

Learning intrinsic shape representations via spectral mesh convolutions

D. Lazzaro, S. Morigi*, P. Zuzolo

Department of Mathematics, University of Bologna, Bologna, Italy

ARTICLE INFO

Communicated by A.D. Jagtap

Keywords:

Graph Neural Networks
Graph-Laplacian
Laplacian eigendecomposition
Surface processing
Mesh denoiser
Physics-Informed Neural Networks

ABSTRACT

We introduce spectral-based convolutional operators embedded within Generalized Graph Neural Networks (G-GNNs). These operators enable deep learning on graphs through a learnable, energy-driven evolution process. This approach empowers us to impose specific properties on the graph convolutional kernel directly derived from the corresponding variational formulations. Our model incorporates both parameterized and non-parameterized graph Laplacian-based energies within the generalized graph convolutional layer to address features like smoothness, sharpness, and compact support. By making appropriate choices within our G-GNN framework, we pave the way for designing novel paradigms for 3D shape representation, reconstruction, and processing, while also enabling effective feature embeddings for intrinsic neural fields.

1. Introduction

Beyond the popular interpretation of a Neural Network (NN) as a black magic box which can be used to enter data and get a solution back, the scientific challenge turns towards the interpretation of NN as an abstract machine which creates a nonlinear mapping between an n -dimensional input data space and a p -dimensional output space, with n usually much larger than p . The different type of input data allows us to categorize them in a simplistic way the NN into Convolutional Neural Networks (CNN), Multilayer Perceptron (MLP), and Graph Neural Network (GNN). CNNs operate on a regular Euclidean data like images (2D grids) and texts (1D sequences) are mainly known for their role in image and video processing, and their architecture is based on three main computational layers: the convolutional layer/Pooling layer/Fully-connected layer. MLPs deal with tabular data (as signal samples) and consist of units and connections. Each unit has an activation, and each link between two units has a weight. The units are organized in layers and distinguished among input/hidden/output units. GNNs play pivotal roles in tackling irregularly structured data (e.g. graphs and polygonal meshes) and usually present a layer structure with input/hidden/output layers. Extending deep neural models to non-Euclidean domains, which is generally referred to as geometric deep learning, has been an emerging research area [1].

These three classes of neural networks share a common goal: to learn the underlying relationships or mappings between data by adjusting their internal network parameters. To achieve this, learning algorithms minimize the error of a context-aware model.

The impressive results achieved by CNN in many fields of image processing are due to an efficient architecture for extracting statistically

significant patterns in large-scale, high-dimensional data sets. CNNs make use of convolutional filters to extract localized spatial features of the input data. These convolutional filters, learnt from the data, have compact support and are translation invariant, i.e. they are able to recognize identical features regardless of their spatial position. A convolution is applied to the input data to filter the information and produce a processed feature map. Let x^ℓ represent a feature map at layer ℓ , A and Θ are small square grids representing the neighboring connections, and the corresponding weights, respectively, then the convoluted feature map at layer $\ell + 1$ reads as

$$x^{\ell+1} = A \odot \Theta x^\ell, \quad (1)$$

where \odot denotes the Hadamard product. Similarly, to collectively aggregate information from graph structure, GNNs use appropriate graph convolutional operators which take into account the structure of the underlying mesh/graph. The general Message Passing Neural Network framework for GNNs (see [2]), typically composes a feature transformation (weight matrix Θ) with a feature aggregation as follows

$$x^{\ell+1} = \text{aggregation}(\text{transformation}(x^\ell)), \quad (2)$$

and its convolutional flavor (see [3,4]) reads as

$$x^{\ell+1} = A x^\ell \Theta. \quad (3)$$

The node features x^ℓ are updated by aggregating the transformed node features, via the graph adjacency map A .

* Corresponding author.

E-mail address: serena.morigi@unibo.it (S. Morigi).

We propose a class of Generalized Graph Convolution (GGC) operators which generalize and extend the classic operator (3), and are defined as nonlinear maps

$$x^{\ell+1} = \text{GGC}(x^\ell; A, \Theta), \quad (4)$$

obtained by solving an energy-minimization problem. This allows to impose properties, such as sharpness, sparsity, and smoothness, which directly derive from the associated variational formulation. In our model, the GGC combines parameterized graph Laplacian-based energies, where the parameters are learned from data. This new definition of learned convolution combines the advantages of spatial connections of the underlying data structure, with the spectral representation of the graph, characterized by the graph Laplacian which intrinsically represents the graph structure itself.

Although generalizable to graph-structured input data, the GGC and the associated Generalized Graph Neural Network (G-GNN) here presented will focus on data meshes which represent piecewise-linear approximations of two-manifolds embedded in \mathbb{R}^3 . Polygonal meshes implicitly include a graph structure and represent the de facto standard explicit surface representations. The success of neural approaches to geometry processing has been evidenced both through their ability to represent complex geometries as well as their utility in end-to-end 3D shape learning, reconstruction, understanding and many other tasks. In particular, intrinsic neural field networks, introduced in [5], are emerging as a new 3D shape representation paradigm. They generalized neural fields on manifolds and their feature embedding is based on spectral properties of the Laplace–Beltrami operator (LBO). Being independent of the extrinsic Euclidean embedding, they inherit the favorable properties of intrinsic representations, such as the invariance under rigid and isometric deformations and reparametrization.

In this geometry processing context, we propose a G-GNN with appropriate GGC layers to approximate the LBO spectra of a given mesh, by using a tailor-made loss function in the spirit of the Physics-Informed Neural Networks (PINNs). The result is a shape-aware LBO-based feature embedding which can improve the performance of intrinsic neural fields.

Moreover, in order to validate the presented G-GNN framework, we propose GGC derived from several energy combinations which give rise to G-GNNs shape representation, reconstruction and denoising by exploiting the optimal mix of high/low-frequency information that the eigendecomposition of the graph Laplacian provides.

To summarize, the main contributions of this paper are:

- we present generalized graph convolutions (GGC) which are spectral-based convolution operators derived from a learnable energy-driven evolution process;
- we proposed a GGC embedding in a Generalized Graph Neural Network (G-GNN) which performs deep learning on graphs and, in particular, on polygonal meshes;
- we applied G-GNN as a learnable framework for solving geometry processing problems such as spectral mesh decomposition, compressed-mode mesh decomposition, feature embeddings for intrinsic neural fields as well as mesh denoising.

The paper is organized as follows. In Section 2 we introduce preliminary background on discrete manifolds and discrete operators on polygonal meshes, and depict the notations used in this paper. In Section 3 we review the eigendecomposition of the graph Laplacian and its applicability to shape analysis. The structure of the proposed Generalized GNN is described in Section 4 and its functionalities as minimization of (learnable) energies are discussed in Section 5. In Section 6 we characterize G-GNN as a new paradigm for object representation and manipulation. Finally, in Section 8 we provide results on the application of G-GNN to shape reconstruction, Laplacian eigendecomposition, shape-based feature embedding, and mesh denoising. Conclusions are drawn in Section 9.

1.1. Related work: convolution on graphs via energies

Graph Neural Networks (GNNs) represent a popular class of neural networks operating on graphs [6,7] and more recently on polygonal meshes [8–10]. Recent literature on GNN classifies the graph convolutional neural networks into two categories. The first one refers to the definition of convolution on graphs based on the spectral graph theory, the other category exploits a spatial domain convolution [6,11], which directly carries out the convolution operations on graphs. In the former the learnable parameters belong to the spectral domain, while in the latter they are defined in the spatial domain. Some popular convolution operators on spectral domain include the Spectral GNN (SGNN) [12], the Chebyshev neural network (ChebNet) [4], the Graph Convolutional Network (GCN) [3] and the p -Laplacian based graph neural networks (p GNNs) [13]. Unified approaches to the definition of convolutional operators on graphs have been proposed in [14], where the authors made the convolutional operators descend from the discretization of the Beltrami flow, and in [15] where the proposed elastic graph convolution follows a ℓ_1 - and ℓ_2 -based graph smoothing. An in-depth introduction to the theory of physics-inspired convolution on graphs has been presented in [16] where GNNs are derived from a class of energy functionals to obtain smoothing and sharpening effects on the features. This allowed for a generalization of GNN architectures thus providing a powerful paradigm to design new GNNs.

Our G-GNN proposal extends the approach in [15], by considering a wider set of energies, and the gradient-flow framework in [16] by using a more flexible GNN architecture with different channel mixing strategies.

2. Preliminary: differential operators on discrete manifolds

A large field of computer graphics and vision concerns the processing of 3D shapes, mathematically described as two-manifolds embedded in \mathbb{R}^3 . Polygonal meshes represent discrete approximations of two-manifolds and extrinsically embed the object shape into Euclidean space by declaring the connectivity between the 3D coordinates explicitly. The most common two-manifold approximation is the triangular mesh, a polygonal mesh composed of triangular faces. A key role in shape analysis is taken by the Laplace Beltrami differential operator, with corresponding discrete Laplacian on meshes.

In Section 2.1 we introduce notations used in this paper, Section 2.2 and Section 2.3 revisit the discrete LBO and its matrix representation, respectively.

2.1. Meshes and graphs

A *polygonal mesh* (or *discrete manifold*) is a triplet of finite sets $X = (\mathcal{V}, \mathcal{E}, \mathcal{T})$, where, denoted by $n_v = \#\mathcal{V}$ and $n_e = \#\mathcal{E}$, the geometry is defined by the set of vertices $\mathcal{V} = \{v_i\}_{i=1}^{n_v}$, with *vertex coordinates* $v = (x, y, z) \in \mathbb{R}^3$, $\forall v \in \mathcal{V}$. The mesh topology is defined by the set of edges and faces: $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V} = \{(i, j), \forall i, j = 1, \dots, n_v\}$, indices to the vertices that define the *edges*; $\mathcal{T} \subseteq \mathcal{V} \times \mathcal{V} \times \mathcal{V} = \{(i, j, k), \forall i, j, k = 1, \dots, n_v\}$, indices to the vertices that make up the *flat facets*. The index set of vertex neighbors v_j of a vertex v_i is denoted by $\mathcal{N}(i) = \{j : (i, j) \in \mathcal{E}\}$.

The definition of polygonal mesh implicitly includes a graph structure consisting of $(\mathcal{V}, \mathcal{E})$, which naturally leads to the definition of the adjacency matrix of a mesh as the adjacency matrix of its underlying undirected graph.

The *adjacency matrix* $\bar{A} \in \mathbb{R}^{n_v \times n_v}$ defined on a mesh X is a symmetric matrix of elements defined on edges (i, j) as

$$\begin{cases} \bar{w}_{i,j} > 0 & \text{if } (i, j) \in \mathcal{E} \\ \bar{w}_{i,j} = 0 & \text{otherwise} \end{cases}.$$

The *degree matrix* $\bar{D} \in \mathbb{R}^{n_v \times n_v}$ associated with the adjacency matrix \bar{A} is the diagonal matrix whose elements are the weighted sum on the connecting edges:

$$\bar{D} = \text{diag}(d_1, \dots, d_{n_v}), \quad d_i = \sum_{j \in \mathcal{N}(i)} \bar{w}_{i,j}, \quad \forall i = 1, \dots, n_v. \quad (5)$$

On graphs, the weights of the adjacency matrix are equal to 1 for each edge and zero otherwise so that each diagonal entry of the \bar{D} matrix is equal to the valence of the corresponding node. On triangular meshes, the weights are usually defined by the so-called cotangent formula, as follows

$$\bar{w}_{i,j} := \frac{1}{2}(\cot\alpha_{i,j} + \cot\beta_{i,j}), \quad \forall (i,j) \in \mathcal{E}, \quad (6)$$

where $\alpha_{i,j}$ and $\beta_{i,j}$ are the two opposite angles to the shared edge (i,j) , see [17].

In the following the matrix \bar{A} is considered in its normalized form

$$A = \bar{D}^{-1/2} \bar{A} \bar{D}^{-1/2}, \quad (7)$$

with elements $w_{ij} = \frac{\bar{w}_{ij}}{\sum_{j \in \mathcal{N}(i)} \bar{w}_{ij}}$, then $\sum_{j \in \mathcal{N}(i)} w_{ij} = 1$, and $D = I_{n_v}$, the identity matrix.

2.2. The discrete Laplace–Beltrami operator

In order to generalize the LBO on discrete manifolds, i.e. polygonal meshes, we introduce some preliminary definitions such as the gradient and divergence operators defined on the vertices and edges of the mesh, respectively.

Let $X = (\mathcal{V}, \mathcal{E}, \mathcal{T})$ be a polygonal mesh, we define the following function spaces:

$$\mathcal{L}^2(\mathcal{V}) = \{f : \mathcal{V} \rightarrow \mathbb{R}, v_i \mapsto f_i\}, \quad \mathcal{L}^2(\mathcal{E}) = \{F : \mathcal{E} \rightarrow \mathbb{R}, (i,j) \mapsto F_{i,j}\}.$$

Let $f \in \mathcal{L}^2(\mathcal{V})$, $F \in \mathcal{L}^2(\mathcal{E})$ be real-valued functions defined on the vertices and edges of X , respectively, we denote as f_i and $F_{i,j}$ the scalar values corresponding to the vertex $v_i \in \mathcal{V}$ and the edge $(i,j) \in \mathcal{E}$, with $\mathcal{L}^2(\mathcal{V}) \cong \mathbb{R}^{n_v}$, $f = (f_i)$, $i = 1, \dots, n_v$, $\mathcal{L}^2(\mathcal{E}) \cong \mathbb{R}^{n_e}$, $F = (F_{i,j})$, $(i,j) \in \mathcal{E}$.

