



ALMA MATER STUDIORUM  
UNIVERSITÀ DI BOLOGNA

DOTTORATO DI RICERCA IN  
MATEMATICA

Ciclo 37

**Settore Concorsuale:** 01/A5 - ANALISI NUMERICA

**Settore Scientifico Disciplinare:** MAT/08 - ANALISI NUMERICA

VARIATIONAL AND DIFFERENTIAL MODELS FOR SHAPE MODELING

**Presentata da:** Giuseppe Antonio Recupero

**Coordinatore Dottorato**

Giovanni Mongardi

**Supervisore**

Serena Morigi

**Co-supervisore**

Valeria Simoncini

Esame finale anno 2025





*Si vede sempre quello che manca,  
"l'erba del vicino è sempre più verde".  
È una filosofia di vita veramente negativa.  
Godiamoci quello che abbiamo, una volta tanto,  
senza pensare a quello che non abbiamo.*

*JULIO VELASCO*



# Abstract

The spread of new technologies led to a crucial role for the modeling of 3D objects, in particular for shape modeling, in a variety of applications, such as architecture, cultural heritage, industrial design, computer graphics, 3D radar scanning and others. Every task demands tailored surface processing of 3D geometric models, which defines the objects' shape and features. We tackled certain surface processing tasks using differential and variational models. The quality of the numerical solution depends on the integrity of the given data, possibly suffering from damage or noise, and on the desired geometric properties to be preserved.

Differential models rely on physics-inspired Partial Differential Equations (PDEs) to process surface data such as position, curvature and normal vectors. They are able to provide smooth, continuous representations of geometric structures and to exploit well-known physics equations. On the other hand, variational models compute the desired surface as the minimum of a suitable energy functional. They are built to encode initial surface features, through a data-fidelity term, and an a priori knowledge about the geometry of the desired result, through regularization or deformation terms. For the numerical solution of the proposed linear and nonlinear PDE models, we applied explicit, implicit or semi-implicit evolutive finite differences schemes. The numerical optimization methods used to solve the proposed variational models range from the gradient descent method on manifolds to the Alternate Direction Method of Multipliers.

A fundamental role in both mathematical approaches is played by the shape descriptors, i.e. the type of representation used for geometric models, based on Euclidean coordinates or on intrinsic representations, like the Differential Coordinates.

The proposed differential and variational models are applied to tackle challenging problems in shape analysis, such as removing noise from surfaces, filling in missing parts of surfaces, transferring textures between surfaces, and segmenting surfaces into meaningful regions.

**Keywords:** Variational models, Numerical optimization, Partial Differential Equations, Mesh processing, Shape analysis, Differential coordinates.



# Contents

<b>Introduction</b>	<b>1</b>
<b>1 Geometric models and surface processing</b>	<b>7</b>
1.1 Surface Representation Models . . . . .	7
1.1.1 Parametric representation . . . . .	7
1.1.2 Implicit representation . . . . .	8
1.1.3 Shape approximation via polygonal mesh . . . . .	9
1.1.4 Mesh parametrization . . . . .	11
1.2 Differential operators for surface processing . . . . .	12
1.2.1 On continuous settings . . . . .	12
1.2.2 On the discrete settings . . . . .	15
1.2.3 Differential Coordinates . . . . .	18
1.3 Surface processing tasks . . . . .	21
1.3.1 Repairing . . . . .	21
1.3.2 Smoothing . . . . .	26
1.3.3 Spectral Analysis . . . . .	26
1.3.4 Deformation . . . . .	27
1.4 Differential Models for Shape Analysis . . . . .	30
1.5 Variational Models for Shape Analysis . . . . .	36
<b>2 Variational recovery in Euclidean coordinates</b>	<b>41</b>
2.1 Variational Recovery Model . . . . .	43
2.1.1 Sparsity-inducing penalty . . . . .	46
2.1.2 Edge-based discretization of the Willmore energy . . . . .	47
2.2 Numerical solution of the optimization problem . . . . .	49
2.3 A practical use of the geometry repair framework . . . . .	55
2.4 Numerical Examples . . . . .	56
<b>3 Variational recovery in differential coordinates</b>	<b>65</b>
3.1 Invariance of shape descriptors . . . . .	67
3.1.1 Inexact invariance under free-form deformation . . . . .	72
3.2 Variational models for Geometric Texture Transfer . . . . .	73
3.2.1 Boundary setting . . . . .	74

3.2.2	Soft Constraints . . . . .	74
3.3	Numerical optimization of the variational GTT models . . . . .	76
3.3.1	Solution of the Laplacian model (3.9) . . . . .	76
3.3.2	Solution of the NCC model (3.10) . . . . .	76
3.3.3	Solution of the MVE model (3.11) . . . . .	77
3.3.4	Algorithm GTT . . . . .	79
3.4	Numerical GTT examples . . . . .	81
<b>4</b>	<b>Variational eigendecomposition of the graph <math>p</math>-Laplacian</b>	<b>91</b>
4.1	Notations and Preliminaries . . . . .	93
4.1.1	The continuous setting: $p$ -Laplacian eigenproblem . . . . .	93
4.1.2	The discrete setting: graph $p$ -Laplacian eigenproblem . . . . .	94
4.2	The variational $p$ -eigendecomposition model, with $p$ -orthogonality constraint	99
4.3	Iterative optimization to solve the $p$ -eigendecomposition problem . . . . .	102
4.3.1	Optimization problems reformulation by change of variable . . . . .	103
4.3.2	Solving (4.42) via Projected Gradient Descent method on Manifold (M-PGD) . . . . .	107
4.3.3	Solving (4.43) via ADMM on Manifold (M-ADMM) . . . . .	109
4.4	Numerical Results . . . . .	119
4.4.1	Example 1: computing 2-Laplacian eigenpairs . . . . .	120
4.4.2	Example 2: computing $p$ -Laplacian eigenpairs . . . . .	122
4.4.3	Example 3: approach to the spectral clustering . . . . .	127
<b>5</b>	<b>Linear PDE models for graph osmotic flow</b>	<b>129</b>
5.1	Linear model: properties and application in image editing . . . . .	130
5.2	Local osmosis model . . . . .	133
5.3	Non-local osmosis model . . . . .	135
5.3.1	Well-posedness . . . . .	138
5.3.2	Regularity in time . . . . .	140
5.4	Consistency . . . . .	141
5.5	Graph discretization and applications . . . . .	145
5.5.1	Space Discretization . . . . .	145
5.5.2	Time Discretization . . . . .	146
5.6	Numerical results . . . . .	148
<b>6</b>	<b>Non-linear PDE models for image osmotic flow</b>	<b>151</b>
6.1	The continuous non-linear model . . . . .	152
6.2	Model discretisation using finite difference schemes . . . . .	157
6.2.1	Semi-discretisation in space . . . . .	158
6.2.2	Time discretisation . . . . .	160
6.3	Numerical results . . . . .	162

---

6.3.1	Shadow removal . . . . .	162
6.3.2	Spot-light removal . . . . .	168
6.3.3	Compact data representation . . . . .	168
<b>7</b>	<b>Linear PDE models for 3D mesh osmotic flow</b>	<b>175</b>
7.1	Cloning via osmosis in Euclidean coordinates . . . . .	175
7.1.1	Numerical Results . . . . .	178
7.2	Inpainting via osmosis in differential coordinates . . . . .	181
7.2.1	Numerical Results . . . . .	183
	<b>Conclusions</b>	<b>191</b>
	<b>Bibliography</b>	<b>191</b>





# Introduction

In recent years, advances in computational geometry have led to powerful tools for analyzing, restoring and modifying 3D surfaces, as digital representations of 3D objects. Such tools play a critical role in applications across various fields, such as cultural heritage preservation, computer graphics, animation, industrial design, just to name a few.

A primary challenge when investigating these data processing problems is determining the most effective representation for surfaces. The two primary approaches for describing a smooth surface in  $\mathbb{R}^3$  are parametric and implicit representations. However, for practical surface processing, continuous surfaces must be approximated by a finite number of simple geometric elements, such as polygonal meshes. In this discrete setting, the geometrical properties of a surface are encoded in shape descriptor functions, such as the straight-forward Euclidean coordinates or the intrinsic differential coordinates. These geometric encoders exhibit different properties, making them more or less suitable for specific surface processing tasks.

Surface processing tasks are concerned with the analysis and modification of geometric models. The mathematical formulation of these tasks is typically formulated using either a differential or a variational approach. Differential models consist in physics-inspired Partial Differential Equations, whose form produces specific modifications or evolution of the geometric model. Their numerical solution can be achieved by applying suitable finite difference schemes, leading for example to explicit, implicit or semi-implicit iterative evolution algorithms. Variational models, on the other hand, interpret the desired solution as the minimum of a suitable energy functional, composed by a fidelity term, depending on the initial data, and by one or more regularization/penalty terms, that favor solution with a-priori knowledge on the geometric properties. The mathematical properties inherent to the functional dictate the optimal optimization method for solving the minimization problem. In both approaches, the models are constructed by employing differential operators defined on the surface, designed to quantify specific geometric properties, such as smoothness, sharpness, curvature, and flatness.

In this thesis, we develop novel differential and variational frameworks for a wide class of surface processing tasks, exploiting both Euclidean and Differential coordinate descriptors of a surface. Specifically, the proposed differential and variational models are applied to address several challenging problems in shape analysis, including the removal of noise from surfaces, the filling of missing parts of surfaces, the transfer of textures between surfaces, and the segmentation of surfaces into meaningful regions.

## Contribution and Outline

This thesis is organized into seven chapters.

In the first chapter, we introduce basic theory on surface processing, starting from surface representation models, setting the main notations for the rest of the thesis. Then, we report the definitions of differential operators on surfaces, both in the continuous domain and in the discrete mesh approximation, and of shape descriptors able to encode the local geometric details of a surface. We proceed with a mathematical formalization of geometric processing tasks, such as denoising, hole-filling, spectral analysis, cloning and geometric texture transferring and we conclude with an overview of differential and variational models, highlighting how they can be applied to solving such tasks.

The next two chapters focus on two variational frameworks for specific mesh processing tasks, exploiting shape descriptors as Euclidean coordinates or Differential coordinates. In particular:

- In Chapter 2, we deal with tasks such as denoising, hole filling and completion, useful for the creation and/or correction of virtual twins of 3D scanned objects, whose utility has increased in the last years. We describe a unified variational model that acts directly on the Euclidean coordinates of the mesh vertices and is able to recover an accurate mesh representation of the object, starting from damaged or noisy data. The non-convex optimization problem involves a quadratic fidelity term and two regularization terms: a discrete approximation of the Willmore energy forcing local sphericity and suited for the recovery of rounded features, and a non-convex approximation of the  $\ell^1$  penalty favoring sparsity in the normal variation. The resolution efficiently exploits the Alternating Direction Method of Multipliers (ADMM), avoiding any domain parameterization or implicit function approaches. The restoration results are precise, even with high noise or large areas with missing data.
- In Chapter 3, we analyze Differential coordinates, namely Laplacian coordinates, Normal Controlled Coordinates and Mean Value Encoding. These descriptors encode the underlying local geometry of an object by describing the relative position of a vertex with respect to its neighbors with different levels of invariance to rigid transformations and uniform scaling. This makes them a useful alternative to the classic Euclidean coordinates. We use them for Geometric Texture Transfer, a task whose goal is to transfer the fine-grained details of a textured surface into a base surface. We propose a variational model involving two terms: the first one recovers vertex positions from differential coordinates, the second one is a soft-constraint that preserves the original underlying shape of the surface. Without strong assumptions of equivalence in local connectivity between the two meshes, our approach requires just boundary matching and is solved via non-linear least

squares numerical methods. Results highlight the better quality of the mean value encoding, due to their good invariance properties.

In Chapters 4-7 we investigate properties and applications in surface processing of two differential equations, namely the  $p$ -Laplacian eigenvalue problem and the osmosis model.

- In Chapter 4, we study the  $p$ -Laplacian eigenproblem on graphs, which has applications in data clustering, spectral graph theory, dimensionality reduction and other problems. Since the  $p$ -Laplacian is a non-linear generalization of the Laplace operator, its eigenfunctions better capture the underlying geometry of the data. The problem of computing multiple eigenpairs is approached incrementally through a sequence of variational models that involve a generalization of the graph Rayleigh quotient, with a non-linear constraint of  $p$ -orthogonality. A simple reformulation allows us to take advantage of linear constraints. To solve the variational problem, we propose two different optimization algorithms: project gradient descent on manifold and Alternate Direction Method of Multipliers. We demonstrate the effectiveness and accuracy of the proposed algorithms and compare them in terms of efficiency. Finally, we show how  $p$ -eigenfunctions can be used for mesh segmentation tasks.
- In Chapter 5, we recall the definition of the linear second-order diffusion-transport Osmosis PDE model, reporting how it can be applied for image editing tasks. Then we define a non-local version of the Osmosis PDE model, where each differential operator is replaced by an integral operator depending on kernel functions, that weights the long-range interactions between points in the domain. We study well-posedness of both local and non-local problems and regularity of their solutions. Then, we observe that, upon suitable rescaling of the kernel, the non-local operators are good approximations of the corresponding differential operators, as the kernel support shrinks. Consequently, we conjecture that a succession of non-local solutions converges to the local solution. Finally, we use the non-local formulation to derive a graph discretization of the model, observing that it can be applied as an editing tool for data functions defined on meshes, without changing the underlying geometry.
- In Chapter 6, we present a non-linear variant of the osmosis model, involving a scalar diffusivity function with suitable properties, which allows to balance the diffusion intensity over different regions of the image while preventing smoothing artifacts. We show that this generalization respects conservation properties already proven in the linear case and it can be paired with a variational formulation. Conditional or unconditional stability and convergence results are proven for explicit and semi-implicit iterative schemes, respectively, given a proper spatial discretization

of the differential operators. The model efficiently provides good results for image editing tasks such as shadow/light removal and compact data representation.

- Finally, in Chapter 7, the Osmosis PDE model is used to alter the geometry of surfaces and perform mesh cloning and inpainting.

The first task consists in replacing a region of a surface with a different patch, with both approximated as meshes. A pre-processing step builds a mixed triangulation that acts as the common domain where osmosis is defined. The osmosis evolution, guided by the Euclidean coordinates of the two meshes, converges to the Euclidean coordinates of the desired surface.

For the second task, osmosis is applied on the Normal Controlled Coordinates of a surface with a damaged, obtaining a shape representation of the repaired surface.

In both cases, the unconditional stability of the implicit osmosis evolution scheme provides fast convergence, with a low computational cost.

## Related Publications

Chapter 2 refers to:

- [61] M. Huska, S. Morigi, G.A. Recupero, Sparsity-aided variational mesh restoration, International Conference on Scale Space and Variational Methods in Computer Vision. Cham: Springer International Publishing, (2021).
- [25] L. Calatroni, M. Huska, S. Morigi, and G. A. Recupero, A unified surface geometric framework for feature-aware denoising, hole filling and context-aware completion, Journal of Mathematical Imaging and Vision, 65 (2023).

Chapters 3-7 refer respectively to:

- [62] M. Huska, S. Morigi, and G. A. Recupero, Geometric texture transfer via local geometric descriptors, Applied Mathematics and Computation, 451 (2023)
- [69] A. Lanza, S. Morigi, G.A. Recupero, Variational graph p-Laplacian eigendecomposition under p-orthogonality constraints, Computational Optimization and Applications (2024)
- [26] L. Calatroni, S. Morigi, S. Parisotto, and G. A. Recupero, Fast and stable schemes for non-linear osmosis filtering, Computers & Mathematics with Applications, 133 (2023), pp. 30–47.
- J. Fadili, G.A. Recupero, R. Zantout, Non local and graph approximation of the osmosis model: properties and consistency, (in progress)

- 
- M. Huska, S. Morigi, G.A. Recupero, Linear PDE osmotic flow for 3D surfaces, International Conference on Scale Space and Variational Methods in Computer Vision (submitted)



# Chapter 1

## Geometric models and surface processing

Surface processing consists in applying algorithms to geometric models of 3D objects. A fundamental role in surface processing is played by the shape descriptors, i.e. the type of representation used for geometric models, which can be based on Euclidean coordinates or alternately on intrinsic representations, like the Differential Coordinates. The representation of 3D objects is a widely explored research field and the different possible approaches present strengths and weaknesses. We deal with the representation of boundary surfaces of a non-degenerate 3D object, i.e. an object with an interior and an exterior, without any infinitely thin features. Mathematically, a surface  $\mathcal{S}$  is an orientable continuous 2D manifold embedded in  $\mathbb{R}^3$ , with or without boundary.

We mainly distinguish two surface representation models: explicit (parametric) and implicit, and we will exploit the triangular meshes to describe the corresponding approximations.

### 1.1 Surface Representation Models

#### 1.1.1 Parametric representation

A surface  $\mathcal{S}$  can be represented through the couple  $(\Omega, X)$ , with  $\Omega \subset \mathbb{R}^2$  a compact parametric domain and  $X : \Omega \rightarrow \mathbb{R}^3$  a continuous parametric vector function. The surface  $\mathcal{S}$  consists in the set  $X(\Omega) \subset \mathbb{R}^3$ , defined by the parameters  $(t, s) \in \Omega$  and by the coordinates functions  $x(t, s), y(t, s), z(t, s)$  as follows:

$$\mathcal{S} : \quad X(t, s) = \begin{pmatrix} x(t, s) \\ y(t, s) \\ z(t, s) \end{pmatrix}. \quad (1.1)$$

Parametric surfaces can capture even fine details and are easy to sample, simply by sampling the 2D parametric domain  $\Omega$ . Similarly, geodesic neighborhoods are eas-



ily found. Moreover, a deformed version of a surface can be represented as a simple composition between the parametric function  $X$  and a deformation function.

A weakness consists in the difficult computation of spatial properties, such as the point membership classification problem.

In Computer Aided Design (CAD), the standard parametric shape representation is given by spline / NURBs parametric surfaces. Surface patches are described as tensor product and represented by linear combinations of spline basis functions  $N_i^n(\cdot) : I \times J \rightarrow \mathbb{R}^3$ , with  $I, J$  intervals in  $\mathbb{R}$ , which are non-negative, with compact support [101]. Spline surfaces allow a natural shape representation and provide easy geometric modeling of the represented shape. A deformation of the surface, in fact, follows from an intuitive adjustment of the control points.

This approach has the strong topological limitation of producing rectangular-shaped patches, making it necessary to use a composition of a large number of patches to represent arbitrary topology shapes.

A possible generalization of spline surfaces is given by subdivision surfaces. They depend on a coarse control mesh, repeatedly refined, with the addition of vertex adjustment steps based on local averaging rules.

In general, an arbitrary mesh may not be representable as the result of subsequent refinement. In such cases, regardless of topological properties, a pre-processing remeshing step may be needed, with the risk of having artifacts or loss of information.

### 1.1.2 Implicit representation

A surface  $\mathcal{S}$  can be implicitly represented as the level set  $c \in \mathbb{R}$  of a scalar function  $F : \mathbb{R}^3 \rightarrow \mathbb{R}$

$$\mathcal{S} = \{(x, y, z) \in \mathbb{R}^3 \mid F(x, y, z) = c\}. \quad (1.2)$$

Implicit surfaces can be open or closed, depending on the nature of the function  $F$ . In case of closed surface, the level  $c$  is conventionally set to 0, in order to identify internal and external points as the ones where  $F$  assumes negative or positive values, respectively. In general, several implicit scalar functions with the same zero-level set can represent the same surface. Such surfaces cannot present self-intersection and, if  $F$  is continuous, they do not have any holes.

Many surface processing applications implicitly define the geometric models by means of the signed distance  $d : \mathbb{R}^3 \rightarrow \mathbb{R}$  which reads as

$$d(x, y, z) = \begin{cases} \min\{\|(\bar{x}, \bar{y}, \bar{z}) - (x, y, z)\|_2, \text{ with } (\bar{x}, \bar{y}, \bar{z}) \in \mathcal{S}\} & \text{if } (x, y, z) \text{ is outside } \mathcal{S}; \\ -\min\{\|(\bar{x}, \bar{y}, \bar{z}) - (x, y, z)\|_2, \text{ with } (\bar{x}, \bar{y}, \bar{z}) \in \mathcal{S}\} & \text{if } (x, y, z) \text{ is inside } \mathcal{S}; \\ 0 & \text{if } (x, y, z) \in \mathcal{S}. \end{cases} \quad (1.3)$$

These minima exist since the norm is a continuous function and  $S$  is a compact set in  $\mathbb{R}^3$ . The signed distance function is smooth and allows to easily find the iso-surface at a distance  $c$  from  $\mathcal{S}$ , as the set  $\{(x, y, z) \in \mathbb{R}^3 \mid d(x, y, z) = c\}$ .

Implicit representation is particularly useful for surface processing tasks that involve boolean operations, such as union, intersection or difference of surfaces, since the resulting implicit function is a combination of min and max operators applied to the implicit functions. This allows to easily represent surfaces with different topologies or even to change topology during a deformation process, since it only implies a change in scalar values of the implicit function.

However, it is difficult to find geodesic neighborhoods, compute a parametrization and even visualize the surface.

### 1.1.3 Shape approximation via polygonal mesh

To apply surface processing tasks to given geometric models, it is necessary first to convert continuous, smooth surfaces to discrete, computationally manageable representations, such as the polygonal meshes, consisting of a collection of simple geometric elements.

The process that approximates parametric and implicit surfaces is called surface meshing and discretizes  $\mathcal{S}$  as a polygonal mesh  $\mathcal{M}$  defined as follows:

**Definition 1.1.1.** *A mesh  $\mathcal{M} = (V, E, T)$  consists in a set of vertices  $V$ , a set of undirected edges  $E$  and a set of triangular faces  $T$*

$$V = \{v_i\}_{i=1}^{n_V} \in \mathbb{R}^{n_V \times 3}, \quad E = \{e_{i,j}, (i,j) \in \mathcal{E}\} \in \mathbb{N}^{n_E \times 2}, \quad T = \{\tau_i\}_{i=1}^{n_T} \in \mathbb{N}^{n_T \times 3}. \quad (1.4)$$

*Two connected vertices  $v_i, v_j$  identify an edge  $e_{i,j}$  and an edge  $e_{j,i}$ , while a face  $\tau = (i, j, k)$  is defined by its three vertices  $v_i, v_j, v_k$ . Conventionally, faces are consistently oriented in the whole mesh, in clockwise or counter-clockwise order, such that every edge  $(i, j)$  belonging to two faces  $\tau_1$  and  $\tau_2$  is counted in both directions, as  $\tau_1 = (i, j, k_1)$ ,  $\tau_2 = (j, i, k_2)$ .*

We assume the mesh to be a 2-dimensional manifold, i.e. :

- it does not contain self-intersection;
- it does not contain non-manifold edges (one edge belongs to two faces, or to one face if it is on the surface boundary);
- it does not contain non-manifold vertices (one vertex is incident to just one fan of triangles).

We introduce the concepts of first disk and first ring that identify, for a given vertex  $v_i$ , its face neighbors and vertex neighbors, respectively:

$$\text{First disk:} \quad \mathcal{D}(v_i) = \{\tau_m \mid v_i \in \tau_m\}, \quad (1.5)$$

$$\text{First ring:} \quad \mathcal{N}(v_i) = \{v_j \mid (v_i, v_j) \in E\}. \quad (1.6)$$

With abuse of notation, we denote as  $\mathcal{N}(i)$  the set of indices of the vertices in  $\mathcal{N}(v_i)$ .

For meshes having the topology of the disk, possibly with boundaries, it is possible to define a parametrization  $\Omega$  onto the plane. Mesh parametrization of 3D models is an important component in various computer graphics and geometry processing applications. It involves computing a bijective mapping between a piecewise-linear triangulated surface (mesh) and a suitable parameter domain. Details on the mesh parametrization process are given in Section 1.1.4.

A mesh  $\mathcal{M}$  with associated parametrization  $\Omega$  onto the plane can be interpreted as a parametric surface  $(\tilde{\Omega}, \tilde{X})$ , where  $\tilde{\Omega}$  is the convex hull of the set of parametric points, while  $\tilde{X}$  is a piecewise linear approximation of  $X$ . A generic parametric point  $(t, s) \in \tilde{\Omega}$  belonging to the triangle  $((t_i, s_i), (t_j, s_j), (t_k, s_k))$ , with barycentric coordinates  $(\alpha, \beta, \gamma)$  s.t.

$$(t, s) = \alpha(t_i, s_i) + \beta(t_j, s_j) + \gamma(t_k, s_k) \quad (1.7)$$

is mapped into the point

$$\tilde{X}(t, s) = \bar{v} = \alpha v_i + \beta v_j + \gamma v_k \quad (1.8)$$

belonging to the face  $(v_i, v_j, v_k)$  of the mesh  $\mathcal{M}$ .

