In: *Electronic Colloquium on Computational Complexity (ECCC)*. Vol. 18. 2011, p. 77.

[13] James Atwood and Don Towsley. "Diffusion-convolutional neural networks". In: *Advances in Neural Information Processing Systems*. 2016, pp. 1993–2001.

[16] Mathieu Aubry, Ulrich Schlickewei, and Daniel Cremers. "The wave kernel signature: A quantum mechanical approach to shape analysis". In: *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*. IEEE. 2011, pp. 1626–1633.

[17] L Babai. *Automorphism groups, isomorphism, reconstruction. Chapter 27 of the Handbook of Combinatorics, 1447–1540. RL Graham, M. Grötschel, L. Lovász Eds*. 1995.

[18] László Babai. "Graph isomorphism in quasipolynomial time". In: *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*. ACM. 2016, pp. 684–697.

[19] László Babai, D. Yu. Grigoryev, and David M. Mount. "Isomorphism of Graphs with Bounded Eigenvalue Multiplicity". In: *Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing*. STOC '82. San Francisco, California, USA: ACM, 1982, pp. 310–324. ISBN: 0-89791-070-2. DOI: 10.1145/800070.802206. URL: http://doi.acm.org/10.1145/800070.802206.

[20] Mikhail Belkin and Partha Niyogi. "Laplacian eigenmaps for dimensionality reduction and data representation". In: *Neural computation* 15.6 (2003), pp. 1373–1396.

[21] Mikhail Belkin and Partha Niyogi. "Towards a theoretical foundation for Laplacian-based manifold methods." In: *COLT*. Vol. 3559. Springer. 2005, pp. 486–500.

[22] Jean-David Benamou et al. "Iterative Bregman projections for regularized transportation problems". In: *SIAM Journal on Scientific Computing* 37.2 (2015), A1111–A1138.

[24] Jon Louis Bentley. "Multidimensional binary search trees used for associative searching". In: *Communications of the ACM* 18.9 (1975), pp. 509–517.

[25] Alexander C Berg, Tamara L Berg, and Jitendra Malik. "Shape matching and object recognition using low distortion correspondences". In: *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*. Vol. 1. IEEE. 2005, pp. 26–33.

[26] Matthew Berger et al. "A survey of surface reconstruction from point clouds". In: *Computer Graphics Forum*. Vol. 36. 1. Wiley Online Library. 2017, pp. 301–329.

[27] Florian Bernard, Christian Theobalt, and Michael Moeller. "Tighter Lifting-Free Convex Relaxations for Quadratic Matching Problems". In: *arXiv preprint arXiv:1711.10733* (2017).

[28] Florian Bernard et al. "Fast correspondences for statistical shape models of brain structures." In: *Medical Imaging: Image Processing*. 2016, 97840R.

[29] P.J. Besl and N.D. McKay. "A Method for Registration of 3-D Shapes". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 14.2 (1992), pp. 239–256. ISSN: 0162-8828. DOI: http://doi.ieeecomputersociety.org/10.1109/34.121791.

[30] Federica Bogo et al. "FAUST: Dataset and evaluation for 3D mesh registration". In: *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*. IEEE. 2014, pp. 3794–3801.

[31] E Bogomolny, O Bohigas, and C Schmit. "Distance matrices and isometric embeddings". In: *arXiv preprint arXiv:0710.2063* (2007).

[32] Davide Boscaini et al. "Learning shape correspondence with anisotropic convolutional neural networks". In: *NIPS*. 2016.

[33] Amit Boyarski et al. "Efficient Deformable Shape Correspondence via Kernel Matching". In: *arXiv preprint arXiv:1707.08991* (2017).

[34] Doug M Boyer et al. "Algorithms to automatically quantify the geometric similarity of anatomical surfaces". In: *Proceedings of the National Academy of Sciences* 108.45 (2011), pp. 18221–18226.

[35] Leo Breiman. "Random forests". In: *Machine learning* 45.1 (2001), pp. 5–32.

[36] Emmanuel Briand. "When is the algebra of multisymmetric polynomials generated by the elementary multisymmetric polynomials". In: *Contributions to Algebra and Geometry* 45.2 (2004), pp. 353–368.

[37] Andrew Brock et al. "Generative and discriminative voxel modeling with convolutional neural networks". In: *arXiv preprint arXiv:1608.04236* (2016).

[38] Alexander M. Bronstein, Michael M. Bronstein, and Ron Kimmel. "Generalized multidimensional scaling: A framework for isometry-invariant partial surface matching". In: *Proceedings of the National Academy of Sciences of the United States of America* 103.5 (2006), pp. 1168–1172. DOI: 10.1073/pnas.0508601103. eprint: http://www.pnas.org/content/103/5/1168.full.pdf+html. URL: http://www.pnas.org/content/103/5/1168.abstract.

[39] Michael M Bronstein et al. "Geometric deep learning: going beyond euclidean data". In: *IEEE Signal Processing Magazine* 34.4 (2017), pp. 18–42.

[40] David S Broomhead and David Lowe. *Radial basis functions, multi-variable functional interpolation and adaptive networks*. Tech. rep. Royal Signals and Radar Establishment Malvern (United Kingdom), 1988.

[41] Benedict J Brown and Szymon Rusinkiewicz. "Global non-rigid alignment of 3-D scans". In: *ACM Transactions on Graphics (TOG)* 26.3 (2007), p. 21.

[42] Joan Bruna et al. "Spectral Networks and Locally Connected Networks on Graphs". In: (2013), pp. 1–14. arXiv: 1312.6203. URL: http://arxiv.org/abs/1312.6203.

[43] Joan Bruna et al. "Spectral networks and locally connected networks on graphs". In: *arXiv preprint arXiv:1312.6203* (2013).

[44] Yu D Burago and Viktor A Zalgaller. *Geometry III: theory of surfaces*. Vol. 48. Springer Science & Business Media, 2013.