The *mesh gradient* operator $\nabla_X : \mathcal{L}^2(\mathcal{V}) \rightarrow \mathcal{L}^2(\mathcal{E})$, at the edge $(i,j) \in \mathcal{E}$, is defined as

$$(\nabla_X f)_{i,j} := \begin{cases} \sqrt{w_{i,j}}(f_i - f_j) & \text{if } (i,j) \in \mathcal{E}, \\ 0 & \text{otherwise,} \end{cases} \quad (8)$$

which represents the directional derivative of f at v_i with respect to the direction defined by the edge (i,j) , and $w_{i,j}$ is the weight associated to the edge in the adjacency matrix A .

The vector of directional derivatives (mesh gradient) at the vertex $v_i \in \mathcal{V}$ is then defined as

$$(\nabla_X f)_i := \{(\nabla_X f)_{i,j}, \forall j : (i,j) \in \mathcal{E}\} \in \mathbb{R}^{\#\mathcal{N}(i)}. \quad (9)$$

Given the functions $f \in \mathcal{L}^2(\mathcal{V})$ and $F \in \mathcal{L}^2(\mathcal{E})$ on X . The *mesh divergence* operator, $\text{div} : \mathcal{L}^2(\mathcal{E}) \rightarrow \mathcal{L}^2(\mathcal{V})$, at the vertex $v_i \in \mathcal{V}$ is the adjoint of the mesh gradient, it satisfies $\langle \nabla f, g \rangle = \langle f, \text{div}(g) \rangle$, and is defined as

$$(\text{div} F)_i := \sum_{(i,j) \in \mathcal{E}} \sqrt{w_{i,j}}(F_{i,j} - F_{j,i}). \quad (10)$$

The discrete LBO on meshes is then defined by combining the two Eqs. (8) and (10).

Let $f \in \mathcal{L}^2(\mathcal{V})$ be a function on the vertices \mathcal{V} , the *mesh Laplacian operator* $\Delta_X : \mathcal{L}^2(\mathcal{V}) \rightarrow \mathcal{L}^2(\mathcal{V})$ at the vertex v_i is defined as

$$(\Delta_X f)_i := \frac{1}{2}(\text{div}(\nabla_X f))_i = \sum_{(i,j) \in \mathcal{E}} w_{i,j}(f_i - f_j). \quad (11)$$

When the adjacent matrix is normalized, then formula (11) can be rewritten as

$$(\Delta_X f)_i = (f_i - \sum_{(i,j) \in \mathcal{E}} w_{i,j} f_j), \quad (12)$$

which 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. The mesh Laplacian can be generalized to the *mesh p -Laplacian* ($p \in (1, \infty)$), defined at v_i as follows

$$(\Delta_p f)_i := \frac{1}{2}(\text{div}(\|\nabla_X f\|^{p-2} \nabla_X f))_i = \sum_{(i,j) \in \mathcal{E}} w_{i,j}^{p-1} |f_i - f_j|^{p-2} (f_i - f_j),$$

(13)

where $\|\cdot\|^{p-2}$ is element-wise, i.e.

$$\|(\nabla_X f)_i\|^{p-2} = (\|(\nabla_X f)_{i,1}\|^{p-2}, \|(\nabla_X f)_{i,2}\|^{p-2}, \dots, \|(\nabla_X f)_{i,\#\mathcal{N}(i)}\|^{p-2}).$$

In general, the p -Laplacian is a nonlinear operator, which implies $\Delta_p(\alpha f) \neq \alpha \Delta_p(f)$, for $\alpha \in \mathbb{R}$, and for $p = 2$ the ordinary mesh Laplacian is recovered.

2.3. Laplacian matrix representation

Given a polygonal mesh $X = (\mathcal{V}, \mathcal{E}, \mathcal{T})$, the mesh Laplacian operator (11) can be represented by the symmetric $n_v \times n_v$ *unnormalized Laplacian matrix*

$$\bar{L} = \bar{D} - \bar{A}, \quad (14)$$

where \bar{A} and \bar{D} are the adjacency matrix and the degree matrix, respectively. By applying to the adjacency matrix the normalization strategies adopted in (7) we get the *normalized Laplacian* defined as

$$L = \bar{D}^{-1/2} \bar{L} \bar{D}^{-1/2} = I - \bar{D}^{-1/2} \bar{A} \bar{D}^{-1/2} = I - A. \quad (15)$$

Given a function $f \in \mathcal{L}^2(\mathcal{V})$, sampled on the vertices of the mesh X , the matrix–vector product Lf approximates $\Delta_X f$, the Laplacian applied to a function f .

Remark 1. The mesh Laplacian operator (11) on unstructured meshes is more commonly represented as $\bar{L} = B^{-1}L$, where B is the diagonal matrix whose entries are the areas of the Voronoi regions (per-vertex area) around the mesh vertices defined as

$$(B)_{i,i} = \text{area}(v_i) = \frac{1}{3} \sum_{j,k:(i,j,k) \in \mathcal{T}} a_{i,j,k}, \quad (16)$$

where $a_{i,j,k} := \sqrt{s_{i,j,k}(s_{i,j,k} - \ell_{i,j})(s_{i,j,k} - \ell_{j,k})(s_{i,j,k} - \ell_{k,i})}$ is the area of triangle $(i,j,k) \in \mathcal{T}$ given by the Heron formula, $\ell_{i,j} := \|v_i - v_j\|_2$ is the edge length and $s_{i,j,k} := \frac{1}{2}(\ell_{i,j} + \ell_{j,k} + \ell_{k,i})$ is the semi-perimeter of triangle (i,j,k) .

The Laplacian matrix L , with weights defined as in (6), and the diagonal matrix B as in (16), correspond to the stiffness and lumped mass matrices of the linear FEM Laplacian \bar{L} , respectively. It can be proved that the discrete LBO, defined in (11) for a triangular mesh X , converges to the continuous LBO of the underlying manifold, when the mesh is infinitely refined, see [18]. The product matrix $B^{-1}L$ is symmetric only for diagonal matrix B .

3. Spectral analysis on meshes

In Riemannian geometry, it is common to use the orthogonal eigenbasis of the Laplacian to define an analogy of the Fourier transform on general domains. Analogously, in the discrete setting, by exploiting the eigendecomposition of the mesh Laplacian matrix, we can perform spectral analysis of data measured and thus generalize the Fourier analysis in the Euclidean domain.

As an appropriate discrete representation of the Laplace operator on Riemannian manifolds, the normalized Laplacian matrix L on a mesh X , defined in (15) with cotangent weights (6), is real symmetric positive semidefinite. Analogously to the continuous case L admits *eigendecomposition* on a compact domain (see [19,20]), which reads as

$$L = \Phi \Lambda \Phi^T, \quad (17)$$

where Λ is a diagonal matrix of real, non-negative *eigenvalues* $\lambda_i \in \mathbb{R}$, $\Lambda = \text{diag}(\lambda_1, \dots, \lambda_{n_v})$, $0 = \lambda_1 \leq \dots \leq \lambda_{n_v}$,

(18)

and Φ is a matrix of corresponding *orthonormal eigenvectors*

$$\Phi = (\phi_1, \dots, \phi_{n_v}), \quad (19)$$

such that $L\phi_i = \lambda_i\phi_i$, $\forall i = 1, \dots, n_v$, $\phi_i \in \mathbb{R}^{n_v}$. The first constant eigenfunction ϕ_1 has associated eigenvalue $\lambda_1 = 0$.

It can be shown that the eigenfunction ϕ_i is a critical point of the Rayleigh quotient defined as

$$R(L, \phi_i) = \frac{\langle L\phi_i, \phi_i \rangle}{\langle \phi_i, \phi_i \rangle}. \quad (20)$$

where $R(L, \phi_i)$ is the corresponding eigenvalue λ_i . Note that the quotient is constant under scaling, i.e. $R(L, \alpha\phi_i) = R(L, \phi_i)$ for any non-zero $\alpha \in \mathbb{R}$.

Given the decomposition (17) of L , if the mesh Laplacian is weighted by the area (eventually diagonal) matrix B , that is $\tilde{L} = B^{-1}L$, then the generalized eigendecomposition of (\tilde{L}, B^{-1}) reads as

$$\tilde{L}\Phi = B^{-1}\Phi\Lambda. \quad (21)$$

The eigenvectors are orthonormal with respect to the B -scalar product; i.e., $\langle \phi_i, \phi_j \rangle_B = \phi_i^T B \phi_j = \delta_{ij}$, and satisfy the identity $\tilde{L}\phi_i = \lambda_i B^{-1}\phi_i$, $\forall i$. The generalized Rayleigh quotient is

$$R(\tilde{L}, B^{-1}, \phi_i) = \frac{\langle \tilde{L}\phi_i, \phi_i \rangle}{\langle B^{-1}\phi_i, \phi_i \rangle}. \quad (22)$$

It is also useful to see that (21) is conceptually equivalent to a ‘‘regular’’ eigendecomposition $B\tilde{L}\Phi = \Phi\Lambda$ on the matrix L .

Eigenvalues and eigenvectors computations are known to be extremely computationally intensive for large-scale meshes. Efficient numerical computation of the eigenvectors of the mesh Laplacian matrix involves the generalization of the Dirichlet energy on a discrete domain, defined as $E^{Dir}(\phi) := \sum_i \|(\nabla_X \phi)_i\|^2 = \text{tr}(\phi^T \Delta_X \phi)$, where $\text{tr}(\cdot)$ denotes trace of a matrix. Given the mesh $X = (\mathcal{V}, \mathcal{E}, \mathcal{T})$, to compute the K lowest eigenpairs, the eigendecomposition problem can be recast as an optimization problem requiring the minimization of the total energy subject to orthonormality conditions

$$\Phi^* = \arg \min_{\Phi \in \mathbb{R}^{n_v \times K}} \text{tr}(\Phi^T \Delta_X \Phi), \text{ s.t. } \Phi^T \Phi = I. \quad (23)$$

We will denote by $\Phi_K \in \mathbb{R}^{n_v \times K}$, $K < n_v$, the solution of (23); the columns of Φ_K represent the first K eigenvectors of Δ_X and the diagonal elements of Λ the corresponding K eigenvalues. It is easy to observe that by applying the first-order optimality conditions for the Lagrangian function associated to Eq. (23), yields to the eigenvalue problem $\Delta_X \Phi = \Phi\Lambda$.

When Δ_X is instead discretized by \tilde{L} then the solution to the problem (23), obtained by imposing in (23) the B-orthogonality constraint $\Phi^T B \Phi = I$, satisfies the eigendecomposition (21).

The discrete Laplacian eigenfunctions are dense and have global spatial support (i.e., they are null only at some isolated points). Compressed Modes (CM), introduced in [21], are instead spatially sparse orthogonal eigenfunctions $(\psi_1, \dots, \psi_{n_v})$ of the mesh Laplacian. In [22] the authors aimed at computing the K lowest of them by solving the variational problem

$$\Psi^* = \arg \min_{\Psi \in \mathbb{R}^{n_v \times K}} \sum_{l=1}^{n_v} \left(\langle \psi_l, \Delta_X \psi_l \rangle + \frac{1}{\mu} \|\psi_l\|_p^p \right) \text{ s.t. } \Psi^T \Psi = I. \quad (24)$$

In (24) the first term of the sum enforces Ψ to be eigenfunctions of the Laplacian, while the constraint imposes orthonormality. The additional ℓ_p -norm, with $0 < p \leq 1$, is used to achieve spatial sparsity. Such functions verify local support and completeness.

The mesh Laplacian eigenfunctions are the smoothest functions in the sense of the Dirichlet energy. In particular, they can be interpreted as a generalization of the functions of the standard Fourier basis, and its eigenvalues are referred to as graph frequencies. Low frequencies correspond to the smallest eigenvalues. These frequencies represent smooth, slowly varying functions and encode macroscopic shape information. Conversely, high frequencies correspond to the largest eigenvalues. These eigenfunctions typically exhibit rapid oscillations and capture

microscopic details of the shape. The role of the graph Fourier transform is taken by Φ^T , and that of the inverse graph Fourier transform is assumed by Φ .

In addition, the eigenfunctions $\{\phi_i\}_{i=1}^{n_v}$ form an orthonormal basis of $\mathcal{L}^2(\mathcal{V})$, and thus allow us to expand any function living on $\mathcal{L}^2(\mathcal{V})$, while the Laplacian eigenvalues encode intrinsic geometric information of the domain. Then any function $f \in \mathcal{L}^2(\mathcal{V})$ defined at the vertices of X can be represented as a linear combination of the eigenbasis as

$$f = \sum_{i=1}^{n_v} \langle f, \phi_i \rangle \phi_i = \Phi_{n_v} \Phi_{n_v}^T f, \quad (25)$$

where $\langle f, \phi_i \rangle$ is the spectral coefficient obtained as a projection of the function f along the direction of the i th eigenvector ϕ_i . The above expression represents a transform from the original function f in terms of the new basis composed by the eigenvectors of L .

