Geometric deep learning: going beyond Euclidean data

Michael M. Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, Pierre Vandergheynst

Many signal processing problems involve data whose underlying structure is non-Euclidean, but may be modeled as a manifold or (combinatorial) graph. For instance, in social networks, the characteristics of users can be modeled as signals on the vertices of the social graph [1]. Sensor networks are graph models of distributed interconnected sensors, whose readings are modelled as time-dependent signals on the vertices. In genetics, gene expression data are modeled as signals defined on the regulatory network [2]. In neuroscience, graph models are used to represent anatomical and functional structures of the brain. In computer graphics and vision, 3D objects are modeled as Riemannian manifolds (surfaces) endowed with properties such as color texture. Even more complex examples include networks of operators, e.g., functional correspondences [3] or difference operators [4] in a collection of 3D shapes, or orientations of overlapping cameras in multi-view vision ("structure from motion") problems [5].

The complexity of geometric data and the availability of very large datasets (in the case of social networks, on the scale of billions) suggest the use of machine learning techniques. In particular, deep learning has recently proven to be a powerful tool for problems with large datasets with underlying Euclidean structure.

The purpose of this paper is to overview the problems arising in relation to geometric deep learning and present solutions existing today for this class of problems, as well as key difficulties and future research directions.

I. Introduction

"Deep learning" refers to learning complicated concepts by building them from simpler ones in a hierarchical or multi-layer manner [6]. Artificial neural networks (ANNs) are popular realizations of such deep multi-layer hierarchies. In the past few years, the growing computational power of modern GPU-based computers and the availability of large training datasets have allowed successfully training ANNs with many layers and degrees of freedom [7]. This has led to qualitative breakthroughs on a wide variety of tasks, from speech recognition [8], [9] and machine translation [10] to image analysis and computer vision [11], [12], [13], [14], [15], [16], [17]. Nowadays, deep learning methods are widely used in commercial applications, including Siri speech recognition in Apple iPhone, Google text translation, and Mobileye vision-based technology for autonomously driving cars.

Constructions that leverage the statistical properties of the data, in particular stationarity and compositionality through local statistics, which are present in natural images, video, and speech [18], [19], are one of the key reasons for the success of deep neural networks in these domains. These statistical properties have been related to physics [20] and formalized in specific classes of convolutional neural networks (CNNs) [21], [22], [23]. For example, one can think of images as functions on the Euclidean space (plane), sampled on a grid. In this setting, stationarity is owed to shift-invariance, locality is due to the local connectivity, and compositionality stems from the multi-resolution structure of the grid. These properties are exploited by convolutional architectures [24], which are built of alternating convolutional and downsampling (pooling) layers. The use of convolutions has a two-fold effect. First, it allows extracting local features that are shared across the image domain and greatly reduces the number of parameters in the network with respect to generic deep architectures (and thus also the risk of overfitting), without sacrificing the expressive capacity of the network. Second, as we will show in the following, the convolutional architecture itself imposes some priors about the data, which appear very suitable especially for natural images [25], [22].

While deep learning models have been particularly successful when dealing with signals such as speech, images, or video, in which there is an underlying Euclidean structure, recently there has been a growing interest in trying to apply learning on non-Euclidean geometric data, for example, in computer graphics and vision [26], [27], [28], natural language processing [29], and biology [30], [31]. The non-Euclidean nature of such data implies that there are no such familiar properties as global parameterization, common system of coordinates, vector space structure, or shift-invariance. Consequently, basic operations like linear combination or convolution that are taken for granted in the Euclidean case are even not well defined on non-Euclidean domains. This major obstacle that has so far precluded the use of successful deep learning methods such as convolutional networks on generic non-Euclidean geometric data. As a result, the quantitative and qualitative breakthrough that deep learning methods have brought into speech recognition, natural language processing, and computer vision has not yet come to fields dealing with functions defined on more general geometric data.

II. Geometric learning problems

Broadly speaking, we can distinguish between two classes of geometric learning problems. In the first class of problems,

MB is with USI Lugano, Switzerland, Tel Aviv University, and Intel Perceptual Computing, Israel. JB is with Courant Institute, NYU and UC Berkeley, USA. YL with with Facebook AI Research and NYU, USA. AS is with Facebook AI Research, USA. PV is with EPFL, Switzerland.

the goal is to characterize the *structure* of the data. For example, given a set of data points with some underlying lower dimensional structure embedded into a high-dimensional Euclidean space, we may want to recover that lower dimensional structure. This is often referred to as *manifold learning*[1] or *non-linear dimensionality reduction*, and is an instance of unsupervised learning. Many methods for non-linear dimensionality reduction consist of two steps: first, they start with constructing a representation of local affinity of the data points (typically, a sparsely connected graph). Second, the data points are embedded into a low-dimensional space trying to preserve some criterion of the original affinity. For example, spectral embeddings tend to map points with many connections between them to nearby locations, and MDS-type methods try to preserve global information such as graph geodesic distances. Examples of such methods include different flavors of multidimensional scaling (MDS) [34], locally linear embedding (LLE) [35], stochastic neighbor embedding (t-SNE) [36], spectral embeddings such as Laplacian eigenmaps [37] and diffusion maps [38], and deep models [39]. Most recent approaches [40], [41], [42] tried to apply the successful word embedding model [43] to graphs. Instead of embedding the points, sometimes the graph can be processed directly, for example by decomposing it into small sub-graphs called *motifs* [44] or *graphlets* [45].

In some cases, the data are presented as a manifold or graph at the outset, and the first step of constructing the affinity structure described above is unnecessary. For instance, in computer graphics and vision applications, one can analyze 3D shapes represented as meshes by constructing local geometric descriptors capturing e.g. curvature-like properties [46], [47]. In network analysis applications such as computational sociology, the topological structure of the social graph representing the social relations between people carries important insights allowing, for example, to classify the vertices and detect communities [48]. In natural language processing, words in a corpus can be represented by the co-occurrence graph, where two words are connected if they often appear near each other [49].

The second class of problems deals with analyzing *functions* defined on a given non-Euclidean domain (these two classes are related, since understanding the properties of functions defined on a domain conveys certain information about the domain, and vice-versa, the structure of the domain imposes certain properties on the functions on it). We can further break down such problems into two subclasses: problems where the domain is *fixed* and those where *multiple domains* are given. Resorting again to the social network example, assume that we are given the geographic coordinates of users at different time, represented as a time-dependent signal on the vertices of the graph. An important application in location-based social networks is to predict the position of the user given his or her past behavior, as well as that of his or her friends [50]. In this problem, the domain (social graph) is assumed to be fixed; methods of *signal processing on graphs*, which have previously been reviewed in Signal Processing Magazine [51], can be applied to this setting, in particular, in order to define an operation similar to convolution in the spectral domain. This, in turn, allows generalizing CNN models to graphs [52], [53].

In computer graphics and vision applications, finding similarity and correspondence between shapes are examples of the second sub-class of problems: each shape is modeled as a manifold, and one has to work with multiple such domains. In this setting, a generalization of convolution in the spatial domain using local charting [54], [26], [28] appears to be more appropriate.

The main focus of this review is on this second class of problems, namely learning functions on non-Euclidean structured domains, and in particular, attempts to generalize the popular CNNs to such settings. We will start with an overview of Euclidean deep learning, summarizing the important assumptions about the data, and how they are realized in convolutional network architectures. For a more in-depth review of CNNs and their and applications, we refer the reader to [7] and references therein.

Going to the non-Euclidean world, we will then define basic notions in differential geometry and graph theory. These topics are insufficiently known in the signal processing community, and to our knowledge, there is no introductory-level reference treating these so different structures in a common way. One of our goals in this review is to provide an accessible overview of these models resorting as much as possible to the intuition of traditional signal processing. We will emphasize the similarities and the differences between Euclidean and non-Euclidean domains, and distinguish between spatial- and spectral domain learning methods.

Finally, we will show examples of selected problem from the fields of network analysis, computer vision, and graphics, and outline current main challenges and potential future research directions.

III. Deep learning on Euclidean domains

Geometric priors: Consider a compact d-dimensional Euclidean domain $\Omega = [0,1]^d \subset \mathbb{R}^d$ on which square-integrable functions $f \in L^2(\Omega)$ are defined (for example, in image analysis applications, images can be thought of as functions on the unit square $\Omega = [0,1]^2$). We consider a generic supervised learning setting, in which an unknown function $y: \; L^2(\Omega) \to \mathcal{Y}$ is observed on a training set

$$\{(f_i \in L^2(\Omega), y_i = y(f_i))\}_{i \in \mathcal{I}}. \quad (1)$$

In a supervised *classification* setting, the target space $\mathcal{Y}$ can be thought discrete with $|\mathcal{Y}|$ being the number of classes. In a *multiple object recognition* setting, we can replace $\mathcal{Y}$ by the K-dimensional simplex, which represents the posterior class probabilities $p(y|x)$. In *regression* tasks, we may consider $\mathcal{Y} = \mathbb{R}^m$.

In the vast majority of computer vision and speech analysis tasks, there are several crucial prior assumptions on the unknown function y. As we will see in the following, these assumptions are effectively exploited by convolutional neural network architectures.

[1] Note that the notion of "manifold" in this setting can be considerably more general than a classical smooth manifold; see e.g. [32], [33]

Notation

$\mathbb{R}^m$	m-dimensional Euclidean space
$a, \mathbf{a}, \mathbf{A}$	Scalar, vector, matrix
Ω, x	Arbitrary domain, coordinate on it
$f \in L^2(\Omega)$	Square-integrable function on Ω
$\mathcal{T}_v$	Translation operator
$\tau, \mathcal{L}_\tau$	Deformation field, operator
$\hat{f}$	Fourier transform of f
$f \star g$	Convolution of f and g
$\mathcal{X}, T\mathcal{X}, T_x\mathcal{X}$	Manifold, its tangent bundle, tangent space at x
$\langle\cdot,\cdot\rangle_{T\mathcal{X}}$	Riemannian metric
$f \in L^2(\mathcal{X})$	Scalar field on manifold $\mathcal{X}$
$F \in L^2(T\mathcal{X})$	Tangent vector field on manifold $\mathcal{X}$
$\nabla, \mathrm{div}, \Delta$	Gradient, divergence, Laplace operators
$\mathcal{V}, \mathcal{E}, \mathcal{F}$	Vertices and edges of a graph, faces of a mesh
$f \in L^2(\mathcal{V})$	Functions on vertices of a graph
$F \in L^2(\mathcal{E})$	Functions on edges of a graph
ϕ_i, λ_i	Laplacian eigenfunctions, eigenvalues
$h_t(\cdot,\cdot)$	Heat kernel
$\mathbf{\Phi}_k$	Matrix of first k Laplacian eigenvectors
$\mathbf{\Lambda}_k$	Diagonal matrix of first k Laplacian eigenvalues
ξ	point-wise nonlinearity
$w_{l,l'}(x), \mathbf{W}_{l,l'}$	Convolutional filter in spatial and spectral domain

Stationarity: A *translation operator*[2]

$$\mathcal{T}_v f(x) = f(x-v), \quad x, v \in \Omega, \tag{2}$$

acts on functions $f \in L^2(\Omega)$. Our first assumption is that the function y is either *invariant* or *covariant* with respect to translations, depending on the task. In the former case, we have $y(\mathcal{T}_v f) = y(f)$ for any $f \in L^2(\Omega)$ and $v \in \Omega$. This is typically the case in object classification tasks. In the latter, we have $y(\mathcal{T}_v f) = \mathcal{T}_v y(f)$, which is well-defined when the output of the model is a space in which translations can act upon (for example, in problems of object localization, semantic segmentation, or motion estimation). Equivalently, we can describe this property in terms of the statistics of natural images. If one considers that natural images are drawn from an underlying probability distribution, the invariance/covariance property implies that this distribution describes a stationary source [18].

Local deformations and scale separation: Similarly, a deformation $\mathcal{L}_\tau$, where $\tau : \Omega \to \Omega$ is a smooth vector field, acts on $L^2(\Omega)$ as $\mathcal{L}_\tau f(x) = f(x - \tau(x))$. Deformations can model local translations, changes in point of view, rotations and frequency transpositions [22].

Most tasks studied in computer vision are not only translation invariant/covariant, but also stable with respect to local deformations [55], [22]. In tasks that are translation invariant we have

$$|y(\mathcal{L}_\tau f) - y(f)| \approx \|\nabla\tau\|, \tag{3}$$

for all f, τ. Here, $\|\nabla\tau\|$ measures the smoothness of a given deformation field. In other words, the quantity to be predicted does not change much if the input image is slightly deformed. In tasks that are translation covariant, we have

$$|y(\mathcal{L}_\tau f) - \mathcal{L}_\tau y(f)| \approx \|\nabla\tau\|. \tag{4}$$

This property is much stronger than the previous one, since the space of local deformations has a high dimensionality, as opposed to the d-dimensional translation group.

It follows from (3) that we can extract sufficient statistics at a lower spatial resolution by downsampling demodulated localized filter responses without losing approximation power. An important consequence of this is that long-range dependencies can be broken into multi-scale local interaction terms, leading to hierarchical models in which spatial resolution is progressively reduced. To illustrate this principle, denote by

$$Y(x_1, x_2; v) = \mathrm{Prob}(f(u) = x_1 \text{ and } f(u+v) = x_2) \tag{5}$$

the joint distribution of two image pixels at an offset v from each other. In the presence of long-range dependencies, this joint distribution will not be separable for any v. However, the deformation stability prior states that $Y(x_1, x_2; v) \approx Y(x_1, x_2; v(1+\epsilon))$ for small ϵ. In other words, whereas long-range dependencies indeed exist in natural images and are critical to object recognition, they can be captured and down-sampled at different scales. This principle of stability to local deformations has been exploited in the computer vision community in models other than CNNs, for instance, deformable parts models [56].

In practice, the Euclidean domain Ω is discretized using a regular grid with n points; the translation and deformation operators are still well-defined so the above properties hold in the discrete setting.

Convolutional neural networks: Stationarity and stability to local translations are both leveraged in convolutional neural networks (see insert IN1). A CNN consists of several *convolutional layers* of the form $\mathbf{g} = C_W(\mathbf{f})$, acting on a p-dimensional input $\mathbf{f}(x) = (f_1(x), \dots, f_p(x))$ by applying a bank of filters $W = (w_{l,l'})$, $l = 1, \dots, q, l' = 1, \dots, p$ and point-wise non-linearity ξ,

$$g_l(x) = \xi\left(\sum_{l'=1}^{p} (f_{l'} \star w_{l,l'})(x)\right), \tag{6}$$

and producing a q-dimensional output $\mathbf{g}(x) = (g_1(x), \dots, g_q(x))$ often referred to as the *feature maps*. Here,

$$(f \star w)(x) = \int_\Omega f(x - x')w(x')dx' \tag{7}$$

denotes the standard convolution. According to the local deformation prior, the filters W have compact spatial support.