[45] Rainer Ernst Burkard et al. "The quadratic assignment problem". In: *Handbook of Combinatorial Optimization*. Kluwer Academic Publishers, 1998.

[46] Jin-Yi Cai, Martin Fürer, and Neil Immerman. "An optimal lower bound on the number of variables for graph identification". In: *Combinatorica* 12.4 (1992), pp. 389–410.

[47] Jonathan C Carr et al. "Reconstruction and representation of 3D objects with radial basis functions". In: *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*. ACM. 2001, pp. 67–76.

[48] Emilio Carrizosa, Vanesa Guerrero, and Dolores Romero Morales. "Visualizing proportions and dissimilarities by Space-filling maps: a Large Neighborhood Search approach". In: *Computers & Operations Research* (2016).

[49] Ken Chatfield et al. "Return of the devil in the details: Delving deep into convolutional nets". In: *arXiv preprint arXiv:1405.3531* (2014).

[50] Qifeng Chen and Vladlen Koltun. "Robust Nonrigid Registration by Convex Optimization". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015, pp. 2039–2047.

[51] Özgün Çiçek et al. "3D U-Net: learning dense volumetric segmentation from sparse annotation". In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer. 2016, pp. 424–432.

[52] Taco S. Cohen and Max Welling. "Steerable CNNs". In: 1990 (2016), pp. 1–14. arXiv: 1612.08498. URL: http://arxiv.org/abs/1612.08498.

[53] Taco S Cohen et al. "Spherical CNNs". In: *arXiv preprint arXiv:1801.10130* (2018).

[54] Taco Cohen and Max Welling. "Group equivariant convolutional networks". In: *International conference on machine learning*. 2016, pp. 2990–2999.

[55] Ronald R Coifman and Stéphane Lafon. "Diffusion maps". In: *Applied and computational harmonic analysis* 21.1 (2006), pp. 5–30.

[56] Timothee Cour, Praveen Srinivasan, and Jianbo Shi. "Balanced graph matching". In: *Advances in Neural Information Processing Systems*. 2007, pp. 313–320.

[57] Trevor F Cox and Michael AA Cox. *Multidimensional scaling*. Chapman and hall/CRC, 2000.

[58] Marco Cuturi. "Sinkhorn distances: Lightspeed computation of optimal transport". In: *Advances in Neural Information Processing Systems*. 2013, pp. 2292–2300.

[59] George Cybenko. "Approximation by superpositions of a sigmoidal function". In: *Mathematics of control, signals and systems* 2.4 (1989), pp. 303–314.

[60] Jifeng Dai et al. "Deformable Convolutional Networks". In: *arXiv preprint arXiv:1703.06211* (2017).

[61] E Brian Davies. *Linear operators and their spectra*. Vol. 106. Cambridge University Press, 2007.

[62] Edilson De Aguiar et al. "Performance capture from sparse multi-view video". In: *ACM Transactions on Graphics (TOG)*. Vol. 27. 3. ACM. 2008, p. 98.

[63] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. "Convolutional neural networks on graphs with fast localized spectral filtering". In: *Advances in Neural Information Processing Systems*. 2016, pp. 3837–3845.

[64] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. "Convolutional neural networks on graphs with fast localized spectral filtering". In: *Advances in Neural Information Processing Systems*. 2016, pp. 3844–3852.

[65] Yichuan Ding and Henry Wolkowicz. "A low-dimensional semidefinite relaxation for the quadratic assignment problem". In: *Mathematics of Operations Research* 34.4 (2009), pp. 1008–1022.

[66] John D Dixon and Brian Mortimer. *Permutation groups*. Vol. 163. Springer Science & Business Media, 1996.

[67] Manfredo P Do Carmo et al. "Differential Geometry". In: *Mathematical Models*. Springer, 2017, pp. 155–180.

[68] Brendan L Douglas. "The Weisfeiler-Lehman method and graph isomorphism testing". In: *arXiv preprint arXiv:1101.5211* (2011).

[69] David K Duvenaud et al. "Convolutional networks on graphs for learning molecular fingerprints". In: *Advances in neural information processing systems*. 2015, pp. 2224–2232.

[70] Nadav Dym and Yaron Lipman. "Exact Recovery with Symmetries for Procrustes Matching". In: *arXiv preprint arXiv:1606.01548* (2016).

[73] Yuval Eldar et al. "The farthest point strategy for progressive image sampling". In: *Image Processing, IEEE Transactions on* 6.9 (1997), pp. 1305–1315.

[74] Wei Feng et al. "Feature correspondences using morse smale complex". In: *The Visual Computer* 29.1 (2013), pp. 53–67.

[75] Matthias Fey and Jan Eric Lenssen. "Fast Graph Representation Learning with PyTorch Geometric". In: *arXiv preprint arXiv:1903.02428* (2019).

[76] Marcelo Fiori and Guillermo Sapiro. "On spectral properties for graph matching and graph isomorphism problems". In: *Information and Inference: A Journal of the IMA* 4.1 (2015), pp. 63–76.

[77] Martin A Fischler and Robert C Bolles. "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography". In: *Communications of the ACM* 24.6 (1981), pp. 381–395.

[78] Martin A Fischler and Robert C Bolles. "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography". In: *Readings in computer vision*. Elsevier, 1987, pp. 726–740.

[79] Fajwel Fogel et al. "Convex relaxations for permutation problems". In: *Advances in Neural Information Processing Systems*. 2013, pp. 1016–1024.

[80] F. Fogel et al. "Convex Relaxations for Permutation Problems". In: *SIAM Journal on Matrix Analysis and Applications* 36.4 (2015), pp. 1465–1488. DOI: 10.1137/130947362.

[81] Marguerite Frank and Philip Wolfe. "An algorithm for quadratic programming". In: *Naval Research Logistics (NRL)* 3.1-2 (1956), pp. 95–110.