We consider now the problem of constructing a low-dimensional representation for data lying on a high-dimensional space. To this end, we can apply the spectral transform as a shape approximation of a 3D object, by considering the expansion of the $x, y, z \in \mathcal{L}^2(\mathcal{V})$ coordinate functions of the mesh. For a mesh with n_v vertices, they are represented by a matrix $M \in \mathbb{R}^{n_v \times 3}$ whose columns specify the x, y , and z coordinates of the mesh vertices. Low-dimensional representations of a shape can be achieved by keeping only the leading spectral transform coefficients (both low-frequency and high-frequency components) and discarding the remaining ones. Given the coordinate functions M associated with the mesh X , we denote by $X^{(K)}$ the mesh reconstructed by using only the $K \leq n_v$ leading coefficients which is characterized by the vertex coordinates

$$M^{(K)} = \sum_{i=1}^K \langle M, \phi_i \rangle \phi_i = \Phi_K \Phi_K^T M. \quad (26)$$

This represents a sort of compression of the surface geometry since only K out of n_v coefficients need to be stored to approximate the original shape. We can quantify the information loss by measuring the Shape Recovery (SR) error defined as

$$\begin{aligned} SR(M, M^{(k)}) &:= \frac{1}{n_v} \|M - M^{(k)}\|_F = \left\| \sum_{i=K+1}^{n_v} \langle M, \phi_i \rangle \phi_i \right\|_2 \\ &= \sqrt{\sum_{i=K+1}^{n_v} \langle M, \phi_i \rangle^2}. \end{aligned} \quad (27)$$

4. Extending graph neural networks with GGC layers

Graph neural networks have been proposed to process data not organized on regular lattices, best represented by graph data structures, designed as a generalization of CNNs on regular grids. GNNs have shown great capacity in learning representations mainly for graphs [23], and, more recently, also for unstructured polygonal 3D meshes [10,24]. GNN-based mesh processing methods typically involve encoding the mesh as a graph, where each vertex represents a node and each edge represents a connection between two nodes. GNNs process mesh data acting on the features assigned to their nodes taking into account the structure of the underlying mesh through appropriate mesh convolutional operators.

Given a 3D mesh $X = (\mathcal{V}, \mathcal{E}, \mathcal{T})$ with n_v nodes, at each node i is associated a n_f -dimensional feature vector x_i . We denote by $x \in \mathbb{R}^{n_v \times n_f}$, $x = [x_1, x_2, \dots, x_{n_v}]^T$ the node embedding matrix representing the *feature map* of the mesh X . The features can represent geometric coordinates of the vertices as well as additional properties such as colors, normals, and so on.

We propose a Generalized-Graph Neural Network (G-GNN) to leverage the inherent suitability of GNNs for mesh representation while incorporating a node embedding strategy based on an energy-driven feature propagation process. A G-GNN is a graph convolutional network (GCN), built for a fixed mesh, where at each node the classical graph

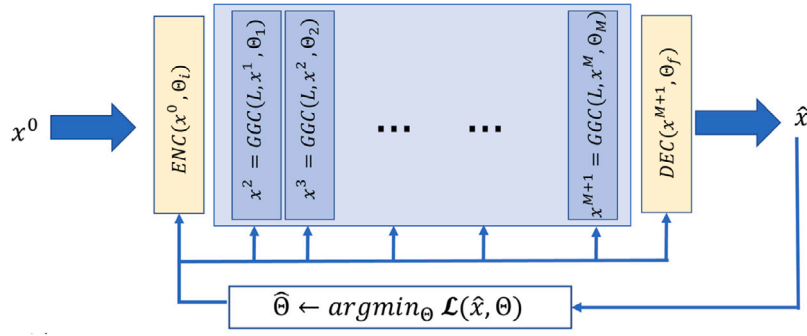


Fig. 1. The general design pipeline for a G-GNN architecture.

Table 1

Non-parametric (left) and parametric (right) energy terms.

Non-Parametric energy	Parametric energy
$E^{Dir}(x) := \frac{1}{2} \sum_{(i,j) \in \mathcal{E}} \ (\nabla_X x)_{i,j}\ _F^2$	$E_{\theta}^{Dir}(x) := \frac{1}{2} \sum_{(i,j) \in \mathcal{E}} \ W(\nabla_X x)_{i,j}\ _F^2$
$E^{fid}(x; y) := \frac{1}{2} \ x - y\ _F^2$	$E_{\theta}^{fid}(x; y) := \frac{1}{2} \ (x - y)W\ _F^2$
$E^{TV_p}(x) := \sum_{(i,j) \in \mathcal{E}} \ (\nabla_X x)_{i,j}\ _F^p$	$E_{\theta}^{TV_p}(x) := \sum_{(i,j) \in \mathcal{E}} \ W(\nabla_X x)_{i,j}\ _F^p$
$E^{\ell_p}(x) := \ x\ _p^p$	$E_{\theta}^{\ell_p}(x) := \ xW\ _p^p$

convolution (3) is replaced with a Generalized Graph Convolution (4) which embeds graph Laplacian-based algorithms.

A G-GNN, from a mathematical point of view, learns a nonlinear mapping F that, applied to a low-dimensional (d) input $x^0 \in \mathbb{R}^{n_v \times d}$, working in the feature space, produces the output features $\hat{x} \in \mathbb{R}^{n_v \times n_{out}}$. The mapping F is the composition of $M + 2$ functions

$$F(x^0, \Theta) = DEC_{\theta_f} \circ GGC_{\theta_M} \circ GGC_{\theta_{M-1}} \circ \dots \circ GGC_{\theta_1} \circ ENC(x^0, \Theta_1), \quad (28)$$

each parameterized by a set of trainable weights Θ used for feature transformation, where $ENC : \mathbb{R}^{n_v \times d} \rightarrow \mathbb{R}^{n_v \times n_f}$, $GGC : \mathbb{R}^{n_v \times n_f} \rightarrow \mathbb{R}^{n_v \times n_f}$, $DEC : \mathbb{R}^{n_v \times n_f} \rightarrow \mathbb{R}^{n_v \times n_{out}}$.

The architecture of the proposed G-GNN is shown in Fig. 1. The encoder (ENC) and decoder (DEC) blocks that enclose the M hidden layers allow to transformation of the data in the appropriate feature dimensional space n_f . Nonlinear activations could be added between consecutive GGN layers. However, the present work does not consider any activation in-between. The M hidden GGC layers are represented within the blue boxes in Fig. 1; the j th GGC layer is characterized by learnable weights Θ_j . The design of some specific GGC layers will be described in Section 5. Finally the result \hat{x} of the mapping F can be interpreted as an energy evolution applied to initial data x^0 , as will be described in detail in Section 5.

5. Evolution of G-GNN as minimization of a (learnable) energy

The evolution of a G-GNN can be represented as the minimization of a (learnable) energy. In this section, we characterize GGC operators which derive from the minimization of variational formulations, which combine several (parametric) as well as nonparametric energy terms. This will allow to impose of properties on the node features directly derived from the associated energy terms. The advantages compared to the classic algorithmic approaches of numerical optimization are attributable to the integration of learnable parameters that weigh the effect of the operators based on the conformation of the analyzed mesh.

Let $X = (\mathcal{V}, \mathcal{E}, \mathcal{T})$ be a mesh and A be its associated adjacency matrix, $x \in \mathbb{R}^{n_v \times n_f}$ denotes a feature map. Table 1 reports the energy terms that will be considered in the design of some GGC proposals. We denoted by $\|x\|_F^2 = tr(x^T x) = \langle x, x \rangle$ the Frobenius norm of the matrix $x \in \mathbb{R}^{m \times n}$, and by $\|x\|_p = (\sum_{i=1}^m \sum_{j=1}^n |x_{i,j}|^p)^{\frac{1}{p}}$ the p -norm of a matrix x . If we want to induce a smoothing or sharpening behavior over the features, then we can resort to the non-negative discrete vector Dirichlet energy E^{Dir} (Table 1 - left panel first row) which can be rewritten as $E^{Dir}(x) = tr(x^T \Delta_X x)$ and measures the smoothness of a vector field on the mesh. Features are smooth if the Dirichlet energy is small. A smoothness control is instead obtained by the generalization of the Total Variation p -norm (Table 1 - left panel third row) with appropriate parameter $p \in \mathbb{R}, p > 0$, E^{TV_p} tunes the feature variation over the entire mesh. These are considered internal energies since they are related to the mesh topology. In contrast, the second and fourth energy terms in Table 1 (left panel) represent external energies in the feature space. These terms are independent of pairwise interactions. Specifically, the energy term E^{fid} forces the optimal feature map x^* to stay close to the available data y , while E^{ℓ_p} provides greater control over feature regularization.

The right panel in Table 1 reports the correspondent learnable energy terms. We can generalize the Dirichlet energy E_{θ}^{Dir} on the left panel in Table 1 rewritten as $E_{\theta}^{Dir}(x) = tr(W^T x^T \Delta_X x W)$, by weighting the norm of the intrinsic gradient ∇_X operator. We treat the elements of W as learnable weights which have the effect of magnification or minification of the Laplacian frequencies. The relationship between W and Θ will be defined in the following results.

In the remainder of this section, we propose three different GGC operators. These operators are derived by combining appropriate energy terms to achieve a specific behavior in the feature evolution of the G-GNN. In particular, the smoothing GGC (S-GGC), the sparsity-inducing (p-GGC) and the enhancing GGC (E-GGC). Each GGC operator will characterize a different task of the $(\ell + 1)$ th GGC layer of the network by updating the features x^{ℓ} at the ℓ th layer, associated with all the vertices of the mesh, to produce the output feature matrix $x^{\ell+1}$. For each GGC operator, we propose both a non-parametric energy-based derivation, defined as

$$x^{\ell+1} = GGC(x^{\ell}; L)\Theta, \quad (29)$$

and its associated parametric energy-based derivation,

$$x^{\ell+1} = GGC(x^{\ell}; L, \Theta), \quad (30)$$

which imply the nonlinear integration of the weight matrix W as a posteriori transformation — non-parametric case (29) — or inside the energy terms — parametric case (30).

The following results represent only some of the possible evolutions of energies that can be induced in the mesh-based neural network. The proposal would pave the way for new and promising characterizations of G-GNN for mesh processing.

The proofs of the Propositions 5.1–5.6 are postponed to Appendix.

Proposition 5.1. [Non-Parametric S-GGC] Let $L \in \mathbb{R}^{n_v \times n_v}$ be a normalized graph Laplacian matrix, $\Theta \in \mathbb{R}^{n_f \times n_f}$ matrix of learnable parameters, and $\alpha, \lambda \in \mathbb{R}_{++}$. Then the non-parametric Smoothing Generalized Graph Convolution operator at layer ℓ reads as

$$x^{\ell+1} = \text{S-GGC}(x^\ell; L; \Theta), \quad (31)$$

where $\text{S-GGC}(x^\ell; L) = x^K$ is obtained, starting from $x^0 = x^\ell$, by iterating

$$x^{k+1} = (I - 2\alpha L - \lambda\alpha)x^k, \quad k = 1, \dots, K. \quad (32)$$

The sequence $(x^k)_{k \in \mathbb{N}}$, under the condition on the step-size $0 < \alpha < 2/(\lambda + 2\|L\|_2)$ converges to the optimal minimizer of the energy functional

$$E(x) := \frac{\lambda}{2} \|x\|_F^2 + \text{tr}(x^T Lx). \quad (33)$$

Remark 2. The S-GGC convolutional operator in (31) reduces to the well-known forward-propagation operator GCN at the ℓ th layer, see [3], by replacing L in (31)–(32) with its definition in (15)

$$x^{\ell+1} = (I - 2\alpha(I - A) - \lambda\alpha)x^\ell \Theta \quad (34)$$

$$= Ax^\ell \Theta, \quad (35)$$

where we set $\lambda = 0$ and $\alpha = 1/2$ in (35).

Proposition 5.2. [Parametric S-GGC] Let $L \in \mathbb{R}^{n_v \times n_v}$ be a normalized graph Laplacian matrix, $\Theta = \{\theta_1, \theta_2\}$, with $\theta_i \in \mathbb{R}^{n_f \times n_f}$, $i = 1, 2$, matrices of learnable parameters, $y \in \mathbb{R}^{n_v \times n_f}$ is assigned and $\alpha, \lambda \in \mathbb{R}_{++}$. Then the parametric Smoothing Generalized Graph Convolution operator at layer ℓ reads as

$$x^{\ell+1} = \text{S-GGC}(x^\ell; L, \Theta) \quad (36)$$

where $\text{S-GGC}(x^\ell; L, \Theta) = x^K$ is obtained starting from $x^0 = x^\ell$, by iterating

$$x^{k+1} = x^k - \alpha[\lambda(x^k - y)\theta_1 + 2Lx^k\theta_2], \quad k = 1, \dots, K. \quad (37)$$

The sequence $(x^k)_{k \in \mathbb{N}}$, under the condition on the step-size $0 < \alpha < 2/(\lambda\|\theta_1\|_2 + 2\|L\|_2\|\theta_2\|_2)$ converges to the optimal minimizer of the energy functional

$$E_\Theta(x) := \frac{\lambda}{2} \|(x - y)W_1\|_F^2 + \text{tr}(W_2^T x^T LxW_2), \quad (38)$$

with $\theta_* = W_* W_*^T$.