The so-called tessellation process produces a polygonal mesh from a parametric surface  $\mathcal{S} = (\Omega, X)$ , and it simply consists of the following steps:

- discretize the parametric domain  $\Omega$  into a grid set of points  $(t_i, s_i)_{i=1}^{n_V}$ ,
- connect the points through edges to form a 2-dimensional planar triangulation, obtaining the sets  $E$  and  $T$  (for example via Delaunay triangulation);
- compute the vertex coordinates as  $v_i = X(t_i, s_i)$ .

The approximation error occurring when representing a smooth surface as a triangulated mesh is of order  $O(h^2)$ , with  $h$  the maximum edge length. To reduce this error, the common strategy is to refine the mesh, uniformly or adaptively increasing the resolution in regions of high curvature to better capture the surface geometry.

On the other hand, if  $\mathcal{S}$  is an implicit surface  $\mathcal{S} = \{F(x, y, z) = 0\}$ , the most popular method to obtain its mesh approximation is the so-called Marching Cubes algorithm [80] which briefly proceeds as follows:

- the bounding box of the surface is sampled through a 3D grid of points  $(x_i, y_j, z_k)$ , which forms a set of cubes, called voxels;
- on each voxel, examine the values of the implicit function  $F$  at the corners, searching for a sign change;
- interpolate along the edges connecting two points with opposite signs, obtaining a point with zero value, that represents a vertex of the desired mesh;
- create the triangulation, connecting vertices in the same voxel, depending on their position.

Memory consumption of this process is high and grows cubically as the edge length decreases. A more efficient alternative is given by adopting adaptive data structures, where a hierarchical process leads to a voxel sampling whose granularity depends on surface curvature and distance from the surface.

#### 1.1.4 Mesh parametrization

A parametrization of a parametric surface  $\mathcal{S}$  is a function putting this surface in one-to-one correspondence with a 2D parametric domain  $\Omega$ .

The parametrization can be hard to compute. In fact, it needs topological and metric coherence between  $\Omega$  and  $\mathcal{S}$ . A change in the shape of  $\mathcal{S}$  may require a correction in the parametrization to preserve a consistent metric and avoid stretching effects.

Computing a parametrization  $(\Omega, X)$  corresponding to a surface  $\mathcal{S}$ , approximated by a mesh  $\mathcal{M} = (V, E, T)$  requires to univocally determine the inverse mapping  $X^{-1}$ , which maps the vertex coordinates  $v_i, i = 1, \dots, n_V$  to the position of the parameter points  $(t_i, s_i)_{i=1}^n \in \Omega$ .

All the most widely used methods for constructing a parametrization of a triangulated surface relies on Tutte's barycentric mapping theorem [121], from graph theory, which states:

**Theorem 1.1.1.** *Given a triangulated surface homeomorphic to a disk, if the  $(t, s)$  coordinates at the boundary vertices lie on a convex polygon, and if the coordinates of the internal vertices are a convex combination of their neighbors, then the  $(t, s)$  coordinates form a valid parametrization (without self-intersections).*

Denoting by  $\{\bar{v}_i, i = 1, \dots, n_b\}$  the boundary vertices of the mesh, which map to the convex polygonal boundary of the domain  $\Omega$  formed by the points  $(t_i, s_i)_{i=1}^{n_b}$ , the parametrization of all the inner vertices are determined by solving the two following linear systems with  $n - n_b$  unknowns  $(t_j, s_j)_{j=n_b+1}^n$ :

$$\begin{aligned}
\sum_{j=n_b+1}^n w_{ij} t_j &= - \sum_{j=1}^{n_b} w_{ij} t_j, \\
\sum_{j=n_b+1}^n w_{ij} s_j &= - \sum_{j=1}^{n_b} w_{ij} s_j,
\end{aligned} \tag{1.9}$$

where the weights  $w_{ij}$  ensure the convex combination condition, respecting the constraints

$$\begin{cases} w_{ij} > 0 & \text{if } (\bar{v}_i, \bar{v}_j) \in E \\ w_{ii} = - \sum_{j \neq i} w_{ij} & \\ w_{ij} = 0 & \text{otherwise} \end{cases}$$

Chosen the appropriate weights, the linear systems in (1.9) can be interpreted as  $L_w t = 0$  and  $L_w s = 0$ , plus constraint on boundary points, which ensures that the position of the internal points  $(t_i, s_i)_{i=n_b+1}^n$  is uniquely defined.

## 1.2 Differential operators for surface processing

In general, all mesh processing tasks consist in properly modifying vertices, edges and faces, according to the desired goal. To this aim, some tools for manipulating surfaces represented by meshes rely on appropriately discretized differential operators, able to describe geometric properties and perform physics-inspired evolutive processes.

We analyze differential operators for a parametric surface  $\mathcal{S}$  embedded in  $\mathbb{R}^3$ , first in the continuous setting, then, analogously, in the discrete setting, where the parametric representation of the surface is approximated by a polygonal mesh.

### 1.2.1 On continuous settings

Let  $\mathcal{S} = (\Omega, X)$  be a parametric surface, with  $x, y, z$  differentiable coordinate functions. Then the partial derivatives of the parametric function  $X$  with respect to the parameters  $(u, v) \in \Omega$  are defined as follows.

**Definition 1.2.1.** *The **partial derivatives**  $X_t(t_0, s_0)$  and  $X_s(t_0, s_0)$  at the point  $(t_0, s_0)$  are the tangent vectors to the iso-parameter curves  $C_t(\tau) = X(t_0 + \tau, s_0)$  and  $C_v(\tau) = X(t_0, s_0 + \tau)$ , respectively,*

$$X_t = \begin{pmatrix} \partial_t x(t_0, s_0) \\ \partial_t y(t_0, s_0) \\ \partial_t z(t_0, s_0) \end{pmatrix}, \quad X_s = \begin{pmatrix} \partial_s x(t_0, s_0) \\ \partial_s y(t_0, s_0) \\ \partial_s z(t_0, s_0) \end{pmatrix}.$$

Assuming a regular parametrization, i.e.  $X_t \times X_s \neq 0$ , then the differential coordinates define the tangent plane and the normal vector.

The **tangent plane** to  $\mathcal{S}$  at the point  $X(t_0, s_0)$  is  $T_{X(t_0, s_0)}\mathcal{S} = \text{span}\{X_t, X_s\}$ .

The **normal vector** to  $\mathcal{S}$  at the point  $X(t_0, s_0)$  is

$$\mathbf{n} = \frac{X_t \times X_s}{\|X_t \times X_s\|}.$$

The direction of the normal depends on the orientation of the parametrization. Conventionally, for closed surfaces the parametrization is chosen such that the normal points outward the surface, while for open surfaces there is no natural orientation.

The **Jacobian matrix** of  $X$ , defined as  $J = [X_t, X_s]$ , encodes the metrics of the surface, describing how distances, angles and areas are transformed passing from the parametric domain to the surface. The directional derivative of  $X$  at a point  $(t_0, s_0)$  in a direction  $\bar{w} = (t_w, s_w)$  is then given by  $w = J\bar{w}$ .

It follows easily that, given two unit direction vectors  $\bar{w}_1, \bar{w}_2 \in \mathbb{R}^2$ , the scalar product between the corresponding tangent vectors  $w_1, w_2 \in \mathbb{R}^3$  is

$$w_1^T w_2 = (J\bar{w}_1)^T (J\bar{w}_2) = \bar{w}_1^T (J^T J) \bar{w}_2.$$

The above equivalence defines an inner product on the tangent space of  $\mathcal{S}$ , relying on  $J^T J$ , which consists in the first fundamental form or metric tensor of the parametrization  $X$ , as detailed in the following definition.

**Definition 1.2.2.** *The **first fundamental form** or **metric tensor** of  $X$  is the matrix  $\mathbf{I} = (g_{ij})_{i,j=1,2}$  defined as*

$$\mathbf{I} = J^T J = \begin{bmatrix} X_t^T X_t & X_t^T X_s \\ X_s^T X_t & X_s^T X_s \end{bmatrix}. \quad (1.10)$$

The length of a tangent vector  $w$  is therefore  $\|w\| = \sqrt{\bar{w}^T \mathbf{I} \bar{w}}$ . Moreover, given a curve  $\bar{\gamma} : [a, b] \rightarrow \Omega$  in the parameter space, the length of the corresponding curve  $\gamma = X \circ \bar{\gamma}$  on the surface  $\mathcal{S}$  is

$$l(a, b) = \int_a^b \sqrt{(t_\tau, s_\tau) \mathbf{I} (t_\tau, s_\tau)^T} d\tau.$$

Analogously, the surface area  $A$  of a region  $X(D)$ , with  $D \subset \Omega$  parameter region, is

$$A = \int \int_D \sqrt{\det(\mathbf{I})} dt ds.$$

These are intrinsic properties, because they depend only on first-order properties of the surface and are invariant under isometries. To get insights on the curvature of the

surface, we consider a tangent vector  $\xi = t_\tau X_t + s_\tau X_s$  at a point  $p \in \mathcal{S}$ , with parametric counterpart  $\bar{\xi} = (t_\tau, s_\tau)$ .

The **normal curvature**  $\kappa_n(\bar{\xi})$  at  $p$  is given by

$$\kappa_n(\bar{\xi}) = \frac{\bar{\xi}^T \mathbf{II} \bar{\xi}}{\bar{\xi}^T \mathbf{I} \bar{\xi}}$$

where  $\mathbf{II}$  represents the second fundamental form, defined as

$$\mathbf{II} = \begin{bmatrix} X_{tt}^T \mathbf{n} & X_{ts}^T \mathbf{n} \\ X_{ts}^T \mathbf{n} & X_{ss}^T \mathbf{n} \end{bmatrix}, \quad (1.11)$$

where  $\mathbf{n}$  is the normal versor.

Considering all the possible directions of  $\bar{\xi} \in \mathbb{R}^2$ , we are able to identify two orthogonal **principal directions**  $\xi_1$  and  $\xi_2$ , with unitary norm, which achieve **maximum** and **minimum curvature**  $\kappa_1$  and  $\kappa_2$ , respectively. Consequently, two useful measures of curvature can be defined.

The **Mean curvature**  $H$  is defined as the average of the principal curvatures:

$$H = (\kappa_1 + \kappa_2)/2.$$

The **Gaussian curvature**  $K$  is defined as the product of the principal curvatures

$$K = \kappa_1 \kappa_2.$$

Gaussian curvature is an intrinsic property, because it depends only on the first fundamental form. These provide insights into the geometric properties of the surface  $\mathcal{S}$ .

Now, we consider differential operators that act on functions defined on the surface  $\mathcal{S}$ . In particular, we consider a scalar function  $f : \mathcal{S} \rightarrow \mathbb{R}$  and a vector field  $\phi : \mathcal{S} \rightarrow \mathbb{R}^3$ .

The classical Euclidean gradient and divergence operator have corresponding analogs that consider the specific geometry of the surface  $\mathcal{S}$ , encoded into the metric tensor  $\mathbf{I}$ . Exterior calculus tools provide the following definitions, expressed in local coordinates:

$$\nabla_S f : \mathcal{S} \rightarrow \mathbb{R}^3, \quad (\nabla_S f)_i = \sum_j g^{ij} \partial_j f \quad (1.12)$$

$$\text{div}_S \phi : \mathcal{S} \rightarrow \mathbb{R}, \quad \text{div}_S \phi = \frac{1}{\sqrt{\det(\mathbf{I})}} \sum_i \partial_i \left( \sqrt{\det(\mathbf{I})} \phi_i \right) \quad (1.13)$$

where  $g^{ij}$  are the components of the inverse of the metric tensor  $\mathbf{I}$ .

The **Laplace-Beltrami** operator generalizes the Laplace operator  $\Delta = \text{div} \circ \nabla$  to scalar functions  $f$  defined on surfaces. In local coordinates, it is defined as

$$\Delta_S f = \sum_i \frac{1}{\sqrt{\det(\mathbf{I})}} \partial_i \left( \sqrt{\det(\mathbf{I})} \sum_j g^{ij} \partial_j f \right). \quad (1.14)$$

Laplace-Beltrami operator differs from classical Euclidean Laplacian because of the metric tensor  $\mathbf{I}$ , which encodes the underlying geometry.

When applied to the three coordinates of the parametric function  $X = (x, y, z)$ , the Laplace-Beltrami operator relates to the mean curvature  $H$ . In fact, it holds

$$\Delta_S X = -2H\mathbf{n}$$

where  $\mathbf{n}$  is the normal vector, considered in the outward direction. This motivates the use of the Laplace-Beltrami operator on geometric models as surface curvature indicator.

Finally, we introduce the  $p$ -Laplacian operator, a non-linear generalization of the classical Laplace operator.

**Definition 1.2.3.** *The  $p$ -Laplacian operator, for  $p \in (1, +\infty)$ , is defined for smooth function  $f : \mathbb{R}^n \rightarrow \mathbb{R}$ , as*

$$\Delta_p f := \operatorname{div} (\|\nabla f\|^{p-2} \nabla f) , \quad (1.15)$$

with  $\nabla$  and  $\operatorname{div}$  denoting the gradient and divergence operators, respectively, and with

$$\|\nabla f\|^{p-2} = (\|\nabla f\|^2)^{\frac{p-2}{2}} = \left( \sum_{i=1}^n \left( \frac{\partial f}{\partial x_i} \right)^2 \right)^{\frac{p-2}{2}} . \quad (1.16)$$

It is immediate to verify that (1.15) reduces to the definition of the linear Laplace operator for  $p = 2$ , whereas for  $p \neq 2$  the operator  $\Delta_p$  is non-linear, as we have

$$\Delta_p(cf) = \operatorname{div} (\|\nabla(cf)\|^{p-2} \nabla(cf)) = |c|^{p-2} c \Delta_p(f) \neq c \Delta_p(f) \quad \forall c \in \mathbb{R} \setminus \{0, \pm 1\} . \quad (1.17)$$

Besides being of mathematical interest, the  $p$ -Laplacian operator appears in many physical models, for example in non-Newtonian fluid dynamics [122], in phase-field models [2] or in game theory [98].

### 1.2.2 On the discrete settings

The emerging field of Discrete Differential geometry studies discrete analogs of smooth geometric objects, providing an essential link between analytic descriptors and computation.

The goal is to define the approximation of these operators, relying on discrete localization on vertices, edges or faces. For a mesh  $\mathcal{M}$ , we define the sets of scalar real-valued functions  $f$  and  $g$  with domains the discrete sets of vertices and edges, respectively

$$\mathcal{F}_V := \{f : V \rightarrow \mathbb{R}\}, \quad \mathcal{F}_E := \{g : E \rightarrow \mathbb{R}\}. \quad (1.18)$$

The sets  $\mathcal{F}_V$  and  $\mathcal{F}_E$  are clearly homeomorphic to the sets  $\mathbb{R}^{n_V}$  and  $\mathbb{R}^{n_E}$  and we can think to functions  $f \in \mathcal{F}_V$  and  $g \in \mathcal{F}_E$  as  $n_V$ -dimensional and  $n_E$ -dimensional vectors,



respectively - i.e.,  $f = \{f_i\}_{i=1,\dots,n_V}$  and  $g = \{g_{i,j}\}_{(i,j) \in E}$ . We endow  $\mathcal{F}_V$  and  $\mathcal{F}_E$  with the standard scalar products  $\langle f^{(1)}, f^{(2)} \rangle = \sum_{i=1}^{n_V} f_i^{(1)} f_i^{(2)}$ ,  $\langle g^{(1)}, g^{(2)} \rangle = \sum_{(i,j) \in E} g_{i,j}^{(1)} g_{i,j}^{(2)}$  and norms  $\|f\|^2 = \langle f, f \rangle$ ,  $\|g\|^2 = \langle g, g \rangle$ , respectively.

First, we deal with the discretization of the normal field. Noting that every face  $\tau$  of a mesh has an intuitive notion of a normal vector, we consider the following normal field definition.

**Definition 1.2.4.** *Given a mesh  $\mathcal{M} = (V, E, T)$ , the **discrete normal field** is the piecewise-constant function  $\mathcal{N} : \mathbb{R}^{n_V \times 3} \rightarrow \mathbb{R}^{n_T \times 3}$ , whose value on a face  $\tau = (i, j, k)$  is*

$$\mathcal{N}_\tau = \frac{(v_j - v_i) \times (v_k - v_i)}{\|(v_j - v_i) \times (v_k - v_i)\|}. \quad (1.19)$$

As in the continuous setting, normals do not have a naturally prescribed direction. However, meshes are usually defined with a consistent face orientation, such that the normal vectors do not flip passing by two neighbor triangles. In case of closed mesh, externally oriented normals are preferred.

One can extend the normal field definition to a vertex  $v_i$ , by averaging the value of the face normals of its first disk:

$$\mathbf{n}_{v_i} = \frac{\sum_{\tau \in \mathcal{D}(v_i)} \alpha_\tau \mathcal{N}_\tau}{\|\sum_{\tau \in \mathcal{D}(v_i)} \alpha_\tau \mathcal{N}_\tau\|}.$$

The weights  $\alpha_\tau$  can be defined as constants equal to 1, efficient but imprecise for irregular meshes, or as the triangle area, more expensive but more robust.

The discrete approximation  $L_w : \mathcal{F}_V \rightarrow \mathbb{R}^{n_V}$  of the Laplace-Beltrami operator acts on functions  $f \in \mathcal{F}_V$ , locally taking the difference between its value at a vertex  $v_i$  and a weighted average of its values at the first-order neighbor vertices:

$$(L_w(f))_i = f_i - \sum_{j \in \mathcal{N}(i)} w_{ij} f_j = \sum_{j \in \mathcal{N}(i)} w_{ij} (f_i - f_j), \quad (1.20)$$

where  $w_{ij} > 0$  are the weights normalized as  $w_{ij} = \bar{w}_{ij} / \sum_{j \in \mathcal{N}(v_i)} \bar{w}_{ij}$ , so that  $\sum_{j \in \mathcal{N}(v_i)} w_{ij} = 1$ .

The linearity of the operator allows to represent it as a matrix  $L_w \in \mathbb{R}^{n_V \times n_V}$ , with

$$L_w^{ij} = \begin{cases} w_{ij} & \text{if } j \in \mathcal{N}(i) \\ -\sum_{k \in \mathcal{N}(i)} w_{ik} & \text{if } i = j \\ 0 & \text{otherwise.} \end{cases} \quad (1.21)$$

Denoting as  $W$  the symmetric weight matrix and  $D$  the degree matrix, diagonal with values  $D_{ii} = \sum_j w_{ij}$ , we have  $L_w = D - W$ .

Many definitions are proposed for the weights  $w_{ij}$ . The **uniform formulation**

$$w_{ij} = 1/|\mathcal{N}(v_i)| \quad (1.22)$$

depends only on the valence  $d_i = |\mathcal{N}(v_i)|$  of each vertex and does not explicitly encode geometric information. The associated  $L$  is called **umbrella operator**.

The **scale-dependent umbrella operator** uses instead  $\bar{w}_{ij} = 1/\|e_{ij}\|_2$ , where  $\|e_{ij}\|_2 = \|v_i - v_j\|_2$  is the length of the edge connecting vertices  $v_i$  and  $v_j$ , so that the weights decrease for distant vertices, as expected. A whole class of weights  $w_{ij} = \phi(\|v_i - v_j\|_2)$  has been proposed, depending on the chosen function  $\phi$ . A common definition in spectral graph theory applications [89] is the heat kernel  $\phi(x) = \exp(-x^2/\sigma^2)$ , with  $\sigma > 0$  as a scaling parameter.

The choice of weights that more faithfully represent the discretization of the Laplace-Beltrami operator from Riemannian geometry consists in the **cotangent formulation** [86]

$$w_{ij} = (\cot \alpha_{ij} + \cot \beta_{ij})/(2A_i) \quad (1.23)$$

where  $A_i$  is the area of a local averaging domain around the vertex  $v_i$  (for example a barycentric cell or a Voronoi cell), and  $\alpha_{ij}$  and  $\beta_{ij}$  are the angles opposite to the edge  $e = (i, j)$ . This definition is not intrinsic, because it may give different results for isometric surfaces with different triangulation.

Other different definitions of the discrete Laplace-Beltrami Operator are described in literature, see [102] for comparison. For example, **mean value weights** are introduced in [48] as

$$w_{ij} = \frac{1}{2A_i}(\tan(\alpha_{ij}/2) + \tan(\beta_{ij}/2)). \quad (1.24)$$

If  $f$  is singly replaced by the three coordinate functions, then the Laplacian on a vertex  $v_i$  is the vector that connects the vertex  $v_i$  to the average of its neighbors  $v_j$ . The uniform formulation, despite being efficient and intrinsic, has the weakness of returning non-zero vectors even for planar configuration of vertices, if they are not uniformly distributed. Since the Laplacian is related to the mean curvature, this would mean that a flat mesh has at least one direction with non-zero curvature, that is non-consistent. This is why cotangent expression is often preferred.

In general, the discrete Laplace-Beltrami Operator provides an estimation of the mean curvature at vertex  $v_i$  as

$$H(v_i) = \frac{1}{2}\|\Delta v_i\| \quad (1.25)$$

encoding useful information about the shape and the texture details of the mesh. These properties allow to define isometric-invariant shape descriptors, used in many mesh processing tasks, such as smoothing and surface parametrization, as detailed in the following sections, but also in shape analysis and surface matching.

Finally, we consider the discretization of the  $p$ -Laplacian operator on a mesh domain, defined as

$$(\Delta_p f)_i = \sum_{j \in \mathcal{N}(i)} w_{i,j}^{p/2} \psi_p(f_j - f_i), \quad i = 1, \dots, n, \quad (1.26)$$

with function  $\psi_p : \mathbb{R} \rightarrow \mathbb{R}$  being

$$\psi_p(x) := \frac{1}{p} (|x|^p)' := |x|^{p-1} \text{sign}(x) = |x|^{p-2} x. \quad (1.27)$$

The function  $\psi_p$  is the duality map for the  $L^p$  Banach space. In fact, if  $f \in L^p(\Omega)$ , with  $\Omega$  open bounded subset of  $\mathbb{R}^d$ , then  $\psi_p(f) \in (L^p(\Omega))^* = L^q(\Omega)$ , where  $q$  is the Hölder conjugate of  $p$  (i.e.  $1/p + 1/q = 1$ ). The duality map  $\psi_p$  has a key role in Banach space regularization [107], as a way to transform a primal optimization problem, involving regularization and/or data-fidelity terms in  $L^p$  norm, in an equivalent dual problem with simpler constraints and more efficient resolution.

The generalization (1.26) of the Laplacian operator for different  $p$  values also holds in graph context, where the operator attracted attention from the machine learning community, e.g., the authors in [22] proved the relationship between graph  $p$ -Laplacian and Cheeger cuts, while the case of  $p \in (2, \infty)$  arises in semi-supervised learning [113].

The  $p$ -Laplacian also appears in signal processing and variational filtering strategies which involve the  $p$ -norm of the gradient of an objective function. Moreover, it has been used to build regularization terms of variational models for scene recognition [77] and human activity recognition [78], showing notable improvements when using different values of  $p$  with respect to the classical Laplacian operator.

### 1.2.3 Differential Coordinates

We have described differential operators, highlighting how they encode geometric properties of the surface. This ability allows to represent a mesh not only via the Euclidean coordinates of its vertices, but also via different kinds of so-called Differential Coordinates.

Many geometry processing tasks, like editing, morphing, deformation, blending, and Geometric Texture Transferring, benefit enormously from using such alternative intrinsic mesh representations, because they are easier to adapt to the considered tasks.

We briefly resume the three main alternative local shape descriptors, how they can be computed on a given mesh  $\mathcal{M}$ , which is denoted as the direct problem, and how to solve the ill-posed shape-from-operator inverse problem, i.e. how to recover information about the mesh structures, in particular the vertices, from the operators and the descriptors.