[82] Ohad Fried et al. "IsoMatch: Creating informative grid layouts". In: *Computer graphics forum*. Vol. 34. 2. Wiley Online Library. 2015, pp. 155–166.

[83] Mituhiro Fukuda et al. "Exploiting Sparsity in Semidefinite Programming via Matrix Completion I: General Framework". In: *SIAM J. on Optimization* 11.3 (Mar. 2000), pp. 647–674. ISSN: 1052-6234. DOI: 10.1137/S1052623400366218. URL: http://dx.doi.org/10.1137/S1052623400366218.

[84] William Fulton and Joe Harris. *Representation theory: a first course*. Vol. 129. Springer Science & Business Media, 2013.

[85] Thomas Funkhouser and Michael Kazhdan. "Shape-based retrieval and analysis of 3D models". In: *ACM SIGGRAPH 2004 Course Notes*. ACM. 2004, p. 16.

[86] Thomas Funkhouser and Philip Shilane. "Partial matching of 3 D shapes with priority-driven search". In: *ACM International Conference Proceeding Series*. Vol. 256. 2006, pp. 131–142.

[87] Andrew H Gee and Richard W Prager. "Polyhedral combinatorics and neural networks". In: *Neural computation* 6.1 (1994), pp. 161–180.

[88] Natasha Gelfand et al. "Robust Global Registration". In: *Proceedings of the Third Eurographics Symposium on Geometry Processing*. SGP '05. Vienna, Austria: Eurographics Association, 2005. ISBN: 3-905673-24-X. URL: http://dl.acm.org/citation.cfm?id=1281920.1281953.

[89] Justin Gilmer et al. "Neural Message Passing for Quantum Chemistry". In: *International Conference on Machine Learning*. 2017, pp. 1263–1272.

[90] Justin Gilmer et al. "Neural message passing for quantum chemistry". In: *arXiv preprint arXiv:1704.01212* (2017).

[91] Daniela Giorgi, Silvia Biasotti, and Laura Paraboschi. "Shape retrieval contest 2007: Watertight models track". In: *SHREC competition* 8.7 (2007).

[92] Manfred Göbel. "Computing bases for rings of permutation-invariant polynomials". In: *J. Symb. Comput.* 19.4 (1995), pp. 285–291.

[93] Marco Gori, Gabriele Monfardini, and Franco Scarselli. "A new model for earning in raph domains". In: *Proceedings of the International Joint Conference on Neural Networks* 2.January (2005), pp. 729–734. DOI: 10.1109/IJCNN.2005.1555942.

[94] John C Gower and Garmt B Dijksterhuis. *Procrustes problems*. Vol. 3. Oxford University Press Oxford, 2004.

[95] Martin Grohe. *Descriptive complexity, canonisation, and definable graph structure theory*. Vol. 47. Cambridge University Press, 2017.

[96] Martin Grohe and Martin Otto. "Pebble games and linear equations". In: *The Journal of Symbolic Logic* 80.3 (2015), pp. 797–844.

[97] Robert Grone et al. "Positive definite completions of partial Hermitian matrices". In: *Linear algebra and its applications* 58 (1984), pp. 109–124.

[98] Paul Guerrero et al. "PCPNET: Learning Local Shape Properties from Raw Point Clouds". In: *arXiv preprint arXiv:1710.04954* (2017).

[99] Kan Guo, Dongqing Zou, and Xiaowu Chen. "3D Mesh Labeling via Deep Convolutional Neural Networks". In: *ACM Trans. Graph.* 35.1 (2015).

[100] Yanrong Guo et al. "Robust anatomical correspondence detection by hierarchical sparse graph matching". In: *IEEE transactions on medical imaging* 32.2 (2013), pp. 268–277.

[102] Will Hamilton, Zhitao Ying, and Jure Leskovec. "Inductive representation learning on large graphs". In: *Advances in Neural Information Processing Systems*. 2017, pp. 1024–1034.

[103] William L Hamilton, Rex Ying, and Jure Leskovec. "Representation learning on graphs: Methods and applications". In: *arXiv preprint arXiv:1709.05584* (2017).

[104] Jason S. Hartford et al. "Deep Models of Interactions Across Sets". In: *ICML*. 2018.

[105] Jason Hartford et al. "Deep Models of Interactions Across Sets". In: *arXiv preprint arXiv:1803.02879* (2018).

[106] Vishakh Hegde and Reza Zadeh. "Fusionnet: 3d object classification using multiple data representations". In: *arXiv preprint arXiv:1607.05695* (2016).

[107] Christoph Helmberg. "Semidefinite programming for combinatorial optimization". In: (2000).

[108] Mikael Henaff, Joan Bruna, and Yann LeCun. "Deep Convolutional Networks on Graph-Structured Data". In: June (2015). ISSN: 1506.05163. arXiv: 1506.05163. URL: http://arxiv.org/abs/1506.05163.

[109] Mikael Henaff, Joan Bruna, and Yann LeCun. "Deep convolutional networks on graph-structured data". In: *arXiv preprint arXiv:1506.05163* (2015).

[110] Kurt Hornik. "Approximation capabilities of multilayer feedforward networks". In: *Neural networks* 4.2 (1991), pp. 251–257.

[111] Gary B Huang et al. *Labeled faces in the wild: A database for studying face recognition in unconstrained environments*. Tech. rep. Technical Report 07-49, University of Massachusetts, Amherst, 2007.

[112] Qixing Huang, Fan Wang, and Leonidas Guibas. "Functional Map Networks for Analyzing and Exploring Large Shape Collections". In: *ACM Trans. Graph.* 33.4 (July 2014), 36:1–36:11. ISSN: 0730-0301. DOI: 10.1145/2601097.2601111. URL: http://doi.acm.org/10.1145/2601097.2601111.