In the following results, reported in Propositions 5.3 and 5.4, we leverage the generalized soft-thresholding (GST) function, introduced in [25] to solve the ℓ_p -norm minimization problem, which is denoted by $T_p^{GST}(\cdot)$ and defined as follows

$$T_p^{GST}(z; \beta) = \begin{cases} 0 & \text{if } |z| \leq \hat{s}(\beta) \\ \text{sign}(z)s^* & \text{if } |z| > \hat{s}(\beta), \end{cases} \quad (39)$$

where $\hat{s}(\beta)$ is the thresholding value given by

$$\hat{s}(\beta) = (2\beta(1-p))^{1/(2-p)} + \beta p(2\beta(1-p))^{(p-1)/(2-p)}, \quad (40)$$

and s^* is the non-zero solution of the nonlinear equation

$$s^* - z + \beta p(s^*)^{p-1} = 0. \quad (41)$$

Proposition 5.3. [Non-parametric p-GGC] Let $L \in \mathbb{R}^{n_v \times n_v}$ be a normalized graph Laplacian matrix, $\Theta \in \mathbb{R}^{n_f \times n_f}$ matrix of learnable parameters, $0 < p \leq 1$ be assigned, and $\alpha, \lambda \in \mathbb{R}_{++}$. Then the non-parametric p-sparsity-inducing Generalized Graph Convolution operator at layer ℓ reads as

$$x^{\ell+1} = \text{p-GGC}(x^\ell; L; \Theta) \quad (42)$$

where $\text{p-GGC}(x^\ell; L) = x^K$ is obtained starting from $x^0 = x^\ell$, by iterating

$$x^{k+1} = T_p^{GST}((I - 2\alpha L)x^k; \alpha\lambda) \quad k = 1, \dots, K. \quad (43)$$

The sequence $(x^k)_{k \in \mathbb{N}}$ generated from (43), under the condition on the step-size $0 < \alpha < 1/\|L\|_2$, converges to a local minimizer of the energy functional

$$E(x) := \lambda \|x\|_p^p + \text{tr}(x^T Lx). \quad (44)$$

Proposition 5.4. [Parametric p-GGC] Let $L \in \mathbb{R}^{n_v \times n_v}$ be a normalized graph Laplacian matrix, $\Theta = \{\theta_1, \theta_2\}$, with $\theta_i \in \mathbb{R}^{n_f \times n_f}$, $i = 1, 2$, matrices of learnable parameters, $0 < p \leq 1$ is assigned and $\alpha, \lambda \in \mathbb{R}_{++}$. Then the parametric p-sparsity-inducing Generalized Graph Convolution operator at layer ℓ reads as

$$x^{\ell+1} = \text{p-GGC}(x^\ell; L, \Theta), \quad (45)$$

where $\text{p-GGC}(x^\ell; L, \Theta) = x^K$ is obtained starting from $x^0 = x^\ell$, by iterating

$$x^{k+1} = T_p^{GST}(x^k - 2\alpha Lx^k\theta_2; \lambda\alpha W_1) \quad k = 1, \dots, K. \quad (46)$$

The sequence $(x^k)_{k \in \mathbb{N}}$, under the condition on the step-size $0 < \alpha < 1/(\|L\|_2\|\theta_2\|)$, converges to a minimizer of the energy functional

$$E_\Theta(x) := \lambda \|xW_1\|_p^p + \text{tr}(W_2^T x^T LxW_2) \quad (47)$$

with $\theta_2 = W_2 W_2^T$, and $\theta_1 = W_1 = \text{diag}(w_1, \dots, w_{n_f})$.

Proposition 5.5. [Non parametric E-GGC] Let $L \in \mathbb{R}^{n_v \times n_v}$ be a normalized graph Laplacian matrix, $\Theta \in \mathbb{R}^{n_f \times n_f}$ matrix of learnable parameters, $y \in \mathbb{R}^{n_f \times n_f}$, $p > 1$ and $\lambda \in \mathbb{R}_{++}$ be assigned. Then the non-parametric Enhancing Generalized Graph Convolution operator at layer ℓ reads as

$$x^{\ell+1} = \text{E-GGC}(x^\ell; L; \Theta) \quad (48)$$

where $\text{E-GGC}(x^\ell; L) = x^K$ is obtained starting from $x^0 = x^\ell$, by iterating

$$x^{k+1} = D^{-1}(Mx^k + \lambda y), \quad k = 1, \dots, K, \quad (49)$$

with matrices M and $D = \text{diag}(d_1, \dots, d_n)$ defined by

$$(M)_{ij} = (1 - L_{ij})^{\frac{p}{2}} \|x_i - x_j\|^{p-2} \in \mathbb{R}, \quad d_i = \sum_{j \in \mathcal{N}(i)} M_{ij} + \lambda \in \mathbb{R}. \quad (50)$$

The sequence $(x^k)_{k \in \mathbb{N}}$ converges to a minimizer of the energy functional

$$E(x) := \frac{\lambda}{2} \|x - y\|_F^2 + \frac{1}{p} \sum_{(i,j) \in \mathcal{E}} \|(\nabla_X x)_{i,j}\|_F^p. \quad (51)$$

Proposition 5.6. [Parametric E-GGC] Let $L \in \mathbb{R}^{n_v \times n_v}$ be a normalized graph Laplacian matrix, $\Theta = \{\theta_1, \theta_2\}$, with $\theta_i \in \mathbb{R}^{n_f \times n_f}$, $i = 1, 2$, diagonal matrices of learnable parameters, with $\theta_1 = W_1 W_1^T$, and $\theta_2 = W_2$, $p > 1$, $y \in \mathbb{R}^{n_f \times n_f}$ be assigned, and $\lambda \in \mathbb{R}_{++}$. Then the parametric Enhancing Generalized Graph Convolution operator at layer ℓ reads as

$$x^{\ell+1} = \text{E-GGC}(x^\ell; L, \Theta) \quad (52)$$

where $\text{E-GGC}(x^\ell; L, \Theta) = x^K$ is obtained starting from $x^0 = x^\ell$, by iterating

$$x^{k+1} = (Mx^k\theta_2 + \lambda y\theta_1)D^{-1}, \quad k = 1, \dots, K, \quad (53)$$

with $M \in \mathbb{R}^{n_f \times n_f}$ and $D = D\theta_2 + \lambda\theta_1 = \text{diag}(d_i(\theta_2)_i + \lambda(\theta_1)_i)$, matrices defined by elements

$$M_{i,j} = (1 - L_{ij})^{\frac{p}{2}} \|x_i - x_j\|^{p-2}, \quad d_i = \sum_{j \in \mathcal{N}(i)} M_{i,j}, \quad (54)$$

respectively. The sequence $(x^k)_{k \in \mathbb{N}}$ converges to a minimizer of the energy functional

$$E_\Theta(x) := \frac{\lambda}{2} \|(x - y)W_1\|_F^2 + \sum_{(i,j) \in \mathcal{E}} \|W_2(\nabla_X x)_{i,j}\|_F^p. \quad (55)$$

6. G-GNN as a new paradigm for object representation and manipulation

In order to evaluate the performance of the proposed GGC operators, presented in Propositions 5.1–5.6, we consider five different goals in the context of 3D surface processing. According to the different GGC involved, given a mesh $X = (\mathcal{V}, \mathcal{E}, \mathcal{T})$ with the associated graph-Laplacian matrix L , we aim to present special G-GNNs with $x^0 = X$, as valid paradigms for 3D object representation, reconstruction, and manipulation. In the remainder of this section, we introduce the five different goals, named TASKs, while in Section 8 we report the results produced for each of the tasks.

In particular, as a first task (denoted by TASK A), we apply G-GNN with smoothing GGC (S-GGC, 1 layer) to approximate N_e eigenfunctions of the eigendecomposition (Φ, Λ) of the graph-Laplacian $L = \Phi^T \Lambda \Phi$ associated with the mesh X . The spectrum of the Laplace–Beltrami operator of any 2D or 3D manifold (surface or solid) contains intrinsic shape information called ‘Shape-DNA’, it represents a fingerprint or signature of the object itself. It can be used to identify and compare objects like surfaces and solids independently of their representation, position and (if desired) independently of their size [26]. TASK A performs challenging computation of a good Shape-DNA for an object represented by a polygonal mesh X .

The performance in approximating the eigendecomposition will be assessed by the following quantitative measures:

- *Eigenfunctions Orthogonality Recovery (EOR) error*

$$EOR(\hat{x}) := \frac{1}{N_e} \|\hat{x}^T \hat{x} - I\|_F^2, \quad (56)$$

- *Eigenvalues Recovery (ER) error*

$$ER(\hat{x}; L) := \frac{1}{N_e} \|L\hat{x} - R(L, \hat{x})\hat{x}\|_F^2 = \frac{1}{N_e} \sum_{j=1}^{N_e} \left(L\hat{x}^j - \frac{\langle L\hat{x}^j, \hat{x}^j \rangle}{\langle \hat{x}^j, \hat{x}^j \rangle} \hat{x}^j \right). \quad (57)$$

where R is the Rayleigh quotient defined in (20), and with \hat{x}^j we denote the j th column of the feature matrix $\hat{x} \in \mathbb{R}^{n_v \times N_e}$.

A deep learning solution of the eigenvalue problem for differential operators is presented in [27]. However, it is limited to structured 1D and 2D domains and realized by a fully connected network, thus not directly applicable to 3D meshes.

A fundamental challenge in machine learning and pattern recognition lies in developing effective representations for complex data. The named TASK B consists of applying G-GNN with smoothing GGC (S-GGC, 1 layer) to provide a shape preserving low-dimensional representation for data lying on a high-dimensional space. This is achieved by reconstructing the object shape using a small number of well-representative eigenfunctions. It is indeed well known that using a reduced number of eigenfunctions corresponding to the smallest eigenvalues allows the approximation of the object shape in an improved manner as the number of eigenfunctions increases. Instead of the first N_e eigenfunctions the proposed G-GNN will automatically select the “best” N_e eigenfunctions, in the sense of best fitting of the given shape which includes also local geometry of the input surface. To evaluate the reconstruction accuracy we consider the SR error defined in (25), which measures the mean square error between the original and the mesh reconstructed through a low-dimension eigenbasis composed by N_e approximated eigenfunctions. Drawing on the connection between the graph Laplacian and the Laplace–Beltrami operator on manifolds, several methods have been proposed for representing high-dimensional data using eigendecomposition [28,29]. However, the computed Laplacian eigenfunctions, regardless of whether they are calculated via their connection to the heat equation or through variational methods, always correspond to the smallest eigenvalues. These first eigenfunctions

(associated with the smallest eigenvalues) do not capture the most significant features that represent the shape and details of an object.

In TASK C, a G-GNN with sparsity GGC (p-GGC, 1 layer) is proposed to approximate $N_e \ll n_v$ eigenfunctions of the L_p -Compressed Mode eigendecomposition (Ψ, Λ_p) of the graph-Laplacian L associated with a given mesh X , defined by the optimization problem (24). We will show how the parameter $0 < p < 1$ offers a useful control on the width of the compact support of the eigenfunctions, as theoretically established in [22]. Unlike the variational method proposed in [22], which requires the tuning of several model parameters, the proposed G-GNN is automatic and more efficient in producing TASK C.

The objective of TASK D is to evaluate the performances of a G-GNN endowed with smoothing GGC (S-GGC, 1 layer) in providing a new intrinsic feature embedding of X - as input to neural field networks [30]. In [5], the authors introduced the Intrinsic Neural Field (INF), a neural field $\mathcal{F}_\theta : X \rightarrow \mathbb{R}^{n_{out}}$ for the solution of inverse problems, that combines the advantages of neural fields with the intrinsic shape representation properties of the Laplace–Beltrami operator. The idea behind the INFs is to replace the coordinate-based input of the MLP neural network f_θ with a feature embedding $\gamma : X \rightarrow \mathbb{R}^{n_f}$ so that

$$\mathcal{F}_\theta(v) = (f_\theta \cdot \gamma)(v) = f_\theta(\gamma_1(v), \gamma_2(v), \dots, \gamma_{n_f}(v)).$$

The choice proposed in [5] of taking the first n_f eigenfunctions Φ of the LBO spectrum as feature embedding γ allowed to overcome the difficulty of coordinate-based neural fields of learning high-frequency functions, so-called “spectral bias” phenomenon, which makes them poorly suited for vision and graphics tasks. However, the weakness of this LBO feature embedding is that it requires to compute all the n_v eigenfunctions by classical numerical methods and then ad hoc select the most representative n_f eigenfunctions for the given manifold, which do not correspond to the first n_f eigenfunctions. We aim to overcome this problem by computing a shape-aware LBO feature embedding $\gamma(v)$ by means of a parametric G-GNN.

Tasks A–D are performed in a single instance modality, which means that a separate G-GNN is optimized for each object.

Finally, in TASK E, we validate the G-GNN framework for the solution of a classic surface processing problem as the 3D mesh denoising which consists of removing noise from corrupted 3D meshes while preserving existing geometric features. At this aim, the G-GNN architecture consists of several E-GGC layers, and the samples of the training set consist of the same input mesh X , perturbed with different noise levels, which involves a unique graph Laplacian L matrix. The noisy meshes $\tilde{\mathcal{V}}$ have been synthetically corrupted in the normal direction n following the degradation model

$$\tilde{\mathcal{V}} = \mathcal{V} + V n, \quad (58)$$

where $V \in \mathbb{R}^{n_v \times 3}$ is Gaussian noise distributed with zero mean and standard deviation $\sigma = \gamma_n \bar{l}_e$ where \bar{l}_e is the average edge length and $\gamma_n \geq 0$ represents the noise level. For a detailed discussion on mesh denoising approaches we refer the reader to [31]. Among the various mesh denoising methods investigated, the variational methods were found to be particularly effective because they can well preserve sharp features while suppressing noise significantly [32,33]. However, they involve the tuning of several regularization parameters, which leads them to be neither efficient nor automatic. For what concerns the methods based on GNN, interesting strategies have been presented in [8–10,34]. All of them apply the GNN only for the solution of a sub-task of the entire denoising pipeline. In [8,10] after a pre-processing, a GNN is applied to predict a noise-reduced normal field and then a post-processing phase produces a noise-reduced mesh via vertex updating normal fields. In [9] the learning is both on the face normal and the vertex dual domain of 3D mesh. In [34] a Laplacian smoothing is applied to the input noised mesh, and then a GNN learns the vertex displacement to be added to the smoothed mesh to obtain the output. Using a naïve realization of the G-GNN proposal we will show how to use it for an effective mesh denoiser task that could be further improved

Table 2
Mesh datasets with n_v vertices n_e edges and n_t triangles.