Additionally, a downsampling or *pooling* layer $\mathbf{g} = P(\mathbf{f})$ may be used, defined as

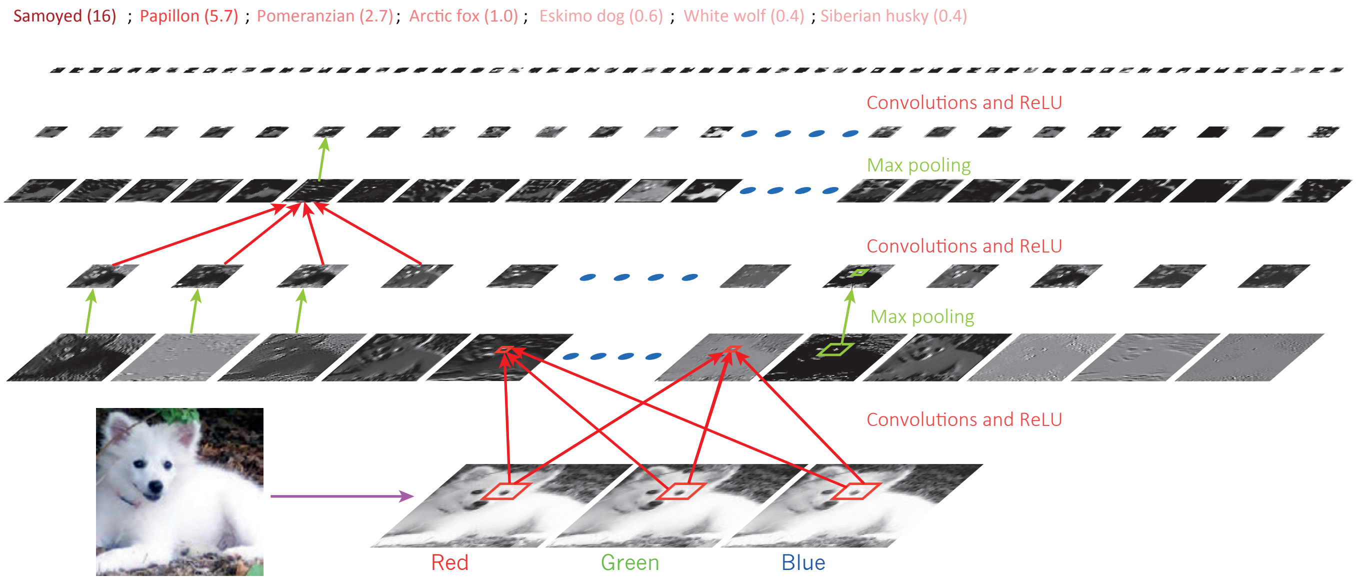
$$g_l(x) = \mathcal{G}(\{f_l(x') : x' \in \mathcal{N}(x)\}), \; l = 1, \dots, q, \tag{8}$$

[2] We assume periodic boundary conditions to ensure that the operation is well-defined over $L^2(\Omega)$.

[IN1] Convolutional neural networks: CNNs are currently among the most successful deep learning architectures in a variety of tasks, in particular, in computer vision. A typical CNN used in computer vision applications (see FIGS1) consists of multiple convolutional layers (6), passing the input image through a set of filters W followed by point-wise non-linearity ξ (typically, half-rectifiers $\xi(z) = \max(0, z)$ are used, although practitioners have experimented with a diverse range of choices [6]). The model can also include a bias term, which is equivalent to adding a constant coordinate to the input.

A network composed of K convolutional layers put together $U(f) = (C_{W^{(K)}} \ldots \circ C_{W^{(2)}} \circ C_{W^{(1)}})(f)$ produces pixel-wise features that are covariant w.r.t. translation and approximately covariant to local deformations. Typical computer vision applications requiring covariance are semantic image segmentation [14] or motion estimation [57].

In applications requiring invariance, such as image classification [13], the convolutional layers are typically interleaved with pooling layers (8) progressively reducing the resolution of the image passing through the network. Alternatively, one can integrate the convolution and downsampling in a single linear operator (convolution with stride). Recently, some authors have also experimented with convolutional layers which increase the spatial resolution using interpolation kernels [58]. These kernels can be learnt efficiently by mimicking the so-called *algorithme à trous* [59], also referred to as *dilated convolution*.


Samoyed (16) ; Papillon (5.7) ; Pomeranzian (2.7); Arctic fox (1.0) ; Eskimo dog (0.6) ; White wolf (0.4) ; Siberian husky (0.4)
Convolutions and ReLU
Max pooling
Convolutions and ReLU
Max pooling
Convolutions and ReLU
Red
Green
Blue

[FIGS1] Typical convolutional neural network architecture used in computer vision applications (figure reproduced from [7]).

where $\mathcal{N}(x) \subset \Omega$ is a neighborhood around x and $\mathcal{G}$ is a permutation-invariant function such as a L_p-norm (in the latter case, the choice of $p = 1, 2$ or ∞ results in average-, energy-, or max-pooling).

A convolutional network is constructed by composing several convolutional and optionally pooling layers, obtaining a generic hierarchical representation

$$U_{\boldsymbol{\Theta}}(f) = (C_{W^{(K)}} \cdots P \cdots \circ C_{W^{(2)}} \circ C_{W^{(1)}})(f) \qquad (9)$$

where $\boldsymbol{\Theta} = \{W^{(1)}, \ldots, W^{(K)}\}$ is the hyper-vector of the network parameters (all the filter coefficients). The model is said to be *deep* if it comprises multiple layers, though this notion is rather vague and one can find examples of CNNs with as few as a couple and as many as hundreds of layers [17]. The output features enjoy translation invariance/covariance depending on whether spatial resolution is progressively lost by means of pooling or kept fixed. Moreover, if one specifies the convolutional tensors to be complex wavelet decomposition operators and uses complex modulus as point-wise nonlinearities, one can provably obtain stability to local deformations [21]. Although this stability is not rigorously proved for generic compactly supported convolutional tensors, it underpins the empirical success of CNN architectures across a variety of computer vision applications [7].

In supervised learning tasks, one can obtain the CNN parameters by minimizing a task-specific cost L on the training set $\{f_i, y_i\}_{i \in \mathcal{I}}$,

$$\min_{\boldsymbol{\Theta}} \sum_{i \in \mathcal{I}} L(U_{\boldsymbol{\Theta}}(f_i), y_i). \qquad (10)$$

If the model is sufficiently complex and the training set is sufficiently representative, when applying the learned model to previously unseen data, one expects $U(f) \approx y(f)$. Although (10) is a non-convex optimization problem, stochastic optimization methods offer excellent empirical performance. Understanding the structure of the optimization problems (10) and finding efficient strategies for its solution is an active area of research in deep learning [60], [61], [62], [63].

A key advantage of CNNs explaining their success in numerous tasks is that the geometric priors on which CNNs are based result in a learning complexity that avoids the curse of dimensionality. Thanks to the stationarity and local deformation priors, the linear operators at each layer have a constant number of parameters, independent of the input size n (number of pixels in an image). Moreover, thanks to the multiscale hierarchical property, the number of layers grows at a rate $\mathcal{O}(\log n)$, resulting in a total learning complexity of $\mathcal{O}(\log n)$ parameters.

IV. The geometry of manifolds and graphs

Our main goal is to generalize CNN-type constructions to non-Euclidean domains. In this paper, by non-Euclidean domains, we refer to two prototypical structures: manifolds and graphs. While arising in very different fields of mathematics (differential geometry and graph theory, respectively), in our context, these structures share several common characteristics that we will try to emphasize throughout our review.

Manifolds: Roughly, a manifold is a space that is locally Euclidean. One of the simplest examples is a spherical surface modeling our Earth: around a point, it seems to be planar, which has led generations of people to believe in the flatness of the Earth. Formally speaking, a (differentiable) *d-dimensional manifold* $\mathcal{X}$ is a topological space where each point x has a neighborhood that is topologically equivalent (homeomorphic) to a d-dimensional Euclidean space, called the *tangent space* and denoted by $T_x\mathcal{X}$ (see Figure IV, top) The collection of tangent spaces at all points (more formally, their disjoint union) is referred to as the *tangent bundle* and denoted by $T\mathcal{X}$. On each tangent space, we define an inner product $\langle \cdot, \cdot \rangle_{T_x\mathcal{X}} : T_x\mathcal{X} \times T_x\mathcal{X} \to \mathbb{R}$, which is additionally assumed to depend smoothly on the position x. This inner product is called a *Riemannian metric* in differential geometry and allows performing local measurements of angles, distances, and volumes. A manifold equipped with a metric is called *Riemannian manifold*.

It is important to note that the definition of a Riemannian manifold is completely abstract and does not require a geometric realization in any space. However, a Riemannian manifold can be realized as a subset of a Euclidean space (in which case it is said to be *embedded* in that space) by using the structure of the Euclidean space to induce a Riemannian metric. The celebrated *Nash Embedding Theorem* guarantees that any sufficiently smooth Riemannian manifold can be realized in a Euclidean space of sufficiently high dimension [64]. An embedding is not necessarily unique; two different realizations of a Riemannian metric are called *isometries*.

Two-dimensional manifolds (surfaces) embedded into $\mathbb{R}^3$ are used in computer graphics and vision to describe boundary surfaces of 3D objects, colloquially referred to as ‘3D shapes’. This term is somewhat misleading since ‘3D’ here refers to the dimensionality of the embedding space rather than the manifold. Thinking of such a shape as made of infinitely thin material, inelastic deformations that do not stretch or tear it are isometric. Isometries do not affect the metric structure of the manifold and consequently, preserve any quantities that can be expressed in terms of the Riemannian metric (called *intrinsic*). Conversely, properties related to the specific realization of the manifold in the Euclidean space are called *extrinsic*.

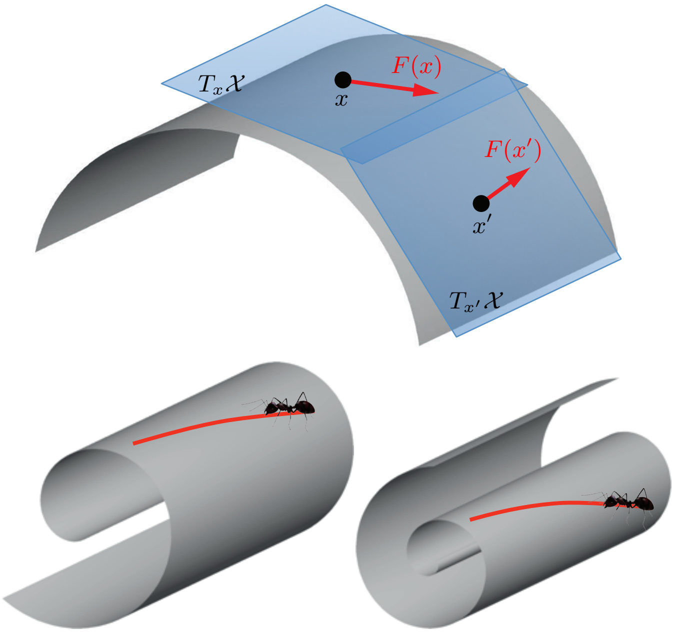


Fig. 1. Top: tangent space and tangent vectors on a two-dimensional manifold (surface). Bottom: Examples of isometric deformations.

As an intuitive illustration of this difference, imagine an insect that lives on a two-dimensional surface (Figure IV, bottom). The surface can be placed in the Euclidean space in any way, but as long as it is transformed isometrically, the insect would not notice any difference. The insect in fact does not even know of the existence of the embedding space, as its only world is 2D. This is an intrinsic viewpoint. A human observer, on the other hand, sees a surface in 3D space - this is an extrinsic point of view.

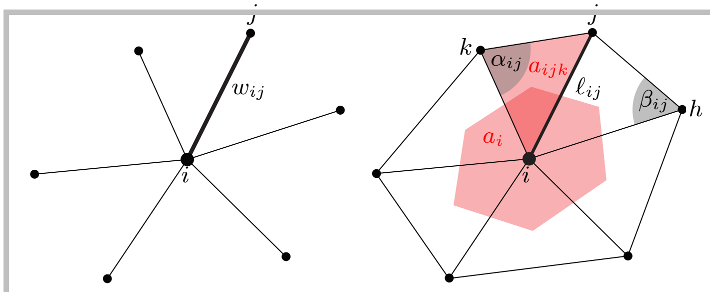
Calculus on manifolds: Our next step is to consider functions defined on manifolds. We are particularly interested in two types of functions: A *scalar field* is a smooth real function $f : \mathcal{X} \to \mathbb{R}$ on the manifold. A *tangent vector field* $F : \mathcal{X} \to T\mathcal{X}$ is a mapping attaching a tangent vector $F(x) \in T_x\mathcal{X}$ to each point x. As we will see in the following, tangent vector fields are used to formalize the notion of infinitesimal displacements on the manifold. We define the Hilbert spaces of scalar and vector fields on manifolds, denoted by $L^2(\mathcal{X})$ and $L^2(T\mathcal{X})$, respectively, with the following inner products:

$$\langle f, g \rangle_{L^2(\mathcal{X})} = \int_{\mathcal{X}} f(x) g(x) dx; \tag{15}$$

$$\langle F, G \rangle_{L^2(T\mathcal{X})} = \int_{\mathcal{X}} \langle F(x), G(x) \rangle_{T_x\mathcal{X}} dx; \tag{16}$$

dx denotes here a d-dimensional volume element induced by the Riemannian metric.

In calculus, the notion of derivative describes how the value of a function changes with an infinitesimal change of its argument. One of the big differences distinguishing classical calculus from differential geometry is a lack of vector space



Undirected graph

Triangular mesh

[FIGS2] Two commonly used discretizations of a two-dimensional manifold: a graph and a triangular mesh.

[IN2] Laplacian on discrete manifolds: In computer graphics and vision applications, two-dimensional manifolds are commonly used to model 3D shapes. There are several common ways of discretizing such manifolds. First, the manifold is assumed to be sampled at n points. Their embedding coordinates $\mathbf{x}_1, \dots, \mathbf{x}_n$ are referred to as *point cloud*. Second, a graph is constructed upon these points, acting as its vertices. The edges of the graph represent the local connectivity of the manifold, telling whether two points belong to a neighborhood or not, e.g. with Gaussian edge weights

$$w_{ij} = e^{-\|\mathbf{x}_i - \mathbf{x}_j\|^2/2\sigma^2}. \tag{11}$$

This simplest discretization of the manifold, however, does not capture correctly the geometry of the underlying continuous manifold (for example, the graph Laplacian would typically not converge to the continuous Laplacian operator of the manifold with the increase of the sampling density [65]). A geometrically consistent discretization is possible with an additional structure of *faces* $\mathcal{F} \in \mathcal{V} \times \mathcal{V} \times \mathcal{V}$, where $(i, j, k) \in \mathcal{F}$ implies $(i, j), (i, k), (k, j) \in \mathcal{E}$. The collection of faces represents the underlying continuous manifold as a polyhedral surface consisting of small triangles glued together. The triplet $(\mathcal{V}, \mathcal{E}, \mathcal{F})$ is referred to as *triangular mesh*. To be a correct discretization of a manifold (a *manifold mesh*), every edge must be shared by exactly two triangular faces; if the manifold has a boundary, any boundary edge must belong to exactly one triangle.

On a triangular mesh, the simplest discretization of the Riemannian metric is given by assigning each edge a length $\ell_{ij} > 0$, which must additionally satisfy the triangle inequality in every triangular face. The mesh Laplacian is given by formula (25) with

$$\begin{aligned} w_{ij} &= \frac{-\ell_{ij}^2 + \ell_{jk}^2 + \ell_{ik}^2}{8a_{ijk}} + \frac{-\ell_{ij}^2 + \ell_{jh}^2 + \ell_{ih}^2}{8a_{ijh}}; & (12) \\ a_i &= \frac{1}{3} \sum_{jk:(i,j,k) \in \mathcal{F}} a_{ijk}, & (13) \end{aligned}$$

where $a_{ijk} = \sqrt{s_{ijk}(s_{ijk} - \ell_{ij})(s_{ijk} - \ell_{jk})(s_{ijk} - \ell_{ik})}$ is the area of triangle ijk given by the Heron formula, and $s_{ijk} = \frac{1}{2}(\ell_{ij} + \ell_{jk} + \ell_{ki})$ is semi-perimeter of triangle ijk. The vertex weight a_i is interpreted as the local area element (shown in red in FIG2b). Note that the weights (12-13) are expressed solely in terms of the discrete metric ℓ and thus intrinsic. When the mesh is infinitely refined under some technical conditions, such a construction can be shown to converge to the continuous Laplacian of the underlying manifold [66].