[113] Benoit Huet, Andrew DJ Cross, and Edwin R Hancock. "Graph matching for shape retrieval". In: *Advances in Neural Information Processing Systems*. 1999, pp. 896–902.

[114] Jean-Pierre Imhof. "Computing the distribution of quadratic forms in normal variables". In: *Biometrika* 48.3/4 (1961), pp. 419–426.

[115] Sergey Ivanov and Evgeny Burnaev. "Anonymous Walk Embeddings". In: *arXiv preprint arXiv:1805.11921* (2018).

[116] Varun Jain, Hao Zhang, and Oliver Van Kaick. "Non-Rigid Spectral Correspondence of Triangle Meshes". In: *International Journal of Shape Modeling* 13 (2007), pp. 101–124. ISSN: 0218-6543. DOI: 10.1142/S0218654307000968.

[117] Tao Ju. "Robust repair of polygonal models". In: *ACM Transactions on Graphics (TOG)* 23.3 (2004), pp. 888–895.

[118] Felix Kälberer, Matthias Nieser, and Konrad Polthier. "QuadCover-Surface Parameterization using Branched Coverings". In: *Computer Graphics Forum*. Vol. 26. 3. Wiley Online Library. 2007, pp. 375–384.

[119] Evangelos Kalogerakis, Aaron Hertzmann, and Karan Singh. "Learning 3D mesh segmentation and labeling". In: *ACM Transactions on Graphics (TOG)* 29.4 (2010), p. 102.

[120] Evangelos Kalogerakis et al. "3D Shape Segmentation with Projective Convolutional Networks". In: *arXiv preprint arXiv:1612.02808* (2016).

[121] Tero Karras et al. "Progressive growing of gans for improved quality, stability, and variation". In: *arXiv preprint arXiv:1710.10196* (2017).

[122] David G Kendall. "Shape manifolds, Procrustean metrics, and complex projective spaces". In: *Bulletin of the London Mathematical Society* 16.2 (1984), pp. 81–121.

[123] Nicolas Keriven and Gabriel Peyré. "Universal Invariant and Equivariant Graph Neural Networks". In: *CoRR* abs/1905.04943 (2019). arXiv: 1905.04943. URL: http://arxiv.org/abs/1905.04943.

[124] Itay Kezurer et al. "Tight Relaxation of Quadratic Matching". In: *Comput. Graph. Forum* 34.5 (Aug. 2015), pp. 115–128. ISSN: 0167-7055. DOI: 10.1111/cgf.12701. URL: http://dx.doi.org/10.1111/cgf.12701.

[125] Itay Kezurer et al. "Tight relaxation of quadratic matching". In: *Computer Graphics Forum*. Vol. 34. 5. Wiley Online Library. 2015, pp. 115–128.

[126] Vladimir G Kim, Yaron Lipman, and Thomas Funkhouser. "Blended intrinsic maps". In: *ACM Transactions on Graphics (TOG)*. Vol. 30. 4. ACM. 2011, p. 79.

[127] Vladimir G Kim, Yaron Lipman, and Thomas Funkhouser. "Blended intrinsic maps". In: *ACM Transactions on Graphics* 30.4 (2011), p. 1. ISSN: 07300301. DOI: 10.1145/2010324.1964974.

[128] Thomas N Kipf and Max Welling. "Semi-supervised classification with graph convolutional networks". In: *arXiv preprint arXiv:1609.02907* (2016).

[129] Roman Klokov and Victor Lempitsky. "Escape from Cells: Deep Kd-Networks for The Recognition of 3D Point Cloud Models". In: *arXiv preprint arXiv:1704.01222* (2017).

[130] Vladimir Kolmogorov. "Convergent tree-reweighted message passing for energy minimization". In: *IEEE transactions on pattern analysis and machine intelligence* 28.10 (2006), pp. 1568–1583.

[131] Risi Kondor and Shubhendu Trivedi. "On the generalization of equivariance and convolution in neural networks to the action of compact groups". In: *arXiv preprint arXiv:1802.03690* (2018).

[132] Risi Kondor et al. "Covariant compositional networks for learning graphs". In: *arXiv preprint arXiv:1801.02144* (2018).

[133] JJ Kosowsky and Alan L Yuille. "The invisible hand algorithm: Solving the assignment problem with statistical physics". In: *Neural networks* 7.3 (1994), pp. 477–490.

[134] Hanspeter Kraft and Claudio Procesi. "Classical invariant theory, a primer". In: *Lecture Notes, Version* (2000).

[135] Nils M. Kriege, Fredrik D. Johansson, and Christopher Morris. *A Survey on Graph Kernels*. 2019. arXiv: 1903.11835 [cs.LG].

[136] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems*. 2012, pp. 1097–1105.

[137] Joseph B Kruskal and Myron Wish. *Multidimensional scaling*. Vol. 11. Sage, 1978.

[139] Dmitry Laptev et al. "TI-POOLING: transformation-invariant pooling for feature learning in Convolutional Neural Networks". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 289–297.

[140] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep learning". In: *nature* 521.7553 (2015), p. 436.

[141] Yann LeCun et al. "Backpropagation applied to handwritten zip code recognition". In: *Neural computation* 1.4 (1989), pp. 541–551.

[142] Tao Lei et al. "Deriving neural architectures from sequence and graph kernels". In: *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org. 2017, pp. 2024–2033.

[143] Marius Leordeanu and Martial Hebert. "A spectral technique for correspondence problems using pairwise constraints". In: *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*. Vol. 2. IEEE. 2005, pp. 1482–1489.

[144] Ron Levie et al. "CayleyNets: Graph Convolutional Neural Networks with Complex Rational Spectral Filters". In: (2017), pp. 1–12. ISSN: 1063-6919. DOI: 10.1109/CVPR.2017.576. arXiv: 1705.07664. URL: http://arxiv.org/abs/1705.07664.