(i) The simplest form of differential coordinates is represented by the **Laplacian Coordinates**, here named LAP, which rely on the discretization  $L_w$  of the continuous linear Laplacian operator at vertex  $v_i$ , as defined in (1.21). The Laplacian Coordinates  $\delta_L \in \mathbb{R}^{n_V \times 3}$  are computed as

$$\delta_L = L_w V. \quad (1.28)$$

As seen in Fig. 1.1(a), the vector  $\delta_L^{(i)}$  represents the displacement from the vertex  $v_i$  and a weighted average of its neighbors. Independently by the choice of uniform or cotangent weights, the matrix has column sums zero. Therefore, if we consider the set of piecewise-constant functions with value 1 on one of the  $K$  connected components of the mesh and 0 on the others, applying the matrix  $L_w$  to such functions returns a null vector. This means that  $L_w$  has rank  $n_V - K$ . It follows that the matrix  $L_w$  is not full rank even for an open connected mesh (where  $K = 1$ ), and, consequently, the operator is never invertible. Therefore, given the  $\delta_L$  coordinates, the inverse process recovers the set of vertices  $V$  by solving for  $V$  the linear system (1.28), with Dirichlet constraint on one vertex, in the least square sense:

$$V^* \in \arg \min_V \|L_w V - \delta_L\|_2^2. \quad (1.29)$$

(ii) A generalization of the Laplacian coordinates consists in the **Normal-Controlled Coordinates** (NCC), introduced in [127]. Fig. 1.1(b) illustrates the construction of these coordinates. The normal to the vertex  $v_i$  defines an orthogonal plane where the neighbor vertices are projected, see Fig. 1.1(b), bottom part. Then the local parametrization of a vertex  $v_i \in V$  is defined with respect to the projected vertices  $v'_j$  with associated weights

$$\bar{w}_{ij} = \frac{\tan(\gamma_{ij}/2) + \tan(\delta_{ij}/2)}{\|v_i - v'_j\|}, \quad (1.30)$$

where the angles  $\gamma_{ij}$  and  $\delta_{ij}$  are shown on Fig.1.1(b). The weights in (1.30), called mean-value coordinates [48], involve the angles formed by the edges of the first ring of the vertex, in the local projected plane. Then, for each vertex  $v_i$  the associated NCC is a vector in  $\mathbb{R}^3$ , describing the local geometry feature at  $v_i$ , which is always parallel to the vertex normal, independently by its definition.

Given the Euclidean coordinates  $V = (V^{(x)}, V^{(y)}, V^{(z)}) \in \mathbb{R}^{n_V \times 3}$ , the associated NCC denoted by  $\delta_N = (\delta_N^{(x)}, \delta_N^{(y)}, \delta_N^{(z)}) \in \mathbb{R}^{n_V \times 3}$  are obtained by the linear mapping

$$\delta_N = N_w V, \quad (1.31)$$

where the weight matrix  $N_w \in \mathbb{R}^{n_V \times n_V}$  is sparse, non-symmetric with elements:

$$(N_w)_{ij} = \begin{cases} 1 & \text{if } i = j \\ -w_{ij} & \text{if } (i, j) \in E \\ 0 & \text{otherwise,} \end{cases} \quad (1.32)$$

and  $w_{ij}$  are the normalized weights  $w_{ij} = \bar{w}_{ij} / \sum_{j \in \mathcal{N}(v_i)} \bar{w}_{ij}$ .

As for  $L_w$ , also  $N_w$  has column sums zero and rank  $n_V - K$ . Therefore, given the  $\delta_N$  coordinates, the inverse process recovers a set of coordinates  $V$  by solving the linear system (1.31), with Dirichlet constrain on one vertex, in the least square sense:

$$V^* \in \arg \min_V \|N_w V - \delta_N\|_2^2. \quad (1.33)$$

(iii) Finally, we consider the **Mean Value Encoding** (MVE) coordinates, introduced for planar triangulation in [48] and then generalized in  $\mathbb{R}^3$  [67], as an evolution of Pyramid Coordinates [111]. The construction of the MVE coordinates is illustrated in Fig. 1.1(c). For each vertex  $v_i \in V$ , an approximated normal  $n_i$  is computed as

$$n_i = \frac{\sum_{j=1}^{|\mathcal{N}(v_i)|} (v_j - l) \times (v_{j+1} - l)}{\left\| \sum_{j=1}^{|\mathcal{N}(v_i)|} (v_j - l) \times (v_{j+1} - l) \right\|}, \quad l = \frac{1}{|\mathcal{N}(v_i)|} \sum_{j \in \mathcal{N}(i)} v_j. \quad (1.34)$$

Then the associated MVE representation is uniquely obtained by first computing the set of coefficients  $(w_{ij}, b_{ij}), j \in \mathcal{N}(i)$ , defined for each half-edge  $w_{ij} \neq w_{ji}, b_{ij} \neq b_{ji}$  as follows

$$w_{ij} = \bar{w}_{ij} / \sum_{j \in \mathcal{N}(v_i)} \bar{w}_{ij}, \quad \bar{w}_{ij} = \frac{\tan(\gamma_{ij}/2) + \tan(\delta_{ij}/2)}{\|v'_i - v'_j\|} \quad (1.35)$$

$$b_{ij} = \frac{c_{ij}}{\sqrt{1 - c_{ij}^2}}, \quad c_{ij} = \frac{(v_i - v_j) \cdot n_i}{\|v_i - v_j\|}. \quad (1.36)$$

Given the Euclidean coordinates  $V = (V^{(x)}, V^{(y)}, V^{(z)})$ , the associated MVE-coords  $\delta_M = (\delta_M^{(x)}, \delta_M^{(y)}, \delta_M^{(z)})$  are obtained by the following non-linear mapping

$$\delta_M = (\mathcal{V}_1(V), \mathcal{V}_2(V), \mathcal{V}_3(V)) \quad \text{with} \quad \mathcal{V}_i(V) = \sum_{j \in \mathcal{N}(i)} w_{ij} (\|v'_i - v'_j\| b_{ij} + (v_j - v'_j) \cdot n_i) n_i. \quad (1.37)$$

Note that  $\delta_M$  represents the displacement of  $v_i$  from  $v'_i$ , its projection onto the local projection plane orthogonal to  $n_i$ .

Given the  $\delta_M$  coordinates, the inverse process recovers the Euclidean Coordinates of  $v_i \in V$  by the following closed-form expression as a function of the rest of the neighborhood vertices

$$v_i = v'_i + \delta_M = F_i(V; w, b) = \sum_{j \in \mathcal{N}(i)} w_{ij} \left( v_j + \|N_i\| \sum_{k \in \mathcal{N}(i)} w_{ik} (v_k - v_j) \|b_{ij} n_i\| \right), \quad (1.38)$$

with  $N_i = I_3 - n_i n_i^T$ . Note that  $F_i(V; w, b)$  has no direct dependence on  $v_i$ , but only on the first-ring vertices of  $v_i$  and on the normal vector  $n_i$  at  $v_i$ . As mentioned in [67],

MVE coordinates depend continuously on the data, thus small variations of vertices in  $\mathcal{N}(v_i)$  result in a small variation of  $v_i$ .

Given the MVE coordinates, the inverse process recovers  $V$  by solving the non-linear least squares minimization problem:

$$V^* \in \arg \min_V \|V - F(V)\|_2^2 = \sum_i \|v_i - F_i(V; w, b)\|_2^2 \quad (1.39)$$

with the non-linear function  $F_i(V)$  defined in (1.38), and the components of the vectors  $w$  and  $b$  defined in (1.35) and (1.36), respectively.

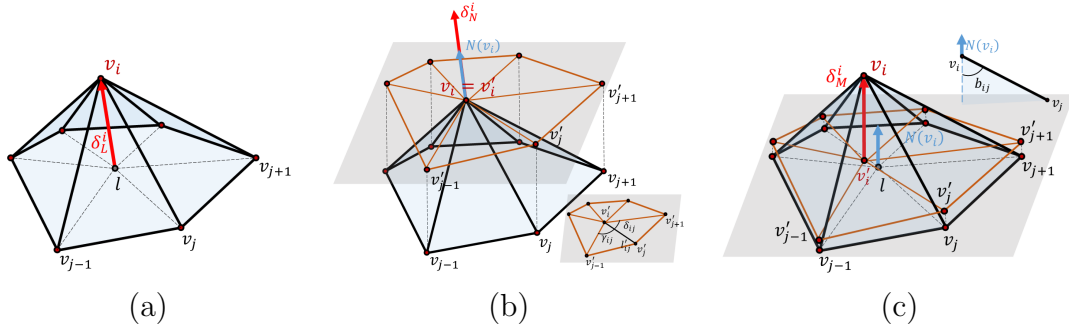


Figure 1.1: Local shape descriptors construction: (a) LAP:  $\delta_L^i$ , (b) NCC:  $\delta_N^i$ , with projection plane, (c) MVE:  $\delta_M^i$ , with cotangent  $b_{ij}$ .

In the following Sections, we describe how some of these Differential Coordinates inspired methods for solving different mesh processing tasks.

Furthermore, in Chapter 3 we propose a variational model for Geometric Texture Transferring, using LAP, NCC and MVE encodings. After analyzing invariance/equivariance properties of LAP, NCC and MVE with respect to affine transformations, we describe the construction of the models, explore different resolution methods and finally compare the obtained results.

## 1.3 Surface processing tasks

Surface processing includes a wide variety of tasks which range from the acquisition to the reconstruction, analysis and editing of surfaces. In this Section, we present some of the main surface processing problems that apply to surfaces approximated with triangular meshes. Figures 1.2-1.3 report a visualization of the analyzed tasks.

### 1.3.1 Repairing

A geometric model can be acquired from a real 3D object through a scanning process. The result may be affected by artifacts that make it unsuitable for successive processing.

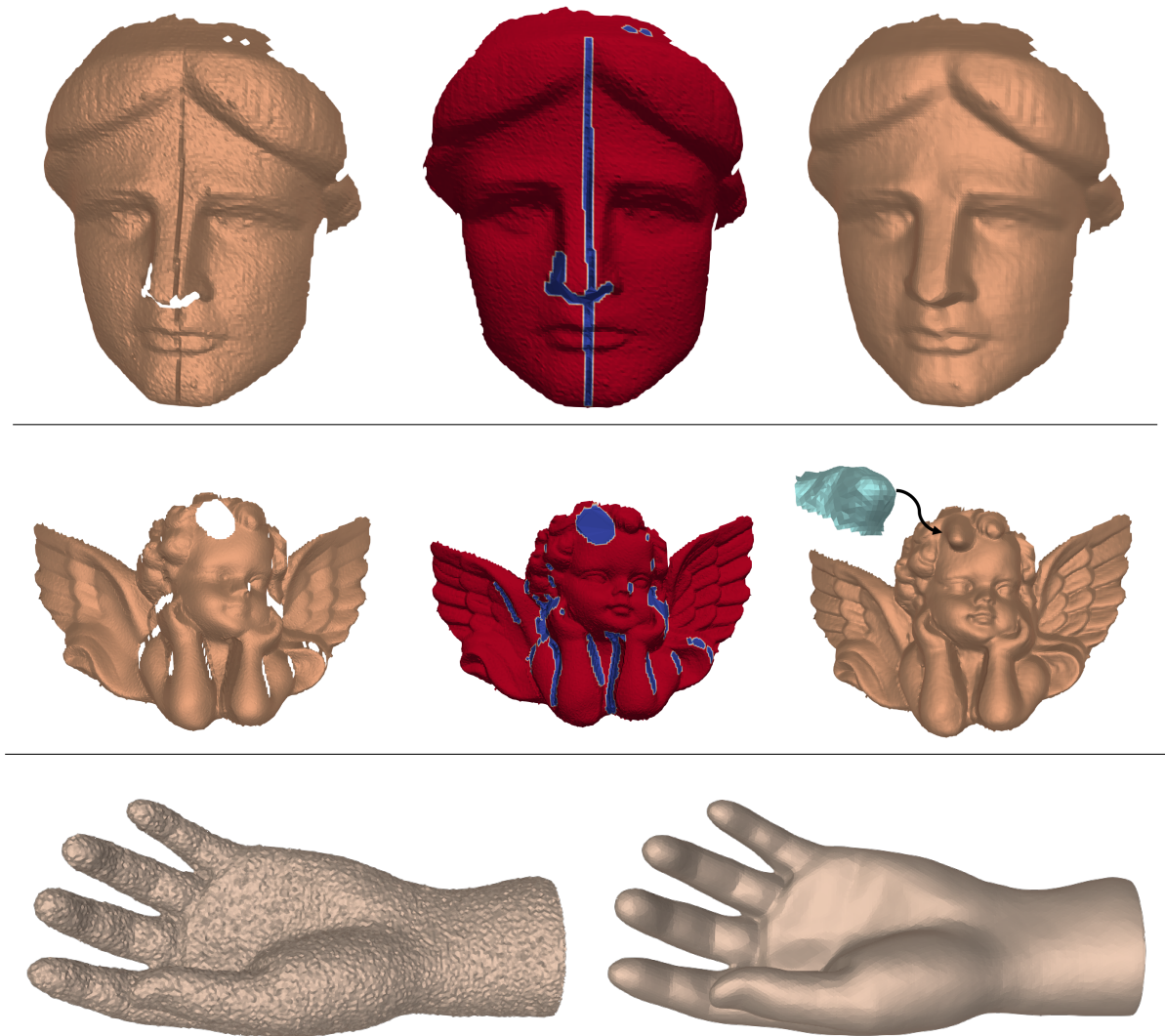


Figure 1.2: Mesh processing tasks. First row: hole-filling/inpainting. Second row: completion. Third row: denoising.

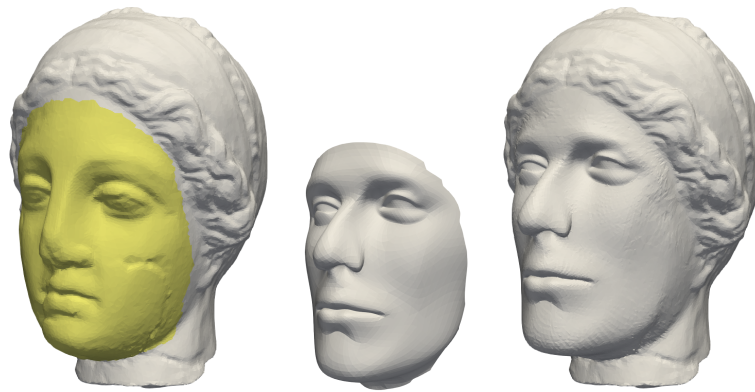
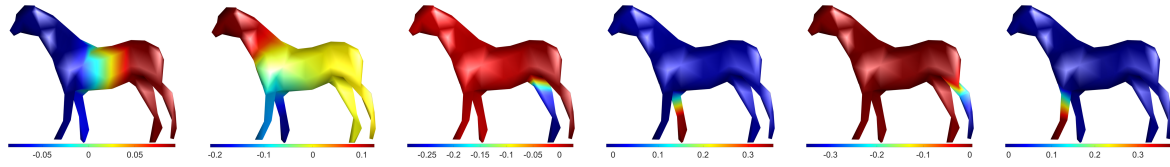


Figure 1.3: Mesh processing tasks. First row: spectral analysis. Second row: cloning. Third row: Geometric Texture Transfer.



In such cases, it is necessary to apply algorithms that repair the model in its problematic regions while keeping its shape and features in the others.

Many kinds of artifacts may occur in a mesh, like inconsistent face orientation, non-manifold vertices, non-manifold edges, gaps, overlaps, self-intersections. For each type, there exists a variety of possible correction methods, that we can categorize as (i) volumetric and (ii) surface-oriented.

(i) Volumetric algorithms convert the input model into a volumetric representation, partitioning the embedding space into blocks classified as inside the surface, outside the surface or intersected by the surface, similarly to an implicit representation. For their nature, such methods eliminate these kinds of artifacts, in an automatic and quite robust way. On the negative aspects, they require a resampling of the model, losing information about the original connectivity structure and with the risk of deleting or modifying features of the object. Moreover, to achieve good precision it is required to have a fine partitioning, returning an object with many triangles, with consequent high memory cost.

(ii) Surface-oriented algorithms act directly on the original mesh by adding/removing/modifying only a few triangles around the region affected by the artifacts, preserving the information in the furthest areas. However, identifying where the artifacts are located is not trivial and it often requires a user interaction with a manual pre-processing. In some pathological cases, gaps or face intersections are impossible to be removed.

A further type of artifacts we consider consists of holes, intuitively recognizable as regions  $S_D$  of the surface  $\mathcal{S}$  that have not been captured by the scanning process or that were missing even in the original 3D object.

In the first case, the local lack of data derives from occlusions, surface reflection and scanner placement constraints that are not avoidable, despite the remarkable progress achieved in the fields of 3D scanning. The second case is common in the context of digital restoration of cultural heritage art-works, where the scanned object itself (e.g., the archaeological findings) may be incomplete and damaged due to the fact that some of its (missing) parts have been ruined over time due to wear and tear. In both scenarios, to facilitate the downstream processing of its digital content, the object shape needs to be repaired.

We distinguish two types of hole-filling tasks, depending on the known information about the missing region:

- *Smooth hole filling/Inpainting*: as shown in Fig.1.2 (first row) inpainting is the process of recovering a missing or damaged region in the surface by filling it in a

plausible way using available information. The result of a surface-inpainting operation depends on the specific application considered. In digital cultural heritage restoration, for instance, surface inpainting is understood as the recovery of the holes in the data or the removal of the scratches/cracks possibly present in the scanned objects. In prototype manufacturing, the goal is shifted towards a water-proof virtual reconstruction, so that the related operation is rather interpreted as a smooth hole filling. In either case, all damaged areas should be filled in a seamless way that is minimally distinguishable from their surrounding regions.

- *Context-aware completion*: when a priori knowledge on the missing/damaged parts of the scanned model is known, it is desirable that the completion of the damaged areas occurs by pasting known data - such as template patches - automatically or semi-automatically under user guidance. This allows, for example, to repair a damaged part of an artifact by filling the region of interest with a patch taken from a valid/undamaged region of the model itself or even from other 3D geometric models. The completion process is shown in Fig.1.2 (second row).

Notable volumetric approaches tackle the problem using an implicit representation of the surface, via signed distance functions [38, 5], Radial Basis Functions implicit interpolations [29] or Moving Least Squares [125].

On the other hand, in surface-oriented approaches  $\mathcal{S}$  is approximated as a mesh  $\mathcal{M}_0$  and the hole-filling problem can be formulated as follows.

*Given a mesh  $\mathcal{M}_0 = (V_0, E_0, T_0)$  with a damaged region  $S_D$ , identified by the set  $b_0$  of its boundary vertices in  $\mathcal{M}$ , the hole-filling algorithm has the goal of defining a repaired mesh*

$$\mathcal{M}^* = (V^*, E^*, T^*), \quad \text{with} \quad V^* = V_0 \cup V_{\mathcal{P}} \quad (1.40)$$

*where  $V_{\mathcal{P}}$  is the set of vertices of a patch  $\mathcal{P}$  that closes the hole in  $S_D$ .*

The union operation in (1.40) is well-defined, only with the compatibility assumption that the boundary  $b_{\mathcal{P}}$  of the patch and boundary  $b_0$  of the damaged region have the same number of vertices. If this is not the case, a suitable subdivision process can be preliminarily applied. The patch is then deformed such that the vertices in  $b_0$  and  $b_{\mathcal{P}}$  are in the same position and the hole is properly closed.

Among such methods, it is worth citing [74], where new triangles are created optimizing a quality function depending on angles and areas of the filling patch, then refined to make the triangulation uniform as in the surrounding part of the mesh, and finally smoothed.

In Chapter 2, we propose an algorithm that tackles the hole-filling task, but also context-aware completion, enclosing the hole with a patch and finally applying smoothing on the common boundary through a variational approach.

### 1.3.2 Smoothing

Smoothing is a geometry processing task that consists of removing high-frequency details of a surface, returning a smoother version of the same object. We distinguish two types of smoothing: (i) denoising and (ii) fairing.

(i) Denoising, as shown in Fig.1.2 (third row), consists in removing the undesired noise that corrupts a surface, while preserving original features, including edges, creases and corners. Noise is an unavoidable consequence of the acquisition errors that occur when scanning a 3D object.

*The mesh denoising problem aims at recovery a mesh  $\mathcal{M}^* = (V^*, E_0, T_0)$ , as accurate as possible to an unperturbed (and unknown) original mesh  $\mathcal{M}_{GT} = (V_{GT}, E_0, T_0)$ , starting from a perturbed mesh  $\mathcal{M}_0 = (V_0, E_0, T_0)$  corrupted by additive noise.*

While the connectivity encoded in  $E_0$  and  $T_0$  remains the same, the vertices degradation model reads as follows:

$$V_0 = V_{GT} + \varepsilon, \quad (1.41)$$

where  $\varepsilon \in \mathbb{R}^{n_V \times 3}$  is a random noise vector under a Gaussian distribution.

Such degradation modifies the surface locally increasing its roughness, mathematically interpretable as its normal deviation or curvature, in particular in smooth regions. For this reason, many denoising models aim at reducing the curvature, with the crucial point of distinguishing which high-frequency details represent noise and which represent features of the underlying original surface.

(ii) Fairing, on the other hand, returns as-smooth-as-possible surface patches, exploiting the concept of curvature or higher-order derivatives. While denoising methods focus on the high-frequency details, surface fairing follows the principle of simplest shape, removing all kinds of details in a selected Region of Interest (ROI) of the surface  $\mathcal{S}$ , while fixing its boundary and preserving the low-frequency global shape.

### 1.3.3 Spectral Analysis

Spectral methods exploit differential properties of meshes to analyze their geometry in the frequency domain, providing insights into shape details and enabling advanced processing techniques.

In 1D signal processing, the signal frequency spectrum is commonly analyzed through the Fourier transform, which decomposes data  $f$ , sampled in  $n$  points, into a linear combination of basis functions  $(\phi_i)$ , that are eigenfunctions of the Laplacian operator.

Ordering the basis by increasing frequencies, i.e. increasing eigenvalues, we get the decomposition

$$f = \sum_{i=1}^n \langle f, \phi_i \rangle \phi_i. \quad (1.42)$$

The same scheme is valid also in surface processing, using as data the  $x, y, z$  coordinates of the surface  $\mathcal{S}$  and as basis the eigenfunctions of the Laplace-Beltrami operator defined in (1.14). The first basis functions encode low-frequency information about the global shape of the object, while the last ones describe the high-frequency details.

*The eigendecomposition problem for a mesh  $\mathcal{M}$  approximating a surface  $\mathcal{S}$  consists in computing the Manifold Harmonic Basis constituted by the set of the  $n_V$  normalized eigenfunctions  $\mathcal{B} = \{\mathbf{e}_1, \dots, \mathbf{e}_{n_V}\}$  of the discrete Laplacian  $L_w$ , solving for  $\phi \in \mathbb{R}^{n_V}$  the equation*

$$L_w \phi = \lambda \phi.$$

The decomposition acts on the vertex coordinates  $V \in \mathbb{R}^{n_V \times 3}$  as

$$V = \sum_{i=1}^{n_V} \langle \mathbf{e}_i, V \rangle \mathbf{e}_i. \quad (1.43)$$

A visualization of the eigenfunctions is reported in Fig.1.3 (first row), while methods for their computation are discussed in Section 1.4.

This spectral decomposition allows for a low-dimensional approximation of the mesh, obtained by truncating at  $k < n_V$  the sum in (1.43). In fact, cutting the sum in (1.43) to the first  $k < n_V$  terms excludes high-frequency information, usually corresponding to noise, achieving a denoised mesh. The same principle applies also in compression, since the surface can be well-represented by the  $n_V - k$  low-frequency eigenpairs [63].

Another application is segmentation/classification [89, 60]. The values of the first  $k < n_V$  eigenfunctions on each vertex of a mesh represent a new coordinate system in the embedding space  $\mathbb{R}^k$ . On this new system, one can apply any clustering algorithm (for example  $K$ -means), returning a segmentation into  $k$  regions [30]. Alternatively, one can consider a Principal Component Analysis interpretation of the Laplacian eigenvectors of the mesh.

### 1.3.4 Deformation

Surface deformation includes different kinds of operations that act on one or more meshes, returning an output surface mesh that can be intuitively viewed as a composition or modification of their geometric features and shape.

For example, morphing is the (continuous) deformation that transforms a source surface into a target surface, while preserving geometric or topological constraints. It is commonly used in animations to interpolate between different 3D models. When the two surfaces  $\mathcal{S}_0, \mathcal{S}_1$  are represented by two parametrizations  $X_0, X_1$  on the same domain  $\Omega$ , the intermediate surfaces at time  $\tau \in [0, 1]$  are

$$\mathcal{S}_\tau = X_\tau(t, s) := f(\tau)X_0(t, s) + (1 - f(\tau))X_1(t, s)$$

with  $f : [0, 1] \rightarrow [0, 1]$  a smooth blending function, that may be linear or non-linear (see for example cubic interpolation, harmonic interpolation or spherical linear interpolation).