An embedding of the mesh (amounting to specifying the vertex coordinates $\mathbf{x}_1, \dots, \mathbf{x}_n$) induces a discrete metric $\ell_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\|_2$, whereby (12) become the *cotangent weights*

$$w_{ij} = \tfrac{1}{2}\left(\cot \alpha_{ij} + \cot \beta_{ij}\right) \tag{14}$$

ubiquitously used in computer graphics [67].

structure on the manifold, prohibiting us from naïvely using expressions like $f(x+dx)$. The conceptual leap that is required to generalize such notions to manifolds is the need to work locally in the tangent space.

To this end, we define the *differential* of f as an operator $df : T\mathcal{X} \to \mathbb{R}$ acting on tangent vector fields. At each point x, the differential can be identified with a linear form $df(x) = \langle \nabla f(x), \cdot \rangle_{T_x\mathcal{X}}$ acting on tangent vectors $F(x) \in T_x\mathcal{X}$, which model a small displacement around x. The change of the function value as the result of this displacement is given by applying the form to the tangent vector, $df(x)F(x) = \langle \nabla f(x), F(x) \rangle_{T_x\mathcal{X}}$, and can be thought of as an extension of the notion of the classical directional derivative.

The operator $\nabla f : L^2(\mathcal{X}) \to L^2(T\mathcal{X})$ in the definition above is called the *intrinsic gradient*, and is similar to the classical notion of the gradient defining the direction of the steepest change of the function at a point, with the only difference that the direction is now a tangent vector. Similarly, the *intrinsic divergence* is an operator $\mathrm{div}F : L^2(T\mathcal{X}) \to L^2(\mathcal{X})$ acting on tangent vector fields and (formal) adjoint to the gradient operator [68],

$$\langle F, \nabla f \rangle_{L^2(T\mathcal{X})} = \langle -\mathrm{div}F, f \rangle_{L^2(\mathcal{X})}. \tag{17}$$

Physically, a tangent vector field can be thought of as a flow of material on a manifold. The divergence measures the net flow of a field at a point, allowing to distinguish between field ‘sources’ and ‘sinks’. Finally, the *Laplacian* (or *Laplace-Beltrami operator* in differential geometric jargon) $\Delta : L^2(\mathcal{X}) \to L^2(\mathcal{X})$ is an operator

$$\Delta f = -\mathrm{div}(\nabla f) \tag{18}$$

acting on scalar fields. Employing relation (17), it is easy to see that the Laplacian is self-adjoint (symmetric),

$$\langle \nabla f, \nabla f \rangle_{L^2(T\mathcal{X})} = \langle \Delta f, f \rangle_{L^2(\mathcal{X})} = \langle f, \Delta f \rangle_{L^2(\mathcal{X})}. \tag{19}$$

The lhs in equation (19) is known as the *Dirichlet energy* in physics and measures the smoothness of a scalar field on the manifold (see insert IN3). The Laplacian can be interpreted as the difference between the average of a function on an infinitesimal sphere around a point and the value of the function at the point itself. It is one of the most important operators in mathematical physics, used to describe phenomena as diverse as heat diffusion (see insert IN4), quantum mechanics, and wave propagation. As we will see in the following, the Laplacian plays a center role in signal processing and learning on non-Euclidean domains, as its eigenfunctions generalize the classical Fourier bases, allowing to perform spectral analysis on manifolds and graphs.

It is important to note that all the above definitions are *coordinate free*. By defining a basis in the tangent space, it is possible to express tangent vectors as d-dimensional vectors and the Riemannian metric as a $d \times d$ symmetric positive-definite matrix.

Graphs and discrete differential operators: Another type of constructions we are interested in are graphs, which are popular models of networks, interactions, and similarities between different objects. For simplicity, we will consider *weighted undirected graphs*, formally defined as a pair $(\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{1, \dots, n\}$ is the set of n vertices, and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of edges, where the graph being undirected implies that $(i,j) \in \mathcal{E}$ iff $(j,i) \in \mathcal{E}$. Furthermore, we associate a weight $a_i > 0$ with each vertex $i \in \mathcal{V}$, and a weight $w_{ij} \geq 0$ with each edge $(i,j) \in \mathcal{E}$.

Real functions $f : \mathcal{V} \to \mathbb{R}$ and $F : \mathcal{E} \to \mathbb{R}$ on the vertices and edges of the graph, respectively, are roughly the discrete analogy of continuous scalar and tangent vector fields in differential geometry.[3] We can define Hilbert spaces $L^2(\mathcal{V})$ and $L^2(\mathcal{E})$ of such functions by specifying the respective inner products,

$$\langle f, g \rangle_{L^2(\mathcal{V})} = \sum_{i \in \mathcal{V}} a_i f_i g_i; \qquad (20)$$

$$\langle F, G \rangle_{L^2(\mathcal{E})} = \sum_{i \in \mathcal{E}} w_{ij} F_{ij} G_{ij}. \qquad (21)$$

Let $f \in L^2(\mathcal{V})$ and $F \in L^2(\mathcal{E})$ be functions on the vertices and edges of the graphs, respectively. We can define differential operators acting on such functions analogously to differential operators on manifolds [69]. The *graph gradient* is an operator $\nabla : L^2(\mathcal{V}) \to L^2(\mathcal{E})$ mapping functions defined on vertices to functions defined on edges,

$$(\nabla f)_{ij} = f_i - f_j, \qquad (22)$$

automatically satisfying $(\nabla f)_{ij} = -(\nabla f)_{ji}$. The *graph divergence* is an operator $\mathrm{div} : L^2(\mathcal{E}) \to L^2(\mathcal{V})$ doing the converse,

$$(\mathrm{div} F)_i = \frac{1}{a_i} \sum_{j:(i,j) \in \mathcal{E}} w_{ij} F_{ij}. \qquad (23)$$

It is easy to verify that the two operators are adjoint w.r.t. the inner products (20–21),

$$\langle F, \nabla f \rangle_{L^2(\mathcal{E})} = \langle \nabla^* F, f \rangle_{L^2(\mathcal{V})} = \langle -\mathrm{div} F, f \rangle_{L^2(\mathcal{V})}. \qquad (24)$$

The *graph Laplacian* is an operator $\Delta : L^2(\mathcal{V}) \to L^2(\mathcal{V})$ defined as $\Delta = -\mathrm{div}\, \nabla$. Combining definitions (22-23), it can be expressed in the familiar form

$$(\Delta f)_i = \frac{1}{a_i} \sum_{(i,j) \in \mathcal{E}} w_{ij} (f_i - f_j). \qquad (25)$$

Note that formula (25) captures the intuitive geometric interpretation of the Laplacian as the difference between the local average of a function around a point and the value of the function at the point itself.

Denoting by $\mathbf{W} = (w_{ij})$ the $n \times n$ matrix of edge weights (it is assumed that $w_{ij} = 0$ if $(i,j) \notin \mathcal{E}$), by $\mathbf{A} = \mathrm{diag}(a_1, \dots, a_n)$ the diagonal matrix of vertex weights, and by $\mathbf{D} = \mathrm{diag}(\sum_{j \neq i} w_{ij})$ the *degree matrix*, the graph Laplacian application to a function $f \in L^2(\mathcal{V})$ represented as a column vector $\mathbf{f} = (f_1, \dots, f_n)^\top$ can be written in matrix-vector form as

$$\mathbf{\Delta f} = \mathbf{A}^{-1}(\mathbf{D} - \mathbf{W})\mathbf{f}. \qquad (26)$$

The choice of $\mathbf{A} = \mathbf{I}$ in (26) is referred to as the *unnormalized graph Laplacian*; another popular choice is $\mathbf{A} = \mathbf{D}$ producing the *random walk Laplacian* [70].

Discrete manifolds: As we mentioned, there are many practical situations in which one is given a sampling of points arising from a manifold but not the manifold itself. In computer graphics applications, reconstructing a correct discretization of a manifold from a point cloud is a difficult problem of its own, referred to a *meshing* (see insert IN2). In manifold learning problems, the manifold is typically approximated as a graph capturing the local affinity structure. We warn the reader that the term "manifold" as used in the context of generic data science is not geometrically rigorous, and can have less structure than a classical smooth manifold. For example, a set of points that "looks locally Euclidean" in practice may have self intersections, infinite curvature, different dimensions depending on the scale and location at which one looks, extreme variations in density, and "noise" with confounding structure.

Fourier analysis on non-Euclidean domains: The Laplacian operator is a self-adjoint positive-semidefinite operator, admitting on a compact domain[4] an eigendecomposition with a discrete set of orthonormal eigenfunctions $\phi_0, \phi_1, \dots$ (satisfying $\langle \phi_i, \phi_j \rangle = \delta_{ij}$) and non-negative real eigenvalues $0 = \lambda_0 \leq \lambda_1 \leq \dots$ (referred to as the *spectrum* of the Laplacian),

$$\Delta \phi_i = \lambda_i \phi_i, \quad i = 0, 1, \dots \qquad (31)$$

The eigenfunctions are the smoothest functions in the sense of the Dirichlet energy (see insert IN3) and can be interpreted as a generalization of the standard Fourier basis (given, in fact, by the eigenfunctions of the 1D Euclidean Laplacian, $-\frac{d^2}{x^2} e^{i\omega x} = \omega^2 e^{i\omega x}$) to a non-Euclidean domain. It is important to emphasize that the Laplacian eigenbasis is intrinsic due to the intrinsic construction of the Laplacian itself.

A smooth square-integrable function f on the domain can be decomposed into *Fourier series* as

$$f(x) = \sum_{i \geq 0} \underbrace{\langle f, \phi_i \rangle_{L^2(\mathcal{X})}}_{\hat{f}_i} \phi_i(x), \qquad (32)$$

[3] It is tacitly assumed here that F is *alternating*, i.e., $F_{ij} = -F_{ji}$.

[4] In the Euclidean case, the Fourier transform of a function defined on a finite interval (which is a compact set) or its periodic extension is discrete. In practical settings, all domains we are dealing with are compact.

[IN3] Physical interpretation of Laplacian eigenfunctions: Given a function f on the domain $\mathcal{X}$, the *Dirichlet energy*

$$\mathcal{E}_{\mathrm{Dir}}(f) \quad = \quad \int_{\mathcal{X}} \|\nabla f(x)\|^2 dx = \int_{\mathcal{X}} f(x)\Delta f(x) dx, \quad (27)$$

measures how smooth it is (the last identity in (27) stems from (19)). We are looking for an orthonormal basis on $\mathcal{X}$, containing k smoothest possible functions, by solving the optimization problem

$$\begin{aligned} &\min_{\phi_0} \ \mathcal{E}_{\mathrm{Dir}}(\phi_0) \quad \text{s.t.} \quad \|\phi_0\| = 1 && (28)\\ &\min_{\phi_i} \ \mathcal{E}_{\mathrm{Dir}}(\phi_i) \quad \text{s.t.} \quad \|\phi_i\| = 1, \quad i = 1, 2, \ldots k-1 \\ &\qquad\qquad\qquad\qquad\quad \phi_i \perp \mathrm{span}\{\phi_0, \ldots, \phi_{i-1}\}. \end{aligned}$$

In the discrete setting, when the domain is sampled at n points, problem (28) can be rewritten as

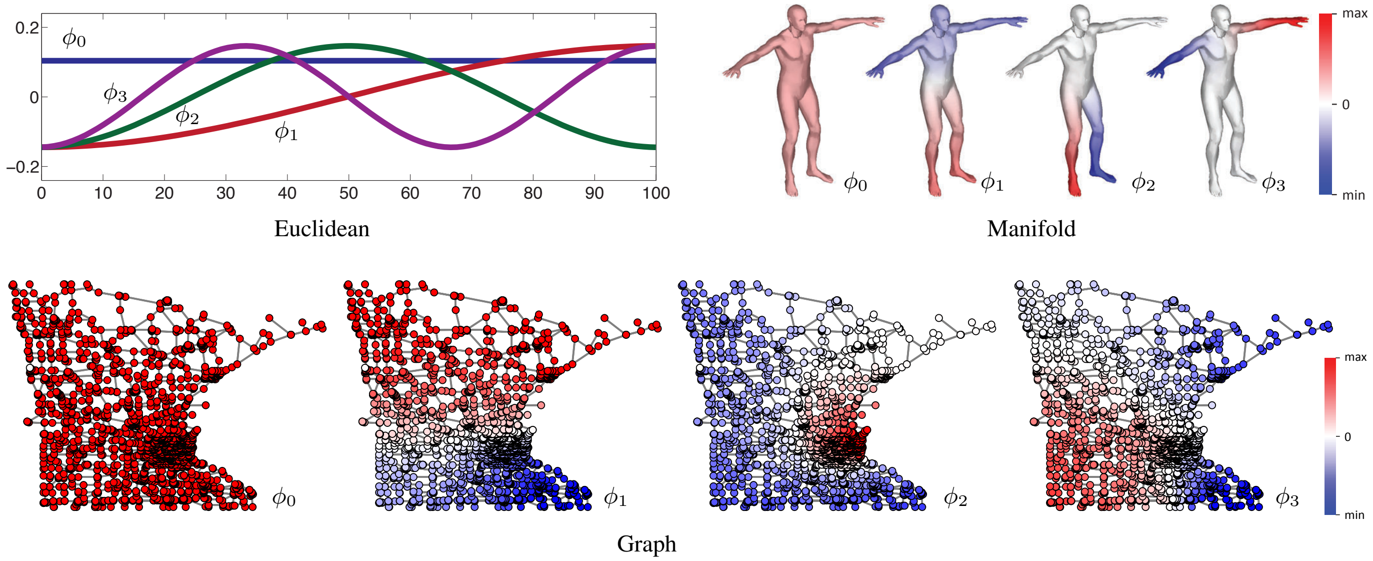
$$\min_{\mathbf{\Phi}_k \in \mathbb{R}^{n \times k}} \ \mathrm{trace}(\mathbf{\Phi}_k^\top \mathbf{\Delta} \mathbf{\Phi}_k) \quad \text{s.t.} \quad \mathbf{\Phi}_k^\top \mathbf{\Phi}_k = \mathbf{I}, \quad (29)$$

where $\mathbf{\Phi}_k = (\boldsymbol{\phi}_0, \ldots \boldsymbol{\phi}_{k-1})$. The solution of (29) is given by the first k eigenvectors of $\mathbf{\Delta}$ satisfying

$$\mathbf{\Delta}\mathbf{\Phi}_k = \mathbf{\Phi}_k \mathbf{\Lambda}_k, \quad (30)$$

where $\mathbf{\Lambda}_k = \mathrm{diag}(\lambda_0, \ldots, \lambda_{k-1})$ is the diagonal matrix of corresponding eigenvalues. The eigenvalues $0 = \lambda_0 \leq \lambda_1 \leq \ldots \lambda_{k-1}$ are non-negative due to positive-semidefiniteness of the Laplacian and can be interpreted as 'frequencies', where $\boldsymbol{\phi}_0 = \mathrm{const}$ with the corresponding eigenvalue $\lambda_0 = 0$ play the role of the DC.