[145] Hao Li, Robert W Sumner, and Mark Pauly. "Global Correspondence Optimization for Non-Rigid Registration of Depth Scans". In: *Computer Graphics Forum*. Vol. 27. 5. Wiley Online Library. 2008, pp. 1421–1430.

[146] Yujia Li et al. "Gated Graph Sequence Neural Networks". In: 1 (2015), pp. 1–20. ISSN: 10797114. DOI: 10.1103/PhysRevLett.116.082003. arXiv: 1511.05493. URL: http://arxiv.org/abs/1511.05493.

[147] Zhuwen Li, Qifeng Chen, and Vladlen Koltun. "Combinatorial optimization with graph convolutional networks and guided tree search". In: *Advances in Neural Information Processing Systems*. 2018, pp. 537–546.

[148] Yaron Lipman and Thomas Funkhouser. "Möbius voting for surface correspondence". In: *ACM Transactions on Graphics (TOG)*. Vol. 28. 3. ACM. 2009, p. 72.

[149] Jingen Liu, Jiebo Luo, and Mubarak Shah. "Recognizing realistic actions from videos "in the wild"". In: *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE. 2009, pp. 1996–2003.

[150] Eliane Maria Loiola et al. "A survey for the quadratic assignment problem". In: *European journal of operational research* 176.2 (2007), pp. 657–690.

[151] Jonathan Long, Evan Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 3431–3440.

[152] Zhi-Quan Luo et al. "Semidefinite relaxation of quadratic optimization problems". In: *Signal Processing Magazine, IEEE* 27.3 (2010), pp. 20–34.

[153] Vince Lyzinski et al. "Graph Matching: Relax at Your Own Risk". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2016). ISSN: 01628828. DOI: 10.1109/TPAMI.2015.2424894. arXiv: 1405.3133.

[154] Vince Lyzinski et al. "Graph Matching: Relax at Your Own Risk." In: *IEEE Trans. Pattern Anal. Mach. Intell.* 38.1 (2016), pp. 60–73. URL: http://dblp.uni-trier.de/db/journals/pami/pami38.html#LyzinskiFFVPS16.

[155] Wolfgang Maier. "Tooth morphology and dietary specialization". In: *Food acquisition and processing in primates*. Springer, 1984, pp. 303–330.

[162] Jonathan Masci et al. "Geodesic convolutional neural networks on riemannian manifolds". In: *Proceedings of the IEEE international conference on computer vision workshops*. 2015, pp. 37–45.

[163] Daniel Maturana and Sebastian 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.

[164] Tim McInerney and Demetri Terzopoulos. "Deformable models in medical image analysis: a survey". In: *Medical image analysis* 1.2 (1996), pp. 91–108.

[165] Facundo Mémoli. "Gromov–Wasserstein distances and the metric approach to object matching". In: *Foundations of computational mathematics* 11.4 (2011), pp. 417–487.

[166] Facundo Mémoli and Guillermo Sapiro. "A theoretical and computational framework for isometry invariant recognition of point cloud data". In: *Foundations of Computational Mathematics* 5.3 (2005), pp. 313–347.

[167] Facundo Mémoli and Guillermo Sapiro. "Comparing Point Clouds". In: *Proceedings of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*. SGP '04. Nice, France: ACM, 2004, pp. 32–40. ISBN: 3-905673-13-4. DOI: 10.1145/1057432.1057436. URL: http://doi.acm.org/10.1145/1057432.1057436.

[168] J Milnor. "Topology from a differentiable viewpoint(University of Virginia Press, Charlottesville, VA)". In: (1965).

[169] Philipp Mitteroecker and Philipp Gunz. "Advances in geometric morphometrics". In: *Evolutionary Biology* 36.2 (2009), pp. 235–247.

[170] Federico Monti et al. "Dual-Primal Graph Convolutional Networks". In: (2018), pp. 1–11. arXiv: 1806.00770. URL: http://arxiv.org/abs/1806.00770.

[171] Federico Monti et al. "Geometric deep learning on graphs and manifolds using mixture model CNNs". In: *arXiv preprint arXiv:1611.08402* (2016).

[172] Federico Monti et al. "Geometric deep learning on graphs and manifolds using mixture model CNNs". In: *Proc. CVPR*. Vol. 1. 2. 2017, p. 3.

[173] Christopher Morris, Kristian Kersting, and Petra Mutzel. "Glocalized Weisfeiler-Lehman graph kernels: Global-local feature maps of graphs". In: *2017 IEEE International Conference on Data Mining (ICDM)*. IEEE. 2017, pp. 327–336.

[174] Christopher Morris and Petra Mutzel. "Towards a practical k -dimensional Weisfeiler-Leman algorithm". In: *arXiv preprint arXiv:1904.01543* (2019).

[175] Christopher Morris et al. "Weisfeiler and Leman Go Neural: Higher-order Graph Neural Networks". In: *arXiv preprint arXiv:1810.02244* (2018).

[176] MOSEK. *The MOSEK optimization toolbox for MATLAB manual. Version 7.1 (Revision 49).* 2015. URL: http://docs.mosek.com/7.1/toolbox/index.html.

[177] Ryan L Murphy et al. "Relational Pooling for Graph Representations". In: *arXiv preprint arXiv:1903.02541* (2019).

[178] Elizbar A Nadaraya. "On estimating regression". In: *Theory of Probability & Its Applications* 9.1 (1964), pp. 141–142.

[179] Marion Neumann et al. "Propagation kernels: efficient graph kernels from propagated information". In: *Machine Learning* 102.2 (2016), pp. 209–245.

[180] Andy Nguyen et al. "An Optimization Approach to Improving Collections of Shape Maps". In: *Computer Graphics Forum* 30.5 (2011), pp. 1481–1491. ISSN: 1467-8659. DOI: 10.1111/j.1467-8659.2011.02022.x. URL: http://dx.doi.org/10.1111/j.1467-8659.2011.02022.x.