A similar concept is shape editing, where the target mesh is derived from the source mesh through a transformation (such as bending, stretching, or twisting) that alters its global shape while preserving the original local geometric details. It is used to create new shapes while maintaining certain aesthetic or functional properties. For polygonal meshes, the deformation is thought as the application of displacement vectors  $(d_1, \dots, d_n)$  to the vertices  $(v_1^0, \dots, v_n^0)$  of the original mesh:

$$V^1 = V^0 + d. \quad (1.44)$$

*Given a source mesh  $\mathcal{M}_0 = (V^0, E^0, T^0)$ , with fixed vertices  $\{v_i^0, i \in F\}$  and displaced handle vertices  $\{v_i^1 = v_i^0 + d_i, i \in H\}$ , the shape editing task consists in deriving the displacement vectors  $\{d_i, i \notin F \cup H\}$  such that the resulting target mesh  $\mathcal{M}_1 = (V^1, E^0, T^0)$  has different global shape but preserved local details.*

A heuristic approach to mesh deformation consists of a simple transformation propagation [11, 97]. The transformation that moves the handles into the new positions gets extended to the free vertices, but its effect decreases linearly with respect to a distance function depending on the handles and the fixed region. This method is simple and computationally efficient, but may lead to not-intuitive results.

Among the class of model modifiers, which combines two or more surfaces into a novel one, we distinguish boolean operations, such as union, intersection or difference, cloning/blending and geometric texture transfer.

Boolean operations are easily defined between implicit surfaces, via an appropriate application of max and min operators to the respective implicit functions. For example, if  $\mathcal{S}_1 = \{f_1 = 0\}$  and  $\mathcal{S}_2 = \{f_2 = 0\}$ , with  $\mathcal{S}_1, \mathcal{S}_2$  closed and  $f_1, f_2$  signed distance functions as defined in (1.3) we have:

$$\begin{aligned} \mathcal{S}_1 \cap \mathcal{S}_2 &= \{\max(f_1, f_2) = 0\}, & \mathcal{S}_1 \cup \mathcal{S}_2 &= \{\min(f_1, f_2) = 0\}, \\ \mathcal{S}_1 \setminus \mathcal{S}_2 &= \{\max(f_1, -f_2) = 0\}, & \mathcal{S}_2 \setminus \mathcal{S}_1 &= \{\max(-f_1, f_2) = 0\}. \end{aligned}$$

A slightly different task is cloning/blending, visualized in Fig.1.3 (second row). It refers to the process of replacing a region of interest ROI of a given surface  $\mathcal{S}_1$  with a patch  $P$  extracted from another surface  $\mathcal{S}_2$ , thus creating a new 3D object  $\mathcal{S}^*$  with seamless transition on the boundary between the two shapes, without noticeable sharp edges or discontinuities.

When the surfaces are approximated as meshes, the cloning process is formalized as follows:

*Given two meshes  $\mathcal{M}_1, \mathcal{M}_2$  and two sub-meshes  $ROI \subset \mathcal{M}_1, P \subset \mathcal{M}_2$ , the cloning task consists in computing a mesh*

$$\mathcal{M}^* = (\mathcal{M}_1 \setminus ROI) \cup ROI^* \quad (1.45)$$

*where  $ROI^*$  has the same geometric shape and detail of the patch  $P$ , with smooth transition on the boundary  $b_{ROI^*}$ .*

Finally, we consider the Geometric Texture Transfer (GTT) modifier, for which we propose a model in Chapter 3. This is a fundamental task in computer graphics and geometric modeling, and it consists in making the macrostructure of an object (base surface) appear wrinkled, wavy and embossed.

In computer graphics for example, the bump mapping technique [15] produces this effect as an apparent variation of the local geometry due to a variation of the normal in a point, which is used in the local lighting model that produces a variation in shading and thus rendering of the surface. The geometric normal of the object remains unchanged and so does the geometry of the object.

Instead, in many engineering and industrial design applications an effective surface deformation is required. In these contexts, GTT extracts a fine detail pattern (geometric texture) from a source surface, and transfers it to a target surface. This process, as shown in Fig.1.3 (third row), alters the local geometry by applying small-scale features, often for enhancing visual details, without changing the overall shape.

The GTT problem can be formalized similarly to the cloning task.

*Given a base mesh  $\mathcal{M}_I$ , with a patch  $P \subset \mathcal{M}_I$ , and a geometric texture  $\mathcal{M}_S$ , the desired textured mesh is computed as*

$$\mathcal{M}_T = (\mathcal{M}_I \setminus P) \cup P^* \quad (1.46)$$

*where  $P^*$  has the global shape of the original patch  $P$  and the local texture of  $\mathcal{M}_S$ .*

The unknown  $P^*$  can have a completely new connectivity, or be a deformation of the initial  $P$ , with the same underlying triangulation.

In the latter case, many proposed methods rely on a consistent surface parametrization between source and target shape. In this context, following the theory of classical displacement mapping [37], surface texturing is a 3-dimensional extension of traditional image-based texture mapping [43]. Let us denote the mesostructure surface  $\mathcal{S}_T$  by the parametric form  $X_T(t, s) : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}^3$ , the macrostructure base surface  $\mathcal{S}_I$  by  $X_I(t, s) : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}^3$ , and the displacement (geometric texture) by a height map function  $X_S(t, s) : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$ . The height map is a grayscale texture image in the texture coordinates  $t, s \in [0, 1]^2$ . Displacement mapping allows to obtain a mesostructure  $\mathcal{S}_T$  by mapping on a macrostructure base surface  $\mathcal{S}_I$  a geometric texture  $\mathcal{S}_S$  in the direction of the macrostructure normal vector  $N_I(t, s)$ . Assuming that both  $\mathcal{S}_I$  and  $\mathcal{S}_S$  share a common parametrization domain, the displacement mapping reads as

$$X_T(t, s) = X_I(t, s) + N_I(t, s)X_S(t, s). \quad (1.47)$$

A natural extension from the texture image was presented in [42] where the displacement map is described via trivariate functions over a volume parametric form. A further surface normals perturbation is proposed in [15], using a wrinkle function to address specific computer graphics visualization purposes.

In Chapter 3, we propose a model for GTT that works also on surfaces with arbitrary topology, without requiring a bijective relation between the base surface and the texture.

## 1.4 Differential Models for Shape Analysis

Differential methods are widely used in the fields of geometry and mesh processing, offering powerful tools for analyzing, representing, and manipulating the geometry of surfaces. These techniques exploit differential geometry concepts described in Section 1.2 for their ability to encode information about shape and details of a surface. The core advantage is to provide smooth, continuous representations of geometric structures and to act on these structures in an interpretative way for solving various 3D modeling tasks.

In general, differential approaches consist in Partial Differential Equations derived from physical principles (e.g. heat equation) or geometric properties (e.g. mean curvature flow). The desired result can be obtained as the solution of the PDE or as the steady-state of an evolutive scheme, following a time-discretization step.

Differential methods usually act on scalar or vector fields defined over the surface, represented continuously, as in parametric surfaces, or discretely, as a mesh. Such fields may be the parametric representation, the 3D vertex coordinate, or geometric properties of the surface such as curvature and normal vectors.

In the following, we provide a comprehensive overview of the differential models of interest. These models, characterized by their specific PDE expressions and the types of variables involved, offer effective solutions to the processing tasks enumerated in Section 1.3.

One of the most widely used PDEs in image and surface processing is the anisotropic second-order non-linear **diffusion equation**,

$$\begin{cases} \frac{\partial f(x,t)}{\partial t} = \operatorname{div}(D(x,t)\nabla f(x,t)) & \forall (x,t) \in \Omega \times [0,T] \\ f(x,0) = f_0(x) & \forall x \in \Omega \end{cases} \quad (1.48)$$

with  $\Omega$  an open subset of  $\mathbb{R}^n$  and  $f_0$  initial data, and suitable Dirichlet or Neumann boundary constraints on  $\partial\Omega$ . The diffusion tensor  $D \in \mathbb{R}^{n \times n}$  is a symmetric positive definite matrix, which varies on  $x \in \Omega \subset \mathbb{R}^n$  and  $t \in [0, T]$ . The tensor  $D$  guides the diffusion in time of the data  $f$  along specific directions. Its value may depend on  $(x, t)$  directly, as in the linear case, or through the data itself, as  $D = D(f(x, t))$ , in the non-linear case. Specific definitions of the tensor generate different diffusion models.

An example of isotropic diffusion is the **Perona-Malik diffusion** [100] model

$$\frac{\partial f(x,t)}{\partial t} = \operatorname{div}(c(|\nabla f(x,t)|)\nabla f(x,t)) \quad (1.49)$$

where the tensor in (1.48) is replaced by a scalar function  $c : \mathbb{R} \rightarrow \mathbb{R}_+$ ,  $D(x, t) = c(|\nabla f(x, t)|)$ . Setting  $c$  as a decreasing function, such as the originally proposed

$$c(t) = \exp(-(c/K)^2) \quad \text{or} \quad c(t) = \frac{1}{1 + (c/K)^2}$$

the model produces an edge-stopping diffusion effect, because the diffusion is controlled by local information on  $f$  in the different regions of the domain.

This PDE model is non-linear since the diffusion intensity depends on the gradient of the function  $f$ .

In the simple case of a constant diffusion coefficient  $c(x, t) \equiv c > 0$ , the generic diffusion PDE (1.48) reduces to the linear **diffusion flow**

$$\frac{\partial f(x,t)}{\partial t} = c\Delta f(x,t). \quad (1.50)$$

The isotropic diffusion effect of the model (1.50) can be exploited for surface smoothing. In fact, if  $f$  represents a smooth surface in 3-dimensional Euclidean space, and  $\Delta$  is the Laplace-Beltrami operator (see (1.25) for the mean curvature), then (1.50) is the non-linear parabolic PDE so-called **mean curvature flow** and  $\frac{\partial f(x,t)}{\partial t}$  stands for the normal velocity of the surface. In the field of differential geometry, mean curvature flow is an example of a geometric flow of hypersurfaces in a Riemannian manifold. Intuitively, a family of surfaces evolves under mean curvature flow if the normal component of the velocity at which a point on the surface moves is given by the mean curvature of the surface.



While the conventional diffusion equation is a linear parabolic partial differential equation and does not develop singularities (when applied forward in time), mean curvature flow may develop singularities because it is a non-linear parabolic equation. To mitigate this issue, additional constraints must be imposed on the surface to ensure regularity under mean curvature flow.

This simple PDE model, although powerful, may cause shrinkage effects. This is avoided by **Taubin smoothing** [120], which alternates smoothing steps with inflating steps, characterized by a negative coefficient  $c_2 < 0$ .

These isotropic models act on the surface uniformly, with the risk of smoothing important features like edges, corners or creases. On the other hand, applying the Perona-Malik model (1.49), diffusion is enhanced in smooth regions, where the gradient is small, while it is reduced in regions with high-gradient magnitudes. This method outperforms mean curvature flow, but the price to pay is a more challenging resolution due to its non-linearity.

Due to their smoothing effect, mean curvature flow, Taubin smoothing and Perona-Malik model are exploited for mesh denoising, with an appropriate tuning of the diffusivity function  $c(x, t)$  and of the stopping time  $T$ .

Rather than analyzing the time-dependent diffusion equation, one may focus on the corresponding steady-state solution. In the linear case, the Laplace equation  $\Delta f = 0$ , whose solutions are the harmonic functions, is usually coupled with boundary conditions as follows

$$\begin{cases} \Delta f = 0 & \text{on } \Omega \\ f = f_0 & \text{on } \partial\Omega \end{cases} \quad (1.51)$$

Because of the smoothing effect of the Laplacian  $\Delta$ , the solution is the smoothest possible function of a given boundary. In surface processing, interpreting  $f$  as the Euclidean coordinates, the equation describes the **minimal surface** fairing model. To the aim of finding the surface with zero mean curvature, the minimal surface fairing method consists in computing the surface with minimal mean curvature, given a fixed boundary position.

Drawing inspiration from these models and techniques borrowed from image processing, a variety of non-linear and anisotropic approaches [119, 82, 88] have been developed in recent years to address the challenges of mesh denoising and fairing. For example, while the  $\Delta$  operator indicates local smoothness or stretching of the data  $f$ , the bi-harmonic operator  $\Delta^2$ , which involves the fourth-order partial derivatives, measures the smoothness of the curvature, which we can think of as bending. The equation

$$-\kappa_s \Delta f + \kappa_b \Delta^2 f = 0 \quad (1.52)$$

with additional boundary constraints, minimizes stretching and bending effects, tuned by corresponding stiffness parameters  $\kappa_s$  and  $\kappa_b$ . This equation is used for mesh deformation,

to derive the optimal displacement vectors  $d$ , defined in (1.44), given their fixed values on a region  $F$  and on handles  $H$ .

While the differential models in (1.51-1.52) aim at minimizing certain smoothness characteristics of the function  $f$ , by setting to zero the value of the involved differential operators, Poisson models exploit the known information given by a guiding vector field  $\phi$  as follows:

$$\Delta f = \text{div}(\phi) . \quad (1.53)$$

Compared to the Laplace equation  $\Delta f = 0$ , Poisson models describe equilibrium states influenced by internal sources, described by the field  $\phi$ .

Poisson models have been widely used in mesh processing. For example, in [65], the vector field  $\phi$  is computed from a discrete sampling of points of an unknown surface  $\mathcal{S}$  and then used through (1.53) to derive the corresponding implicit function  $f$ . In [137], the authors perform mesh editing, starting from a mesh with vertices  $V_0$ , setting  $\phi$  as a modification of  $\nabla V_0$  and interpreting  $f$  as the edited vertices  $V^*$ .

An analog approach characterizes the Laplacian coordinates inverse reconstruction problem, discussed in Sec.1.2.3. In [114], the deformation consists in solving for  $V'$  the linear system

$$L_w V' = T(V') \delta_L$$

where  $T$  is a transformation linearly dependent from the original positions  $V$  and the unknown deformed  $V'$ .

Normal-Controlled Coordinates are instead used in [127], solving the system

$$N_w V' = \delta'_N$$

where  $\delta'_N$  are the NCC of the deformed mesh. Since they are unknown, an iterative algorithm is proposed, where at each step  $k$  the vertex coordinates  $V^{(k)}$  are computed from  $\delta_N^{(k-1)}$ , while the updated  $\delta_N^{(k)}$  derives from the new vertex normals.

A further example of time-independent PDE involving the Laplace operator is represented by the eigendecomposition problem

$$\Delta f = \lambda f . \quad (1.54)$$

We have described in Sec.1.3.3 how, in the mesh processing context, solving this equation for the discrete Laplace-Beltrami operator  $L_w$  allows to perform a spectral decomposition of the mesh.

In general, solving the problem  $Lf = \lambda f$  is highly expensive for any discrete linear operator  $L$ . In the case  $L = L_w$ , the sparsity of the matrix allows to have a reasonable cost

of  $O(n_V)$ , but when  $n_V$  is large the computational methods become numerically unstable, due to adjacent eigenvalues becoming too close. However, the matrix is symmetric and positive semi-definite and therefore the basis  $\mathcal{B}$  is orthogonal.

Moreover, as seen in Sec.1.2.3, the matrix  $L_w$  in (1.20) has column sums zero and, for meshes with a single component, rank  $n_V - 1$ . Therefore, the first eigenvalue is 0, with corresponding constant eigenfunction.

Since in many applications we are interested only in the first  $m$  eigenfunctions, an alternative resolution approach computes them one after the other. First, we notice that  $\phi$  is an eigenfunction if and only if it is a critical value of the Rayleigh quotient

$$R(f) = \frac{f^T L f}{f^T f}$$

with corresponding eigenvalue  $R(\phi)$ . Therefore, exploiting the known orthogonality, we solve the  $m - 1$  optimization problems

$$\phi_{k+1} = \arg \min_f R(f) \quad \text{s.t. } \langle f, \phi_i \rangle = 0 \quad \forall i = 1, \dots, k \quad (1.55)$$

where the first eigenfunction  $\phi_1$  is the trivial constant.

Many authors have tried to extend the analysis of Laplacian properties to the non-linear case of the  $p$ -Laplacian operator. For example, the  $p$ -Laplacian flow, defined as

$$\frac{\partial f}{\partial t}(x, t) = \Delta_p f(x, t) \quad (1.56)$$

has been successfully applied to mesh decimation [87], and to the construction of Graph Neural Networks [50].

At the same time, spectral analysis of the  $p$ -Laplacian operator is one of the many non-linear eigenproblems that have gained in popularity over the last years with applications in data clustering, spectral graph theory, dimensionality reduction and machine learning problems [36, 81, 27, 22, 113, 96].

The  $p$ -Laplacian eigendecomposition problem, which is one of the most classical examples of non-linear eigenvalue problem, has been studied both in the continuous and discrete setting [64, 41, 51]. However, there are few results related to the spectrum of such non-linear operator, even on its dimensionality. For the  $p$ -Laplacian, countability or finiteness of its spectrum are open problems [140]. In the discrete setting the number of eigenvalues of the  $p$ -Laplacian can also exceed the dimension of the space, as shown in [1, 39] with some simple examples.

Although for the special case  $p = 2$  analytic solutions exist, there are no general analytic solutions for the non-linear  $p$ -Laplacian eigenproblem when  $p \in (1, 2)$  or  $p \in (2, +\infty)$ .

The eigenpair computation is hard to perform, because non-linearity adds several difficulties, such as the loss of orthogonality. In Chapter 4 we briefly describe some methods presented in literature and we propose a novel model, computing  $p$ -Laplacian eigenpairs using an approach similar to (1.55), exploiting a generalization of the notions of Rayleigh quotient and orthogonality to the case  $p \neq 2$ . Furthermore, we show spectral clustering results on different meshes and for different values of  $p$ , proving that  $p$ -Laplacian is better than its linear counterpart in providing valuable information about curvature and topology of the different regions of the object, facilitating mesh segmentation.

The PDE models described up to this point involve only a diffusion term, in various forms (linear, non-linear, isotropic, anisotropic) and in various settings (time-evolutive or as steady-states).

Notable importance have PDE models where the diffusion term is accompanied by a transport term, combining the effects of spreading (diffusion) and directed movement (transport). The general form of such transport-diffusion PDEs reads as

$$\begin{cases} \frac{\partial f(x,t)}{\partial t} = \operatorname{div}(D(x,t)\nabla f(x,t)) - \operatorname{div}(\mathbf{d}(x,t) f(x,t)) & \forall (x,t) \in \Omega \times [0, T] \\ f(x, 0) = f_0(x) & \forall x \in \Omega \end{cases} \quad (1.57)$$

to which it must be added a boundary condition on  $\partial\Omega$ . The novel function  $\mathbf{d} : \Omega \times [0, T] \rightarrow \mathbb{R}^n$ , called drift, is a velocity vector field guiding the transport effect.

In Chapters 5 and 6, we focus on a specific diffusion-transport PDE, called osmosis, originally defined in [129] as

$$\frac{\partial f(x,t)}{\partial t} = \Delta f(x,t) - \operatorname{div}(\mathbf{d}(x) f(x,t)) \quad (1.58)$$

This PDE is a particular case of (1.57), with  $D(x,t) \equiv I_n$ , and  $\mathbf{d}$  constant in time. As many other diffusion-transport PDEs, it allows to describe many physical, biological, and chemical processes where substances are both dispersed and carried by a flow. For example, in biology [18] osmosis indicates a transport process where molecules pass through a semipermeable membrane in such a way that, at its steady state, the liquid concentration on each side of the membrane may differ. Due to its non-trivial (i.e. non-constant) steady states, defined by the drift vector field, such a process can be seen indeed as the non-symmetric counterpart of standard diffusion processes in the sense that, during evolution, the probability of moving from inside to outside the cell through the membrane is not equal to the probability of performing the reverse process [53]. Although being defined through a guiding vector field as in Poisson models [99], the osmosis model is invariant under multiplicative changes.

In Chapter 5 we report the definition of the linear osmosis model, obtained from (1.57) setting  $D(x,t) \equiv I_n$ , its conservation properties and how it was applied to solve different image processing tasks [129].

Furthermore, we analyze its theoretical properties in terms of wellposedness, and, most importantly, we derive a non-local version where the differential operators have been replaced by corresponding integral operators. After studying the wellposedness and regularity of the non-local solutions, we study their convergence to solutions of the local model. From the non-local model, we derive an appropriate discretization of the PDE model into graph domains, reporting the same properties and applications already shown in the image context.

In Chapter 6, we present a non-linear version of the osmosis model, involving a non-linear diffusivity function promoting isotropic edge-stopping diffusion. It respects the same conservation properties, both in the continuous and in the image domain. Finally, we validate our model on exemplar image processing tasks, such as shadow/light-spot removal and compact data representation, observing that the novel diffusivity function avoids the appearance of smoothing artifacts resulting in the linear model.

To our knowledge, the osmosis model has not been previously applied to surface processing tasks. In Chapter 7 we propose a framework that employs the osmosis model for tasks such as mesh cloning and inpainting, showing its potential accuracy.

## 1.5 Variational Models for Shape Analysis

In numerical analysis, variational models aim to represent the desired result of a given task as a minimizer or maximizer of an appropriate energy function, defined to favor solutions that verify specific properties.

In mesh processing, each task has the goal of producing a surface with particular geometric shapes or details, starting from an initial surface  $\mathcal{S}_0$ . Many variational models act directly on the mesh approximation  $\mathcal{M}_0$  of the surface looking for the resulting vertex set  $V^*$  as solutions of a minimization problem:

$$V^* \in \arg \min_V \{ \mathcal{J}(V; \lambda) = \mathcal{F}(V; V_0) + \lambda R(V) \}, \quad (1.59)$$

where

- the data fidelity term  $\mathcal{F}(V; V_0)$  measures the distance from the vertices  $V_0$  of the initial mesh, usually defined as  $\mathcal{F}(V; V_0) = \|V - V^0\|_2^2$ ;
- one or more penalty/regularization terms  $R(V)$  promote surfaces that satisfy the desired geometrical properties, exploiting this a priori knowledge;
- the regularization parameter  $\lambda > 0$ , balancing the effects of the two terms.

The definition of the data term and the penalty term is a critical aspect of problem formulation. As we will demonstrate, each task necessitates specific choices for these terms. Once the model is defined, the optimization problem is solved using an appropriate minimization method, taking into account the functional properties of the model, such as continuity, differentiability, convexity, and linearity.

**Denoising** In mesh denoising, the goal is to solve the inverse problem corresponding to the noise degradation model defined in (1.41). The noise corruption in surface scanning processes is usually represented by realizations of Gaussian distributions with zero mean and variance  $\sigma^2$ . Maximum likelihood estimation suggests defining the fidelity term as the corresponding log-likelihood function, that is

$$\mathcal{F}(V; V^0) = \frac{\|V - V^0\|_2^2}{2\sigma^2}. \quad (1.60)$$

To define the penalty function we first observe that noise affects the surface locally enhancing its roughness, which can be identified as its curvature or normal deviation. For this reason, common penalty functions acts on  $\|\nabla \mathcal{N}\|$ , with  $\mathcal{N}$  the triangle normals of Def. 1.2.4. Since  $\mathcal{N} \in \mathbb{R}^{n_T \times 3}$  is constant on each triangle  $\tau$ ,  $\nabla \mathcal{N}$  has non-zero values only on edges, locally estimating normal deviation.

The ideal penalty function consists in the sparsity-inducing  $\ell_0$  pseudo norm, applied for example in [55, 118]:

$$R(V) = \|\nabla \mathcal{N}\|_{\ell_0} \quad \text{with} \quad \|(x_1, \dots, x_n)\|_{\ell_0} := \# \{i = 1, \dots, n \text{ s.t. } x_i \neq 0\}.$$

However, its combinatorial nature makes the computation inefficient and results may have spurious overshoots and fold-backs.

A convex relaxation of the  $\ell_0$  pseudo-norm is the  $\ell_1$ -norm penalty, firstly used in signal processing methods and then applied to mesh denoising (see [6]):

$$R(V) = \|\nabla \mathcal{N}\|_{\ell_1} \quad \text{with} \quad \|(x_1, \dots, x_n)\|_{\ell_1} := \sum_{i=1}^n |x_i|.$$

It overcomes computational challenges, but it may produce undesired shrinkage or stair-case effect, in particular in the presence of high-level noise.

Recent works have analyzed classes of sparsity-promoting parametrized nonconvex regularizers, proving their useful theoretical properties and providing excellent results [90, 70, 116].

In Chapter 2, we propose a mesh processing method that, among other goals, tackles denoising using one of such regularizers, namely the Minimax Concave Penalty [59], which approximates the  $\ell_0$  pseudo-norm and is able to control sparsity more accurately than the  $\ell_1$  norm, avoiding artifacts and removing noise.