The Laplacian eigendecomposition can be carried out in two ways. First, equation (30) can be rewritten as a generalized eigenproblem $(\mathbf{D} - \mathbf{W})\mathbf{\Phi}_k = \mathbf{A}\mathbf{\Phi}_k\mathbf{\Lambda}_k$, resulting in $\mathbf{A}$-orthogonal eigenvectors, $\mathbf{\Phi}_k^\top \mathbf{A} \mathbf{\Phi}_k = \mathbf{I}$. Alternatively, introducing a change of variables $\mathbf{\Psi}_k = \mathbf{A}^{1/2}\mathbf{\Phi}_k$, we can obtain a standard eigendecomposition problem $\mathbf{A}^{-1/2}(\mathbf{D} - \mathbf{W})\mathbf{A}^{-1/2}\mathbf{\Psi}_k = \mathbf{\Psi}_k\mathbf{\Lambda}_k$ with orthogonal eigenvectors $\mathbf{\Psi}_k^\top \mathbf{\Psi}_k = \mathbf{I}$. When $\mathbf{A} = \mathbf{D}$ is used, the matrix $\tilde{\mathbf{\Delta}} = \mathbf{A}^{-1/2}(\mathbf{D} - \mathbf{W})\mathbf{A}^{-1/2}$ is referred to as the *normalized symmetric Laplacian*.



[FIGS3] Example of the first four Laplacian eigenfunctions $\phi_0, \ldots, \phi_3$ on a Euclidean domain (1D line, top left) and non-Euclidean domains (human shape modeled as a 2D manifold, top right; and Minnesota road graph, bottom) domains. In the Euclidean case, the result is the standard Fourier basis comprising sinusoids of increasing frequency. In all cases, the eigenfunction ϕ_0 corresponding to zero eigenvalue is constant ('DC').

where the projection on the basis functions producing a discrete set of Fourier coefficients generalizes the *analysis* (forward transform) stage in classical signal processing, and summing up the basis functions with these coefficients is the *synthesis* (inverse transform) stage.

A centerpiece of classical Euclidean signal processing is the property of the Fourier transform diagonalizing the convolution operator, colloquially referred to as the *Convolution Theorem*, allowing to express the convolution $f \star g$ of two functions in the spectral domain as the element-wise product of their Fourier transforms,

$$\widehat{(f \star g)}(\omega) \quad = \quad \int_{-\infty}^{\infty} f(x)e^{-i\omega x}dx \int_{-\infty}^{\infty} g(x)e^{-i\omega x}dx.(33)$$

Unfortunately, in the non-Euclidean case we cannot even define the operation $x - x'$ on the manifold or graph, so the notion of convolution does not directly extend to this case. One possibility to generalize convolution to non-Euclidean domains is by using the convolution theorem as a definition,

$$(f \star g)(x) \;\; = \;\; \sum_{i \geq 0} \langle f, \phi_i \rangle_{L^2(\mathcal{X})} \langle g, \phi_i \rangle_{L^2(\mathcal{X})} \phi_i(x). \quad (34)$$

One of the key differences of such a construction from the classical convolution is the lack of shift-invariance. In terms of signal processing, it can be interpreted as a position-dependent filter. While parametrized by a fixed number of coefficients in the frequency domain, the spatial representation of the filter can vary dramatically at different points.

The discussion above also applies to graphs instead of manifolds, where one only has to replace the inner product in equations (32, 34) with the discrete one (20). All the sums over i would become finite, as the graph Laplacian $\boldsymbol{\Delta}$ has n eigenvectors. In matrix-vector notation, the generalized convolution $f \star g$ can be expressed as $\mathbf{G}\mathbf{f} = \boldsymbol{\Phi}\, \mathrm{diag}(\hat{\mathbf{g}}) \boldsymbol{\Phi}^\top \mathbf{f}$, where $\hat{\mathbf{g}} = (\hat{g}_1, \ldots, \hat{g}_n)$ is the spectral representation of the filter and $\boldsymbol{\Phi}$ denotes the Laplacian eigenvectors (30). The lack of shift invariance results in the absence of circulant (Toeplitz) structure in the matrix $\mathbf{G}$, which characterizes the Euclidean setting. Furthermore, it is easy to see that the convolution operation commutes with the Laplacian, $\mathbf{G}\boldsymbol{\Delta}\mathbf{f} = \boldsymbol{\Delta}\mathbf{G}\mathbf{f}$.

Uniqueness and stability: Finally, it is important to note that the Laplacian eigenfunctions are not uniquely defined. To start with, they are defined up to sign (i.e., $\Delta(\pm\phi) = \lambda(\pm\phi)$). Thus, even isometric domains might have different Laplacian eigenfunctions. Furthermore, if a Laplacian eigenvalue has multiplicity, then the associated eigenfunctions can be defined as orthonormal basis spanning the corresponding eigen-subspace (or said differently, the eigenfunctions are defined up to an orthogonal transformation in the eigen-subspace). A small perturbation of the domain can lead to very large changes in the Laplacian eigenvectors, especially those associated with high frequencies. At the same time, the definition of heat kernels (36) and diffusion distances (38) does not suffer from these ambiguities – for example, the sign ambiguity disappears as the eigenfunctions are squared. Heat kernels also appear to be robust to domain perturbations.

V. Frequency-domain learning methods

We have now finally got to our main goal, namely, constructing a generalization of the CNN architecture on non-Euclidean domains. We will start with the assumption that the domain on which we are working is fixed, and for the rest of this section will use the problem of classification of a signal on a graph as the prototypical application.

Spectral CNN: We have seen that convolutions are linear operators that commute with the Laplacian operator. Therefore, given a weighted graph, a first route to generalize a convolutional architecture is by first restricting our interest on linear operators that commute with the graph Laplacian [71]. This property, in turn, implies operating on the spectrum of the graph weights, given by the eigenvectors of the graph Laplacian.

Similarly to the convolutional layer (6) of a classical Euclidean CNN, we define a *spectral convolutional layer* as

$$\mathbf{g}_l = \xi \left(\sum_{l'=1}^{q} \boldsymbol{\Phi}_k \mathbf{W}_{l,l'} \boldsymbol{\Phi}_k^\top \mathbf{f}_{l'} \right), \quad (39)$$

where the $n \times p$ and $n \times q$ matrices $\mathbf{F} = (\mathbf{f}_1, \ldots, \mathbf{f}_p)$ and $\mathbf{G} = (\mathbf{g}_1, \ldots, \mathbf{g}_q)$ represent the p- and q-dimensional input and output signals on the vertices of the graph, respectively (we use $n = |\mathcal{V}|$ to denote the number of vertices in the graph), $\mathbf{W}_{l,l'}$ is a $k \times k$ diagonal matrix of spectral multipliers representing a filter in the frequency domain, and ξ is a nonlinearity applied on the vertex-wise function values. Using only the first k eigenvectors in (39) sets a cutoff frequency which depends on the intrinsic regularity of the graph and also the sample size. Typically, $k \ll n$, since only the first Laplacian eigenvectors describing the smooth structure of the graph are useful in practice.

If the graph has an underlying group invariance, such a construction can discover it. In particular, standard CNNs can be redefined from the spectral domain (see insert IN5). However, in many cases the graph does not have a group structure, or the group structure does not commute with the Laplacian, and so we cannot think of each filter as passing a template across $\mathcal{V}$ and recording the correlation of the template with that location.

We should stress that a fundamental limitation of the spectral construction is its limitation to a single domain. The reason is that spectral filter coefficients (39) are *basis dependent*. It implies that if we learn a filter w.r.t. basis $\boldsymbol{\Phi}_k$ on one domain, and then try to apply it on another domain with another basis $\boldsymbol{\Psi}_k$, the result could be very different (see Figure V). It is possible to construct compatible orthogonal bases across different domains resorting to a joint diagonalization procedure [72], [73]. However, such a construction requires the knowledge of some correspondence between the domains. In applications such as social network analysis, for example, where dealing with two time instances of a social graph in which new vertices and edges have been added, such a correspondence can be easily computed and is therefore a reasonable assumption. Conversely, in computer graphics applications, finding correspondence between shapes is in itself a very hard problem, so assuming known correspondence between the domains is a rather unreasonable assumption.

Assuming that $k = O(n)$ eigenvectors of the Laplacian are kept, a convolutional layer (39) requires $pqk = O(n)$ parameters to train. We will see next how the global and local regularity of the graph can be combined to produce layers with constant number of parameters, i.e., such that the number of learnable parameters per layer does not depend upon the size of the input.

The non-Euclidean analogy of pooling is *graph coarsening*, in which only a fraction $\alpha < 1$ of the graph vertices is retained. The eigenvectors of graph Laplacians at two different resolutions are related by the following multigrid property: Let $\boldsymbol{\Phi}$, $\bar{\boldsymbol{\Phi}}$ denote the $n \times n$ and $\alpha n \times \alpha n$ matrices of Laplacian eigenvectors of the original and the coarsened graph, respectively. Then,

$$\bar{\boldsymbol{\Phi}} \approx \mathbf{P} \boldsymbol{\Phi} \begin{pmatrix} \mathbf{I}_{\alpha n} \\ \mathbf{0} \end{pmatrix}, \quad (40)$$

where $\mathbf{P}$ is a $\alpha n \times n$ binary matrix whose ith row encodes the position of the ith vertex of the coarse graph on the original graph. It follows that strided convolutions can be

[IN4] Heat diffusion on non-Euclidean domains: An important application of spectral analysis, and historically, the main motivation for its development by Fourier, is the solution of partial differential equations (PDEs). In particular, we are interested in heat propagation on non-Euclidean domains. This process is governed by the *heat diffusion equation*, which in the simplest setting of homogeneous and isotropic diffusion has the form

$$\begin{cases} f_t(x,t) = -c\Delta f(x,t) \\ f(x,0) = f_0(x) \quad \text{(Initial condition)} \end{cases} \tag{35}$$

with additional boundary conditions if the domain has a boundary. $f(x,t)$ represents the temperature at point x at time t. Equation (35) encodes the *Newton's law of cooling*, according to which the rate of temperature change of a body (lhs) is proportional to the difference between its own temperature and that of the surrounding (rhs). The proportion coefficient c is referred to as the *thermal diffusivity constant*. The solution of (35) is given by applying the *heat operator* $H^t = e^{-t\Delta}$ to the initial condition and can be expressed in the spectral domain as

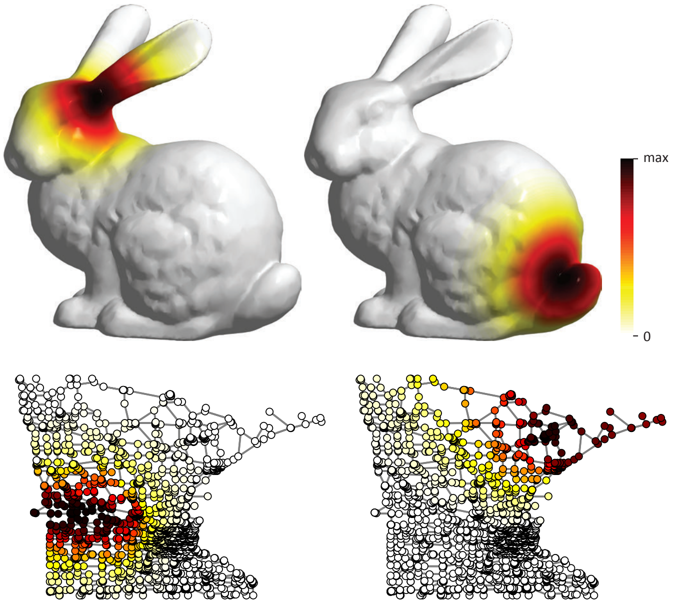
$$\begin{aligned} f(x,t) &= e^{-t\Delta} f_0(x) = \sum_{i\geq 0} \langle f_0, \phi_i \rangle_{L^2(\mathcal{X})} e^{-t\lambda_i} \phi_i(x) \quad (36) \\ &= \int_{\mathcal{X}} f_0(x') \underbrace{\sum_{i\geq 0} e^{-t\lambda_i} \phi_i(x)\phi_i(x')}_{h_t(x,x')} dx'. \end{aligned}$$

$h_t(x,x')$ is known as the *heat kernel* and represents the solution of the heat equation with an initial condition $f_0(x) = \delta_{x'}(x)$, or, in signal processing terms, an 'impulse response'. In physical terms, $h_t(x,x')$ describes how much heat flows from a point x to point x' in time t. In the Euclidean case, the heat kernel is *shift-invariant*, $h_t(x,x') = h_t(x-x')$, allowing to interpret the integral in (36) as a convolution $f(x,t) = (f_0 \star h_t)(x)$. In the spectral domain, convolution with the heat kernel amounts to low-pass filtering (with frequency response $e^{-t\lambda}$). Larger values of diffusion time t result in lower effective cutoff frequency and thus smoother solutions in space (corresponding to the intuition that longer diffusion smoothes more the initial heat distribution).

The 'cross-talk' between two heat kernels positioned at points x and x' allows to measure an intrinsic distance

$$\begin{aligned} d_t^2(x,x') &= \int_{\mathcal{X}} (h_t(x,y) - h_t(x',y))^2 dy \qquad (37) \\ &= \sum_{i\geq 0} e^{-2t\lambda_i} (\phi_i(x) - \phi_i(x'))^2 \qquad (38) \end{aligned}$$

referred to as the *diffusion distance* [38]. Note that interpreting (37) and (38) as spatial- and frequency-domain norms $\|\cdot\|_{L^2(\mathcal{X})}$ and $\|\cdot\|_{\ell^2}$, respectively, their equivalence is the consequence of the *Parseval identity*. Unlike *geodesic distance* that measures the length of the shortest path on the manifold or graph, the diffusion distance has an effect of averaging over different paths. It is thus more robust to perturbations of the domain, for example, introduction or removal of edges in a graph, or 'cuts' on a manifold.



[FIGS4] Examples of heat kernels on non-Euclidean domains (manifold, top; and graph, bottom). Observe how moving the heat kernel to a different location changes its shape, which is an indication of the lack of shift-invariance.

generalized using the spectral construction by keeping only the low-frequency components of the spectrum. This property also allows us to interpret (via interpolation) the local filters at deeper layers in the spatial construction to be low frequency. However, since in (39) the non-linearity is applied in the spatial domain, in practice one has to recompute the graph Laplacian eigenvectors at each resolution and apply them directly after each pooling step.

The spectral construction (39) assigns a degree of freedom for each eigenvector of the graph Laplacian. In most graphs, individual high-frequency eigenvectors become highly unstable. However, similarly as the wavelet construction in Euclidean domains, by appropriately grouping high frequency eigenvectors in each octave one can recover meaningful and stable information. As we shall see next, this principle also entails better learning complexity.

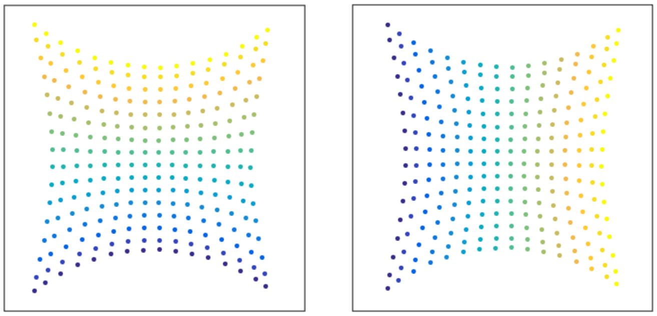
Learning with smooth spectral multipliers: In order to achieve a good generalization, it is important to adapt the learning complexity to reduce the number of free parameters of the model. On Euclidean domains, this is achieved by learning convolutional kernels with small spatial support, which enables the model to learn a number of parameters independent of the input size. In order to achieve a similar learning complexity in the spectral domain, it is thus necessary to restrict the class of

[IN5] Rediscovering standard CNNs using correlation kernels: In situations where the graph is constructed from the data, a straightforward choice of the edge weights (11) of the graph is the covariance of the data. Let $\mathbf{F}$ denote the input data distribution and

$$\mathbf{\Sigma} = \mathbb{E}(\mathbf{F} - \mathbb{E}\mathbf{F})(\mathbf{F} - \mathbb{E}\mathbf{F})^\top \quad (41)$$

the data covariance matrix. If each point has the same variance $\sigma_{ii} = \sigma^2$, then diagonal operators on the Laplacian simply scale the principal components of $\mathbf{F}$.