Mesh	n_v	n_e	n_t
bunny1	152	450	300
bunny2	2503	7503	5001
cat	35 290	70 576	35 288
teddy	502	1500	1000
horse	152	194	300
hand	2500	7500	5000
bimba	7478	22 428	14 952
dragon	9602	28 800	19 200
fertility	8799	26 415	17 610

by exploiting a suitable mixture of the GGC layers, using the ad hoc energies proposed.

Section 8 will delve into the details of the conducted tasks and present the results, which serve as evidence for the effectiveness of the proposed G-GNN.

7. Loss function and implementation details

For the addressed problems (TASKs A-E) the G-GNN weight parameters

$$\Theta = \{\theta_i, \theta_1, \theta_2, \theta_3, \dots, \theta_M, \theta_f\}$$

are obtained by minimizing the following loss function

$$\mathcal{L}(\Theta; \hat{x}) = \gamma_1 \mathcal{L}_1(\Theta; \hat{x}) + \gamma_2 \mathcal{L}_2(\Theta; \hat{x}) + \mathcal{L}_3(\Theta; \hat{x}) \quad (59)$$

where $\hat{x} \in \mathbb{R}^{n_v \times n_{out}}$ is the G-GNN output as indicated in Fig. 1 and γ_1, γ_2 are hyperparameters which balance the loss terms and are tuned by a rough grid search. The three components of the multiple loss (59) are defined as follows

- $\mathcal{L}_1(\Theta; \hat{x}) = \|P\hat{x}\|_F^2$, where $(P\hat{x})_j := L(\hat{x}^j) + R(L, \hat{x}^j)\hat{x}^j$ with $\hat{x} \in \mathbb{R}^{n_v \times N_e}$, \hat{x}^j denoting the j th column of \hat{x} , and $R(\cdot)$ is the Rayleigh quotient defined in (20). \mathcal{L}_1 penalizes deviations from the eigenvalues of L .
- $\mathcal{L}_2(\Theta; \hat{x}) = \|\hat{x}^T \hat{x} - I\|_F^2$, pushes the orthonormality of distinct eigenfunctions of L .
- $\mathcal{L}_3(\Theta; \hat{x}) = \|\hat{\mathcal{V}} - \mathcal{V}\|_F^2$, where, according to (26), $\hat{\mathcal{V}} = \hat{x}\hat{x}^T \mathcal{V}$ with $\hat{x} \in \mathbb{R}^{n_v \times N_e}$ for TASKs A-D, while $\hat{\mathcal{V}} = \hat{x}$ with $\hat{x} \in \mathbb{R}^{n_v \times 3}$ and $\mathcal{V} = \hat{\mathcal{V}}$ noisy mesh following (58), for TASK E. It forces the 3D shape reconstruction produced by the G-GNN towards the given shape \mathcal{V} .

To avoid scale differences, a preliminary rescaling has been performed to the set of vertices of each input mesh, to fit the mesh into a bounding box centered at the origin, with edges of unit length.

We performed the associated network training and inference using a machine equipped with an Nvidia Tesla V100 GPU and leveraging Python 3.10.9, Pytorch 1.13.1, and Pytorch Geometric 2.3.0. The training time depends on two factors: the dimensionality of the mesh (n_v) and the number of epochs the G-GNN is trained for. Interestingly, the specific task itself did not seem to significantly impact the training duration in our observations. In particular, for all meshes except the cat mesh, we have run the G-GNN for a number of epochs between 3000 and 6000, having a duration of training lower than 15 min. When N_e is small, such as 8 or 15, training takes even less than 5 min. Instead, we need approximately 10,000 epochs to obtain good results for the cat mesh in each of the tasks it is involved with a training time lower than 45 min.

The ENC block in (28) employs the *LeakyReLU*, as activation function, with default *negative - slope* = 0.01 with He weight initialization from a uniform distribution [35], while neither the GGC layers nor the DEC block employ any activation function and leverage the Xavier weight initialization [36]. We used the Adam optimizer with default

parameters. In the experimental results, the hyperparameter K of the GGC layer kernel, which represents the number of iterations for the computation of the GGC operator (see Propositions 5.1–5.6) is set to be $K < 10$.

8. Experimental results

In this section, in order to show the effectiveness of the G-GNN proposal, we show the results obtained in the various TASKs A-E described in Section 6.

Details on the polygonal meshes used in the reported experiments are summarized in Table 2.

8.1. Task A: LBO eigendecomposition approximation

The objective of this task is to approximate N_e eigenfunctions of the eigendecomposition (Φ, Λ) of the discrete Laplacian associated with the cat mesh, see Table 2.

The G-GNN with one parametric S-GGC layer (defined as in 5.2) is applied in single instance modality, to compute $N_e = 15$ features \hat{x} which approximate the eigenfunctions $\Phi_{15} \in \mathbb{R}^{n_v \times 15}$. In particular, according to (37) in Proposition 5.2, the number of features are exactly the number of eigenfunctions $n_f = N_e$, neglecting y .

The computed eigenfunctions are illustrated in Fig. 3, by using false colors on the mesh domain, blue for negative, and red for positive values. For what concerns the accuracy errors defined in (57) and (56), we obtain $ER = 3.27 \times 10^{-7}$ and $EOR = 2.83 \times 10^{-5}$.

The corresponding eigenvalues, computed by leveraging the Rayleigh quotient defined in (20), are shown in Fig. 2 as red diamonds, while the blue dots indicate the first 15 eigenvalues of the discrete LBO, sorted in ascending order, computed using standard Lancsoz bidiagonalization iterative method [37,38]. We notice how the 15 eigenfunctions carried out by the G-GNN over the 35,290 total eigenfunctions which characterize the cat mesh, are associated with eigenvalues of varying magnitude corresponding to both low frequencies (small values) and higher frequencies (large values) associated with those essential global and local geometry details such that the G-GNN retrieves a good shape reconstruction.

However, although we enforce the model using a tailor-made loss function to preserve orthogonality and eigendecomposition, a limitation of the proposed approach is that this cannot be explicitly guaranteed. Inevitably, the G-GNN produces features that only approximate the eigenfunctions, being however extremely useful as an intrinsic shape paradigm in many geometry processing tasks.

8.2. Task B: 3D shape reconstruction

This task aims to reconstruct the object shape by using a small number of eigenfunctions, N_e , from the eigendecomposition (Φ, Λ) of the discrete Laplacian associated with the given object mesh.

The G-GNN with one parametric S-GGC layer is applied in single instance modality, to compute the features $\hat{x} \in \mathbb{R}^{n_v \times N_e}$ which approximate the eigenfunctions, and then reconstruct the mesh through $\hat{\mathcal{V}} = \hat{x}\hat{x}^T \mathcal{V}$, see (25). We report in Table 4 some reconstructions of the mesh models, bunny1, horse and teddy, obtained by progressively increasing the number of spectral coefficients, i.e., by increasing N_e from the second row to the fifth row: $N_e = 8$, $N_e = 15$, $N_e = 50\%n_v$, and $N_e = n_v$, with n_v reported in Table 2. Each column shows the reconstructions obtained using the G-GNN with one parametric S-GGC layer, on the left, while on the right the reconstructions achieved by solving the variational eigendecomposition in (23), using simple algebraic standard approach implemented in MATLAB. The *Shape Recovery* error SR, defined in (25), is reported below the reconstructed meshes, distinguishing with SR_L the reconstruction errors due to the application of the eigendecomposition in (23). As expected, the visual quality of the reconstructions increases as the N_e increases, whereas

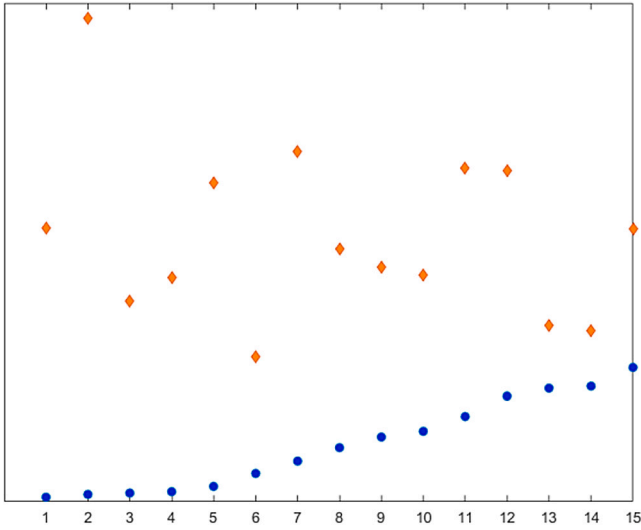


Fig. 2. TASK A - First 15 eigenvalues of the graph Laplacian associated with the cat mesh computed by the algorithm in [38] (blue dots, $\lambda_i \in [10^{-4}, 4 \times 10^{-3}]$) and by the G-GNN network (red diamonds, $\lambda_i \in [4 \times 10^{-3}, 10^{-2}]$).

Table 3

TASK B - Comparison of the errors for the reconstruction of the mesh bunny1 using $N_e = 152$ eigenfunctions applying G-GNN with a parametric and a non-parametric S-GGC layer.

	SR	ER	EOR
Non parametric S-GGC	9.84×10^{-05}	1.94×10^{-03}	6.57×10^{-03}
Parametric S-GGC	1.11×10^{-04}	2.20×10^{-03}	9.46×10^{-05}

the SR error decreases, for both the reconstruction methods. However, the G-GNN-based reconstructions exhibit more shape details than those obtained through the variational eigendecomposition, even for very small values of N_e .

In Table 6, we show the reconstructions of the cat mesh using an increasing set of eigenfunctions, $N_e = \{5, 15, 25, 35\}$ out of $n_v = 35290$. The quality of the reconstructions can be evaluated by the SR values reported below each mesh. In Table 6 last column, we report also the reconstructions obtained by the first N_e eigenfunctions which are associated to the N_e smallest eigenvalues. The obtained poor reconstructions only capture the skeleton of the object. A lot more eigenfunctions are needed to improve the accuracy of the reconstruction and capture characteristic shape details.

In Table 3 we report a comparison between a G-GNN with a non-parametric S-GGC layer and with a parametric S-GGC layer in computing all the eigenfunctions of the graph Laplacian associated to the mesh bunny1. The SR and ER errors are quite similar, while the parametric S-GGC layer allows for a lower EOR error with respect to the non-parametric S-GGC. This can be attributed to the weights θ_1 in (38) which allow to better induce the orthogonality of the computed eigenfunctions.

8.3. Task C: LBO L_p compressed mode (CM) approximation

The objective of this task is to approximate $N_e = 50$ eigenfunctions of the L_p -Compressed Mode eigendecomposition (Ψ, Λ_p) of the discrete Laplacian associated with bunny2 and hand meshes. The G-GNN with one parametric p-GGC is applied in single instance modality with several features $n_f = N_e$, where we set $p = 0.1$ for bunny2 mesh and $p = 0.5$ for hand mesh. Table 5 reports, in the first two columns, the original meshes and their reconstructions $M^{(50)}$ according to (26), and in the remaining columns 4 out of the 50 eigenfunctions for each mesh. To highlight the local support of the eigenfunctions, a thresholded

colormap is used: eigenfunctions' values below and above the threshold have been gray-colored. We observe that the highest (red) and lowest (blue) values of the eigenfunctions are localized on small portions on the surface of the whole object, in regions characterizing the shape of the object.

The parametric version of the G-GNN with the p-GGC layer is preferred with respect to the non-parametric p-GGC layer. Even if we get qualitatively similar reconstructions, the results from parametric p-GGC layer present a lower error ($EOR = 8.93 \times 10^{-5}$) than the results obtained by the non-parametric p-GGC layer ($EOR = 1.12 \times 10^{-3}$).

The shape reconstruction results shown in Table 5 represent the best approximations of shape in a low-dimensional space using L_p -CM eigenfunctions obtained by the G-GNN with one p-GNN layer. To measure the performance of the proposed G-GNN, we reconstructed the first $N_e = 50$ L_p -CM eigenfunctions using the variational formulation of the problem (24), using the Alternating Direction Method of Multipliers algorithm for the numerical minimization, as described in [22]. The reconstructed meshes are shown in gray color in Table 5 second column. They are noticeably smoother and lack details, as we expected since they are the first and not the best, so qualitatively the resulting low-dimensional reconstructions are much worse, endorsed by $SR_{L_p} = 2 \times 10^{-3}$ for bunny and $SR_{L_p} = 3 \times 10^{-2}$ for hand.

8.4. Task D: Intrinsic representation

The objective of this task is to evaluate the performances of a G-GNN equipped with an S-GGC operator, to provide a shape-aware feature embedding useful for feeding intrinsic neural fields.

In [5] the INFs have been successfully applied to several graphics applications, including the texture mapping task for a large-scale cat mesh. Such a reconstruction consists of recovering the textured image of the cat given a limited set of posed images. As detailed in [5], the LBO feature embedding is obtained by computing the first 4096 eigenfunctions of the eigendecomposition (Φ, Λ) of the mesh Laplacian by solving the generalized eigenvalue problem (21) using the algorithm in [38]. Then, an ad hoc manual screening is performed to select only 1024 out of the 4096 computed eigenfunctions (specifically the intervals 1 to 256, 1794 to 2304, 3841 to 4096) to include eigenfunctions associated with information at varying scales, from macroscopic to microscopic details of the mesh.