**Fairing** When we deal with fairing, the penalty function must be chosen in order to enforce local sphericity. A commonly adopted fairness prior relies on the Laplace-Beltrami operator and is defined as  $\|\Delta V\|_2^2$ , proposed for smooth hole filling with the so-called least squares meshes, see [115]. An alternative penalty function is the Willmore energy, which favors smoothness, but through a non-linear curvature measure, leading to a more rounded shape filling. In the continuous domain, it is defined as

$$E_W(S) = \frac{1}{2} \int_S (h^2 - \kappa) dA \quad (1.61)$$

where  $h$  and  $\kappa$  are mean and Gaussian curvature, and it measures how far is the surface from being a sphere. In contrast to mean curvature flow, it is scale-invariant.

Numerical approximations of the Willmore energy in digital geometry processing and geometric modeling are mainly based either on finite element discretization and numerical quadrature [34, 54], or on discrete differential geometry approaches. Discrete isometric bending models, derived from an axiomatic treatment of discrete Laplace operators [12], the discrete conformal vertex-based energy well-defined for simplicial surfaces using circumcircles of their faces [16, 17], and the integer linear programming approach [106] all fall into the latter class.

The method introduced in Chapter 2 involves a fairing-promoting penalty function, defined as an alternative edge-based discrete approximation of the Willmore energy (1.61).

**Deformation** In the previous models, the data-fidelity term acts on the entire set of vertices, in order to preserve the global shape of the object, allowing only small local movements. In deformation tasks, conversely, we aim at changing the global shape, while preserving local details.

Consequently, the fidelity term can be defined through the differential coordinates introduced in 1.2.3, such as Laplacian Coordinates, Normal-Controlled Coordinates and Mean Value Encoding, thanks to their ability to encode local features. In these three cases, the fidelity terms are defined as follows

$$\mathcal{F}_1(V; \delta_L) = \frac{1}{2} \|L_w V - \delta_L\|_2^2, \quad (1.62)$$

$$\mathcal{F}_2(V; \delta_N) = \frac{1}{2} \|N_w V - \delta_N\|_2^2, \quad (1.63)$$

$$\mathcal{F}_3(V; w, b) = \frac{1}{2} \|V - F(V; w, b)\|_2^2. \quad (1.64)$$

where  $\delta_L$ ,  $\delta_N$ ,  $(w, b)$  are, respectively, the LAP, NCC, MVE coordinates of the initial mesh.

The global shape is modified by selecting a few vertices in a subset  $H \subset \{1, \dots, n_V\}$ , to be moved into the new desired positions  $\{\bar{v}_i, i \in H\}$ . This condition can be imposed as a hard constraint, searching the minimum of  $\mathcal{F}_1, \mathcal{F}_2, \mathcal{F}_3$  only on the set of vertices that satisfy this prescription, or as a soft constraint, solving an unconstrained minimization problem that involves an additional penalty term

$$R(V; \bar{V}) = \sum_{i \in H} \|v_i - \bar{v}_i\|_2^2. \quad (1.65)$$

In Chapter 3, we exploit these variational model formulations to solve the specific deformation task of Geometric Texture Transferring. The use of geometric descriptors and shared parametrization for the transferring of geometric details has been preliminarily presented in [114] using the Laplacian coordinates and more recently in [94] via mean curvature details. In contrast to the previous approaches we avoid the use of a global parametrization  $(t, s, atlas)$ , and we do not require any mesh refinement of the base surface and texture to adapt one to the other. Even if we mainly handle height-field geometric textures, we detail and showcase the extension to generic 3D texture sample meshes, which is a well-known critical issue in the case of GTT with mesh representation.





## Chapter 2

# Variational recovery in Euclidean coordinates

In this Chapter, we propose a framework able to tackle the problems of hole-filling (including both inpainting and completion) and denoising, as defined in Sec.1.3.1-1.3.2.

An example of the three surface geometry tasks processed by the proposed variational framework is illustrated in Fig.2.1. The original noisy and incomplete scanned *angel* mesh (see Fig.2.1(left)) is denoised while keeping all the holes, see Fig.2.1(center, first row). Then the inpainting tool filled the holes smoothly, as shown in Fig.2.1(center, second row) driven by the inpainting mask illustrated on the left. The large damaged region on the head is recovered by replacing a hair curl patch selected from a different, undamaged, mesh, see the recovered mesh in Fig.2.1(center, third row). Finally, the completion of the damaged part, as well as hole filling is performed and illustrated in Fig.2.1(right).

Following the principles described in Sec. 1.5, we solve these challenging geometric tasks by proposing a unified variational approach encoding a priori knowledge of the particular problem (i.e. the mask operators) directly in the cost functional.

Here, we use the setting described in Sec.1.3.1 for the hole-filling problem, assuming that a corrupted surface  $\mathcal{S}$  embedded into  $\mathbb{R}^3$  is represented by a triangulated mesh  $\mathcal{M}_0 = (V_0, E_0, T_0)$  and possibly characterized by the presence of a damaged (incomplete) region  $S_D \subset \mathcal{S}$ .

The reconstructed triangular mesh  $\mathcal{M}^* = (V^*, E^*, T^*)$  is obtained as in (1.40), closing the hole  $S_D$  with a patch  $P$ , with boundary  $b_P$ . In case of surface completion, the template patch  $P$  is already given as known data coming from another object, otherwise it must be defined in a pre-processing step.

As explained in the following, the aforementioned three geometry processing tasks are addressed through the following unified variational formulation, which includes a fidelity and two penalty functions,

$$\begin{aligned} V^* &\in \arg \min_V \mathcal{J}(V; M_E), \\ \mathcal{J}(V; M_E) &:= \mathcal{F}(V; V_0, \lambda \chi_{S \setminus S_D}) + \mathcal{R}_1(V; M_E) + \mathcal{R}_2(V; M_E^c), \end{aligned} \tag{2.1}$$

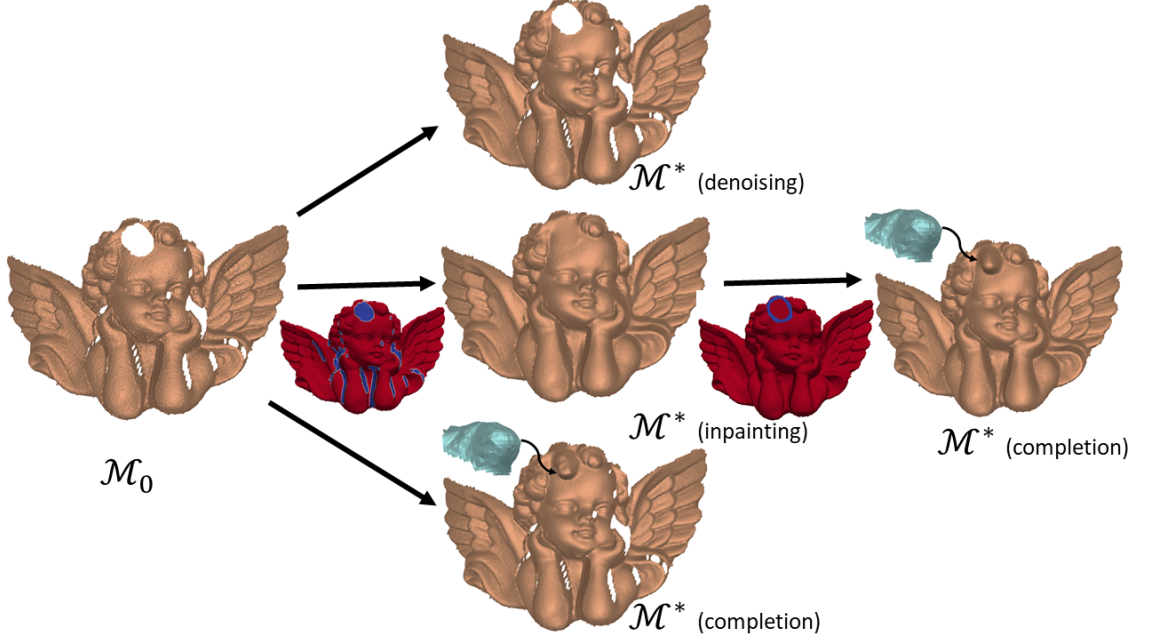


Figure 2.1: Applications of the proposed surface geometry framework: incomplete and noisy surface input  $\mathcal{M}_0$  (left); denoised surface  $\mathcal{M}^*$  (center, first row), inpainting mask  $M_V$  and inpainting result  $\mathcal{M}^*$  (center, second row), context-aware completion of the curly hair detail without inpainting (center, third row); context-aware completion result  $\mathcal{M}^*$  with preliminary inpainting (right). The  $S_D$  region is represented in blue in the masks.

where  $\chi_{S \setminus S_D} : S \rightarrow \{0, 1\}$  denotes the characteristic function of the subset  $S \setminus S_D$ , while the binary mask operators  $M_E \in \{0, 1\}^{n_E}$  and  $M_E^c = \mathbf{1}_{n_E} - M_E$ , characterize the specific surface geometry considered. As a result of the discretization on the triangulated mesh, the role of the characteristic function is played by a mask operator  $M_V \in \{0, 1\}^{n_V}$  whose zero values identify the region  $S_D$ .

The set of vertices  $V^*$  solution of the unconstrained optimization problem (2.1) defines a restored triangulated surface  $\mathcal{M}^* = (V^*, E^*, T^*)$  which provides a solution of the three surface geometry tasks, depending on the particular setup considered.

The proposed approach does not need any global or even local 2D parameterization, nor any sophisticated octree data structures to efficiently solve implicit volumetric computations [110, 95]. The data are explicitly treated as connected samples of a surface embedded in  $\mathbb{R}^3$ .

The functional  $\mathcal{J}(V; M_E)$  in (2.1) is characterized by the presence of the sum of two regularization terms: the sparsity-promoting term  $\mathcal{R}_1(V; M_E)$  and the sphericity-inducing penalty  $\mathcal{R}_2(V; M_E^c)$ . Furthermore, a fidelity term  $F(V; V_0)$ , weighted by the

scalar parameter  $\lambda \geq 0$  is used to control the trade-off between fidelity to the observations and regularity in the solution  $V^*$  of (2.1).

The regularizer  $\mathcal{R}_1$  favors solutions with piece-wise constant normal map and sharp discontinuities. Instead of using the  $\ell_0$  pseudo-norm or the convex  $\ell_1$  norm, analyzed in Sec. 1.5, we then rather consider as regularizer  $\mathcal{R}_1(V; M_E)$  a sparsity-promoting parametrized non-convex term, whose form provides effective control on the sparsity of the normal deviation magnitudes being more accurate than the  $\ell_1$  norm, while mitigating the strong effect and the numerical difficulties of  $\ell_0$  pseudo-norm. Numerical experiments will show its efficiency in handling high levels of noise, producing good-shaped triangles, and faithfully recovering straight and smoothly curved edges.

As far as the  $\mathcal{R}_2$  regularization term is concerned, we choose it to encode a geometric energy, aimed to force local sphericity in correspondence of rounded regions. We considered here the Willmore energy  $E_w(S)$ , defined in (1.61), which is non-negative, and vanishes if and only if  $S$  is a sphere [17]. For compact and closed surfaces, and surfaces whose boundary is fixed up to first order, i.e. positions and normals are prescribed, finding the minima of (1.61) is equivalent to minimize the Willmore bending energy  $E_h(S) = \frac{1}{2} \int_S h^2 dA$  since the two functionals differ only by a constant (the Euler characteristic of the surface  $S$ ), [131]. Here we present a discrete Willmore energy, which, in contrast to traditional approaches, follows an edge-based discrete formulation.

From an algorithmic point of view, we solve the (non-convex) problem (2.1) employing an Alternating Direction Method of Multipliers (ADMM) scheme. This allows us to split the minimization problem into three more tractable sub-problems. Closed-form solutions for two of these problems can be found, while for the third, non-convex, one different optimization solvers can be used. For this substep, we compare standard gradient descent, with heavy ball and Broyden–Fletcher–Goldfarb–Shanno (BFGS) schemes, endowed with suitable backtracking strategy applied to guarantee the convergence to stationary points of the sub-problem considered.

Numerical experiments will demonstrate the effectiveness of the proposed method for the solution of several exemplar mesh denoising, inpainting and completion problems.

In the following, we present the proposed geometric variational model (Sec. 2.1), describe its numerical optimization by means of the ADMM-based scheme (Sec. 2.2), explain how to tackle the three task through the optimization problem (Sec. 2.3) and finally observe experimental results (Sec. 2.4).

## 2.1 Variational Recovery Model

Solving the variational problem (2.1) on surfaces requires the definition of the discrete manifold representing the underlying object of interest as well as the discrete approximation of the first-order differential operators involved.

We thus assume  $\mathcal{M} := (V, E, T)$  to be a triangulated mesh of arbitrary topology

approximating a 2-manifold  $\mathcal{S}$  embedded in  $\mathbb{R}^3$ , with vertices, edges and faces denoted as in (1.4).

Let  $\mathcal{N} : \mathbb{R}^{n_V \times 3} \rightarrow \mathbb{R}^{n_T \times 3}$  be the mapping computing from the vertex positions  $V$  the piecewise-constant normal field  $\mathcal{N}(V)$  over the triangles of the mesh, where the  $m$ -th element  $\mathcal{N}_m(V)$  is the outward unit normal at face  $\tau_m = (v_i, v_j, v_k)$  (see Def.1.2.4).

Notice that the normal vector's sign depends on the orientation of the face. The desire for consistently oriented normals is that adjacent faces have consistent orientation.

We now introduce the discretization of the gradient operator of the normal field on a 3D mesh. Since the normal field is piecewise-constant over the mesh triangles, the gradient operator vanishes to zero everywhere but the mesh edges along which it is constant. Therefore, the gradient operator discretization is represented by a sparse matrix  $D \in \mathbb{R}^{n_E \times n_T}$  defined by

$$D_{ij} = \begin{cases} l_i & \text{if } \tau_j \cap \tau_k = e_i, k > j, \\ -l_i & \text{if } \tau_j \cap \tau_k = e_i, k < j, \\ 0 & \text{otherwise,} \end{cases} \quad (2.2)$$

where  $l_i = \|e_i\|_2$ ,  $i = 1, \dots, n_E$  is the length of  $i$ -th edge.

The matrix  $D$  can be decomposed as  $D = L\bar{D}$ , with  $L = \text{diag}\{l_1, l_2, \dots, l_{n_E}\}$  being the diagonal matrix of edge lengths, whose values may be updated during the iteration scheme considered, and  $\bar{D} \in \mathbb{R}^{n_E \times n_T}$  an edge-length independent sparse matrix.

Key ingredients of the proposed formulation (2.1) are the two operator masks  $M_V$  and  $M_E$ . The role of the mask  $M_E$  is to adapt the recovery according to the surface morphology, while  $M_V$  selects the region to be preserved in the inpainting and completion tasks.  $M_E$  is a sharp detection mask represented by a binary vector  $M_E \in \{0, 1\}^{n_E}$  which has 1s in correspondence with sharp edges. Recalling that the dihedral angle associated with the edge  $e_i$  is the angle between normals to the adjacent triangle faces  $\tau_\ell$  and  $\tau_s$  which share  $e_i$ , we classify  $e_i$  as a sharp edge if the dihedral angle  $\theta_{\ell s} \in [0, 360)$  is greater than a given threshold  $th$ . In formulas

$$(M_E)_i = \begin{cases} 1 & \text{if } (\theta_{\ell s} > th) \\ 0 & \text{otherwise.} \end{cases} \quad (2.3)$$

Given  $M_E$ , its complementary mask is the vector  $M_E^c = \mathbf{1}_{n_E} - M_E$ . Fig.2.2 shows  $M_E$  for three different surface meshes, where we empirically set  $th = 30$ , which typically produces good results.

The influence of the choice of the mask  $M_E$  in realizing the denoising task is shown in Fig.2.3. The perturbed *sharp sphere* is illustrated in Fig.2.3 on the left panel, and the denoised meshes on the right panel, obtained by applying the proposed method under the choice  $M_E = \mathbf{0}_{n_E}$ ,  $M_E = \mathbf{1}_{n_E}$ , in Fig.2.3(b) and Fig.2.3(e), respectively. The space-variant mask  $M_E$  obtained with  $th = 30$ , and illustrated in Fig.2.3(c) is applied to obtain the denoised mesh in Fig.2.3(d).

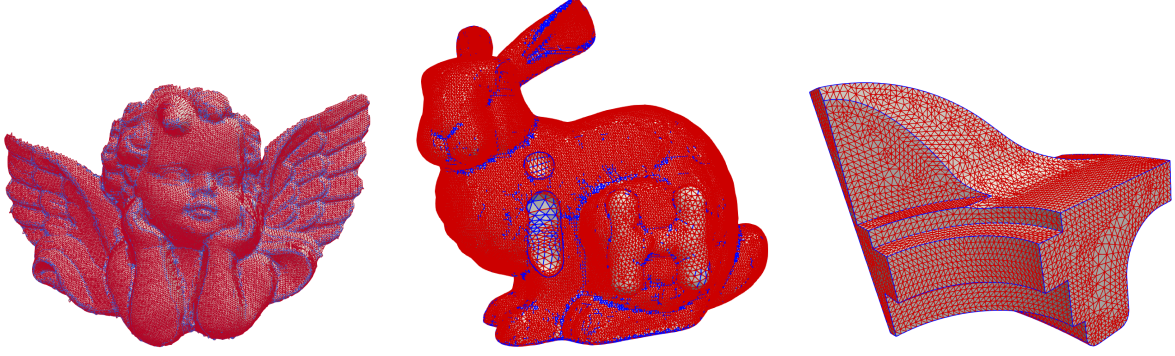


Figure 2.2: Examples of  $M_E$  mask for three different meshes: blue colors represent values 1, while red colors represent 0 values.

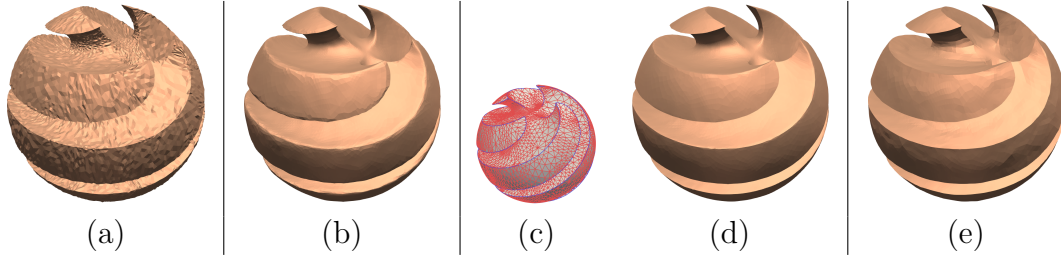


Figure 2.3: Effect of the mask  $M_E$  on the denoising task: original noisy mesh (a); setting  $M_E = \mathbf{0}_{n_E}$  (b); using a space-variant  $M_E$  mask (c)-(d); setting  $M_E = \mathbf{1}_{n_E}$  (e). The perturbed *sharp sphere* on the left panel has been corrupted according to (1.41) with  $\gamma = 0.15$ .

Stemming from the consideration by which a general scanned surface is characterized by sharp as well as rounded features, we specify the form of problem (2.1) to determine solutions  $V^*$  which are close to the given data  $V_0$  according to the observation model,

$$\begin{aligned}
 V^* \in \arg \min_{V \in \mathbb{R}^{n_V \times 3}} \mathcal{J}(V; M_E, \lambda, a) \\
 \mathcal{J}(V; M_E, \lambda, a) := \frac{\lambda}{2} \sum_{i=1}^{n_V} ((M_V)_i (V_i - (V_0)_i))^2 + \\
 + \sum_{j=1}^{n_E} \left\{ (M_E)_j \phi \left( \|(DN(V))_j\|_2; a \right) + (M_E^c)_j \|(DN(V))_j\|_2^2 \right\},
 \end{aligned} \tag{2.4}$$

where  $\|\cdot\|_2$  denotes the Frobenius norm. The functional in (2.4) involves three terms designed to meet three different and competing requirements that arise quite naturally from the intuitive concept of surface recovery:

1. fidelity to the known data, in  $\ell_2$  norm, with mask  $M_V$ , optimal in case of Gaussian

noise distribution, as discussed in Sec. 1.5;

2. a discontinuity-preserving smoothing function, with mask  $M_E$ , depending on a parameter  $a \in \mathbb{R}_+$ , favoring piece-wise constant normals;
3. smooth connection between parts and inside unknown regions, with mask  $M_E^C$ .

The functional  $\mathcal{J}$  in (2.4) is composed of the sum of smooth convex (quadratic) terms and a non-smooth non-convex regularization term, all non-negative, thus  $\mathcal{J}$  is bounded from below by zero, non-smooth and can be convex or non-convex depending on the values of  $M_E$  and  $a$ .

### 2.1.1 Sparsity-inducing penalty

We aim at constructing a parameterized sparsity-promoting regularizer characterized by a tunable degree of non-convexity  $a \in \mathbb{R}_+$  inducing sparsity on the vector of components  $\|(D\mathcal{N})_i\|_2, i = 1, \dots, n_E$ , which represent the normal variation between adjacent triangles sharing the  $i$ -th edge.

Among the class of sparsity-promoting parametrized non-convex regularizers cited in Sec. 1.5, we consider here the Minimax Concave (MC) penalty  $\phi(\cdot; a) : [0, +\infty) \rightarrow \mathbb{R}$ , introduced in [139] and used previously in [59] applied to  $\|(D\mathcal{N})_i\|_2$  in the context of mesh editing, defined by:

$$\phi(t; a) = \begin{cases} -\frac{a}{2}t^2 + \sqrt{2a}t & \text{for } t \in [0, \sqrt{2/a}) , \\ 1 & \text{for } t \in [\sqrt{2/a}, +\infty) \end{cases} \quad (2.5)$$

which, for any value of the parameter  $a$ , satisfies the following assumptions:

- $\phi(t; a) \in \mathcal{C}^1(\mathbb{R}) \cap \mathcal{C}^2(\mathbb{R} \setminus \{\sqrt{2/a}\})$
- $\phi'(t; a) \geq 0$ ,
- $\phi''(t; a) \leq 0, \forall t \in [0, \infty) \setminus \{\sqrt{2/a}\}$
- $\phi(0; a) = 0, \inf_t \phi''(t; a) = -a$ .

We denoted by  $\phi'(t; a)$  and  $\phi''(t; a)$  the first-order and second-order derivatives of  $\phi$  with respect to the variable  $t$ , respectively.

The parameter  $a$  allows tuning the degree of non-convexity, such that  $\phi(\cdot; a)$  mimics the asymptotically constant behavior of the  $\ell_0$  pseudo-norm for  $a \rightarrow \infty$ , while behaves as an  $\ell_1$  regularization term, for values  $a$  approaching to zero, since the quadratic term vanishes more than the linear one. For values of  $a$  in between, the MC penalty function in (2.5) is a sparsity-inducing penalty that preserves sharp features in normal variations

better than  $\ell_0$ -pseudo-norm regularizer, and more accurate than  $\ell_1$  regularizer which tends to produce shrinkage effects.

This motivated us to use it in the construction of the regularizer  $\mathcal{R}_1(V; M_E)$ .

### 2.1.2 Edge-based discretization of the Willmore energy

We consider an edge-based discrete approximation of the Willmore energy (1.61) for open triangulated surfaces  $\mathcal{M}$  represented by polygonal meshes. This energy is a sum of contributions from individual edges

$$E(\mathcal{M}) = \frac{1}{2} \sum_{j=1}^{n_E} \|e_j\|^2 \|(D\mathcal{N})_j\|^2, \quad (2.6)$$

where  $(D\mathcal{N})_j$  measures how the surface “curves” near  $e_j$ . To derive the continuum limit of (2.6) in the limit of vanishing triangle size, we consider  $S$  as a 2-dimensional manifold of arbitrary topology embedded in  $\mathbb{R}^3$  and use the following notation:

- $(\Omega, X)$ , with  $\Omega \subset \mathbb{R}^2$  an open reference domain, is a parametrization of  $S$ ;
- $(t, s)$  are the local coordinates in  $\Omega$ ;
- the tangent space  $T_x S$  at  $x \in \Omega$  is spanned by  $\left\{ r_1 := \frac{\partial X(x)}{\partial t}, r_2 := \frac{\partial X(x)}{\partial s} \right\}$ ;
- the induced metric is given by the first fundamental form  $\mathbf{I}$ , with entries  $g_{ij} = r_i \cdot r_j$ , its inverse is denoted by  $g^{ij}$ , and its determinant is defined as

$$\det(g) \equiv |g| = \frac{1}{2} \epsilon^{ik} \epsilon^{jl} g_{ij} g_{kl} = \frac{1}{2} (g_{ij} g_{kl} - g_{ik} g_{jl}),$$

using Levi-Civita symbol  $\epsilon^{ij}$  and Einstein summation notation;

- The second fundamental form  $\mathbf{II} : T_x S \times T_x S \rightarrow \mathbb{R}$  is the symmetric bilinear form represented by the coefficients  $L_{ij} = -r_i \cdot \partial_j n$ ,  $1 \leq i, j \leq 2$ .