In natural images, since their distribution is approximately stationary, the covariance matrix has a circulant structure $\sigma_{ij} \approx \sigma_{i-j}$ and is thus diagonalized by the standard Discrete Cosine Transform (DCT) basis. It follows that the principal components of $\mathbf{F}$ roughly correspond to the DCT basis vectors organized by frequency. Moreover, natural images exhibit a power spectrum $\mathbb{E}(|\widehat{f}(\omega)|^2) \sim |\omega|^{-2}$, since nearby pixels are more correlated than far away pixels [18]. It results that principal components of the covariance are essentially ordered from low to high frequencies, which is consistent with the standard group structure of the Fourier basis. When applied to natural images represented as graphs with weights defined by the covariance, the spectral CNN construction recovers the standard CNN, without any prior knowledge [74]. Indeed, the linear operators $\mathbf{\Phi}\mathbf{W}_{l,l'}\mathbf{\Phi}^T$ in (39) are by the previous argument diagonal in the Fourier basis, hence translation invariant, hence classical convolutions. Furthermore, Section VII explains how spatial subsampling can also be obtained via dropping the last part of the spectrum of the Laplacian, leading to pooling, and ultimately to standard CNNs.



[FIG5a] Two-dimensional embedding of pixels in 16 × 16 image patches using a Euclidean RBF kernel. The RBF kernel is constructed as in (11), by using the covariance σ_{ij} as Euclidean distance between two features. The pixels are embedded in a 2D space using the first two eigenvectors of the resulting graph Laplacian. The colors in the left and right figure represent the horizontal and vertical coordinates of the pixels, respectively. The spatial arrangement of pixels is roughly recovered from correlation measurements.

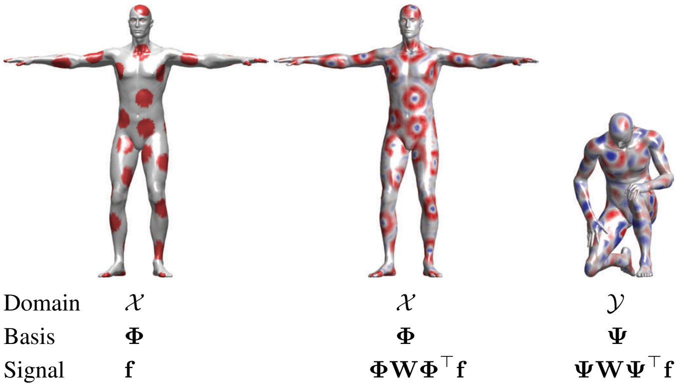


Fig. 2. A toy example illustrating the difficulty of generalizing spectral filtering across non-Euclidean domains. Left: a function defined on a manifold (function values are represented by color); middle: result of the application of an edge-detection filter in the frequency domain; right: the same filter applied on the same function but on a different (nearly-isometric) domain produces a completely different result. The reason for this behavior is that the Fourier basis is domain-dependent, and the filter coefficients learnt on one domain cannot be applied to another one in a straightforward manner.

spectral multipliers to those corresponding to localized filters.

For that purpose, we have to express spatial localization of filters in the frequency domain. In the Euclidean case, smoothness in the frequency domain corresponds to spatial decay, since

$$\int |x|^{2k}|f(x)|^2 dx = \int \left| \frac{\partial^k \hat{f}(\omega)}{\partial \omega^k} \right|^2 d\omega \ , \quad (42)$$

from the Parseval Identity. This suggests that, in order to learn a layer in which features will be not only shared across locations but also well localized in the original domain, one can learn spectral multipliers which are smooth. Smoothness can be prescribed by learning only a subsampled set of frequency multipliers and using an interpolation kernel to obtain the rest, such as cubic splines.

However, the notion of smoothness also requires some geometry in the spectral domain. In the Euclidean setting, such a geometry naturally arises from the notion of frequency; for example, in the plane, the similarity between two Fourier atoms $e^{i\boldsymbol{\omega}^\top \mathbf{x}}$ and $e^{i\boldsymbol{\omega}'^\top \mathbf{x}}$ can be quantified by the distance $\|\boldsymbol{\omega} - \boldsymbol{\omega}'\|$, where $\mathbf{x}$ denotes the two-dimensional planar coordinates, and $\boldsymbol{\omega}$ is the two-dimensional frequency vector. On graphs, such a relation can be defined by means of a dual graph with weights $\tilde{w}_{ij}$ encoding the similarity between two eigenvectors ϕ_i and ϕ_j.

A particularly simple choice consists in choosing a one-dimensional arrangement, obtained by ordering the eigenvectors according to their eigenvalues. [5] In this setting, the spectral multipliers are parametrized as

$$\mathrm{diag}(\mathbf{W}_{l,l'}) = \mathbf{B}\boldsymbol{\alpha}_{l,l'}, \tag{43}$$

where $\mathbf{B} = (b_{ij}) = (\beta_j(\lambda_i))$ is a $k \times q$ fixed interpolation kernel (e.g., $\beta_j(\lambda)$ can be cubic splines) and $\boldsymbol{\alpha}$ is a vector of q interpolation coefficients. In order to obtain filters with constant spatial support (i.e., independent of the input size n), one should choose a sampling step $\gamma \sim n$ in the spectral domain, which results in a constant number $q \sim n\gamma^{-1} = \mathcal{O}(1)$ of coefficients $\boldsymbol{\alpha}_{l,l'}$ per filter.

Even with such a parametrization of the filters, the spectral CNN (39) entails a high computational complexity of performing forward and backward passes, since they require an expensive matrix multiplication step by $\boldsymbol{\Phi}_k$ and $\boldsymbol{\Phi}_k^\top$. While on Euclidean domains such a multiplication can be efficiently carried in $\mathcal{O}(n \log n)$ operations resorting to FFT-type algorithms, for general graphs such algorithms do not exist and the complexity is $\mathcal{O}(n^2)$. We will see in the following how to alleviate this cost by avoiding explicit computation of the Laplacian eigenvectors.

Spectrum-free computation: Another convenient parametric way of representing the convolution filters is via an explicit polynomial expansion [53].

$$g_{\boldsymbol{\alpha}}(\lambda) = \sum_{j=0}^{r-1} \alpha_j \lambda^j, \tag{44}$$

where $\boldsymbol{\alpha}$ is the r-dimensional vector of polynomial coefficients. Applying the polynomial to the Laplacian matrix is expressed as an operation on its eigenvalues,

$$g_{\boldsymbol{\alpha}}(\boldsymbol{\Delta}) \;=\; \boldsymbol{\Phi} g_{\boldsymbol{\alpha}}(\boldsymbol{\Lambda})\boldsymbol{\Phi}^\top, \tag{45}$$

where $g_{\boldsymbol{\alpha}}(\boldsymbol{\Lambda}) = \mathrm{diag}(g_{\boldsymbol{\alpha}}(\lambda_1), \ldots, g_{\boldsymbol{\alpha}}(\lambda_n))$, resulting in filter matrices $\mathbf{W}_{l,l'} = g_{\boldsymbol{\alpha}_{l,l'}}(\boldsymbol{\Lambda})$ whose entries have an explicit form in terms of the eigenvalues.

An important property of this representation is that it automatically yields localized filters, for the following reason. Since the Laplacian is a local operator (working on 1-hop neighborhoods), the action of its jth power action is constrained to j-hops. Since the filter is a linear combination of powers of the Laplacian, overall (44) behaves like a diffusion operator limited to r-hops.

Besides their ease of interpretation, polynomial filters can also be applied very efficiently if one chooses $g_{\boldsymbol{\alpha}}(\lambda)$ as a polynomial that can be computed recursively. For instance, the Chebyshev polynomial $T_j(\lambda)$ of order j may be generated by the recurrence relation

$$\begin{aligned} T_j(\lambda) &= 2\lambda T_{j-1}(\lambda) - T_{j-2}(\lambda); \qquad (46)\\ T_0(\lambda) &= 1;\\ T_1(\lambda) &= \lambda. \end{aligned}$$

A filter can thus be parameterized uniquely via an expansion of order $r-1$ such that

$$g_{\boldsymbol{\alpha}}(\boldsymbol{\Delta}) = \sum_{j=0}^{r-1} \alpha_j \boldsymbol{\Phi} T_j(\tilde{\boldsymbol{\Lambda}})\boldsymbol{\Phi}^\top, \tag{47}$$

where $\tilde{\boldsymbol{\Lambda}} = 2\lambda_n^{-1}\boldsymbol{\Lambda} - \mathbf{I}$ is a rescaling mapping the Laplacian eigenvalues from the interval $[0, \lambda_n]$ to $[-1, 1]$ (necessary since the Chebyshev polynomials form an orthonormal basis in $[-1, 1]$).

Filtering a signal $\mathbf{f}$ can now be written as

$$g_{\boldsymbol{\alpha}}(\boldsymbol{\Delta})\mathbf{f} = \sum_{j=0}^{r-1} \alpha_j T_j(\tilde{\boldsymbol{\Delta}})\mathbf{f}, \tag{48}$$

where $\tilde{\boldsymbol{\Delta}} = 2\lambda_n^{-1}\boldsymbol{\Delta} - \mathbf{I}$ is the rescaled Laplacian. Denoting $\bar{\mathbf{f}}^{(k)} = T_k(\tilde{\boldsymbol{\Delta}})\mathbf{f}$, we can use the recurrence relation (46) to compute $\bar{\mathbf{f}}^{(k)} = 2\tilde{\boldsymbol{\Delta}}\bar{\mathbf{f}}^{(k-1)} - \bar{\mathbf{f}}^{(k-2)}$ with $\bar{\mathbf{f}}^{(0)} = \mathbf{f}$ and $\bar{\mathbf{f}}^{(1)} = \tilde{\boldsymbol{\Delta}}\mathbf{f}$. The computational complexity of this procedure is therefore $\mathcal{O}(rn)$ operations and does not require an explicit computation of the Laplacian eigenvectors.

In [75], this construction was simplified by assuming $r = 2$ and $\lambda_n \approx 2$, resulting in filters of the form

$$\begin{aligned} g_{\boldsymbol{\alpha}}(\mathbf{f}) &= \alpha_0\mathbf{f} + \alpha_1(\boldsymbol{\Delta} - \mathbf{I})\mathbf{f}\\ &= \alpha_0\mathbf{f} - \alpha_1\mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}\mathbf{f}. \qquad (49) \end{aligned}$$

Further constraining $\alpha = \alpha_0 = -\alpha_1$, one obtains filters represented by a single parameter,

$$g_{\alpha}(\mathbf{f}) \;=\; \alpha(\mathbf{I} + \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2})\mathbf{f}. \tag{50}$$

Since the eigenvalues of $\mathbf{I} + \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}$ are now in the range $[0, 2]$, repeated application of such a filter can result in numerical instability. This can be remedied by a renormalization

$$g_{\alpha}(\mathbf{f}) \;=\; \alpha\tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{W}}\tilde{\mathbf{D}}^{-1/2}\mathbf{f}, \tag{51}$$

where $\tilde{\mathbf{W}} = \mathbf{W} + \mathbf{I}$ and $\tilde{\mathbf{D}} = \mathrm{diag}(\sum_{j\neq i} \tilde{w}_{ij})$.

VI. Spatial-domain learning methods

We will now consider the second sub-class of non-Euclidean learning problems, where we are given multiple domains. A prototypical application the reader should have in mind throughout this section is the problem of finding correspondence between shapes, modeled as manifolds. As we have seen, defining convolution in the frequency domain has an inherent drawback of inability to adapt the model across different domains. We will therefore need to resort to an alternative generalization of the convolution in the spatial domain. Furthermore, note that in the setting of multiple domains, there is no immediate way to define a meaningful spatial pooling operation, as the number of points on different domains can vary, and their order be arbitrary.

[5] In the mentioned 2D example, this would correspond to ordering the Fourier basis function according to the sum of the corresponding frequencies $\omega_1 + \omega_2$. Although numerical results on simple low-dimensional graphs show that the 1D arrangement given by the spectrum of the Laplacian is efficient at creating spatially localized filters [71], an open fundamental question is how to define a dual graph on the eigenvectors of the Laplacian in which smoothness (obtained by applying the diffusion operator) corresponds to localization in the original graph.

Charting-based methods: On a Euclidean domain, due to shift-invariance the convolution can be thought of as passing a template at each point of the domain and recording the correlation of the template with the function at that point. Thinking of image filtering, this amounts to extracting a (typically square) patch of pixels, multiplying it element-wise with a template and summing up the results, then moving to the next position in a sliding window manner. Shift-invariance implies that the very operation of extracting the patch at each position is always the same.

One of the major problems in applying the same paradigm to non-Euclidean domains is the lack of shift-invariance, implying that the 'patch operator' extracting a local 'patch' would be position-dependent. Furthermore, the typical lack of meaningful global parametrization for a graph or manifold forces to represent the patch in some local intrinsic system of coordinates. Such a mapping can be represented by defining a set of weighting functions $u_1(x,\cdot),\dots,u_J(x,\cdot)$ localized to positions near x (see examples in Figure VI). Extracting a patch amounts to averaging the function f at each point by these weights,

$$D_j(x)f \quad = \quad \int_{\mathcal{X}} f(x')u_j(x,x')dx', \quad j=1,\dots,J, \tag{52}$$

providing for a spatial definition of an intrinsic equivalent of convolution

$$(f\star g)(x) \quad = \quad \sum_j g_j D_j(x)f, \tag{53}$$

where g denotes the template coefficients applied on the patch extracted at each point. Overall, (52–53) act as a kind of non-linear filtering of f, and the patch operator D is specified by defining the weighting functions u. Several recently proposed frameworks for non-Euclidean CNNs, which we briefly overview here, essentially amount to different choice of these weights.

Diffusion CNN: The simplest local charting on a non-Euclidean domain $\mathcal{X}$ is a one-dimensional coordinate measuring the intrinsic (e.g. geodesic or diffusion) distance $d(x,\cdot)$ [54]. The weighting functions in this case, for example chosen as Gaussians

$$w_i(x,x') \quad = \quad e^{-(d(x,x')-\rho_i)^2/2\sigma^2} \tag{54}$$

have the shape of rings of width σ at distances $\rho_1,\dots,\rho_J$ (Figure VI, left).

Geodesic CNN: Since manifolds naturally come with a low-dimensional tangent space associated with each point, it is natural to work in a local system of coordinates in the tangent space. In particular, on two-dimensional manifolds one can create a polar system of coordinates around x where the radial coordinate is given by some intrinsic distance $\rho(x')=d(x,x')$, and the angular coordinate $\theta(x)$ is obtained by ray shooting from a point at equi-spaced angles. The weighting functions in this case can be obtained as a product of Gaussians

$$w_{ij}(x,x') \quad = \quad e^{-(\rho(x')-\rho_i)^2/2\sigma_\rho^2}\, e^{-(\theta(x')-\theta_j)^2/2\sigma_\theta^2}, \tag{55}$$

where $i=1,\dots,J$ and $j=1,\dots,J'$ denote the indices of the radial and angular bins, respectively. The resulting JJ' weights are bins of width $\sigma_\rho\times\sigma_\theta$ in the polar coordinates (Figure VI, right). The diffusion CNN can be considered as a particular setting where only one angular bin and $\sigma_\theta=\infty$ are used.