[181] Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. "Learning Convolutional Neural Networks for Graphs". In: (2016). ISSN: 1938-7228. arXiv: 1605.05273.

[182] Richard G Ogier and DA Beyer. "Neural network solution to the link scheduling problem using convex relaxation". In: *Global Telecommunications Conference, 1990, and Exhibition.'Communications: Connecting the Future', GLOBECOM'90., IEEE*. IEEE. 1990, pp. 1371–1376.

[183] Mark JL Orr et al. *Introduction to radial basis function networks*. 1996.

[184] Mark JL Orr. "Recent advances in radial basis function networks". In: *Institute for Adaptative and Neural Computation* (1999).

[185] Maks Ovsjanikov, Jian Sun, and Leonidas Guibas. "Global intrinsic symmetries of shapes". In: *Computer Graphics Forum*. Vol. 27. 5. Wiley Online Library. 2008, pp. 1341–1348.

[186] Maks Ovsjanikov et al. "Functional maps: a flexible representation of maps between shapes". In: *ACM Transactions on Graphics (TOG)* 31.4 (2012), p. 30.

[187] Maks Ovsjanikov et al. "One point isometric matching with the heat kernel". In: *Computer Graphics Forum*. Vol. 29. 5. Wiley Online Library. 2010, pp. 1555–1564.

[188] Jooyoung Park and Irwin W Sandberg. "Universal approximation using radial-basis-function networks". In: *Neural computation* 3.2 (1991), pp. 246–257.

[189] Omkar M Parkhi, Andrea Vedaldi, and Andrew Zisserman. "Deep face recognition". In: *British Machine Vision Conference*. Vol. 1. 3. 2015, p. 6.

[190] Ulrich Pinkall and Konrad Polthier. "Computing discrete minimal surfaces and their conjugates". In: *Experimental mathematics* 2.1 (1993), pp. 15–36.

[191] J. Pokrass et al. "Sparse modeling of intrinsic correspondences". In: *Computer Graphics Forum* 32.2 PART4 (2013), pp. 459–468. ISSN: 01677055. DOI: 10.1111/cgf.12066. arXiv: arXiv:1209.6560v1.

[192] Svatopluk Poljak, Franz Rendl, and Henry Wolkowicz. "A recipe for semidefinite relaxation for (0, 1)-quadratic programming". In: *Journal of Global Optimization* 7.1 (1995), pp. 51–73.

[193] Emil Praun and Hugues Hoppe. "Spherical parametrization and remeshing". In: *ACM Transactions on Graphics (TOG)*. Vol. 22. 3. ACM. 2003, pp. 340–349.

[194] Charles Ruizhongtai Qi et al. "Volumetric and Multi-View CNNs for Object Classification on 3D Data". In: *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*. 2016.

[195] Charles R Qi et al. "PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space". In: *arXiv preprint arXiv:1706.02413* (2017).

[196] Charles R. Qi et al. "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation". In: *arXiv preprint arXiv:1612.00593* (2016).

[197] Charles R Qi et al. "Pointnet: Deep learning on point sets for 3d classification and segmentation". In: *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE* 1.2 (2017), p. 4.

[198] Novi Quadrianto, Le Song, and Alex J Smola. "Kernelized sorting". In: *Advances in neural information processing systems*. 2009, pp. 1289–1296.

[199] Raghunathan Ramakrishnan et al. "Quantum chemistry structures and properties of 134 kilo molecules". In: *Scientific data* 1 (2014), p. 140022.

[200] Anand Rangarajan, Steven Gold, and Eric Mjolsness. "A novel optimizing network architecture with applications". In: *Neural Computation* 8.5 (1996), pp. 1041–1060.

[201] Anand Rangarajan et al. "A convergence proof for the softassign quadratic assignment algorithm". In: *Advances in neural information processing systems* (1997), pp. 620–626.

[202] Siamak Ravanbakhsh, Jeff Schneider, and Barnabas Poczos. "Deep learning with sets and point clouds". In: *arXiv preprint arXiv:1611.04500* (2016).

[203] Siamak Ravanbakhsh, Jeff Schneider, and Barnabas Poczos. "Equivariance through parameter-sharing". In: *arXiv preprint arXiv:1702.08389* (2017).

[204] Gernot Riegler, Ali Osman Ulusoys, and Andreas Geiger. "Octnet: Learning deep 3d representations at high resolutions". In: *arXiv preprint arXiv:1611.05009* (2016).

[205] Emanuele Rodola et al. "A game-theoretic approach to deformable shape matching". In: *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE. 2012, pp. 182–189.

[206] Emanuele Rodola et al. "Elastic net constraints for shape matching". In: *Computer Vision (ICCV), 2013 IEEE International Conference on*. IEEE. 2013, pp. 1169–1176.

[207] Emanuele Rodolà et al. "Dense non-rigid shape correspondence using random forests". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2014, pp. 4177–4184.

[208] Mark Rudelson, Roman Vershynin, et al. "Hanson-Wright inequality and sub-gaussian concentration". In: *Electronic Communications in Probability* 18 (2013).

[209] Szymon Rusinkiewicz and Marc Levoy. "Efficient variants of the ICP algorithm". In: *Proceedings Third International Conference on 3-D Digital Imaging and Modeling* (2001), pp. 145–152. ISSN: 08876185. DOI: 10.1109/IM.2001.924423. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=924423.

[210] Raif M. Rustamov. "Laplace-Beltrami Eigenfunctions for Deformation Invariant Shape Representation". In: *Proceedings of the Fifth Eurographics Symposium on Geometry Processing*. SGP '07. Barcelona, Spain: Eurographics Association, 2007, pp. 225–233. ISBN: 978-3-905673-46-3. URL: http://dl.acm.org/citation.cfm?id=1281991.1282022.