When the grid size of the triangulation  $\mathcal{M}$  is sent to 0, the energy (2.6) approximates the Willmore energy as stated by the following proposition.

**Proposition 2.1.1.** *Let  $S \subset \mathbb{R}^3$  be a 2-dimensional manifold,  $\mathcal{M}$  an underlying flat triangulated approximation of  $S$ . Let  $\mathcal{M}_j$  be regular flat triangulated surfaces  $\mathcal{M}_j \subset \mathbb{R}^3$  with  $\text{size}(\mathcal{M}_j) \rightarrow 0$  and  $\mathcal{M}_j \rightarrow S$  for  $j \rightarrow \infty$ . Then, the discrete energy (2.6) approximates the Willmore energy of  $S$ , i.e.*

$$\lim_{j \rightarrow \infty} E(\mathcal{M}_j) = \frac{1}{2} \int_S (h^2 - k) dS. \quad (2.7)$$



*Proof.* Let us first consider the integrand of (1.61) in the continuum, with  $h = \kappa_1 + \kappa_2 = \text{tr}(L_k^i)$  being the mean curvature and  $k = \frac{1}{2}\kappa_1\kappa_2 = \det(L_k^i)$  the Gaussian curvature where  $\kappa_1, \kappa_2$  represent the principal curvatures. The second fundamental form with components  $L_{ij}$  relates with the linear map  $L_i^k$  with respect to the basis of  $T_x S$ , according to the matrix equation:  $[g^{ij}][L_{ij}] = [L_j^i]$ , and we denote  $L_{ij} = \sum_k g_{ik} L_j^k$ ,  $1 \leq i, j \leq 2$ . Following notations in [109], we use the identity

$$g^{ij}g^{kl} = g^{ik}g^{jl} + \epsilon^{il}\epsilon_{mn}g^{mj}g^{nk}$$

in the integrand of (1.61) as

$$h^2 - k = (L_i^i)^2 + \epsilon^{il}\epsilon_{mn}L_l^m L_i^n = (g^{ik}L_{ik})^2 + \epsilon^{il}\epsilon_{mn}(g^{mj}L_{jl})(g^{nk}L_{ki}) = g^{ij}g^{kl}L_{ik}L_{jl}. \quad (2.8)$$

Substituting in (2.8) the Weingarten equations  $\partial_i n = L_i^k r_k$  and  $L_i^k, i = 1, 2$ , we have

$$g^{ij}g^{kl}L_{ik}L_{jl} = L_k^j L_j^k g^{kj} g_{jk} = L_k^j r_j L_j^k r_k g^{kj} = \partial_k n \cdot \partial_j n g^{kj} \quad (2.9)$$

which is the gradient of the normal vector field. Therefore, replacing (2.8-2.9) in (1.61), we get

$$\int_S \partial_k n \cdot \partial_j n g^{kj} dS. \quad (2.10)$$

For sufficiently fine, non-degenerate tessellations  $\mathcal{M}_j$  approximating  $S$ , we consider a partition of the undeformed surface  $S$  into the disjoint union of diamond-shaped tiles,  $\bar{T}$ , associated with each mesh edge  $e$ . Following Meyer et al.[86], one can use the barycenter of each triangle to define these regions or, alternatively, the circumcenters. Over such a diamond partition, the integral (2.10) is defined as the sum over all the diamond tiles, which reads

$$\int_S \partial_k n \cdot \partial_j n g^{kj} dS = \sum_{i=1}^{n_E} \int_{(\bar{T})_i} |\partial_i n|^2 d\bar{T}. \quad (2.11)$$

If the triangles do not degenerate, we can approximate the area of the diamond related to the edge  $e_i$  in  $\mathcal{M}_j$  by  $\|e_i\|^2$ , i.e.  $d\bar{T} \approx \|e_i\|^2$ , which implies that  $\int_{(\bar{T})_i} |\partial_i n|^2 d\bar{T} \approx \|(D\mathcal{N})_i\|^2 \|e_i\|^2$ .  $\square$

The result of the limiting process depends on the triangulations considered. In particular, we assume the triangulations of  $S$  consist of almost equilateral triangles. For our purposes, the discrete Willmore energy will be used based on the observation that (1.61) is invariant under rigid motions and uniform scaling of the surface, which implies that  $E(S)$  itself is a conformal invariant of the surface  $S$ , see [17].

**Remark 1.** *Even if the introduced discrete formulation is very simple when compared with the ones introduced in [105, 17], it practically produces good results. In order to*

validate the effective applicability of the proposed discrete Willmore energy, we evaluated  $E(\mathcal{M})$  in (2.6) on a uniformly tessellated sphere, for decreasing average edge-size  $h = \{0.1208, 0.0308, 0.0076\}$ . The achieved energy values  $E(\mathcal{M}_{h_1}) = 0.1182$ ,  $E(\mathcal{M}_{h_2}) = 0.0075$ ,  $E(\mathcal{M}_{h_3}) = 0.00047$ , tend to zero, as theoretically expected from (1.61), which is zero for continuous sphere.

## 2.2 Numerical solution of the optimization problem

In this section, we illustrate the ADMM-based iterative algorithm used to compute the numerical solution of (2.4).

In order to define the ADMM iteration on triangular mesh surfaces, we first consider a matrix variable  $N \in \mathbb{R}^{n_T \times 3}$  with row components defined as in (1.19), and resort to the variable splitting technique by defining  $t \in \mathbb{R}^{n_E \times 3}$  as  $t := DN$ , where  $D$  is defined in (2.2). The optimization problem (2.4) can be thus reformulated as

$$\begin{aligned} & \{V^*, N^*, t^*\} \in \\ & \arg \min_{V, N, t} \left\{ \frac{\lambda}{2} \sum_{i=1}^{n_V} ((M_V)_i (V_i - (V_0)_i))^2 + \sum_{j=1}^{n_E} [(M_E)_j \phi(\|t_j\|_2; a) + (M_E^c)_j \|t_j\|_2^2] \right\}, \\ & \text{s.t. } t = DN, \quad N = \mathcal{N}(V). \end{aligned} \quad (2.12)$$

We define the augmented Lagrangian functional associated with problem (2.12) as

$$\begin{aligned} \mathcal{L}(V, N, t, \rho_1, \rho_2; \lambda, \beta_1, \beta_2, a) &:= \frac{\lambda}{2} \sum_{i=1}^{n_V} ((M_V)_i (V_i - (V_0)_i))^2 + \\ &+ \sum_{j=1}^{n_E} \left[ (M_E)_j \phi(\|t_j\|_2; a) + (M_E^c)_j \|t_j\|_2^2 - \langle \rho_{1j}, t_j - (DN)_j \rangle + \frac{\beta_1}{2} \|t_j - (DN)_j\|_2^2 \right] + \\ &+ \sum_{m=1}^{n_T} \left[ -\langle \rho_{2m}, N_m - \mathcal{N}_m(V) \rangle + \frac{\beta_2}{2} \|N_m - \mathcal{N}_m(V)\|_2^2 \right], \end{aligned} \quad (2.13)$$

where  $\beta_1, \beta_2 > 0$  are scalar penalty parameters, and  $\rho_1 \in \mathbb{R}^{n_E \times 3}$ ,  $\rho_2 \in \mathbb{R}^{n_T \times 3}$  represent the matrices of Lagrange multipliers associated with the constraints. We now consider the following saddle-point problem:

$$\begin{aligned} & \text{Find} \quad (V^*, N^*, t^*, \rho_1^*, \rho_2^*) \in \mathbb{R}^{n_V \times 3} \times \mathbb{R}^{n_T \times 3} \times \mathbb{R}^{n_E \times 3} \times \mathbb{R}^{n_E \times 3} \times \mathbb{R}^{n_T \times 3} \\ & \text{s.t.} \quad \mathcal{L}(V^*, N^*, t^*, \rho_1, \rho_2) \leq \mathcal{L}(V^*, N^*, t^*, \rho_1^*, \rho_2^*) \leq \mathcal{L}(V, N, t, \rho_1^*, \rho_2^*), \\ & \quad \forall (V, N, t, \rho_1, \rho_2) \in \mathbb{R}^{n_V \times 3} \times \mathbb{R}^{n_T \times 3} \times \mathbb{R}^{n_E \times 3} \times \mathbb{R}^{n_E \times 3} \times \mathbb{R}^{n_T \times 3}. \end{aligned} \quad (2.14)$$

An ADMM-based iterative scheme can now be applied to approximate the solution of the saddle-point problem (2.13)–(2.14). Initializing to zeros both the dual variables

$\rho_1^{(0)}, \rho_2^{(0)}$  and setting  $N_m^{(0)} = \mathcal{N}_m(V^{(0)})$ ,  $m = 1, \dots, n_T$ , the  $k$ -th iteration of the proposed alternating iterative scheme reads:

$$t^{(k+1)} = \arg \min_{t \in \mathbb{R}^{n_E \times 3}} \mathcal{L}(V^{(k)}, N^{(k)}, t; \rho_1^{(k)}, \rho_2^{(k)}) , \quad (2.15)$$

$$N^{(k+1)} = \arg \min_{\substack{N \in \mathbb{R}^{n_T \times 3}, \\ \|N_\tau\|=1}} \mathcal{L}(V^{(k)}, N, t^{(k+1)}; \rho_1^{(k)}, \rho_2^{(k)}) , \quad (2.16)$$

$$V^{(k+1)} = \arg \min_{V \in \mathbb{R}^{n_V \times 3}} \mathcal{L}(V, N^{(k+1)}, t^{(k+1)}; \rho_1^{(k)}, \rho_2^{(k)}) , \quad (2.17)$$

$$\rho_1^{(k+1)} = \rho_1^{(k)} - \beta_1 (t^{(k+1)} - DN^{(k+1)}) , \quad (2.18)$$

$$\rho_2^{(k+1)} = \rho_2^{(k)} - \beta_2 (N^{(k+1)} - \mathcal{N}(V^{(k+1)})) . \quad (2.19)$$

The updates of Lagrangian multipliers  $\rho_1$  and  $\rho_2$  have closed form. In the following, we show in detail how to solve the three minimization sub-problems (2.15), (2.16) and (2.17) for the primal variables  $t$ ,  $N$  and  $V$ , respectively.

#### Sub-problem for $t$ .

The minimization sub-problem for  $t$  in (2.15) can be explicitly rewritten as:

$$t^{(k+1)} = \arg \min_{t \in \mathbb{R}^{n_E \times 3}} \sum_{j=1}^{n_E} \left[ (M_E)_j \phi(\|t_j\|_2; a) + (M_E^c)_j \|t_j\|_2^2 - \langle \rho_{1,j}, t_j - (DN)_j \rangle + \frac{\beta_1}{2} \|t_j - (DN)_j\|_2^2 \right] , \quad (2.20)$$

where we omitted the constant terms in (2.13). Due to the separability property of  $\phi(\cdot; a)$ , problem (2.20) is equivalent to  $n_E$  independent, three-dimensional problems for each  $t_j$ ,  $j = 1, \dots, n_E$  in the form

$$t_j^{(k+1)} = \arg \min_{t_j \in \mathbb{R}^3} \left\{ \phi(\|t_j\|_2; a) + \frac{\alpha}{2} \|t_j - r_j^{(k+1)}\|_2^2 \right\} , \quad (2.21)$$

with

$$r_j^{(k+1)} := \frac{1}{\beta_1 + 2(M_E^c)_j} \left( \beta_1 (DN^{(k)})_j + (\rho_1^{(k)})_j \right)$$

and  $\alpha = \frac{\beta_1 + 2(M_E^c)_j}{(M_E)_j}$ , where we conventionally set  $\frac{x}{0} = 0$ .

Necessary and sufficient conditions for strong convexity of the cost functions in (2.21) are demonstrated in [58]. In particular, problems (2.21) are strongly convex if and only if the following condition holds:

$$\frac{\beta_1 + 2(M_E^c)_j}{(M_E)_j} > a, \forall j = 1, \dots, n_E \implies \beta_1 = \varepsilon \max_j \{(M_E)_j a - 2(M_E^c)_j\} , \quad \text{for } \varepsilon > 1. \quad (2.22)$$

We noticed that the sub-problem is always convex when  $t_j$  has associated  $(M_E)_j = 0$ , as it eliminates  $\phi(\cdot; a)$  from the sub-problem.

Whenever (2.22) holds, the unique minimizers of (2.21) can be obtained by direct computation in closed form as

$$t_j^{(k+1)} = \min(\max(\nu - \zeta / \|r_j\|_2, 0), 1) r_j ,$$

where  $\nu = \frac{\alpha}{\alpha - a}$  and  $\zeta = \frac{\sqrt{2a}}{\alpha - a}$ .

We remark that the condition on  $\beta_1$  in (2.13) only ensures the convexity conditions (2.22) of  $t$ -subproblem (2.21), but does not guarantee convergence of the overall ADMM scheme.

**Sub-problem for  $N$ .** The minimization sub-problem (2.16) for  $N$  can be reformulated as:

$$N^{(k+1)} = \arg \min_{\substack{N \in \mathbb{R}^{n_T \times 3}, \\ \|N_\tau\|=1}} \left\{ \frac{\beta_1}{2} \|t^{(k+1)} - DN\|_2^2 + \langle \rho_1^{(k)}, DN \rangle - \langle \rho_2^{(k)}, N \rangle + \frac{\beta_2}{2} \|N - \mathcal{N}(V^{(k)})\|_2^2 \right\}.$$

The first optimality conditions lead to the following three linear systems, one for each spatial coordinate of  $N \in \mathbb{R}^{n_T \times 3}$

$$\left( D^T D + \frac{\beta_2}{\beta_1} I \right) N = \frac{\beta_2}{\beta_1} \mathcal{N}(V^{(k)}) + \frac{\rho_2^{(k)}}{\beta_1} + D^T \left( t^{(k+1)} - \frac{1}{\beta_1} \rho_1^{(k)} \right). \quad (2.23)$$

Since  $\beta_1, \beta_2 > 0$ , the linear system coefficient matrix is sparse, symmetric, positive definite and identical for all three coordinate vectors. The systems can thus be solved efficiently by applying, e.g., a unique Cholesky decomposition. At each iteration, the edge lengths diagonal matrix  $L$  in  $D = L\bar{D}$ , defined in (2.2), needs to be updated as the vertices  $V$  move to their updated position. For large meshes, an iterative solver warm-started with the solution of the last ADMM iteration, is rather preferred. A normalization is finally applied as  $N$  represents a normal field.

The reconstructed normal map  $N^*$  obtained by solving (2.12) via the proposed ADMM, satisfies the orientation consistency, as proved in [61], thus reducing the foldovers issue. This property is not trivially satisfied by most of the two-stage mesh denoising algorithms (normal smoothing and vertex update). They present the normal orientation ambiguity problem in the vertex updating stage, which provokes ambiguous shifts of the vertex position due to direction inconsistency of the normal vectors [143, 117]. In [79], this issue is solved by an orientation-aware vertex updating scheme.

**Sub-problem for  $V$ .** Omitting the constant terms in (2.13), the sub-problem for  $V$

reads

$$V^{(k+1)} = \arg \min_{V \in \mathbb{R}^{n_V \times 3}} \mathcal{J}_V(V) \quad (2.24)$$

$$\mathcal{J}_V(V) := \frac{\lambda}{2} \sum_{i=1}^{n_V} ((M_V)_i(V_i - (V_0)_i))^2 + \sum_{m=1}^{n_T} \left[ \left\langle \rho_{2_m}^{(k)}, \mathcal{N}_m(V) \right\rangle + \frac{\beta_2}{2} \|N_m^{(k+1)} - \mathcal{N}_m(V)\|_2^2 \right].$$

The functional  $\mathcal{J}_V(V)$  is proper, smooth, non-convex and bounded from below by zero. A minimum can be obtained by applying the gradient descent (GD) algorithm with backtracking satisfying the Armijo condition or using the BFGS algorithm. In order to balance between the slow convergence properties of GD and the high computational costs required to compute the operators involved in the BFGS method, we also considered a heavy-ball type rule, following [138], and its extension with backtracking (covering also non-smooth problems) given in [91]. In particular, the heavy-ball method is a multi-step extension of gradient descent, which, starting from  $\bar{V}^{(0)} = V^{(k)}$ , iterates over  $V$  as follows

$$\bar{V}^{(j+1)} = \bar{V}^{(j)} - \alpha_j \nabla J(\bar{V}^{(j)}) + \delta_j (\bar{V}^{(j)} - \bar{V}^{(j-1)}), \quad j = 1, 2, \dots \quad (2.25)$$

where  $\alpha_j > 0$  is a step-size parameter and  $\delta_j \in [0, 1)$  sets the inertial contribution. Note that for  $\delta_j = 0$ , (2.25) reduces to the gradient descent method. In [91], the convergence of the scheme above to stationary points is proved in the context of non-convex cost functions as the one in (2.24), with an extension also to non-smooth scenarios.

All the numerical optimization methods here considered rely on an easily computable formula for the gradient of the functional  $\mathcal{J}_V$  in (2.24), which is derived in the following.

**Proposition 2.2.1.** *Let  $s_{\tau_m} := \|(v_j - v_i) \times (v_k - v_i)\|_2/2$  be the area of the triangle  $\tau_m = (v_i, v_j, v_k)$  with updated vertices in  $V$ , and  $\mathcal{N}_m(V) = ((v_j - v_i) \times (v_k - v_i))/(2s_{\tau_m})$ . For all triangles  $m = 1, \dots, n_T$ ,*

$$\nabla_{v_i} \mathcal{J}_V(V) = \lambda(M_V)_i(v_i - v_i^0) + \sum_{\tau_m \in \mathcal{D}(v_i)} \frac{\left[ \left( \rho_{2_m}^{(k)} - \beta_2 N_m^{(k+1)} \right) - \left\langle \rho_{2_m}^{(k)} - \beta_2 N_m^{(k+1)}, \mathcal{N}_m(V) \right\rangle \mathcal{N}_m(V) \right]}{2s_{\tau_m}} \times (v_k - v_j). \quad (2.26)$$

*Proof.* The gradient of  $\mathcal{J}_V(V)$  in (2.24) w.r.t vertex  $v_i \in V$ ,  $i = 1, \dots, n_V$  is non-zero only over the triangles sharing  $v_i$  which are contained in the first disk  $\mathcal{D}(v_i)$ . Therefore the sum in (2.24) is reduced to

$$\nabla_{v_i} \mathcal{J}_V(V) = \lambda(M_V)_i(v_i - v_i^0) + \sum_{\substack{\tau_m \in \mathcal{D}(v_i) \\ \tau_m = (v_i, v_j, v_k)}} \nabla_{v_i} \left( \underbrace{\left\langle \frac{z}{q}, (v_j - v_i) \times (v_k - v_i) \right\rangle}_{g_i} \right),$$

where  $z = \rho_{2m}^{(k)} - \beta_2 N_m^{(k+1)}$ ,  $q = \|(v_j - v_i) \times (v_k - v_i)\|_2$ , and the third term in (2.24) reduces to the scalar product  $g_i$  since both  $N_m^{(k+1)}$  and  $\mathcal{N}_m(V)$  have unitary norm. In order to compute  $\nabla_{v_i}(g_i)$ , we resort to the following two properties, which hold for every constant vectors  $w, u \in \mathbb{R}^3$  and can be easily proved:

1.  $\nabla_{v_i} \left( \langle w, (v_j - v_i) \times (v_k - v_i) \rangle \right) = w \times (v_k - v_j) ;$
2.  $\nabla_{v_i} \left( \left\langle \frac{w}{\|(v_j - v_i) \times (v_k - v_i)\|_2}, u \right\rangle \right) = -\langle w, u \rangle \frac{(v_j - v_i) \times (v_k - v_i) \times (v_k - v_j)}{\|(v_j - v_i) \times (v_k - v_i)\|_2^3}.$

To evaluate the product rule derivative, we apply property 1, with  $w = z / \|(v_j - v_i) \times (v_k - v_i)\|_2$  for the left-side term constant, while property 2 is applied with  $w = z$  and  $u = (v_j - v_i) \times (v_k - v_i)$  for a right-side term kept constant. Combining the results leads to the explicit formula for  $\nabla_{v_i} g_i$ :

$$\begin{aligned} \nabla_{v_i} g_i(V) &= \frac{\left( \rho_{2m}^{(k)} - \beta_2 N_m^{(k+1)} \right) \times (v_k - v_j)}{\|(v_j - v_i) \times (v_k - v_i)\|_2} \\ &\quad - \frac{\left\langle \rho_{2m}^{(k)} - \beta_2 N_m^{(k+1)}, (v_j - v_i) \times (v_k - v_i) \right\rangle [(v_j - v_i) \times (v_k - v_i) \times (v_k - v_j)]}{\|(v_j - v_i) \times (v_k - v_i)\|_2^3}, \end{aligned} \quad (2.27)$$

which reduces to (2.26).  $\square$

In Figure 2.4 (first and second rows) we report the graphs showing both the energy decay and the gradient norm decay for the three different algorithms used, i.e. GD (with and without backtracking), BFGS and heavy-ball with backtracking. The plots are related to the meshes **twelve** (first column) and **block** (second column) as representative of the entire set of meshes analyzed in the experimental section, furtherly consisting of the meshes **fandisk**, **foot**, **cube-hole**, **sharp-sphere**, **twelve**, **trim-star**, **hand**, **mannequin**, **julius**, **bunny**, **igea**, **minerva**, **lion**, **shard**, **max**, **mech**.

We remark that the use of the Armijo-type backtracking rule is justified by the difficult expression (2.26), which makes the accurate estimation of the Lipschitz constant  $L_{\mathcal{J}_V}$  of  $\nabla \mathcal{J}_V(V)$  quite challenging. In the proposed strategy, a (typically) initially large step-size  $\alpha_0$  is then reduced depending on whether the following inequality is verified:

$$\mathcal{J}_V(\bar{V}^{(j+1)}) \leq \mathcal{J}_V(\bar{V}^{(j)}) - c_1 \alpha \|\nabla \mathcal{J}_V(\bar{V}^{(j)})\|_2^2 \quad (2.28)$$

with  $c_1 \in (0, 1)$  and where  $\bar{V}^{(j)}$  denotes the  $j$ -th update of  $V$  given by (2.25).