Anisotropic CNN: We have already seen the non-Euclidean heat equation (35), whose heat kernel $h_t(x,\cdot)$ produces localized blob-like weights around the point x (see FIGS4). Varying the diffusion time t controls the spread of the kernel. However, such kernels are *isotropic*, meaning that the heat flows equally fast in all the directions. A more general *anisotropic diffusion* equation on a manifold

$$f_t(x,t) = -\mathrm{div}(\mathbf{A}(x)\nabla f(x,t)), \tag{56}$$

involves the *thermal conductivity tensor* $\mathbf{A}(x)$ (in case of two-dimensional manifolds, a 2×2 matrix applied to the intrinsic gradient in the tangent plane at each point), allowing modeling heat flow that is position- and direction-dependent [76]. A particular choice of the heat conductivity tensor proposed in [77] is

$$\mathbf{A}_{\alpha\theta}(x) = \mathbf{R}_\theta(x)\begin{pmatrix}\alpha & \\ & 1\end{pmatrix}\mathbf{R}_\theta^\top(x), \tag{57}$$

where the 2×2 matrix $\mathbf{R}_\theta(x)$ performs rotation of θ w.r.t. to some reference (e.g. the maximum curvature) direction and $\alpha>0$ is a parameter controlling the degree of anisotropy ($\alpha=1$ corresponds to the classical isotropic case). The heat kernel of such anisotropic diffusion equation is given by the spectral expansion

$$h_{\alpha\theta t}(x,x') = \sum_{i\geq 0} e^{-t\lambda_{\alpha\theta i}}\phi_{\alpha\theta i}(x)\phi_{\alpha\theta i}(x'), \tag{58}$$

where $\phi_{\alpha\theta 0}(x),\phi_{\alpha\theta 1}(x),\dots$ are the eigenfunctions and $\lambda_{\alpha\theta 0},\lambda_{\alpha\theta 1},\dots$ the corresponding eigenvalues of the *anisotropic Laplacian*

$$\Delta_{\alpha\theta}f(x) = -\mathrm{div}(\mathbf{A}_{\alpha\theta}(x)\nabla f(x)). \tag{59}$$

The discretization of the anisotropic Laplacian is a modification of the cotangent formula (14) on meshes or graph Laplacian (11) on point clouds.

The anisotropic heat kernels $h_{\alpha\theta t}(x,\cdot)$ look like elongated rotated blobs (see Figure VI, center), where the parameters α,θ and t control the elongation, orientation, and scale, respectively. Using such kernels as weighting functions u in the construction of the patch operator (52), it is possible to obtain a charting similar to the geodesic patches (roughly, θ plays the role of the angular coordinate and t of the radial one).

A limitation of the spatial generalization of CNNs based on patch operators is the assumption of some local low-dimensional structure in which a meaningful system of coordinates can be defined. While very natural on manifolds (where the tangent space is such a low-dimensional space), such a definition is significantly more challenging on graphs. In particular, defining anisotropic diffusion on general graphs seems to be an intriguing but hard problem.

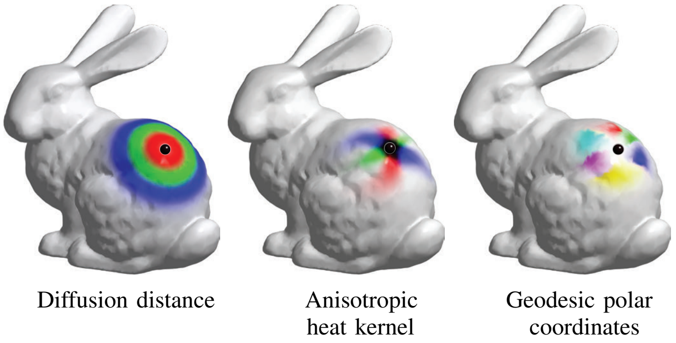


Fig. 3. Example of intrinsic weighting functions used to construct a patch operator at the point marked in black (different colors represent different weighting functions). Diffusion distance (left) allows to map neighbor points according to their distance from the reference point, thus defining a one-dimensional system of local intrinsic coordinates. Anisotropic heat kernels (middle) of different scale and orientations and geodesic polar weights (right) are two-dimensional systems of coordinates.

Pooling: As we already mentioned, unlike the subsetting of learning on a single domain, there is no immediate meaningful interpretation of a spatial pooling in the case of multiple domains. It is however possible to pool point-wise features produced by a network by aggregating all the local information into a single vector. One possibility for such a pooling is computing the statistics of the point-wise features, e.g. the covariance matrix [26]. Note that after such a pooling all the spatial information is lost.

Graph Neural Network: A more general spatial construction on graphs has been proposed [78], [79] and has been simplified in [80], [81]. Given a p-dimensional input signal on the vertices of the graph, represented by the $n \times p$ matrix $\mathbf{F}$, the application of the Laplacian is an intrinsic operation that can be broken down into $\mathbf{WF}$ and $\mathbf{DF}$. The *Graph Neural Network* (GNN) considers a generic point-wise nonlinearity $\eta_{\boldsymbol{\theta}} : \mathbb{R}^p \times \mathbb{R}^p \to \mathbb{R}^q$, parametrized by trainable parameters $\boldsymbol{\theta}$, that is applied to all nodes of the graph:

$$\mathbf{g}_i = \eta_{\boldsymbol{\theta}}\left((\mathbf{Df})_i, (\mathbf{Wf})_i\right) \ . \tag{60}$$

In particular, choosing $\eta(\mathbf{a}, \mathbf{b}) = \mathbf{a} - \mathbf{b}$ one recovers the Laplacian operator $\boldsymbol{\Delta}\mathbf{f}$. More general, nonlinear choices for η yield trainable, task-specific diffusion operators. Similarly as with a CNN architecture, one can stack the resulting GNN layers $\mathbf{g} = C_{\boldsymbol{\theta}}(\mathbf{f})$ and interleave them with graph pooling operators. Some of the previous constructions can be cast as special cases of (60). In particular, Chebyshev polynomials $T_r(\boldsymbol{\Delta})$ can be obtained with r layers of (60).

Because the communication at each layer is local to a vertex neighborhood, one may worry that it would take many layers to get information from one part of the graph to another, requiring multiple hops. However, note that the graphs at each layer of the network need not be the same. Thus we can replace the original neighborhood structure with a one's favorite multiscale coarsening of the input graph, and operate on that to obtain the same flow of information as with the convolutional nets above (or rather more like a "locally connected network" [82]). This also allows producing a single output for the whole graph (for "translation-invariant" tasks), rather than an undifferentiated output per vertex, by connecting each to a special output node.

The GNN model can be further generalized to replicate other operators on graphs. For instance, the point-wise nonlinearity η can depend on the vertex type, allowing extremely rich architectures. Also, one can allow η to use not only $\mathbf{Df}$ and $\mathbf{Wf}$ at each node, but also $\mathbf{W}^s\mathbf{f}$ for several diffusion scales $s > 1$, giving the GNN the ability to learn algorithms such as the power method, therefore accessing spectral properties of the graph.

VII. Spatio-frequency learning methods

The third alternative for constructing convolution-like operations of non-Euclidean domains is jointly in spatial-frequency domain.

Windowed Fourier methods: One of the notable drawbacks of classical Fourier analysis is its lack of spatial localization. By virtue of the *Uncertainty Principle*, one of the fundamental properties of Fourier transforms, spatial localization comes at the expense of frequency localization, and vice versa. In classical signal processing, this problem is remedied by localizing frequency analysis in a window $g(x)$, leading to the definition of the *Windowed Fourier Transform* (WFT, also known as *short-time Fourier transform* or *spectrogram* in 1D signal processing),

$$\begin{aligned} (Sf)(x,\omega) &= \int_{-\infty}^{\infty} f(x')\, \underbrace{g(x'-x)e^{-i\omega x'}}_{g_{x,\omega}(x')} dx' && (61)\\ &= \langle f, g_{x,\omega}\rangle_{L^2(\mathbb{R})}. && (62)\end{aligned}$$

The WFT is a function of two variables: spatial location of the window x and the modulation frequency ω. The choice of the window function g allows to control the tradeoff between spatial and frequency localization (wider windows result in better frequency resolution). Note that WFT can be interpreted as inner products (62) of the function f with translated and modulated windows $g_{x,\omega}$, referred to as the WFT *atoms*.

The generalization of such a construction to non-Euclidean domains requires the definition of translation and modulation operators [83]. While modulation simply amounts to multiplication by a Laplacian eigenfunction, translation is not well-defined due to the lack of shift-invariance. It is possible to resort again to the spectral definition of a convolution-like operation (34), defining translation as convolution with a delta-function,

$$\begin{aligned} (g \star \delta_{x'})(x) &= \sum_{i \geq 0} \langle g, \phi_i\rangle_{L^2(\mathcal{X})} \langle \delta_{x'}, \phi_i\rangle_{L^2(\mathcal{X})} \phi_i(x) \\ &= \sum_{i\geq 0} \hat{g}_i \phi_i(x')\phi_i(x). && (63)\end{aligned}$$

The translated and modulated atoms can be expressed as

$$g_{x',j}(x) = \phi_j(x') \sum_{i\geq 0} \hat{g}_i \phi_i(x)\phi_i(x'), \tag{64}$$

where the window is specified in the spectral domain by its Fourier coefficients $\hat{g}_i$; the WFT on non-Euclidean domains thus takes the form

$$(Sf)(x',j) = \langle f, g_{x',j} \rangle_{L^2(\mathcal{X})} = \sum_{i\geq 0} \hat{g}_i \phi_i(x') \langle f, \phi_i \phi_j \rangle_{L^2(\mathcal{X})}. \quad (65)$$

Due to the intrinsic nature of all the quantities involved in its definition, the WFT is also intrinsic.

The WFT allows expressing a function around a point in the spectral domain, and thus can be regarded as a patch operator $D_j(x)f = (Sf)(x,j)$ of the form (52) and used in an intrinsic convolution-like construction (53). An additional degree of freedom is the definition of the window, which can also be learned [27].

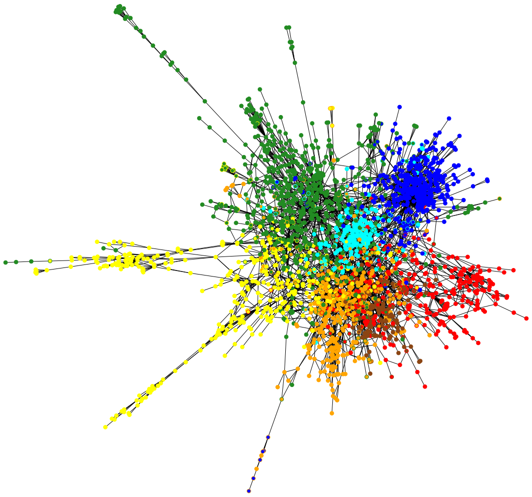
Wavelet methods: Replacing the notion of frequency in time-frequency representations by that of scale leads to wavelet decompositions. Wavelets have been extensively studied in general graph domains [84]. Their objective is to define stable linear decompositions with atoms well localized both in space and frequency that can efficiently approximate signals with isolated singularities. Similarly to the Euclidean setting, wavelet families can be constructed either from its spectral constraints or from its spatial constraints.

The simplest of such families are Haar wavelets. Several bottom-up wavelet constructions on graphs were studied in [85] and [86]. In [87], the authors developed an unsupervised method that learns wavelet decompositions on graphs by optimizing a sparse reconstruction objective. In [88], ensembles of Haar wavelet decompositions are used to define deep wavelet scattering transforms on general domains, obtaining excellent numerical performance. Learning amounts to finding optimal pairings of nodes at each scale, which can be efficiently solved in polynomial time.

VIII. Applications

Network analysis: One of the classical examples used in many works on network analysis are citation networks. Citation network is a graph where vertices represent papers and there is a directed edge (i,j) if paper i cites paper j. Typically, vertex-wise features representing the content of the paper (e.g. histogram of frequent terms in the paper) are available. A prototypical classification application is to attribute each paper to a field. Traditional approaches work vertex-wise, performing classification of each vertex's feature vector individually. More recently, it was shown that classification can be considerably improved using information from neighbor vertices, e.g. with a CNN on graphs [53], [75]. Insert IN4 shows an example of application of Spectral CNN on a citation network.

[IN4] Citation network analysis example: The CORA citation network [89] is a graph containing 2708 vertices representing papers and 5429 edges representing citations. Each paper is described by a 1433-dimensional bag-of-words feature vector and belongs to seven classes. For simplicity, the network is treated as an undirected graph. Applying the spectral CNN with two spectral convolutional layers parametrized according to (51), the authors of [75] obtain classification accuracy of 81.5% (compared to 75.7% previous best result).



[FIGS4a] Classifying research papers in the CORA dataset with Spectral CNN. Shown is the citation graph, where each node is a paper, and an edge represents a citation. Vertex fill color represents the predicted label; vertex outline color represents the groundtruth label (ideally, the two colors should coincide).

Computer vision and graphics: The computer vision community has recently shown an increasing interest in working with 3D geometric data, mainly due to the emergence of affordable range sensing technology such as Microsoft Kinect or Intel RealSense. Many machine learning techniques successfully working on images were tried "as is" on 3D geometric data, represented for this purpose in some way "digestible" by standard frameworks, e.g. as range images [90], [91] or rasterized volumes [92], [93]. The main drawback of such approaches is their treatment of geometric data as Euclidean structures. First, for complex 3D objects, Euclidean representations such as depth images or voxels may lose significant parts of the object or its fine details, or even break its topological structure. Second, Euclidean representations are not intrinsic, and vary when changing pose or deforming the object. Achieving invariance to shape deformations, a common requirement in many vision applications, demands very complex models and huge training sets due to the large number of degrees of freedom involved in describing non-rigid deformations.

In the domain of computer graphics, on the other hand, working intrinsically with geometric shapes is a standard practice. In this field, 3D shapes are typically modeled as Riemannian manifolds and are discretized as meshes. Numerous studies (see, e.g. [94], [95], [96], [97], [3]) have been devoted to designing local and global features e.g. for establishing similarity or correspondence between deformable shapes with guaranteed invariance properties. Two well-studied classes of deformations are *isometries* (metric-preserving transformations, consequently also preserving local areas and angles) and *conformal* (angle-preserving) deformations. The former model suits well inelastic and articulated motions, such as different poses of the human body, but is unable to capture significant shape variability (e.g. matching people of different stature or complexion). The class of conformal maps, on the other hand, is way too large: a classical result in differential geometry known as the *Uniformization Theorem* states that any closed simply-connected surface can be conformally mapped to a sphere [98]. Apparently, there are no other deformation classes that are larger than isometries but smaller than conformal.

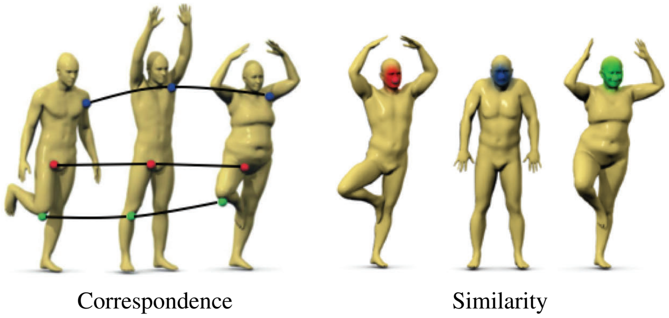


Fig. 4. Left: features used for shape correspondence should ideally manifest invariance across the shape class (e.g., the "knee feature" shown here should not depend on the specific person). Right: on the contrary, features used for shape retrieval should be specific to a shape within the class to allow distinguishing between different people. Similar features are marked with same color. Hand-crafting the right feature for each application is a very challenging task.

Furthermore, different applications in computer vision and graphics may require completely different features: for instance, in order to establish feature-based correspondence between a collection of human shapes, one would desire the descriptors of corresponding anatomical parts (noses, mouths, etc.) to be as similar as possible across the collection. In other words, such descriptors should be invariant to the collection variability. Conversely, for shape classification, one would like descriptors that emphasize the subject-specific characteristics, and for example, distinguish between two different nose shapes (see Figure VIII). Deciding a priori which structures should be used and which should be ignored is often hard or sometimes even impossible. Moreover, axiomatic modeling of geometric noise such as 3D scanning artifacts turns out to be extremely hard.