[211] David Rydh. "A minimal set of generators for the ring of multisymmetric functions". In: *Annales de l'institut Fourier*. Vol. 57. 6. 2007, pp. 1741–1769.

[212] James Saunderson, Pablo A Parrilo, and Alan S Willsky. "Semidefinite descriptions of the convex hull of rotation matrices". In: *arXiv preprint arXiv:1403.4914* (2014).

[213] Franco Scarselli et al. "The graph neural network model". In: *Neural Networks, IEEE Transactions on* 20.1 (2009), pp. 61–80. ISSN: 1045-9227. DOI: 10.1109/TNN.2008.2005605.

[214] Christian Schellewald, Stefan Roth, and Christoph Schnörr. "Evaluation of convex optimization techniques for the weighted graph-matching problem in computer vision". In: *Joint Pattern Recognition Symposium*. Springer. 2001, pp. 361–368.

[215] Florian Schroff, Dmitry Kalenichenko, and James Philbin. "Facenet: A unified embedding for face recognition and clustering". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 815–823.

[216] Kristof Schütt et al. "MolecuLeNet: A continuous-filter convolutional neural network for modeling quantum interactions". In: *Advances in Neural Information Processing Systems*. 2017, pp. 992–1002.

[217] Tianjia Shao et al. "Interpreting concept sketches". In: *ACM Transactions on Graphics (TOG)* 32.4 (2013), p. 56.

[218] Nino Shervashidze et al. "Efficient graphlet kernels for large graph comparison". In: *Artificial Intelligence and Statistics*. 2009, pp. 488–495.

[219] Nino Shervashidze et al. "Weisfeiler-lehman graph kernels". In: *Journal of Machine Learning Research* 12.Sep (2011), pp. 2539–2561.

[220] Martin Simonovsky and Nikos Komodakis. "Dynamic Edge-Conditioned Filters in Convolutional Neural Networks on Graphs". In: *arXiv preprint arXiv:1704.02901* (2017).

[221] Martin Simonovsky and Nikos Komodakis. "Dynamic edge-conditioned filters in convolutional neural networks on graphs". In: *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*. 2017. ISBN: 9781538604571. DOI: 10.1109/CVPR.2017.11. arXiv: 1704.02901.

[222] A. Singer and H.-T. Wu. "Vector diffusion maps and the connection Laplacian". In: *Communications on Pure and Applied Mathematics* 65.8 (2012), pp. 1067–1144. ISSN: 1097-0312. DOI: 10.1002/cpa.21395. URL: http://dx.doi.org/10.1002/cpa.21395.

[223] Ayan Sinha, Jing Bai, and Karthik Ramani. "Deep learning 3D shape surfaces using geometry images". In: *European Conference on Computer Vision*. Springer. 2016, pp. 223–240.

[224] Justin Solomon et al. "Convolutional wasserstein distances: Efficient optimal transportation on geometric domains". In: *ACM Transactions on Graphics (TOG)* 34.4 (2015), p. 66.

[225] Justin Solomon et al. "Entropic Metric Alignment for Correspondence Problems". In: *ACM Trans. Graph.* 35.4 (July 2016), 72:1–72:13. ISSN: 0730-0301. DOI: 10.1145/2897824.2925903. URL: http://doi.acm.org/10.1145/2897824.2925903.

[226] Justin Solomon et al. "Entropic metric alignment for correspondence problems". In: *ACM Transactions on Graphics (TOG)* 35.4 (2016), p. 72.

[227] Justin Solomon et al. "Soft maps between surfaces". In: *Computer Graphics Forum*. Vol. 31. 5. Wiley Online Library. 2012, pp. 1617–1626.

[228] Grant Strong and Minglun Gong. "Self-sorting map: An efficient algorithm for presenting multimedia data in structured layouts". In: *IEEE Transactions on Multimedia* 16.4 (2014), pp. 1045–1058.

[229] Hang Su et al. "Multi-view convolutional neural networks for 3d shape recognition". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015, pp. 945–953.

[230] Gary KL Tam et al. "Registration of 3D point clouds and meshes: a survey from rigid to nonrigid". In: *Visualization and Computer Graphics, IEEE Transactions on* 19.7 (2013), pp. 1199–1217.

[231] Art Tevs et al. "Isometric registration of ambiguous and partial data". In: *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE. 2009, pp. 1185–1192.

[232] W. T. Tutte. "How to draw a graph". In: *Proc. London Math. Soc.* 13.3 (1963), pp. 743–768.

[233] Shinji Umeyama. "An eigendecomposition approach to weighted graph matching problems". In: *IEEE transactions on pattern analysis and machine intelligence* 10.5 (1988), pp. 695–703.

[234] Oliver Van Kaick et al. "A survey on shape correspondence". In: *Computer Graphics Forum*. Vol. 30. 6. Wiley Online Library. 2011, pp. 1681–1707.

[235] Andrea Vedaldi and Karel Lenc. "Matconvnet: Convolutional neural networks for matlab". In: *Proceedings of the 23rd ACM international conference on Multimedia*. ACM. 2015, pp. 689–692.

[236] Petar Veličković et al. "Graph Attention Networks". In: (2017), pp. 1–12. arXiv: 1710.10903. URL: http://arxiv.org/abs/1710.10903.

[237] Saurabh Verma and Zhi-Li Zhang. "Hunt For The Unique, Stable, Sparse And Fast Feature Learning On Graphs". In: *Advances in Neural Information Processing Systems*. 2017, pp. 88–98.

[238] Matthias Vestner et al. "Product Manifold Filter: Non-rigid Shape Correspondence via Kernel Density Estimation in the Product Space". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. 2017, pp. 6681–6690.