We propose an automatic procedure for the selection of the most representative eigenfunctions by employing a G-GNN which provides a certain number of approximated eigenfunctions that enable high-fidelity representations and reconstructions.

In Table 6, we show some texture reconstructions along with the associated 3D shape reconstructions, obtained for $N_e = \{5, 15, 25, 35\}$. In Table 6, left panel reports reconstructions obtained by employing G-GNN, while the right panel illustrates reconstructions obtained by computing the first N_e eigenfunctions of the graph-Laplacian $L = \Phi^T \Lambda \Phi$. After fitting the intrinsic neural field to the data, the texture images generated from 200 novel viewpoints have been compared with the ground-truth images for evaluation. The quality of the resulting texture images has been assessed by the Peak Signal-to-Noise Ratio (PSNR) measure and reported below the corresponding posed image.

We observe that for a given N_e , G-GNN allows not only a better PSNR in the textured images but also lower reconstruction errors, even for a small set of eigenfunctions N_e , such as 5 or 15, out of a total of 35,290. This provides an automatic shape-aware LBO feature embedding, essential to be able to apply the promising INFs with large-scale meshes.

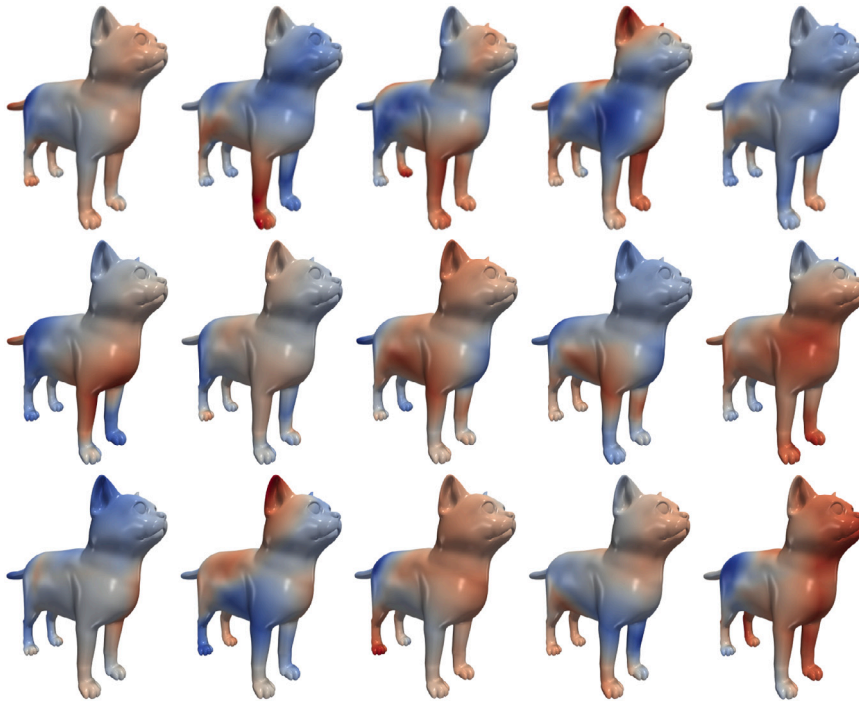


Fig. 3. TASK A - approximation of $N_e = 15$ eigenfunctions of the graph Laplacian associated to the cat mesh by applying a parametric smoothing GGC layer, shown as false colored scalar field onto the original mesh.

8.5. Task E: mesh denoising

We present a vanilla unsupervised learning-based mesh denoising method obtained by applying a G-GNN with non-parametric E-GGC operators, with tuning of the parameter $p > 1$.

We compared the G-GNN proposal with the GNN-based method in [34], named DMP, using the open source code kindly provided by the authors at <https://github.com/astaka-pe/DeepMeshPrior>, [and with the TGV method based on Total Generalized Variation for Denoising, named TGV, [33] using the executable code kindly provided by the authors at <https://github.com/LabZhengLiu/MeshTGV>.

The proposed G-GNN consists of six non parametric E-GGC layers. Our preliminary experiments showed that using more layers did not necessarily lead to improved results, particularly when applying the same GGC operators to each layer. This aligns with the general observation that GNN architectures often do not benefit significantly from a large number of layers. In fact, a relatively small number of layers (less than 10) can often be sufficient to achieve good accuracy.

We set the hyperparameters $\gamma_1 = \gamma_2 = 0$ in the loss function (59) for this task, where $\hat{\mathcal{V}}$ is the set of denoised vertices of the mesh and \mathcal{V} is the set of corrupted vertices of the mesh. We applied the G-GNN both in single instance modality (the network takes a single noisy mesh as input data and directly outputs the denoised mesh without being trained) and in multiple instances, using 100 meshes for training and 20 for testing. The total number of training meshes is constructed by adding a different level of noise to the original uncorrupted meshes, under the degradation model (58).

In Fig. 4 we illustrate the results for a few meshes from the dataset: noise-free meshes in the first row together with their perturbed meshes in the second row. The denoised meshes, obtained by G-GNN in single instance modality with E-GGC layers, are reported, for fixed $\lambda = 100$ value, in the third row for $p = 1.1$ and in the fourth row for $p = 1.8$. For comparison, the sixth and seventh rows illustrate the denoised meshes obtained using the single-instance DMP method proposed in [34] and the TGV method proposed in [33], respectively. From a visual inspection and the SR errors reported below each mesh, we observe that the G-GNN reconstructions are more accurate and better

preserve the shape details. The single instance modality avoids the training phase but requires a high number of epochs in order to achieve an acceptable reconstruction accuracy. In case the denoise procedure must be repeated for different corruptions of the same mesh, multiple instance modality is preferred, which allows for a fast inference phase after appropriate training with different corruptions. In Fig. 4, fifth row, we show the denoised meshes obtained with G-GNN in multiple instances with $p = 1.8$ running for 50 epochs in the training phase, instead of 2000 epochs needed in the single instance case. The proposed unsupervised approach is fast and accurate, and it can be trained on noisy data, without explicit correspondence between noisy and ground-truth meshes and without knowledge of the noise level of corruption. Future work will consider including p-GGC layers in the G-GNN vanilla architecture here considered to face noise removal on objects with sharp features.

9. Conclusion

We introduced Generalized Graph Convolutions in Graph Neural Networks which are energy-driven graph convolutions derived from the solution of different graph Laplacian-based variational regularization problems. A selected set of possible energy-driven graph convolutions have been presented, many more can be designed in future work. In particular, we proposed both parametric and non-parametric forms of S-GGC, which induces a smoothing behavior over the features, p-GGC, to preserve sharpening, and E-GGC, to control enhancing by a real positive parameter p .

We demonstrated their processing power successfully in several graphics applications in geometry processing. Generalized GNNs have been applied to 3D mesh denoising, shape reconstruction, and eigen-decomposition, which is particularly useful to determine the ‘‘Shape-DNA’’ of an object. We additionally employed a G-GNN as a new shape-aware feature embedding to be used in intrinsic neural fields. The presented results are promising and show how the proposed approach brings convenient synergies between the energy dictated by the application context and the adaptation to specific data granted by the learning approach.

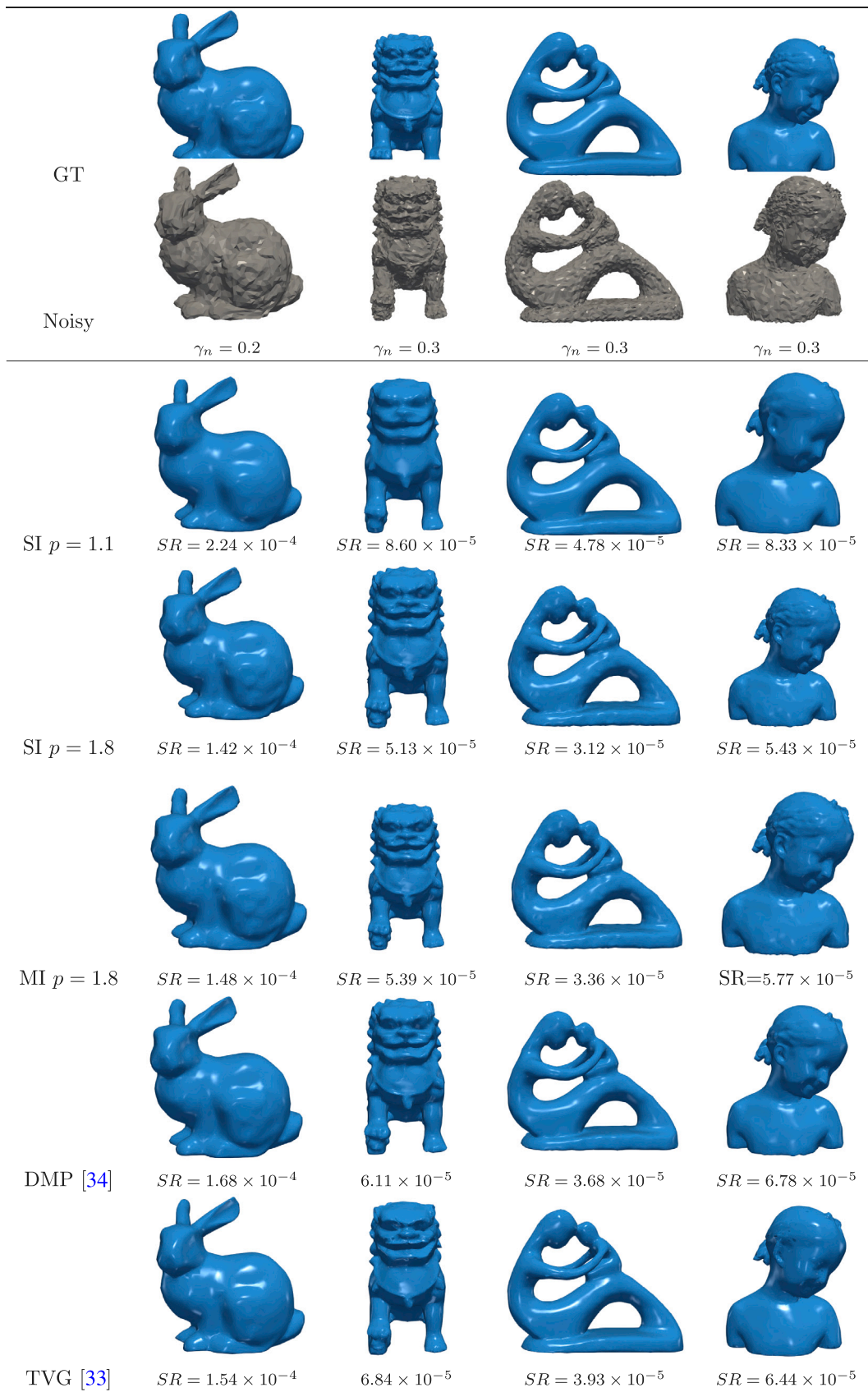




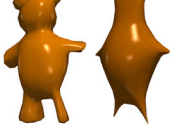

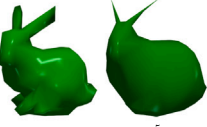


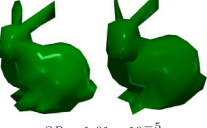


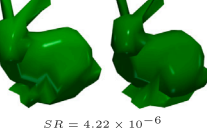
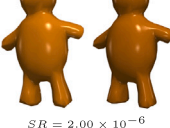



Fig. 4. Task E - mesh denoiser on synthetically perturbed meshes (second row). Comparison among single instance (SI) G-GNN denoised meshes with $p = 1.1$ (third row) and $p = 1.8$ (fourth row), multiple instance (MI) G-GNN denoised meshes with $p = 1.8$ (fifth row), DMP denoised meshes (sixth row), and TVG denoised meshes (seventh row).

Table 4

TASK B - 3D shape reconstructions of meshes described in Table 2 obtained using an increasing number of eigenfunctions, N_e . In each column, on the left, the reconstructions obtained using the G-GNN with one parametric S-GGC layer, on the right, those obtained using the variational eigendecomposition in (23). The SR errors are reported below the reconstructed meshes, denoting by SR_L the reconstruction error for the eigendecomposition (23).

	bunny1	teddy	horse
GT			
N_e 8	 $SR = 4.33 \times 10^{-4}$ $SR_L = 1.01 \times 10^{-1}$	 $SR = 1.45 \times 10^{-3}$ $SR_L = 9.45 \times 10^{-1}$	 $SR = 3.11 \times 10^{-3}$ $SR_L = 1.94 \times 10^{-1}$
N_e 15	 $SR = 6.59 \times 10^{-5}$ $SR_L = 8.60 \times 10^{-2}$	 $SR = 1.80 \times 10^{-4}$ $SR_L = 8.52 \times 10^{-2}$	 $SR = 1.80 \times 10^{-4}$ $SR_L = 8.52 \times 10^{-2}$
N_e 50%	 $SR = 1.01 \times 10^{-5}$ $SR_L = 2.94 \times 10^{-4}$	 $SR = 1.14 \times 10^{-5}$ $SR_L = 1.04 \times 10^{-3}$	 $SR = 1.94 \times 10^{-5}$ $SR_L = 4.1 \times 10^{-3}$
N_e 100%	 $SR = 4.22 \times 10^{-6}$ $SR_L = 6.45 \times 10^{-6}$	 $SR = 2.00 \times 10^{-6}$ $SR_L = 1.11 \times 10^{-5}$	 $SR = 4.22 \times 10^{-6}$ $SR_L = 6.25 \times 10^{-6}$

Future work will include a network adaptation to handle \tilde{L} instead of L , to encompass the area diagonal weight matrix. We expect to be able to produce more accurate reconstructions while penalizing the efficiency. Another future direction will consider the enlargement of the training set by including several poses of the same mesh, which, sharing the same graph-Laplacian L , could contribute towards an improvement of the network performance. Furthermore, it is of interest to construct G-GNN using a mixture of GGC layers derived from different energy terms focusing on different features.