From the convergence plots, we notice that upon a manual selection of a sufficiently small constant step-size  $\alpha$  the convergence of plain GD without backtracking is as good as the one of the heavy-ball algorithm combined with backtracking. However, the former

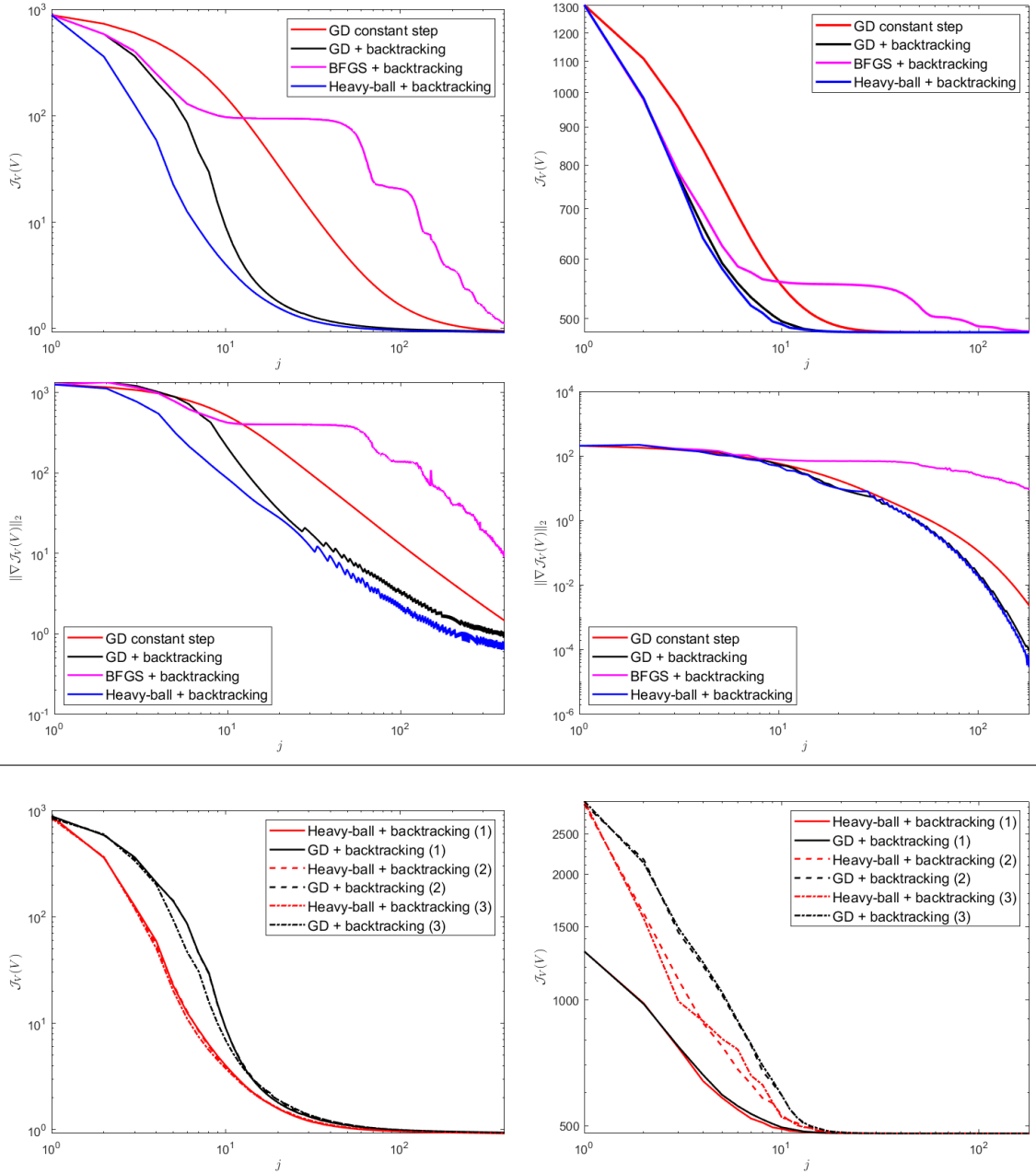


Figure 2.4: First and second rows: plots of the energy  $\mathcal{J}_V$  (first row) and gradient norm (second row) decay for sub-problem (2.24) with GD (with and without backtracking), heavy-ball update with backtracking and BFGS. Third row: energy decay for GD with backtracking and heavy-ball with backtracking, for three different initializations  $V^{(0)}$ . First column mesh **twelve** and second column mesh **block** corrupted by noise level  $\gamma = 0.2$ .

choice is problem-dependent, hence a backtracking strategy automatically adjusting the value of  $\alpha$  to an appropriate size is preferred.

The graphs in Figure 2.4 (third row) show the robustness of the initialization  $\bar{V}^{(0)}$  and show that both GD and heavy-ball with backtracking are consistent, regardless of the chosen initialization. However, the natural and most efficient choice for  $\bar{V}^{(0)}$  is a warm start given by the matrix  $V^{(k)}$  obtained as a solution of the problem (2.24) in the previous ADMM iteration.

The rigorous analysis of the convergence properties of our proposed three-block ADMM scheme following, e.g., [128] is not easy to derive. However, we will provide some evidence of the numerical convergence in Section 2.4.

## 2.3 A practical use of the geometry repair framework

In the following, we provide some details for the practical use of the geometric framework introduced above, adapted to the specific task of denoising, inpainting and completion, as described in Sec.1.3.2 and Sec.1.3.1.

**Feature-aware mesh denoising.** We aim at removing noise and returning a restored surface, which is a 3D mesh that represents as faithfully as possible a piecewise smooth surface, where edges appear as discontinuities in the normal field. To achieve this goal, the natural choice is to set in (2.4)  $M_V = \mathbf{1}_{n_V}$  and define  $M_E$  as in (2.3), so as to distinguish salient edges from smooth regions. In case of severe noise, when the estimate of the mask  $M_E$  may be affected by false edge detections, we suggest recomputing the edge mask  $M_E$  along the ADMM iterations.

**Smooth hole filling/inpainting.** In contrast to techniques for image inpainting, which make use of the given spatial structure of the data (the regular grid of an image), surfaces lack a natural underlying spatial domain, which brings an additional degree of freedom in the setting of the problem. At the same time, vertices' positions encode both function values and the domain of the function to be reconstructed. The initial mesh,  $\mathcal{M}_0 = (V_0, T_0)$  thus has to be set as the original (possibly noisy) incomplete mesh with trivially enclosed and labeled disconnected holes - region  $S_D$  - marked as zeros in  $M_V$ . On the other hand, the mask  $M_E$  can still be defined as in (2.3), by additionally forcing zero values on the edges in  $S_D$ . The proposed geometric repair algorithm then performs simultaneously denoising, outside the holes, and smooth filling in the internal part of the holes, through the regularizer  $\mathcal{R}_2$ .

**Context-aware completion.** In some applications smooth filling of holes is not sufficient: this is the case in archaeology and in general cultural heritage applications where the main goal is the reconstruction of a digital twin of a cultural heritage object.



Some parts of the original 3D model can be damaged or missing but can be completed by means of characteristic parts taken from the object under consideration or from others. Given the original incomplete mesh  $\mathcal{M}_0$  with a region of interest bounded by a curve  $b_0$  and characterized by vertices  $\bar{V} \subset V_0$  and triangles  $\bar{T} \subset T_0$ , together with a template patch  $P = (V_P, T_P)$ , bounded by a curve  $b_P$ , we build a repaired mesh  $\mathcal{M}^*$  by replacing  $(\bar{V}, \bar{T})$  by  $(V_P, T_P)$  and blending the two parts through the proposed variational model.

Note that, in case the region of interest on  $\mathcal{M}_0$  that has to be completed is a hole, then trivially  $(\bar{V}, \bar{T})$  are empty sets.

We assume that the template patch  $P$  is properly aligned in the correct position and that both polygonals  $b_0$  and  $b_P$  are approximants of oriented, closed, simple curves in  $\mathbb{R}^3$  with the same number of vertices. The correct positioning can thus be performed either automatically (by rigid body transformation algorithms) or through user interaction. A narrow band around  $b_0$ , named  $strip(b_0)$ , containing at least 2-disk of triangle neighbors adjacent to  $b_0$ , plays the role of  $S_D$ . Hence  $M_V$  is the characteristic function of  $\mathcal{M}_0 \setminus strip(b_0)$ , i.e. is zeros only on  $strip(b_0)$ .

The operator mask  $M_E$  has values one for each sharp edge in both  $\mathcal{M}_0 \setminus strip(b_0)$  and  $P$ . According to the user desiderata, the blending can be performed in three different ways: edges in  $strip(b_0)$  all zeros in  $M_E$  to force a smooth joint with the template; edges in  $strip(b_0)$  all ones, to keep a sharp connection; edges in  $strip(b_0)$  defined by the spatially adaptive  $M_E$  in (2.3) to maintain geometric continuity  $G^0/G^1$  over the blended region.

The vertices  $V^*$  of the completed surface  $\mathcal{M}^*$  are obtained by minimizing (2.4), properly initialized with  $V^{(0)} = (V_0 \setminus \bar{V}) \cup V_P$ , while maintaining the connectivity defined by  $T^* = (T_0 \setminus \bar{T}) \cup T_P$ . The connectivity  $T^*$  is automatically achieved as we imposed  $b_0 \equiv b_P$ .

We refer the reader to Fig. 2.1 for a visual representation of the three different tasks performed. Moreover, Section 2.4 offers additional insights.

## 2.4 Numerical Examples

We validate the proposed geometric framework both qualitatively and quantitatively on a variety of benchmark triangulated surfaces characterized by different sharpness and smoothness features and on some real datasets.

At the aim of a quantitative validation, meshes  $\mathcal{M}_0 = (V_0, T_0)$  have been synthetically corrupted, to mimic common noise effects described in (1.41). The noisy vertices in  $V_0$  correspond to underlying noise-free vertices  $V_{GT}$  by the following additive degradation model

$$V_0 = V_{GT} + \eta d, \quad (2.29)$$

where the product  $\eta d$  accounts for the noise perturbations. Namely,  $\eta \in \mathbb{R}^{nv}$  is assumed to be at each vertex independently and identically distributed as a zero-mean Gaussian

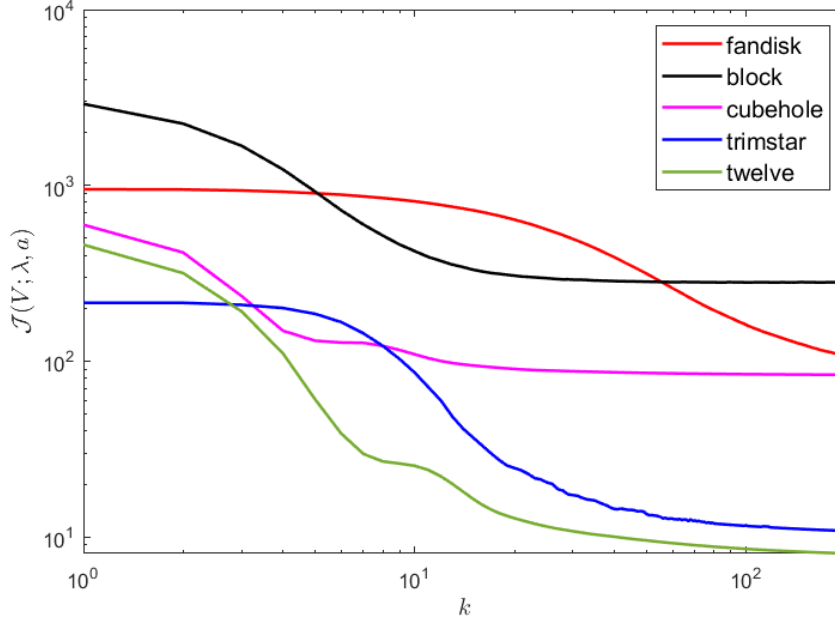


Figure 2.5: Empirical convergence of ADMM algorithm for some reconstructed meshes.

random variable, i.e.  $\eta_i \sim \text{Gauss}(0, \sigma^2)$ ,  $i = 1, \dots, n_V$ , with known variance  $\sigma^2$ , and  $d \in \mathbb{R}^{n_V \times 3}$  is a vector field of noise directions with elements  $d_i \in \mathbb{R}^3$ ,  $\|d_i\| = 1$ ,  $i = 1, \dots, n_V$ , which can be either random directions or the normals to the vertices. The perturbations are thus characterized by a noise level  $\gamma \in \mathbb{R}_+$  defined by  $\sigma = \gamma \bar{l}$ , with  $\bar{l}$  representing the average edge length.

Quantitative evaluation is done in terms of the following error metrics, which measure the discrepancy of the computed  $V^*$ ,  $N^*$  w.r.t. the noise-free mesh  $V_{GT}$ ,  $N_{GT}$ :

- **Mean squared angular error**  $MSAE = \mathbb{E}[\angle(N_{GT}, N^*)^2]$ ,
- **$L_2$  vertex to vertex error**  $E_V = \|V^* - V_{GT}\|_F / n_V$ .

For all the tests, the iterations of the ADMM algorithm are stopped as soon as either of the two following conditions is fulfilled:

$$k > 200, \quad \|V^{(k+1)} - V^{(k)}\|_2 / \|V^{(k)}\|_2 < 10^{-6}. \quad (2.30)$$

Fig. 2.5 shows the energy decay curve versus the number of iterations for some of the meshes reported in this section. We observe that for all meshes considered the energy converges to a stationary value. This represents an empirical validation of the numerical convergence of the proposed ADMM-based minimization scheme. Having performed a comparative analysis between inner solvers in Section 2.2, we used the GD algorithm with backtracking for solving the subproblem for  $V$ , with a warm start strategy allowing us

to restrict to a few numbers (three in our experiments) of GD iterations while achieving good relative accuracy.

With respect to the comparisons shown with competing approaches for mesh repairing, we remark that most of them are based on hierarchical data structures and combined with various heuristic algorithms. On the contrary, the results presented in the following are directly derived from the solution of the proposed unified mathematical optimization problem and do not require any heuristic post-processing procedure.

All the meshes are rendered in a flat shading model and visualized using ParaView software.

**Example 1: feature-aware denoising.** To evaluate the performance of the proposed method for mesh denoising, we compared the results with other state-of-the-art variational methods for mesh denoising, namely the methods introduced in [117, 143, 55, 142], which have been kindly provided by authors of [142] at <https://github.com/bldeng/GuidedDenoising>, and a learning-based approach, presented in [126]. For each method, we show the best results achieved by tuning the corresponding set of parameters.

Fig. 2.6 shows the denoised meshes colored by their mean curvature scalar map, with fixed range, together with zoomed details on mesh edges. From a visual inspection, we notice remarkable overlaps in the denoised meshes obtained from the other compared methods, and severe perturbations of the triangle shapes in the reconstructed meshes. To further demonstrate how robust our approach is w.r.t. to increasing noise perturbation, in Fig. 2.7 we reported qualitative and quantitative results for noise levels  $\gamma = \{0.2, 0.2, 0.3, 0.4, 0.5, 0.6\}$  - from top to bottom. In the last row, the mesh has been corrupted by arbitrary perturbations on the noise directions ( $d_i$ ) in (1.41). Below each recovered surface, we report the quantitative evaluations according to the two error metrics ( $MSAE \times 10^2$ ,  $E_V \times 10^6$ ). Both quantitatively and qualitatively the results confirm the effectiveness of the proposed variational model in preserving sharp features while smoothly recovering rounded parts. Finally, we can comment on the efficiency of our algorithm whose computational time is, on average, one order less than the  $\ell_2 - \ell_0$  denoising method [118] which is the slowest, while it is comparable to the other compared methods.

To improve the estimation of mask  $M_E$  for severe noise, we dynamically updated the edge mask  $M_E$  every three ADMM iterations.

**Example 2: hole filling/inpainting.** We applied our geometric framework for the recovery of various meshes  $\mathcal{M}_0$  which exhibit holes or damaged parts. Fig.2.2 illustrates the basic workflow for the inpainting task on **angel** mesh which takes as input the original eventually noisy mesh  $\mathcal{M}_0$  (Fig.2.2, left) and the inpainting mask  $M_V$ , which can be of arbitrary topology, in the figure the holes to be filled are marked as 0 in  $M_V$  and blue colored. The recovery of **angel** mesh using smooth hole filling is illustrated in Fig.2.2(second row).

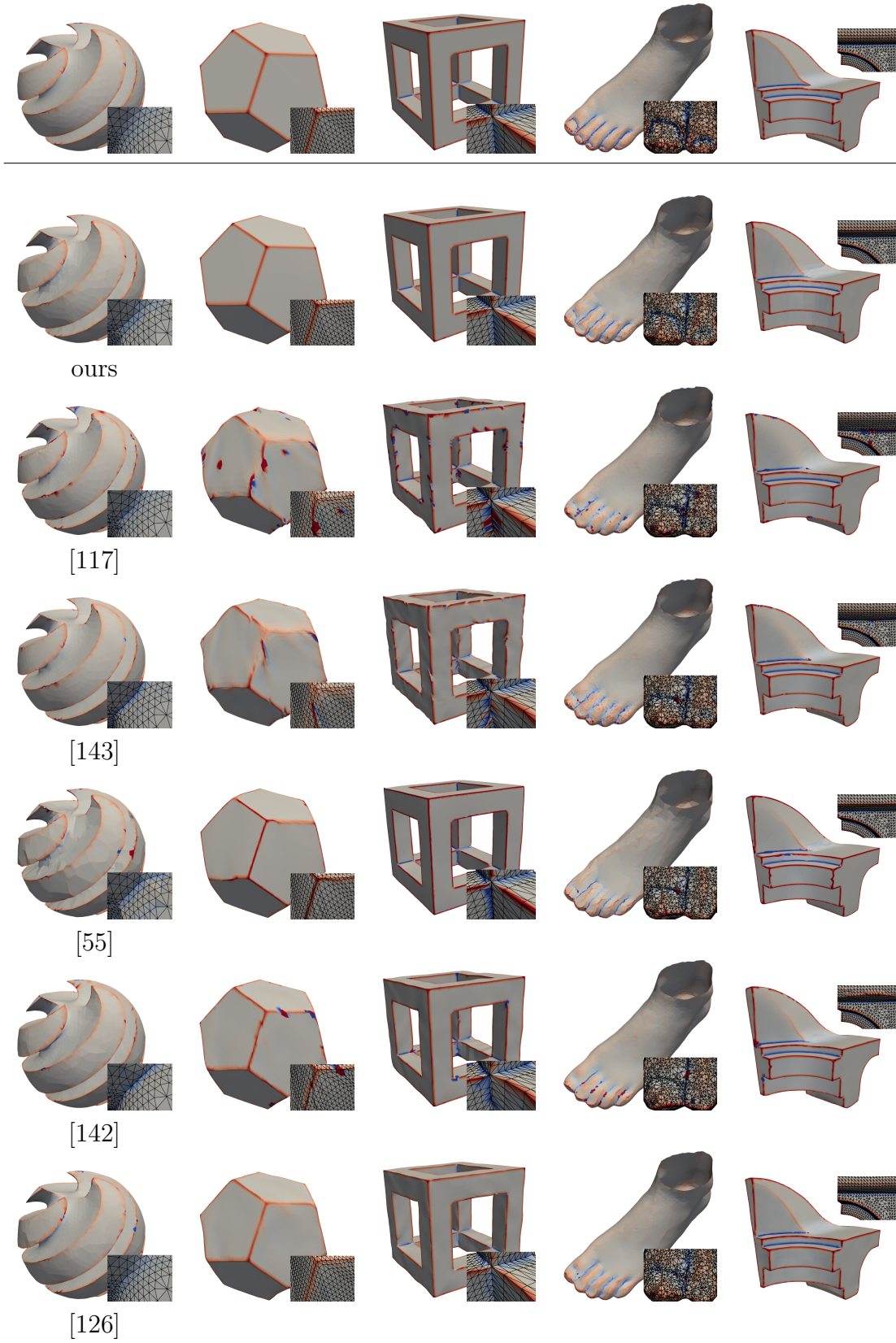


Figure 2.6: Examples of denoising: results of noisy-free input meshes (first row) corrupted by noise levels  $\gamma = \{0.15, 0.3, 0.3, 0.2, 0.2\}$ , from left to right.



















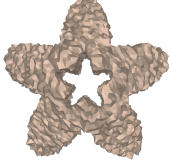





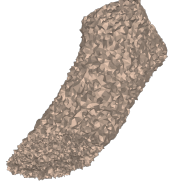





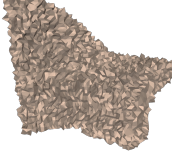
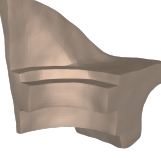
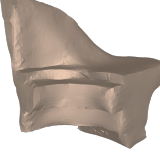
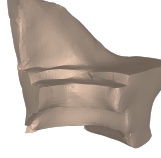
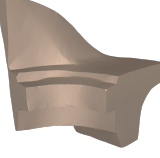
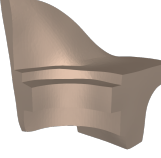






					
$\gamma = 0.2$	(0.62;3.84)	(2.11;8.96)	(0.79;4.24)	(1.37;5.53)	(1.02;4.92)
					
$\gamma = 0.2$	(0.93;130)	(2.40;170)	(1.52;160)	(2.99;210)	(3.70;230)
					
$\gamma = 0.3$	(2.15;6.42)	(3.05;7.15)	(2.19;6.37)	(4.82;14)	(2.25;6.56)
					
$\gamma = 0.4$	(3.98;51.3)	(13.56;72.6)	(10.55;54.3)	(7.97;93.7)	(9.79;62.4)
					
$\gamma = 0.5$	(2.85;41.7)	(9.84;74.4)	(6.18;43.4)	(10.7;71)	(8.33;69.2)
					
$\gamma = 0.6$	(3.17;88.0)	(10.6;144)	(11.8;150)	(5.93;180)	(6.45;143)
					
$\gamma = 0.2$	(2.5;5.9)	(4.51;6.33)	(6.16;6.87)	(4.2;6.53)	(5.34;6.56)
input $V^0$	ours	[117]	[143]	[55]	[142]

Figure 2.7: Examples of denoising: comparison of our denoising framework with related works on meshes synthetically corrupted by noise levels  $\gamma$ . Reported metrics: ( $MSAE \times 10^{-2}$ ;  $E_V \times 10^{-6}$ )

Fig.2.8 (first row), shows the challenging *Igea* mesh which presents a deep groove on the left side of the mouth and a shallower one on the right cheek. Our geometric framework was able to inpaint the shallower hole perfectly, while the deep one was filled in a satisfactory, even if not complete, way. This is justified by the different contribution of Willmore vs sparsity-inducing penalties. The latter acts more strongly with respect to the former, especially for high levels of noise. Hence, adding suitable weights to the two penalties could overcome this disparity.

The data set *minerva* shown in Fig.2.8 (second row, first column) presents a few holes caused by the scanner acquisition, in the head and under the nose. Moreover, a vertical strip has been intentionally added to the inpainting region  $S_D$  in order to remove the groove provoked by the gluing of the two parts of the *minerva*'s face. This dataset has been provided by ENEA, Bologna, Italy, and acquired by a VIVID laser scanner. The dataset presents inherent noise due to the optical acquisition system. The result of repairing the damaged geometry and filling surface holes is illustrated in Fig.2.8(c).

In Fig.2.8 (third row) the inpainting framework has been applied to repair a *shard* from neolithic pottery received by the CEPAM laboratory (CNRS France), obtained by fusion of more fragments. The inpainting region, shown in Fig.2.8 (third row, second column) has been intentionally imposed to eliminate obvious fractures between joined fragments.

**Example 3: context-aware completion.** We finally applied context-aware completion as an editing tool for seamless object fusion. Completion results for the meshes *lion*, *screwdriver*, and *igea* are illustrated in Fig.2.9-2.10. The templates  $P$  smoothly complete the original surfaces.

A critical aspect in context-aware completion is the continuity imposed in the joint region, which we denoted by  $strip(b_0)$ . Conditions for geometric continuity between parametric surfaces are well assessed, while for meshes a rigorous treatment on this topic is still missing. In our framework, according to the user's desiderata, the template  $P$  can be joined to  $\mathcal{M}_0$ , both smoothly, by setting  $M_E(strip(b_0)) \equiv 0$ , in a sharp manner by setting  $M_E(strip(b_0)) \equiv 1$ , or in a blended fashion by simply using the  $M_E$  mask of one of the two meshes (or even a combination of them). Therefore, imposing different continuity conditions for  $strip(b_0)$  means defining in a different way the mask  $M_E$  in correspondence to the  $strip(b_0)$ .

A typical example is shown in Fig.2.11(left panel) where a synthetically created hole on the *fandisk* mesh  $\mathcal{M}_0$  is filled with a similar corner patch - template  $P$  cyan colored. In the right panel, we report details onto the completion area  $\mathcal{M}_0 \cup P$  (a), output  $\mathcal{M}^*$  for  $M_E(strip(b_0)) \equiv 0$  (b),  $\mathcal{M}^*$  for  $M_E(strip(b_0)) \equiv 1$  (c),  $\mathcal{M}^*$  for  $M_E(strip(b_0))$  estimated from dihedral angles (d). Note that the initial boundary  $b_P$  was larger than  $b_0$  and slightly shifted. Nevertheless, the feature-adaptive regularization perfectly respects the continuity of  $strip(b_0)$ , as illustrated in Fig.2.11(d), while a smooth mask - Fig.2.11(b) - destroys the sharp edges, and a non-smooth joint - Fig.2.11(c) - creates artifact features.