[239] S Vichy N Vishwanathan et al. "Graph kernels". In: *Journal of Machine Learning Research* 11.Apr (2010), pp. 1201–1242.

[240] Daniel Vlasic et al. "Articulated mesh animation from multi-view silhouettes". In: *ACM Transactions on Graphics (TOG)*. Vol. 27. 3. ACM. 2008, p. 97.

[241] Joshua T Vogelstein et al. "Fast approximate quadratic programming for graph matching". In: *PLOS one* 10.4 (2015), e0121002.

[242] Hayato Waki et al. "Sums of Squares and Semidefinite Programming Relaxations for Polynomial Optimization Problems with Structured Sparsity". In: *SIAM Journal on Optimization* 17 (2006), pp. 218–242.

[243] Michael Wand et al. *Computing Correspondences in Geometric Data Sets Tutorial*. http://resources.mpi-inf.mpg.de/deformableShapeMatching/EG2011_Tutorial/. Eurographics Association, 2011.

[244] Peng-Shuai Wang et al. "O-cnn: Octree-based convolutional neural networks for 3d shape analysis". In: *ACM Transactions on Graphics (TOG)* 36.4 (2017), p. 72.

[245] Lingyu Wei et al. "Dense Human Body Correspondences Using Convolutional Networks". In: *Computer Vision and Pattern Recognition (CVPR)*. 2016.

[246] Maurice Weiler et al. "3D Steerable CNNs: Learning Rotationally Equivariant Features in Volumetric Data". In: (2018). arXiv: 1807.02547. URL: http://arxiv.org/abs/1807.02547.

[247] Eric W Weisstein. "Normal sum distribution". In: (2000).

[248] Holger Wendland. *Scattered data approximation*. Vol. 17. Cambridge university press, 2004.

[249] Tomas Werner. "A linear programming approach to max-sum problem: A review". In: *IEEE transactions on pattern analysis and machine intelligence* 29.7 (2007).

[250] Helmut Wielandt. "Permutation groups through invariant relations and invariant functions". In: (1969).

[251] Zhenqin Wu et al. "MoleculeNet: a benchmark for molecular machine learning". In: *Chemical science* 9.2 (2018), pp. 513–530.

[253] Yong Xia. "Second order cone programming relaxation for quadratic assignment problems". In: *Optimization Methods & Software* 23.3 (2008), pp. 441–449.

[254] Jianxiong Xiao et al. "Sun database: Large-scale scene recognition from abbey to zoo". In: *Computer vision and pattern recognition (CVPR), 2010 IEEE conference on*. IEEE. 2010, pp. 3485–3492.

[255] Keyulu Xu et al. "How Powerful are Graph Neural Networks?" In: *arXiv preprint arXiv:1810.00826* (2018).

[256] Keyulu Xu et al. "How Powerful are Graph Neural Networks?" In: *International Conference on Learning Representations*. 2019. URL: https://openreview.net/forum?id=ryGs6iA5Km.

[257] Pinar Yanardag and S.V.N. Vishwanathan. "Deep Graph Kernels". In: *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '15*. 2015. ISBN: 9781450336642. DOI: 10.1145/2783258.2783417.

[258] Jiaolong Yang, Hongdong Li, and Yunde Jia. "Go-ICP: Solving 3D Registration Efficiently and Globally Optimally". In: *2013 IEEE International Conference on Computer Vision* (2013), pp. 1457–1464. ISSN: 1550-5499. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6751291.

[259] Dmitry Yarotsky. "Universal approximations of invariant maps by neural networks". In: *arXiv preprint arXiv:1804.10306* (2018).

[260] Li Yi et al. "A scalable active framework for region annotation in 3d shape collections". In: *ACM Transactions on Graphics (TOG)* 35.6 (2016), p. 210.

[261] Li Yi et al. "SyncSpecCNN: Synchronized Spectral CNN for 3D Shape Segmentation". In: *arXiv preprint arXiv:1612.00606* (2016).

[262] Rex Ying et al. "Hierarchical Graph Representation Learning with Differentiable Pooling". In: (2018). DOI: 10.1145/nnnnnnn.nnnnnnn. arXiv: 1806.08804. URL: http://arxiv.org/abs/1806.08804.

[263] *Yobi3d - free 3d model search engine*. https://www.yobi3d.com. Accessed: 2016-10-15. 2016.

[264] Manzil Zaheer et al. "Deep sets". In: *Advances in Neural Information Processing Systems*. 2017, pp. 3391–3401.

[265] Mikhail Zaslavskiy, Francis Bach, and Jean-Philippe Vert. "A path following algorithm for the graph matching problem". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 31.12 (2009), pp. 2227–2242.

[266] Yun Zeng et al. "Dense non-rigid surface registration using high-order graph matching". In: *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. IEEE. 2010, pp. 382–389.

[267] Muhan Zhang and Yixin Chen. "Weisfeiler-Lehman neural machine for link prediction". In: *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM. 2017, pp. 575–583.

[268] Muhan Zhang et al. "An end-to-end deep learning architecture for graph classification". In: *Proceedings of AAAI Conference on Artificial Inteligence*. 2018.

[269] Hong-Kai Zhao et al. "Implicit and nonparametric shape reconstruction from unorganized data using a variational level set method". In: *Computer Vision and Image Understanding* 80.3 (2000), pp. 295–314.

[270] Qing Zhao et al. "Semidefinite programming relaxations for the quadratic assignment problem". In: *Journal of Combinatorial Optimization* 2.1 (1998), pp. 71–109.

[271] Feng Zhou and Fernando De la Torre. "Factorized graph matching". In: *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE. 2012, pp. 127–134.

[272] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. "Fast global registration". In: *European Conference on Computer Vision*. Springer. 2016, pp. 766–782.

[273] Silvia Zuffi and Michael J Black. "The Stitched Puppet: A Graphical Model of 3D Human Shape and Pose". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 3537–3546.