CRedit authorship contribution statement

D. Lazzaro: Writing – original draft, Validation, Software, Methodology. **S. Morigi:** Writing – original draft, Supervision, Methodology, Formal analysis. **P. Zuzolo:** Writing – original draft, Software, Methodology, Formal analysis.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

This work was supported in part by the National Group for Scientific Computation (GNCS-INDAM), Research Projects 2024. This study was carried out within the MICS (Made in Italy – Circular and Sustainable) Extended Partnership and received funding from the European Union Next-GenerationEU (PNRR) – D.D. 1551.11-10-2022, PE00000004. The research of PZ has been funded by PNRR CN-HPC, which is funded by the European Commission under the NextGeneration EU program CUP J33C22001170001. The work of SM and DL was in part supported by PRIN2022_MORIGI, titled “Inverse Problems in the Imaging Sciences (IPIS)” 2022 ANC8HL - CUP J53D23003670006, and PRIN2022_PNRR_CRESCENTINI CUP J53D23014080001.

Appendix

Proof of Proposition 5.1

Proof. The cost function in the optimization problem (33) is strictly convex, bounded from below, and continuous. Hence by standard arguments in convex analysis the optimization problem (33) has a unique solution.

Table 5

TASK C - Four out of the $N_e = 50$ eigenfunctions of the L_p -compressed Mode eigendecomposition (Ψ, A_p) of the discrete Laplacian associated with bunny2 mesh — first row $p = 0.1$ — and hand mesh — second row $p = 0.5$. In the first column the original meshes are reported, in the second column the reconstructed L_p -CM meshes using G-GNN with p-GGC layer – in green – and the variational approach in [22] — in gray.

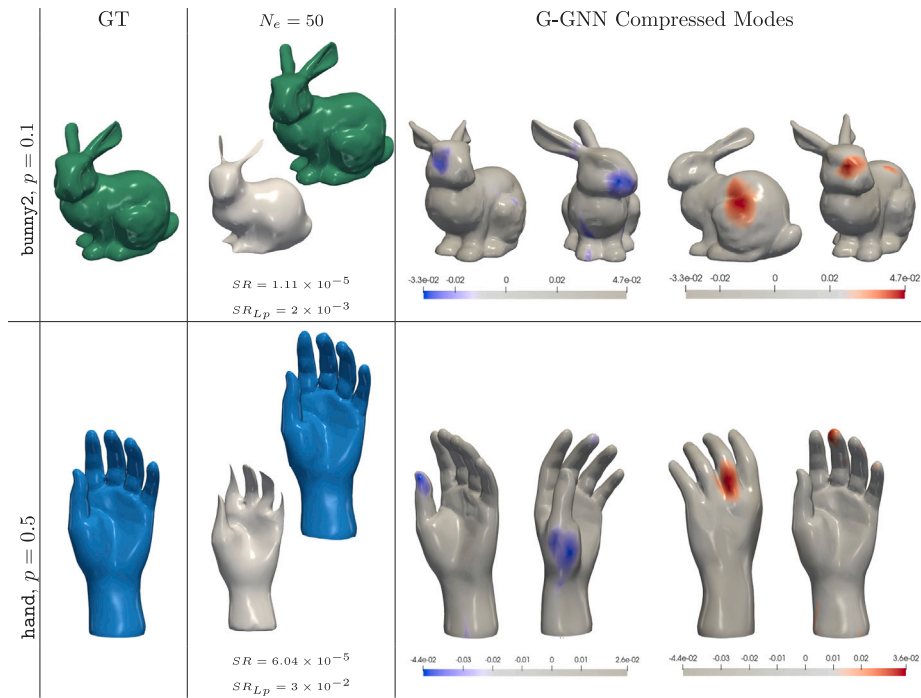


Table 6

TASK D - textured images obtained by using the shape-aware feature embedding provided by G-GNN (left panel), and by LBO feature embedding (right panel). Each row represents a different number of eigenfunctions N_e involved. The last columns of each panel show the object reconstructions obtained for each subset of eigenfunctions.

N_e	G-GNN			INF [5]		
5						
15						
25						
35						

Since the energy in (33) is differentiable, we apply the gradient descent method for its minimization. Considering that the partial gradient of $E(x)$ with respect to x is given by

$$\frac{\partial E}{\partial x}(x) = \frac{\partial \text{tr}(x^T Lx)}{\partial x} + \lambda x = 2Lx + \lambda x, \quad (\text{A.1})$$

then the iteration $(k+1)$ th of the gradient descent method with step-size $\alpha > 0$ updates x^k as follows

$$x^{k+1} = x^k - \alpha(2Lx^k + \lambda x^k) = (I - \alpha 2L - \lambda \alpha)x^k, \quad (\text{A.2})$$

where x^{k+1} converges to the minimizer of (33) when $0 < \alpha < 2/L_E$, with $L_E = \lambda + 2\|L\|_2$ the Lipschitz constant of the energy $E(x)$. \square

Proof of Proposition 5.2

Proof. The energy in (38) is differentiable,

$$\frac{\partial E_{\theta}}{\partial x}(x) = \frac{1}{2} \frac{\partial \|(x-y)W_1\|^2}{\partial x} + \frac{\partial E_{\theta}^{\text{Dir}}(x)}{\partial x} = \lambda(x-y)\theta_1 + 2Lx\theta_2. \quad (\text{A.3})$$

In fact

$$\frac{\partial \|(x-y)W_1\|_F^2}{\partial x} = \frac{\partial \text{tr}(W_1^T(x-y)^T(x-y)W_1)}{\partial x} \quad (\text{A.4})$$

$$= \frac{\partial \text{tr}(W_1^T(x)^T x W_1)}{\partial x} - 2 \frac{\partial \text{tr}(W_1^T y^T x W_1)}{\partial x} \quad (\text{A.5})$$

$$= 2xW_1W_1^T - 2yW_1W_1^T = 2(x-y)\theta_1$$

$$\frac{\partial E_{\theta}^{\text{Dir}}(x)}{\partial x} = \frac{\partial \text{tr}(W_2^T x^T Lx W_2)}{\partial x} = 2LxW_2W_2^T = 2Lx\theta_2. \quad (\text{A.6})$$

Then we can apply the gradient descent method which reads as (37) and converges to a minimizer of functional energy (38) under standard conditions on the α step-size. \square

Proof of Proposition 5.3

Proof. Let $f(x) = \text{tr}(x^T Lx)$ be a differentiable function with $\frac{\partial \text{tr}(x^T Lx)}{\partial x} = 2Lx$, and $g(x) = \lambda\|x\|_p^p$ be a non-differentiable, for $p \leq 1$, and non-convex function, for $p < 1$. Then the minimization of the cost function $f(x) + g(x)$ can be obtained by the proximal gradient algorithm, which iterates over n as

$$z^n = x^n - 2\alpha Lx^n \quad (\text{A.7})$$

$$x^{n+1} = \arg \min_{x \in \mathbb{R}^{n_v \times n_f}} \left\{ \frac{1}{2} \|x - z^n\|^2 + \alpha \lambda \|x\|_p^p \right\}, \quad (\text{A.8})$$

where the parameter $\alpha > 0$ is the step-size. The cost function in (A.8) is separable and thus solvable by applying the generalized soft-thresholding iterative algorithm to the solution of the following $n_v \times n_f$ independent ℓ_p -norm minimization problems

$$s_{il}^* = \arg \min_{s \in \mathbb{R}} \left\{ h(s) := \frac{1}{2}(x - z_{il})^2 + \alpha \lambda |s|^p \right\}. \quad (\text{A.9})$$

For any $z_{il} \in (\hat{s}(\alpha\lambda), +\infty)$, $h(s)$ has one unique minimum $x_{il}^{n+1} = s_{il}^*$ which can be obtained for each component by solving $\bar{s}_{il}^* = T_p^{GST}(z_{il}; \alpha\lambda)$, through (39)–(41).

Under standard assumptions on the step-size α the proximal gradient algorithm converges to a local minimizer, see [39]. \square

Proof of Proposition 5.4

Proof. Consider $f(x) = \text{tr}(W_2^T x^T Lx W_2)$ differentiable and $g(x) = \|xW_1\|^p$ not differentiable, for $p < 1$. Then the minimization of the cost function $f(x) + g(x)$ is obtained by the proximal gradient algorithm, which iterates over n as

$$z^n = x^n - 2\alpha Lx^n \theta_2, \quad (\text{A.10})$$

$$x^{n+1} = \arg \min_{x \in \mathbb{R}^{n_v \times n_f}} \left\{ \frac{1}{2} \|x - z^n\|^2 + \alpha \lambda \|xW_1\|_p^p \right\}. \quad (\text{A.11})$$

The functional in (A.11) is separable and therefore we can solve $n_v \times n_f$ independent ℓ_p -norm minimization scalar problems

$$s_{il}^* = \arg \min_{s \in \mathbb{R}} \left\{ h(s) := \frac{1}{2}(s - z_{il})^2 + \alpha \lambda |w_l|^p |s|^p \right\} \quad (\text{A.12})$$

where the weights w_l define the matrix W_1 and are constant for each feature. Leveraging the Generalized Iterative Soft Thresholding [25] operator T_p^{GST} , defined in (39)–(41), the solution of (A.12) read as

$$\bar{s}_{il}^* = T_p^{GST}(z_{il}; \alpha \lambda |w_l|^p).$$

For any $z_{il} \in (\hat{s}(\alpha \lambda |w_l|^p), +\infty)$, $h(s)$ has one unique minimum which can be obtained by solving the nonlinear Eq. (41). Under the standard assumptions $0 < \alpha < 1/(\|L\|_2 \|\theta_2\|_2)$, the proximal gradient algorithm converges to a local minimizer, see [39]. \square

Proof of Proposition 5.5

Proof. Since $p > 1$, the functional $E(x)$ in (51) is differentiable and convex, the first-order optimality conditions for its minimization read as

$$\frac{\partial E(x)}{\partial x} = \frac{1}{p} \frac{\partial E^{TV_p}(x)}{\partial x} + \lambda E^{fid}(x; y) = \Delta_p(x) + \lambda(x - y) = 0. \quad (\text{A.13})$$

The mesh p -Laplacian $\Delta_p(f)$ is defined in (13) for scalar functions f , and can be defined for $x \in \mathbb{R}^{n_v \times n_f}$ as a matrix of dimension $n_v \times n_f$ with i th row $(\Delta_p(x))_i \in \mathbb{R}^{n_f}$ (which corresponds to vertex v_i)

$$\begin{aligned} (\Delta_p(x))_i &= \sum_{(i,j) \in \mathcal{E}} \sqrt{w_{i,j}} \|(\nabla_X x)_{i,j}\|_F^{p-2} (\nabla_X x)_{i,j} \\ &= \sum_{(i,j) \in \mathcal{E}} \| \sqrt{w_{i,j}} (x_i - x_j) \|_F^{p-2} w_{i,j} (x_i - x_j). \end{aligned}$$

For each node i , the nonlinear problem (A.13) is separable for all its features, thus we have

$$\sum_{(i,j) \in \mathcal{E}} \| \sqrt{w_{i,j}} (x_i - x_j) \|_F^{p-2} w_{i,j} (x_i - x_j) + \lambda x_i = \lambda y_i. \quad (\text{A.14})$$

The nonlinear Eq. (A.14) can be linearized and then solved for x_i by applying the following iterative scheme over k :

$$\left(\sum_{(i,j) \in \mathcal{E}} w_{i,j}^{\frac{p}{2}} \|x_i^k - x_j^k\|^{p-2} + \lambda \right) x_i^{k+1} = \sum_{(i,j) \in \mathcal{E}} w_{i,j}^{\frac{p}{2}} \|x_i^k - x_j^k\|^{p-2} x_j^k + \lambda y_i. \quad (\text{A.15})$$

Let $M \in \mathbb{R}^{n_v \times n_v}$ and $D = \text{diag}(d_1, \dots, d_{n_v})$ be matrices with elements defined in (50), then Eq. (A.15) has the explicit solution form

$$x_i^{k+1} = d_i^{-1} \left[\sum_{(i,j) \in \mathcal{E}} M_{i,j}^k x_j^k + \lambda y_i \right], \quad (\text{A.16})$$

which takes the more compact form (49). The linearization strategy here adopted extends the result for scalar functions in [40] which proves the convergence to the minimizer of the scalar analogous to the cost function (51). \square

Proof of Proposition 5.6

Proof. The energy functional $E_{\theta}(x)$ with $p > 1$ is convex and differentiable, and we have

$$\frac{\partial E_{\theta}(x)}{\partial x} = \frac{1}{p} \frac{\partial E_{\theta}^{TV_p}(x)}{\partial x} + \lambda(x - y)W_1W_1^T = \Delta_p(x)\theta_2 + \lambda(x - y)\theta_1, \quad (\text{A.17})$$

where for each node i th the weighted p -Laplacian is separable for each feature

$$\frac{\partial E_{\theta}^{TV_p}(x)}{\partial x_i} = p \sum_{j=1}^{n_v} \|\sqrt{w_{i,j}}(x_i - x_j)\|^{p-2} w_{i,j} (x_i - x_j) |W_2|^p \quad (\text{A.18})$$

$$= p [A_p(x_i) |(\theta_2)_1|^p, \dots, A_p(x_i) |(\theta_2)_{n_f}|^p]. \quad (\text{A.19})$$