Put in a somewhat oversimplified manner, the computer vision community works with real-world 3D data, but uses Euclidean techniques originally developed for images that are not suitable for geometric data. At the same time, the mathematically rigorous models used in computer graphics to describe geometric objects can hardly deal with noisy data, leading to a tendency to work with idealized synthetic shapes. We believe that the gap between the two communities can be bridged with the development of geometric deep learning methods. By resorting to intrinsic deep neural networks, the invariance to isometric deformations is automatically built into the model, thus vastly reducing the number of degrees of freedom required to describe the invariance class. Roughly speaking, the intrinsic deep model will try to learn 'residual' deformations that deviate from the isometric model.

Intrinsic deep learning can be applied to several problems in 3D shape analysis, which can be divided in two classes. First, problems such as local descriptor learning [26], [77] or correspondence learning [28] (see example in the insert IN5), in which the output of the network is *point-wise*. The inputs to the network are some point-wise features, for example, color texture or simple geometric features. Using a CNN architecture with multiple intrinsic convolutional layers, it is possible to produce non-local features that capture the context around each point. The second type of problems such as shape recognition require the network to produce a *global* shape descriptor, aggregating all the local information into a single vector using e.g. the covariance pooling.

IX. Open problems and future directions

The recent emergence of geometric deep learning methods in various communities and application domains, which we tried to overview in this paper, allow us to proclaim, perhaps with some caution, that we might be witnessing a new field being born. We expect the following years to bring exciting new approaches and results, and conclude our review with a few observations of current key difficulties and potential directions of future research.

Many disciplines dealing with geometric data employ some empirical models or "handcrafted" features. This is a typical situation in computational sociology, where it is common to first come up with a hypothesis and then test it on the data [1], or geometry processing and computer graphics, where axiomatically-constructed features are used to analyze 3D shapes. Yet, such models assume some prior knowledge (e.g. isometric shape deformation model), and often fail to correctly capture the full complexity and richness of the data. In computer vision, departing from "handcrafted" features towards generic models learnable from the data in a task-specific manner has brought a breakthrough in performance and led to an overwhelming trend in the community to favor deep learning methods. Such a shift has not occurred yet in the fields dealing with geometric data due to the lack of adequate methods, but there are first indications of a coming paradigm shift.

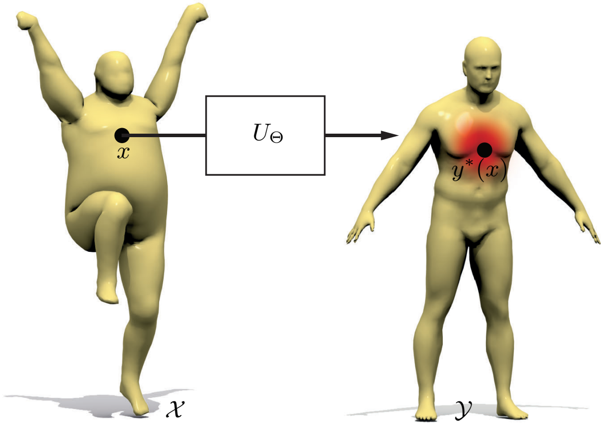
In some applications, geometric data can also be handled as a Euclidean structure, allowing to resort to classical deep learning techniques. In deformation-invariant 3D shape correspondence application we mentioned in the context of computer graphics, 3D shapes can be considered both as 2D manifolds and as subsets of the 3D Euclidean space. The latter representation fails to correctly capture the geometric structure of the data, as it is extrinsic and not invariant under non-rigid deformations. While in principle it is possible to apply classical deep learning to Euclidean representations of non-rigid shapes, such models tend to be very complex and require large amounts of training data [91]. The main contribution of intrinsic deep learning in these settings is using a more suitable model with guaranteed invariance properties that appear to be much simpler than the Euclidean ones.

Another important aspect is generalization capabilities and transfer learning. Generalizing deep learning models to geo-

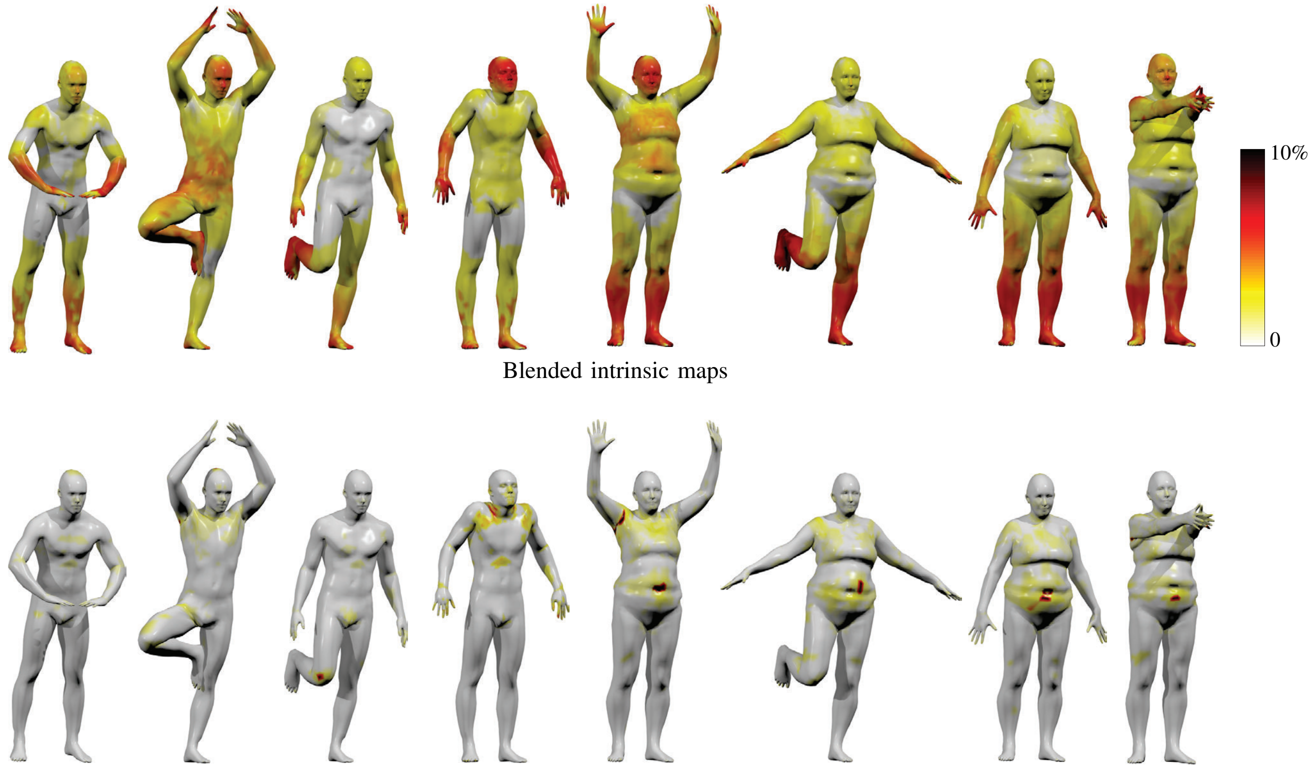
[IN5] 3D shape correspondence application: Finding intrinsic correspondence between deformable shapes is a classical tough problem that underlies a broad range of vision and graphics applications, including texture mapping, animation, editing, and scene understanding [99]. From the machine learning standpoint, correspondence can be thought of as a classification problem, where each point on the query shape is assigned to one of the points on a reference shape (serving as a "label space") [100]. It is possible to learn the correspondence with a deep intrinsic network applied to some input feature vector at each point x of the query shape $\mathcal{X}$, producing an output $U_{\mathbf{\Theta}}(f(x))(y)$, which is interpreted as the conditional probability $p(y|x)$ of x being mapped to y. Using a training set of points with their ground-truth correspondence $\mathcal{T} = \{(x, y^*(x))\}$, supervised learning is performed minimizing the *multinomial regression loss*

$$\min_{\mathbf{\Theta}} \; -\sum_{(x,y^*(x))\in\mathcal{T}} \log U_{\mathbf{\Theta}}(f(x))(y^*(x)) \qquad (66)$$

w.r.t. the network parameters $\mathbf{\Theta}$. The loss penalizes for the deviation of the predicted correspondence from the groundtruth.



[FIGS5a] Learning shape correspondence: an intrinsic deep network $U_{\mathbf{\Theta}}$ is applied point-wise to some input features defined at each point. The output of the network at each point x of the query shape $\mathcal{X}$ is a probability distribution of the reference shape $\mathcal{Y}$ that can be thought of as a soft correspondence.



Blended intrinsic maps

Intrinsic CNN

[FIGS2b] Quality of intrinsic correspondence established between human shapes using a state-of-the-art non-learning approach (blended intrinsic maps [96], first row) and learned using intrinsic deep architecture (Anisotropic CNN [28] with three convolutional layers, second row). SHOT descriptors capturing the local normal vector orientations [101] were used in this example as input features. Shown is the correspondence deviation from the groundtruth at each point, measured in % of geodesic diameter. Hotter colors represent larger errors.

metric data requires not only finding non-Euclidean counterparts of basic building blocks (such as convolutional and pooling layers), but also generalization *across* different domains. Generalization capability is a key requirement in many applications, including computer graphics, where a model is learned on a training set of non-Euclidean domains (3D shapes) and then applied to previously unseen ones. Recalling the approaches we mentioned in this review, spectral formulation of convolution allows designing CNNs on a graph, but the model learned this way on one graph cannot be straightforwardly applied to another one, since the spectral representation of convolution is domain-dependent. The spatial methods, on the other hand, allow generalization across different domains, but the construction of low-dimensional local spatial coordinates on graphs turns to be rather challenging. In particular, the generalization of anisotropic diffusion construction from manifolds to general graphs is an interesting research direction.

In this review, we addressed analysis problems on non-Euclidean domains. Not less important is the question of data *synthesis*; there have been several recent attempts to try to learn a *generative model* allowing to synthesize new images [102] and speech waveforms [103]. Extending such methods to the geometric setting seems a promising direction, though the key difficulty is the need to reconstruct the geometric structure (e.g., an embedding of a 2D manifold in the 3D Euclidean space modeling a deformable shape) from some intrinsic representation [104].

The final consideration is a computational one. All existing deep learning software frameworks are primarily optimized for Euclidean data. One of the main reasons for the computational efficiency of deep learning architectures (and one of the factors that contributed to their renaissance) is the assumption of regularly structured data on 1D or 2D grid, allowing to take advantage of modern GPU hardware. Geometric data, on the other hand, in most cases do not have a grid structure, requiring different ways to achieve efficient computations. It seems that computational paradigms developed for large-scale graph processing are more adequate frameworks for such applications.

Acknowledgement

The authors are grateful to Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodolà for comments on the manuscript and for providing figures FIGS4a and FIGS5b. MB is supported in part by the ERC Starting Grant No. 307047 (COMET) and a Google Faculty Research Award.

References

[1] D. Lazer *et al.*, "Life in the network: the coming age of computational social science," *Science*, vol. 323, no. 5915, p. 721, 2009.

[2] E. H. Davidson *et al.*, "A genomic regulatory network for development," *Science*, vol. 295, no. 5560, pp. 1669–1678, 2002.

[3] M. Ovsjanikov, M. Ben-Chen, J. Solomon, A. Butscher, and L. J. Guibas, "Functional maps: a flexible representation of maps between shapes," *ACM Trans. Graphics*, vol. 31, no. 4, p. 30, 2012.

[4] R. M. Rustamov, M. Ovsjanikov, O. Azencot, M. Ben-Chen, F. Chazal, and L. J. Guibas, "Map-based exploration of intrinsic shape differences and variability," *ACM Trans. Graphics*, vol. 32, no. 4, p. 72, 2013.

[5] M. Arie-Nachimson, S. Kovalsky, I. Kemelmacher-Shlizerman, A. Singer, and R. Basri, "Global motion estimation from point matches," in *Proc. 3DIMPVT*, 2012.

[6] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, in preparation.

[7] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[8] T. Mikolov, A. Deoras, D. Povey, L. Burget, and J. Černockỳ, "Strategies for training large scale neural network language models," in *Proc. ASRU*, 2011, pp. 196–201.

[9] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, and T. N. Sainath, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Sig. Proc. Magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[10] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Proc. NIPS*, 2014.

[11] Y. LeCun, K. Kavukcuoglu, and C. Farabet, "Convolutional networks and applications in vision," in *Proc. ISCAS*, 2010.

[12] D. Cireşan, U. Meier, J. Masci, and J. Schmidhuber, "A committee of neural networks for traffic sign classification," in *Proc. IJCNN*, 2011.

[13] A. Krizhevsky, I. Sutskever, and G. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. NIPS*, 2012.

[14] C. Farabet, C. Couprie, L. Najman, and Y. LeCun, "Learning hierarchical features for scene labeling," *Trans. PAMI*, vol. 35, no. 8, pp. 1915–1929, 2013.

[15] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, "Deepface: Closing the gap to human-level performance in face verification," in *Proc. CVPR*, 2014.

[16] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv:1409.1556*, 2014.

[17] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *arXiv:1512.03385*, 2015.

[18] E. P. Simoncelli and B. A. Olshausen, "Natural image statistics and neural representation," *Annual Review of Neuroscience*, vol. 24, no. 1, pp. 1193–1216, 2001.

[19] D. J. Field, "What the statistics of natural images tell us about visual coding," in *Proc. SPIE*, 1989.

[20] P. Mehta and D. J. Schwab, "An exact mapping between the variational renormalization group and deep learning," *arXiv:1410.3831*, 2014.

[21] S. Mallat, "Group invariant scattering," *Communications on Pure and Applied Mathematics*, vol. 65, no. 10, pp. 1331–1398, 2012.

[22] J. Bruna and S. Mallat, "Invariant scattering convolution networks," *Trans. PAMI*, vol. 35, no. 8, pp. 1872–1886, 2013.

[23] M. Tygert, J. Bruna, S. Chintala, Y. LeCun, S. Piantino, and A. Szlam, "A mathematical motivation for complex-valued convolutional networks," *Neural Computation*, 2016.

[24] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten ZIP code recognition," *Neural Computation*, vol. 1, no. 4, pp. 541–551, 1989.

[25] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio, "Maxout networks," *arXiv:1302.4389*, 2013.

[26] J. Masci, D. Boscaini, M. M. Bronstein, and P. Vandergheynst, "Geodesic convolutional neural networks on Riemannian manifolds," in *Proc. 3DRR*, 2015.

[27] D. Boscaini, J. Masci, S. Melzi, M. M. Bronstein, U. Castellani, and P. Vandergheynst, "Learning class-specific descriptors for deformable shapes using localized spectral convolutional networks," in *Computer Graphics Forum*, vol. 34, no. 5, 2015, pp. 13–23.

[28] D. Boscaini, J. Masci, E. Rodolà, and M. M. Bronstein, "Learning shape correspondence with anisotropic convolutional neural networks," in *Proc. NIPS*, 2016.

[29] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, "A neural probabilistic language model," *JMLR*, vol. 3, pp. 1137–1155, 2003.

[30] B. Alipanahi, A. Delong, M. T. Weirauch, and B. J. Frey, "Predicting the sequence specificities of DNA-and RNA-binding proteins by deep learning," *Nature Biotechnology*, 2015.

[31] D. K. Duvenaud, D. Maclaurin, J. Iparraguirre, R. Bombarell, T. Hirzel, A. Aspuru-Guzik, and R. P. Adams, "Convolutional networks on graphs for learning molecular fingerprints," in *Proc. NIPS*, 2015.