By imposing the optimality conditions for the minimization of the energy $E_{\theta}(x)$, the global minimizer can be obtained by solving

$$A_p(x)\theta_2 + \lambda(x - y)\theta_1 = 0. \quad (\text{A.20})$$

Let M and $D = \text{diag}(d_i)$ be the matrices defined as in (54), then (A.20) can be rewritten as

$$xD\theta_2 - Mx\theta_2 + \lambda x\theta_1 - \lambda y\theta_1 = 0. \quad (\text{A.21})$$

An approximation for x in (A.21) from an initial x^0 is obtained by iterating over k as follows

$$x^{k+1}(D\theta_2 + \lambda\theta_1) = Mx^k\theta_2 + \lambda y\theta_1, \quad (\text{A.22})$$

where the second term of (A.21) is considered at the previous iteration k . Then (52) follows. The linearization strategy here adopted for the nonlinear operator p -Laplacian extends the result for scalar functions in [40] and introduces the weighted p -Laplacian; as in [40] the convergence to the minimizer of the scalar analogous of the cost function (55) follows. \square

References

- [1] M.M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, P. Vandergheynst, Geometric deep learning: Going beyond euclidean data, *IEEE Signal Process. Mag.* 34 (4) (2017) 18–42, <http://dx.doi.org/10.1109/MSP.2017.2693418>.
- [2] J. Gilmer, S.S. Schoenholz, P.F. Riley, O. Vinyals, G.E. Dahl, Neural message passing for quantum chemistry, in: *Proceedings of the 34th International Conference on Machine Learning - Volume 70, ICML '17, JMLR.org*, 2017, pp. 1263–1272.
- [3] T.N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, in: *International Conference on Learning Representations*, 2017.
- [4] M. Defferrard, X. Bresson, P. Vandergheynst, Convolutional neural networks on graphs with fast localized spectral filtering, in: *Proceedings of the 30th International Conference on Neural Information Processing Systems, NIPS '16, Curran Associates Inc., Red Hook, NY, USA*, 2016, pp. 3844–3852.
- [5] L. Koestler, D. Grittner, M. Moeller, D. Cremers, Z. Löhner, Intrinsic neural fields: Learning functions on manifolds, in: *Computer Vision – ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part II, Springer-Verlag, Berlin, Heidelberg*, 2022, pp. 622–639, http://dx.doi.org/10.1007/978-3-031-20086-1_36.
- [6] F. Scarselli, M. Gori, A.C. Tsoi, M. Hagenbuchner, G. Monfardini, The graph neural network model, *IEEE Trans. Neural Netw.* 20 (1) (2009) 61–80, <http://dx.doi.org/10.1109/TNN.2008.2005605>.
- [7] L. Wu, P. Cui, J. Pei, L. Zhao, *Graph Neural Networks: Foundations, Frontiers, and Applications*, Springer Singapore, Singapore, 2022, p. 725.
- [8] M. Armando, J.-S. Franco, E. Boyer, Mesh denoising with facet graph convolutions, *IEEE Trans. Vis. Comput. Graphics* 28 (8) (2022) 2999–3012, <http://dx.doi.org/10.1109/TVCG.2020.3045490>.
- [9] Y. Zhang, G. Shen, Q. Wang, Y. Qian, M. Wei, J. Qin, GeoBi-GNN: Geometry-aware bi-domain mesh denoising via graph neural networks, *Comput. Aided Des.* 144 (2022) 103154, <http://dx.doi.org/10.1016/j.cad.2021.103154>.
- [10] Y. Shen, H. Fu, Z. Du, X. Chen, E. Burnaev, D. Zorin, K. Zhou, Y. Zheng, GCN-denoiser: Mesh denoising with graph convolutional networks, *ACM Trans. Graph.* 41 (1) (2022) <http://dx.doi.org/10.1145/3480168>.
- [11] M. Gori, G. Monfardini, F. Scarselli, A new model for learning in graph domains, in: *Proceedings. 2005 IEEE International Joint Conference on Neural Networks*, 2005, Vol. 2, 2005, pp. 729–734, <http://dx.doi.org/10.1109/IJCNN.2005.1555942>.
- [12] J. Bruna, W. Zaremba, A. Szlam, Y. Lecun, Spectral networks and locally connected networks on graphs, in: *International Conference on Learning Representations (ICLR2014)*, CBLs, April 2014, 2014.
- [13] G. Fu, P. Zhao, Y. Bian, p -Laplacian based graph neural networks, in: K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, S. Sabato (Eds.), *Proceedings of the 39th International Conference on Machine Learning*, in: *Proceedings of Machine Learning Research*, Vol. 162, PMLR, 2022, pp. 6878–6917.
- [14] B. Chamberlain, J. Rowbottom, M.I. Gorinova, M. Bronstein, S. Webb, E. Rossi, GRAND: Graph neural diffusion, in: M. Meila, T. Zhang (Eds.), *Proceedings of the 38th International Conference on Machine Learning*, in: *Proceedings of Machine Learning Research*, Vol. 139, PMLR, 2021, pp. 1407–1418.
- [15] X. Liu, W. Jin, Y. Ma, Y. Li, H. Liu, Y. Wang, M. Yan, J. Tang, Elastic graph neural networks, in: M. Meila, T. Zhang (Eds.), *Proceedings of the 38th International Conference on Machine Learning*, in: *Proceedings of Machine Learning Research*, Vol. 139, PMLR, 2021, pp. 6837–6849.
- [16] F. Di Giovanni, J. Rowbottom, B.P. Chamberlain, T. Markovich, M.M. Bronstein, Understanding convolution on graphs via energies, *Trans. Mach. Learn. Res.* (2023).
- [17] U. Pinkall, K. Polthier, Computing discrete minimal surfaces and their conjugates, *Exp. Math.* 2 (1) (1993) 15–36.
- [18] M. Wardetzky, S. Mathur, F. Kälberer, E. Grinspun, Discrete laplace operators: no free lunch, in: *Proceedings of the Fifth Eurographics Symposium on Geometry Processing, SGP '07, Eurographics Association*, 2007, pp. 33–37.
- [19] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, P.S. Yu, A comprehensive survey on graph neural networks, *IEEE Trans. Neural Netw. Learn. Syst.* 32 (1) (2021) 4–24, <http://dx.doi.org/10.1109/tnnls.2020.2978386>.
- [20] L. Tu, *An Introduction to Manifolds*, in: *Universitext*, Springer New York, 2010.
- [21] F. Barekat, R. Cafilisch, S. Osher, On the support of compressed modes, *SIAM J. Math. Anal.* 49 (4) (2017) 2573–2590, <http://dx.doi.org/10.1137/140956725>.
- [22] M. Huska, D. Lazzaro, S. Morigi, Shape partitioning via L_p compressed modes, *J. Math. Imaging Vision* 60 (7) (2018) 1111–1131, <http://dx.doi.org/10.1007/s10851-018-0799-8>.
- [23] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, M. Sun, Graph neural networks: A review of methods and applications, *AI Open* 1 (2020) 57–81, <http://dx.doi.org/10.1016/j.aiopen.2021.01.001>.
- [24] Y. Wang, J. Ren, D.-M. Yan, J. Guo, X. Zhang, P. Wonka, MGCN: descriptor learning using multiscale GCNs, *ACM Trans. Graph.* 39 (4) (2020) <http://dx.doi.org/10.1145/3386569.3392443>.
- [25] W. Zuo, D. Meng, L. Zhang, X. Feng, D. Zhang, A generalized iterated shrinkage algorithm for non-convex sparse coding, in: *Proceedings - 2013 IEEE International Conference on Computer Vision, ICCV 2013, IEEE*, 2013, pp. 217–224, <http://dx.doi.org/10.1109/ICCV.2013.34>.
- [26] M. Reuter, F.-E. Wolter, N. Peinecke, Laplace–Beltrami spectra as ‘Shape-DNA’ of surfaces and solids, *Comput. Aided Des.* 38 (4) (2006) 342–366, <http://dx.doi.org/10.1016/j.cad.2005.10.011>.
- [27] I. Ben-Shaul, L. Bar, D. Fishelov, N. Sochen, Deep learning solution of the eigenvalue problem for differential operators, *Neural Comput.* 35 (2023) 1–35, http://dx.doi.org/10.1162/neco_a01583.
- [28] C. Xu, H. Lin, H. Hu, Y. He, Fast calculation of Laplace–Beltrami eigenproblems via subdivision linear subspace, *Comput. Graph.* 97 (2021) 236–247, <http://dx.doi.org/10.1016/j.cag.2021.04.019>, URL <https://www.sciencedirect.com/science/article/pii/S0097849321000613>.
- [29] M. Belkin, P. Niyogi, Laplacian eigenmaps for dimensionality reduction and data representation, *Neural Comput.* 15 (6) (2003) 1373–1396, <http://dx.doi.org/10.1162/08997660321780317>.
- [30] Y. Xie, T. Takikawa, S. Saito, O. Litany, S. Yan, N. Khan, F. Tombari, J. Tompkin, V. Sitzmann, S. Sridhar, Neural fields in visual computing and beyond, *Comput. Graph. Forum* (2022) <http://dx.doi.org/10.1111/cgf.14505>.
- [31] W. Tang, Y. Gong, G. Qiu, Feature preserving 3D mesh denoising with a dense local graph neural network, *Comput. Vis. Image Underst.* 233 (2023) 103710, <http://dx.doi.org/10.1016/j.cviu.2023.103710>.
- [32] L. Calatroni, M. Huska, S. Morigi, G.A. Recupero, A unified surface geometric framework for feature-aware denoising, hole filling and context-aware completion, *J. Math. Imaging Vision* 65 (1) (2022) 82–98, <http://dx.doi.org/10.1007/s10851-022-01107-w>.
- [33] Z. Liu, Y. Li, W. Wang, L. Liu, R. Chen, Mesh total generalized variation for denoising, *IEEE Trans. Vis. Comput. Graphics* 28 (12) (2022) 4418–4433, <http://dx.doi.org/10.1109/TVCG.2021.3088118>.
- [34] S. Hattori, T. Yatagawa, Y. Ohtake, H. Suzuki, Deep mesh prior: Unsupervised mesh restoration using graph convolutional networks, in: *CVPR Workshop on Learning To Generate 3D Shapes and Scenes*, 2021.
- [35] K. He, X. Zhang, S. Ren, J. Sun, Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification, in: *2015 IEEE International Conference on Computer Vision, ICCV, 2015*, pp. 1026–1034, <http://dx.doi.org/10.1109/ICCV.2015.123>.
- [36] X. Glorot, Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, in: Y.W. Teh, M. Titterton (Eds.), *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, in: *Proceedings of Machine Learning Research*, Vol. 9, PMLR, 2010, pp. 249–256.
- [37] J. Baglama, L. Reichel, Augmented implicitly restarted lanczos bidiagonalization methods, *SIAM J. Sci. Comput.* 27 (1) (2005) 19–42, <http://dx.doi.org/10.1137/04060593X>.

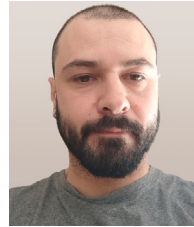
- [38] R.B. Lehoucq, D.C. Sorensen, C. Yang, ARPACK Users' Guide, Society for Industrial and Applied Mathematics, 1998, <http://dx.doi.org/10.1137/1.9780898719628>.
- [39] D. Palomar, Y. Eldar (Eds.), Convex Optimization in Signal Processing and Communications: Gradient-Based Algorithms with Applications to Signal-Recovery Problems, Cambridge University Press, United Kingdom, 2009, <http://dx.doi.org/10.1017/CBO9780511804458>.
- [40] D. Zhou, B. Scholkopf, Regularization on discrete spaces, in: The 27th DAGM Symposium, Vienna, Austria, August 31 - September 2, 2005, Vol. 3663 (2005) 361–368.



Damiana Lazzaro: She is currently Associate Professor of Numerical Analysis at the Department of Mathematics, University of Bologna, Italy. She received the B.S. degree in Mathematics, *summa cum laude*, from the University of Messina in 1988, and the Ph.D. degree in Computational Mathematics from the University of Naples in 1995. Her major research interests include wavelet and multiwavelet theory and applications, signal and image processing, inverse problems, compressed sensing. She is Associated Editor of International Journal of Computer Mathematics.



Serena Morigi is currently a full Professor of Numerical Analysis at the Department of Mathematics, University of Bologna, Italy. In 1996, she received the Ph.D. degree in Applied Mathematics from the University of Padova, Italy. In 1997, she did a postdoctoral fellowship in Mathematics at the Vanderbilt University, Tennessee, USA. Her research interests lie in the field of mathematical image processing and numerical methods for optimization. She is also interested in geometric modeling, surface reconstruction, and geometry processing. She is Associate Editor for international journals in numerical analysis and image processing. She has coordinated/participated several national and international research projects.



Paolo Zuzolo received the M.S. degree in mathematics with the University of Bologna, Bologna, Italy in 2020, and he is currently a Ph.D. student in mathematics with the University of Bologna, Italy. After completing his Master he worked as researcher at CINECA, Bologna, Italy, at the Visualization Information Technology Laboratory. His research interests include numerical methods for inverse problems in image and geometry processing and graph neural networks.