[32] M. B. Wakin, D. L. Donoho, H. Choi, and R. G. Baraniuk, "The multiscale structure of non-differentiable image manifolds," in *Proc. SPIE*, 2005.

[33] N. Verma, S. Kpotufe, and S. Dasgupta, "Which spatial partition trees are adaptive to intrinsic dimension?" in *Proc. Uncertainty in Artificial Intelligence*, 2009.

[34] J. B. Tenenbaum, V. De Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, no. 5500, pp. 2319–2323, 2000.

[35] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, no. 5500, pp. 2323–2326, 2000.

[36] L. Maaten and G. Hinton, "Visualizing data using t-SNE," *JMLR*, vol. 9, pp. 2579–2605, 2008.

[37] M. Belkin and P. Niyogi, "Laplacian eigenmaps for dimensionality reduction and data representation," *Neural Computation*, vol. 15, no. 6, pp. 1373–1396, 2003.

[38] R. R. Coifman and S. Lafon, "Diffusion maps," *App. and Comp. Harmonic Analysis*, vol. 21, no. 1, pp. 5–30, 2006.

[39] R. Hadsell, S. Chopra, and Y. LeCun, "Dimensionality reduction by learning an invariant mapping," in *Proc. CVPR*, 2006.

[40] B. Perozzi, R. Al-Rfou, and S. Skiena, "DeepWalk: Online learning of social representations," in *Proc. KDD*, 2014.

[41] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "LINE: Large-scale information network embedding," in *Proc. WWW*, 2015.

[42] S. Cao, W. Lu, and Q. Xu, "GraRep: Learning graph representations with global structural information," in *Proc. IKM*, 2015.

[43] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv:1301.3781*, 2013.

[44] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon, "Network motifs: simple building blocks of complex networks," *Science*, vol. 298, no. 5594, pp. 824–827, 2002.

[45] N. Pržulj, "Biological network comparison using graphlet degree distribution," *Bioinformatics*, vol. 23, no. 2, pp. 177–183, 2007.

[46] J. Sun, M. Ovsjanikov, and L. J. Guibas, "A concise and provably informative multi-scale signature based on heat diffusion," in *Computer Graphics Forum*, vol. 28, no. 5, 2009, pp. 1383–1392.

[47] R. Litman and A. M. Bronstein, "Learning spectral descriptors for deformable shape correspondence," *Trans. PAMI*, vol. 36, no. 1, pp. 171–180, 2014.

[48] S. Fortunato, "Community detection in graphs," *Physics Reports*, vol. 486, no. 3, pp. 75–174, 2010.

[49] T. Mikolov and J. Dean, "Distributed representations of words and phrases and their compositionality," *Proc. NIPS*, 2013.

[50] E. Cho, S. A. Myers, and J. Leskovec, "Friendship and mobility: user movement in location-based social networks," in *Proc. KDD*, 2011.

[51] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," *IEEE Sig. Proc. Magazine*, vol. 30, no. 3, pp. 83–98, 2013.

[52] M. Henaff, J. Bruna, and Y. LeCun, "Deep convolutional networks on graph-structured data," *arXiv:1506.05163*, 2015.

[53] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Proc. NIPS*, 2016.

[54] J. Atwood and D. Towsley, "Diffusion-convolutional neural networks," *arXiv:1511.02136v2*, 2016.

[55] S. Mallat, "Understanding deep convolutional networks," *Phil. Trans. R. Soc. A*, vol. 374, no. 2065, p. 20150203, 2016.

[56] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *Trans. PAMI*, vol. 32, no. 9, pp. 1627–1645, 2010.

[57] A. Dosovitskiy, P. Fischery, E. Ilg, C. Hazirbas, V. Golkov, P. van der Smagt, D. Cremers, T. Brox *et al.*, "Flownet: Learning optical flow with convolutional networks," in *Proc. ICCV*, 2015.

[58] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, "Striving for simplicity: The all convolutional net," *arXiv:1412.6806*, 2014.

[59] S. Mallat, *A wavelet tour of signal processing*. Academic Press, 1999.

[60] A. Choromanska, M. Henaff, M. Mathieu, G. B. Arous, and Y. LeCun, "The loss surfaces of multilayer networks," in *Proc. AISTATS*, 2015.

[61] I. Safran and O. Shamir, "On the quality of the initial basin in overspecified neural networks," *arXiv:1511.04210*, 2015.

[62] K. Kawaguchi, "Deep learning without poor local minima," in *Proc. NIPS*, 2016.

[63] T. Chen, I. Goodfellow, and J. Shlens, "Net2net: Accelerating learning via knowledge transfer," *arXiv:1511.05641*, 2015.

[64] J. Nash, "The imbedding problem for Riemannian manifolds," *Annals of Mathematics*, vol. 63, no. 1, pp. 20–63, 1956.

[65] M. Wardetzky, S. Mathur, F. Kälberer, and E. Grinspun, "Discrete laplace operators: no free lunch," in *Proc. SGP*, 2007.

[66] M. Wardetzky, "Convergence of the cotangent formula: An overview," in *Discrete Differential Geometry*, 2008, pp. 275–286.

[67] U. Pinkall and K. Polthier, "Computing discrete minimal surfaces and their conjugates," *Experimental Mathematics*, vol. 2, no. 1, pp. 15–36, 1993.

[68] S. Rosenberg, *The Laplacian on a Riemannian manifold: an introduction to analysis on manifolds*. Cambridge University Press, 1997.

[69] L.-H. Lim, "Hodge Laplacians on graphs," *arXiv:1507.05379*, 2015.

[70] U. Von Luxburg, "A tutorial on spectral clustering," *Statistics and Computing*, vol. 17, no. 4, pp. 395–416, 2007.

[71] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," *Proc. ICLR*, 2013.

[72] A. Kovnatsky, M. M. Bronstein, A. M. Bronstein, K. Glashoff, and R. Kimmel, "Coupled quasi-harmonic bases," in *Computer Graphics Forum*, vol. 32, no. 2, 2013, pp. 439–448.

[73] D. Eynard, A. Kovnatsky, M. M. Bronstein, K. Glashoff, and A. M. Bronstein, "Multimodal manifold analysis by simultaneous diagonalization of Laplacians," *Trans. PAMI*, vol. 37, no. 12, pp. 2505–2517, 2015.

[74] N. L. Roux, Y. Bengio, P. Lamblin, M. Joliveau, and B. Kégl, "Learning the 2-d topology of images," in *Proc. NIPS*, 2008.

[75] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv:1609.02907*, 2016.

[76] M. Andreux, E. Rodolà, M. Aubry, and D. Cremers, "Anisotropic Laplace-Beltrami operators for shape analysis," in *Proc. NORDIA*, 2014.

[77] D. Boscaini, J. Masci, E. Rodolà, M. M. Bronstein, and D. Cremers, "Anisotropic diffusion descriptors," in *Computer Graphics Forum*, vol. 35, no. 2, 2016, pp. 431–441.

[78] M. Gori, G. Monfardini, and F. Scarselli, "A new model for learning in graph domains," in *Proc. IJCNN*, 2005.

[79] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Trans. Neural Networks*, vol. 20, no. 1, pp. 61–80, 2009.

[80] Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel, "Gated graph sequence neural networks," *arXiv:1511.05493*, 2015.

[81] S. Sukhbaatar, A. Szlam, and R. Fergus, "Learning multiagent communication with backpropagation," *arXiv:1605.07736*, 2016.

[82] A. Coates and A. Y. Ng, "Selecting receptive fields in deep networks," in *Proc. NIPS*, 2011.

[83] D. I. Shuman, B. Ricaud, and P. Vandergheynst, "Vertex-frequency analysis on graphs," *App. and Comp. Harmonic Analysis*, vol. 40, no. 2, pp. 260–291, 2016.

[84] R. R. Coifman and M. Maggioni, "Diffusion wavelets," *Applied and Computational Harmonic Analysis*, vol. 21, no. 1, pp. 53–94, 2006.

[85] A. D. Szlam, M. Maggioni, R. R. Coifman, and J. C. BremerJr, "Diffusion-driven multiscale analysis on manifolds and graphs: top-down and bottom-up constructions," in *Optics & Photonics 2005*. International Society for Optics and Photonics, 2005, pp. 59 141D–59 141D.

[86] M. Gavish, B. Nadler, and R. R. Coifman, "Multiscale wavelets on trees, graphs and high dimensional data: Theory and applications to semi supervised learning," in *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, 2010, pp. 367–374.

[87] R. Rustamov and L. J. Guibas, "Wavelets on graphs via deep learning," in *Advances in Neural Information Processing Systems*, 2013, pp. 998–1006.

[88] X. Cheng, X. Chen, and S. Mallat, "Deep haar scattering networks," *Information and Inference*, p. iaw007, 2016.

[89] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad, "Collective classification in network data," *AI Magazine*, vol. 29, no. 3, p. 93, 2008.

[90] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller, "Multi-view convolutional neural networks for 3D shape recognition," in *Proc. ICCV*, 2015.

[91] L. Wei, Q. Huang, D. Ceylan, E. Vouga, and H. Li, "Dense human body correspondences using convolutional networks," in *Proc. CVPR*, 2016.

[92] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, "3D shapenets: A deep representation for volumetric shapes," in *Proc. CVPR*, 2015.

[93] C. R. Qi, H. Su, M. Nießner, A. Dai, M. Yan, and L. J. Guibas, "Volumetric and multi-view CNNs for object classification on 3D data," in *Proc. CVPR*, 2016.

[94] A. M. Bronstein, M. M. Bronstein, and R. Kimmel, "Generalized multidimensional scaling: a framework for isometry-invariant partial surface matching," *PNAS*, vol. 103, no. 5, pp. 1168–1172, 2006.

[95] M. M. Bronstein and I. Kokkinos, "Scale-invariant heat kernel signatures for non-rigid shape recognition," in *Proc. CVPR*, 2010.

[96] V. Kim, Y. Lipman, and T. Funkhouser, "Blended intrinsic maps," *ACM Trans. Graphics*, vol. 30, no. 4, p. 79, 2011.

[97] A. M. Bronstein, M. M. Bronstein, L. J. Guibas, and M. Ovsjanikov, "ShapeGoogle: Geometric words and expressions for invariant shape retrieval," *ACM Trans. Graphics*, vol. 30, no. 1, p. 1, 2011.

[98] H. Poincaré, "Sur l'uniformisation des fonctions analytiques," *Acta Mathematica*, vol. 31, no. 1, pp. 1–63, 1908.

[99] S. Biasotti, A. Cerri, A. M. Bronstein, and M. M. Bronstein, "Recent trends, applications, and perspectives in 3D shape similarity assessment," in *Computer Graphics Forum*, 2015.

[100] E. Rodolà, S. Rota Bulo, T. Windheuser, M. Vestner, and D. Cremers, "Dense non-rigid shape correspondence using random forests," in *Proc. CVPR*, 2014.

[101] F. Tombari, S. Salti, and L. Di Stefano, "Unique signatures of histograms for local surface description," in *Proc. ECCV*, 2010.

[102] A. Dosovitskiy, J. Springenberg, M. Tatarchenko, and T. Brox, "Learning to generate chairs, tables and cars with convolutional networks," 2015.

[103] S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," *arXiv:1609.03499*, 2016.

[104] D. Boscaini, D. Eynard, D. Kourounis, and M. M. Bronstein, "Shape-from-operator: Recovering shapes from intrinsic operators," in *Computer Graphics Forum*, vol. 34, no. 2, 2015, pp. 265–274.

Michael M. Bronstein (PhD with distinction 2007, CS Technion, Israel) is an Associate Professor at the USI Lugano, Switzerland and Tel Aviv University, Israel. He also serves as a Principal Engineer at Intel Perceptual Computing. His main research interest is in theoretical and computational methods for geometric data analysis. He authored over 150 papers and the book *Numerical geometry of non-rigid shapes* (Springer, 2008). He was awarded the highly competitive ERC Starting Grant (2012), Google Faculty award (2016), and invited as a Young Scientist to the World Economic Forum, an honor bestowed on forty worlds leading scientists under the age of 40. He is an ACM Distinguished Speaker and Associate editor of CVIU (2013-present). In addition to academic work, Michael is actively involved in commercial technology development and consulting to start-up companies. He was a co-founder and technology executive at Novafora (2005-2009) developing large-scale video analysis methods, and one of the chief technologists at Invision (2009-2012) developing low-cost 3D sensors. Following the multi-million acquisition of Invision by Intel in 2012, Michael has been one of the key developers of the Intel RealSense technology.

Joan Bruna graduated from Universitat Politècnica de Catalunya (Barcelona, Spain) in both Mathematics and Electrical Engineering. He obtained an M.Sc. in applied mathematics from ENS Cachan (France) . He then became a research engineer in an image processing startup, developing realtime video processing algorithms. He obtained his PhD in Applied Mathematics at Ecole Polytechnique (France) , under the supervision of Prof. Stéphane Mallat. He was a postdoctoral researcher at the Courant Institute, NYU, New York, and a postdoctoral fellow at Facebook AI Research. In 2015, he became Assistant Professor at UC Berkeley, Statistics Department, and starting Fall 2016 he joined the Courant Institute (NYU, New York) as Assistant Professor in Computer Science, Data Science and Mathematics (affiliated).

His research interests include invariant Signal representations, high-dimensional statistics and Stochastic processes, Deep Learning, and its applications to signal processing.

Yann LeCun (PhD 1987, Universitè Pierre et Marie Curie) is Director of AI Research at Facebook and Silver Professor at NYU. After a postdoc at the University of Toronto, he joined AT&T Bell Labs in Holmdel in 1988, where he became head of the Image Processing Research Department. He joined NYU as a professor in 2003, after a brief period as a Fellow of the NEC Research Institute in Princeton. Since the late 80's, LeCun has been working on deep learning methods, particularly the convolutional network model, which is the basis of many products and services deployed by companies such as Facebook, Google, Microsoft, Baidu, IBM, NEC, AT&T and others for image and video understanding, document recognition, human-computer interaction, and speech recognition. LeCun has been on the editorial board of IJCV, IEEE PAMI, and IEEE Trans. Neural Networks, was program chair of CVPR 2006 and chair of ICLR 2013 and 2014. He is on the science advisory board of Institute for Pure and Applied Mathematics, and has advised many large and small companies about machine learning technology, including several startups he co-founded. He is the recipient of the 2014 IEEE Neural Network Pioneer Award.

Arthur Szlam is a research scientist at Facebook AI Research. He received the PhD under Prof. Ronald Coifman at Yale University. Prior to joining Facebook, he was a postdoctoral fellow at the Institute for Mathematics and its Applications (IAM) at the University of Minnesota and the Institute for Pure and Applied Mathematics (IPAM) UCLA, and an assistant professor at City College of New York. He is a Sloan Foundation fellow. His research interests are in computational harmonic analysis, the relationships between smoothness, frequency, and scale on graphs and data clouds, and applications to signal processing and machine learning.

Pierre Vandergheynst (PhD in Mathematical Physics 1998, UCL, Belgium) is a Full Professor at the Swiss Federal Institute of Technology (EPFL), Switzerland. His research focuses on harmonic analysis, sparse approximations and mathematical image processing with applications to higher dimensional, complex data processing. He was co-Editor-in-Chief of Signal Processing (2002-2006) and is Associate Editor of the IEEE Transactions on Signal Processing (2007-present). He has been on the Technical Committee of various conferences and was Co-General Chairman of the EUSIPCO 2008 conference. Pierre Vandergheynst is the author or co-author of more than 50 journal papers, one monograph and several book chapters. He is a laureate of the Apple ARTS award and of the 2010-2011 De Boelpaepe prize from the Royal Academy of Sciences of Belgium. Pierre is strongly involved in technology transfer: he co-founded two start-ups and holds numerous patents.