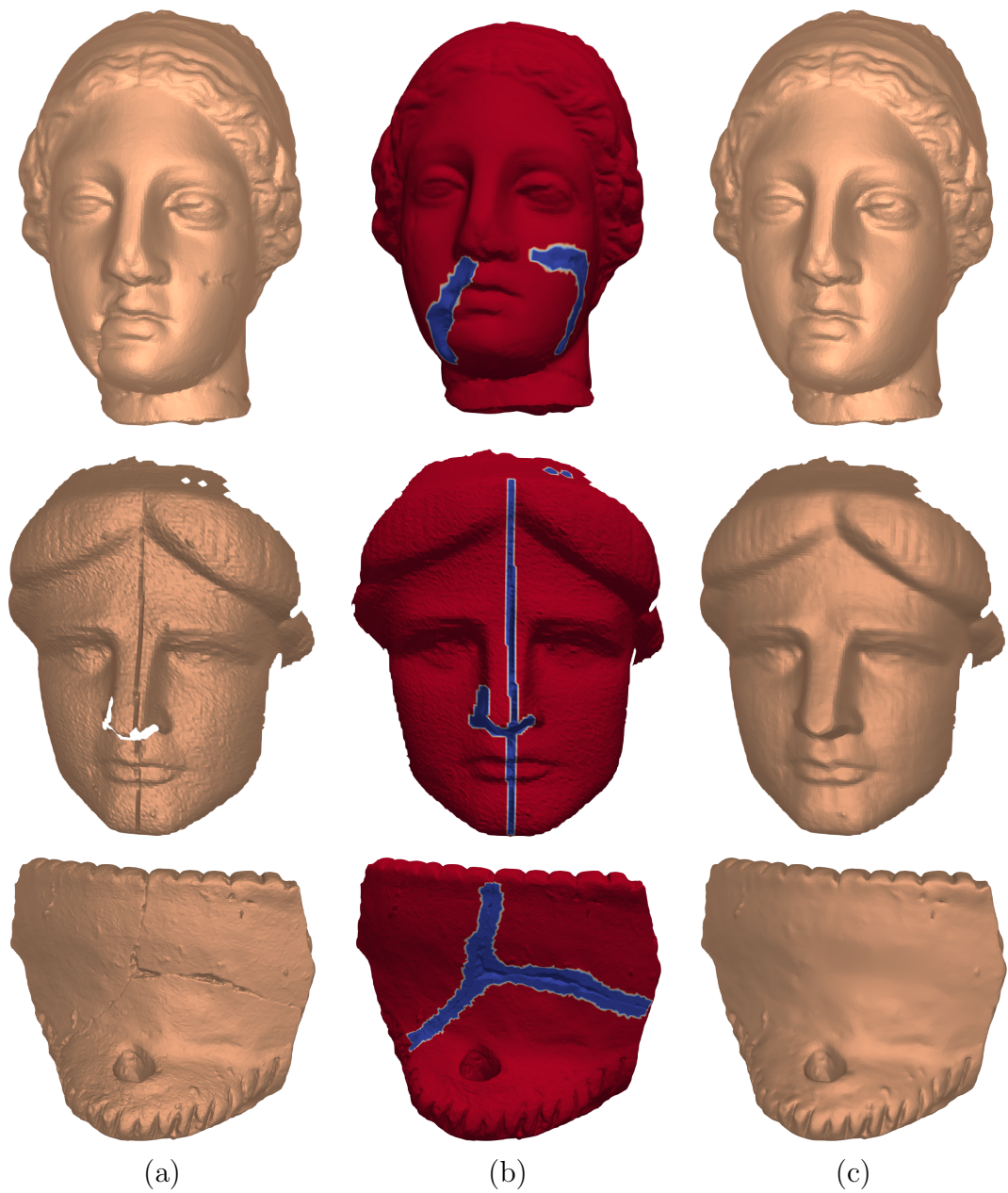


Figure 2.8: Examples of surface inpainting: (a) original damaged object; (b) inpainting mask  $M_V$ ; (c) inpainted surface.

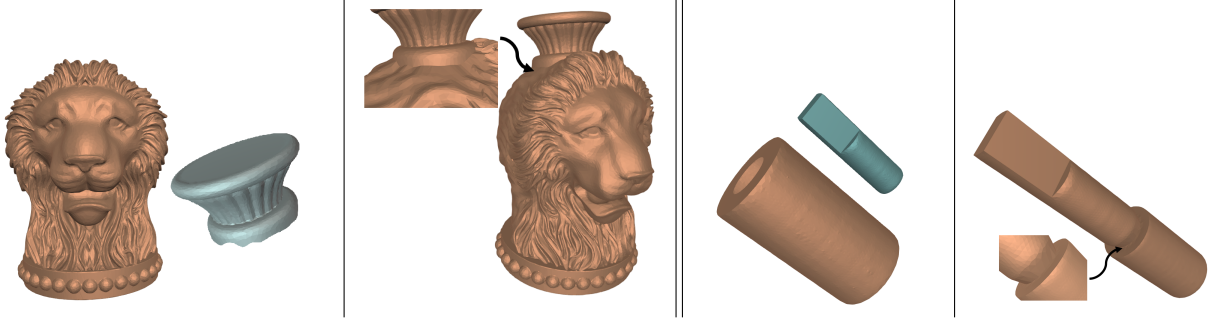


Figure 2.9: Examples of context-aware completion:  $\mathcal{M}_0$  with template patch  $P$ , output  $\mathcal{M}^*$  for two different incomplete meshes.

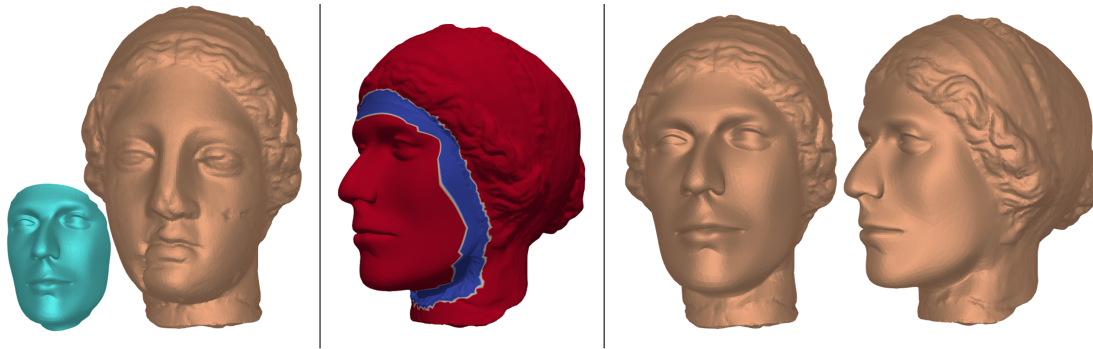


Figure 2.10: Example of context-aware completion:  $\mathcal{M}_0$  with template patch  $P$  (left); mask  $M_V$  (middle); output  $\mathcal{M}^*$  from two different camera points of view (right).



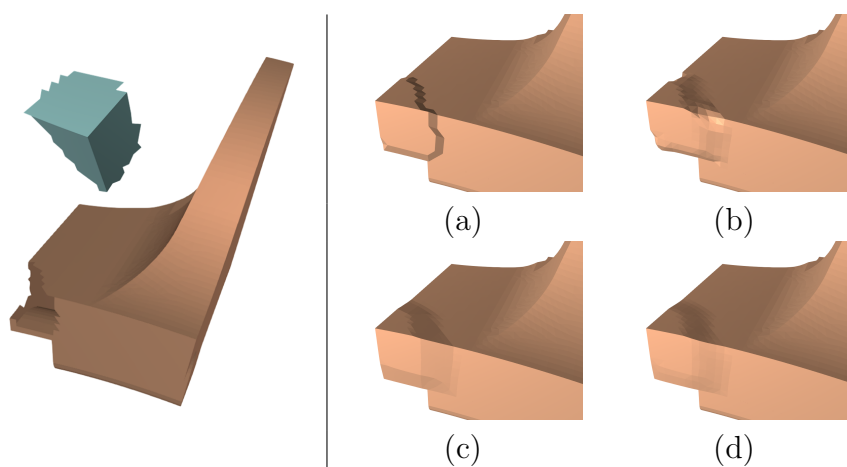


Figure 2.11: Examples of different continuity conditions for  $\text{strip}(b_0)$  in  $M_E$  for context-aware completion:  $\mathcal{M}_0$  with a hole bounded by  $b_0$  and template  $P$  (left). Right, zoom onto the completion area  $\mathcal{M}_0 \cup P$ , i.e.  $V^{(0)}$  (a), output  $\mathcal{M}^*$  for  $M_E(\text{strip}(b_0)) \equiv 0$  (b),  $\mathcal{M}^*$  for  $M_E(\text{strip}(b_0)) \equiv 1$  (c),  $\mathcal{M}^*$  for  $M_E(\text{strip}(b_0))$  estimated from dihedral angles (d).

# Chapter 3

## Variational recovery in differential coordinates

In this Chapter, we focus on a specific mesh deformation task, called Geometric Texture Transfer (GTT), formalized in (1.46).

As seen in Sec.1.3.4, the task has a natural formulation when the base surface and the texture surface are defined as parametric surfaces on the same domain. In this case, the displacement mapping technique allows to define the textured mesh as in (1.47).

However, in most real applications, 3D shapes are commonly represented as arbitrary topology, irregular, triangular meshes, which demand texture atlases generation and charts parametrization, before applying the direct formulation (1.47). For example, in [68], both the base surface and the texture patch are preliminarily mapped onto geometry images.

To avoid the critical parametrization issue, a different strategy consists in using implicit surface representation. For example, in [45] the authors combine the implicit representation of the underlying smooth surface with so-called detail particles, i.e. particle sets containing the offset vectors from the detailed surface. The texture transfer task consists in level-set evolution driven by speed function derived from the detail particles.

Alternatively, spectral shape representation allows to apply a multi-scale empirical mode decomposition (EMD) to the source object, performing the GTT task by adding selected modes to the EMD of the target surface, (see [141]). A similar spectral-based idea is presented in [84] where the geometric texture from the source object is encoded via spectral decomposition of Laplacian, then combined with the spectral decomposition of the target.

With the rapid development of machine learning approaches, new neural implicit representations for 3D geometry have been introduced. In [56, 136] the authors applied a convolutional neural network to synthesize a geometric texture from a sample object and to transfer it over the whole target object. GTT using neural representations is memory/computation demanding and still very challenging in transferring texture on a specific bounded patch of a surface.

In this Chapter, we propose a Geometric Texture Transfer method that does not

require a bijective relation between the base surface and the texture. With respect to GTT using implicit representations, the proposed GTT approaches avoid the timing and accuracy shortcomings of the level-set modeling. Finally, unlike the multi-block procedures proposed in computer graphics literature to deal with the GTT task, we face the problem from a numerical point of view, offering a simple, compact mathematical variational formulation with efficient solutions.

To make the transfer of geometric details as effective as possible, it is first of all essential to represent them at their best. The Euclidean coordinates have been a ubiquitous choice in geometry processing due to their intuitive properties of representing the spatial embedding of a shape. However, they fail to capture the geometric features relevant for many shape analysis tasks [25].

Alternate descriptors as the Differential Coordinates introduced in Sec. 1.2.3 have been effectively used for deformation tasks (see Sec. 1.5), for their ability to encode aspects of extrinsic geometry, and therefore they are potentially suitable also for the GTT task. In this Chapter, we present a further analysis of the properties of invariance and equivariance of Laplacian Coordinates, Normal-Controlled Coordinates and Mean Value Encoding, under affine transformations and free deformations. This represents indeed a key aspect of their use in the context of the GTT task.

Contrarily to the displacement mapping described in 1.2 that acts on parametric surfaces, in our GTT proposal the macrostructure base surface is a Riemannian manifold of arbitrary topology embedded in  $\mathbb{R}^3$  represented by an irregular base mesh  $\mathcal{M}_I$ , defined by its vertices, edges and triangular faces  $(V_I, E_I, T_I)$ , respectively, and the displacement / geometric texture is an open mesh  $\mathcal{M}_S = (V_S, E_S, T_S)$  with boundary  $b_S := \partial\mathcal{M}_S$  defined by a geometry image. We assume that the underlying base mesh is coarse relative to the fine scale of details in the texture. We aim at building a new textured mesh  $\mathcal{M}_T = (V_T, E_T, T_T)$ , which corresponds to the base mesh  $\mathcal{M}_I$  everywhere except in a bounded patch  $P \subset \mathcal{M}_I$  where the geometric texture is mapped on the base mesh, while preserving as much as possible the underlying original shape of  $\mathcal{M}_I$ , i.e.  $V_T|_{\mathcal{M}_I-P} = V_I|_{\mathcal{M}_I-P}$ , and  $V_T|_P$  is displaced by transferring the level of details of  $V_S$ .

The geometric transfer process is illustrated in Fig.3.1 where a column of stones is modeled by mapping on half of a cylindrical shape a geometric textured surface built from a gray-scale image. The involved underlying meshes are represented in Fig.3.1, bottom row.

The Chapter is divided as follows. In Section 3.1 we investigate properties of Laplacian, Normal-Controlled and Mean Value coordinates under 3D transformations; in Section 3.2, associated with the three considered geometric descriptors, we formulate and analyze three variational models for the solution of the GTT problem which involve two terms: one for the inverse reconstruction problem and one for shape preserving interpolation. In Section 3.3 we design a nonlinear optimization algorithm to efficiently solve the GTT models. In Section 3.4 we compare the results of transferring geometric textures

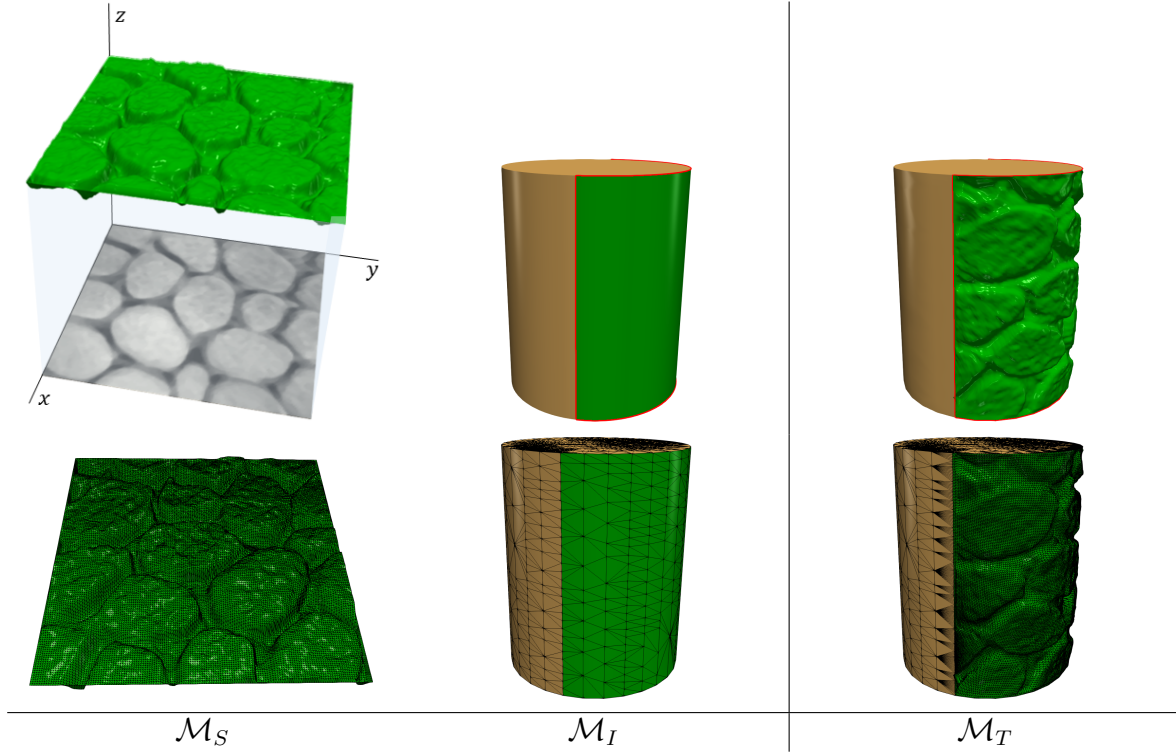


Figure 3.1: The geometric transfer process. First column: geometric texture  $\mathcal{M}_S$ , derived from the texture image; second column: base mesh  $\mathcal{M}_I$ , with the patch  $P \subset \mathcal{M}_I$  colored in green; third column: textured mesh  $\mathcal{M}_T$ .

both qualitatively and quantitatively on a wide range of examples.

### 3.1 Invariance of shape descriptors

Let  $G$  be the affine transformation group (containing translations, rotations, shearing and scaling). A  $n$ -dimensional real representation of a group  $G$  is a map  $T_g : G \rightarrow \mathbb{R}^{n \times n}$ , assigning to each  $g \in G$  an invertible matrix  $T_g$ . We are interested in transformations in homogeneous space  $T_g \in \mathbb{R}^{4 \times 4}$  represented as

$$T_g = \begin{bmatrix} a_1 & b_1 & c_1 & t_x \\ a_2 & b_2 & c_2 & t_y \\ a_3 & b_3 & c_3 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad (3.1)$$

where the principal three-by-three sub-matrix represents linear transformations such as scaling and rotation, while  $t = [t_x, t_y, t_z]^T$  denotes the translation vector. To characterize

the invariance properties of the shape descriptors, we introduce the following concepts of invariance with respect to transformations  $g \in G$ .

Let  $\Omega$  be the domain underlying our data, and  $X(\Omega)$  the space of functions on  $\Omega$ . A function  $f : X(\Omega) \rightarrow Y$  is  $G$ -invariant if

$$f(T_g x) = f(x) \quad (3.2)$$

for all  $g \in G$ , and  $x \in X(\Omega)$ , i.e., its output is unaffected by the group action on the input.

A function  $f : X(\Omega) \rightarrow X(\Omega)$  is  $G$ -equivariant if

$$f(T_g x) = T_g f(x), \quad (3.3)$$

for all  $g \in G$ , i.e., group action on the input affects the output in the same way.

We focus now on the main transformations involved in the geometric transfer task, such as translation, rotation along an axis, and isotropic scaling, represented as  $T_g$  by (3.1). In Fig. 3.2, left, and Fig. 3.3, left, we summarize the properties of  $G$ -invariance and  $G$ -equivariance of the shape descriptors presented in Section 1.2.3, i.e. Laplacian Coordinates, Normal-Controlled Coordinates and Mean Value Encoding; with the symbol  $\checkmark$  we denote satisfied, and by  $\times$  not satisfied. The difference between the two concepts of  $G$ -invariance and  $G$ -equivariance maps is illustrated in the right panels of Fig. 3.2 and Fig. 3.3.

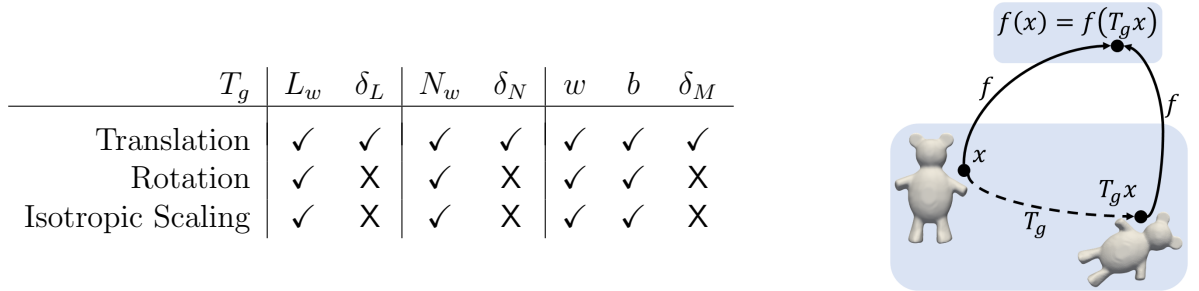


Figure 3.2: Geometric descriptor  $G$ -invariance  $f(T_g x) = f(x)$ . Left: table for all descriptors. Right: Illustration of  $G$ -invariance map  $f$ .

*Laplacian coordinates.* A transformation  $T_g$  changes the geometry of the mesh, leaving unchanged its local connectivity, thus the uniform weights  $w_{ij}$  in (1.22) remain invariant and not equivariant. In case the cotangent weights  $w_{ij}$  in (1.23) are used, they remain invariant since the angles in the local neighborhood are neither modified by rigid transformation, nor by isotropic scaling. Therefore, the Laplacian  $L_w$  is an intrinsic linear operator that remains invariant under isometric transformations - such as translations and rotations in  $\mathbb{R}^3$  - which do not change the metric. A different behavior is expected for the LAP coordinates, as shown in Prop. 3.1.1.

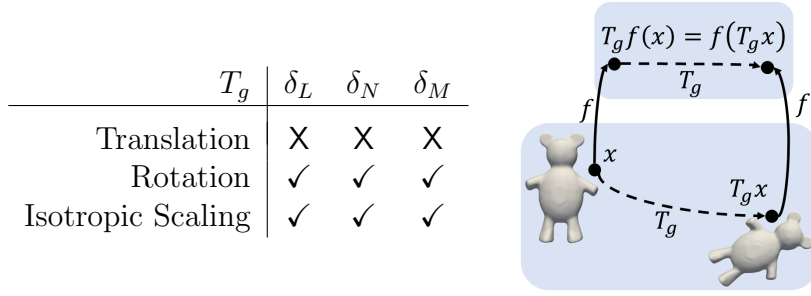


Figure 3.3: Geometric descriptor G-equivariance  $f(T_g x) = T_g f(x)$ . Left: table for all descriptors. Right: Illustration of G-equivariance map  $f$ .

**Proposition 3.1.1.** *Let  $\delta_L$  be the map defined in (1.28) and consider the transformation  $T_g$  of (3.1). Then  $\delta_L$  is  $G$ -invariant for  $T_g$  translation, and  $G$ -equivariant for  $T_g$  being a rotation along an axis or an isotropic scaling.*

*Proof.* Let  $\delta_L$  be the vector of LAP coords of the mesh  $\delta_L = L_w(V)$  and  $\delta'_L$  the vector of the LAP coords of the transformed mesh  $\delta'_L = L_w(T_g V)$ . If  $T_g = T$ , with  $T$  the translation matrix obtained by replacing in  $T_g$  of (3.1) the principal submatrix with the  $I_3$  identity matrix, then

$$(\delta'_L)_i = (L_w(TV))_i = \sum w_{ij}(v_i + t - v_j - t) = (L_w(V))_i = (\delta_L)_i.$$

If  $T_g = S$ , with  $S$  the diagonal matrix obtained by setting  $a_1 = b_2 = c_3 = s \in \mathbb{R}$  and  $t_x = t_y = t_z = 0$  in (3.1), which represents the isotropic scaling of a factor  $s$ , then, due to the linearity of  $L_w$ ,

$$(\delta'_L)_i = (L_w(SV))_i = \sum w_{ij}(sv_i - sv_j) = (sL_w(V))_i = (S\delta_L)_i.$$

If  $T_g = R$ , with  $R$  an orthogonal rotation matrix, and  $t_x = t_y = t_z = 0$ , then

$$(\delta'_L)_i = (L_w(RV))_i = Rv_i - \sum w_{ij}Rv_j = R\left(v_i - \sum w_{ij}v_j\right) = R(L_w(V))_i = (R\delta_L)_i.$$

□

*Normal-controlled coordinates.* Similarly to the Laplacian matrix  $L_w$ , the weights  $w_{ij}$  of  $N_w$  in (1.30) remain unchanged under translation and rotation, which do not affect the shape of the projected neighborhood. Isotropic scaling does not alter  $\gamma_{ij}$  and  $\delta_{ij}$  in (1.30), while affects  $l_{ij}$ . Nonetheless, the weight normalization  $w_{i,j}$  from  $\bar{w}_{i,j}$  in (1.30) allows to preserve the invariance under isotropic scaling. Following the same arguments of Prop. 3.1.1, one can show that the map  $\delta_N$  defined in (1.31) is invariant for translation and equivariant for rotation and scaling.

*Mean Value Encoding.* The weights  $w_{ij}$ , defined in (1.35), differ from NCC weights only for the choice of the local projection plane. Thus the  $w_{ij}$  are invariant under all the affine transformations. The components  $b_{ij}$ , as illustrated in Fig. 1.1(c), define the angle between the edge  $e_{ij}$  and the vertex normal  $n_i$ , which remains preserved both for rigid transformation and isotropic scaling. The coordinates  $\delta_M$  combine non-linearly the invariant weights  $w_{ij}$ ,  $b_{ij}$  with the normal vector  $n_i$ , thus the invariance/equivariance of  $\delta_M$  is less trivial to verify.

**Proposition 3.1.2.** *Let  $\delta_M$  be the map defined in (1.37). Then  $\delta_M$  is  $G$ -invariant for  $T_g$  translation, and  $G$ -equivariant for  $T_g$  being a rotation along an axis or an isotropic scaling. Let  $F$  be the map defined in (1.38). Then  $F$  is  $G$ -invariant for  $T_g$  translation, rotation and uniform scaling.*

*Proof.* Let us define the average distance from origin as

$$d(v_i) := -\frac{1}{|\mathcal{N}(i)|} \sum_{j \in \mathcal{N}(i)} \langle n_i, v_j \rangle.$$

For  $T_g$  defining translation, for each vertex we have  $T_g v_i = v_i + t$  for  $t = [t_x, t_y, t_z]^T \in \mathbb{R}^3$  and  $d(v_i + t) = d(v_i) - \langle n_i, t \rangle$ . Then

$$(v_i + t)' = v_i + t - (d(v_i) - \langle n_i, t \rangle + \langle v_i + t, n_i \rangle) n_i = v_i' + t. \quad (3.4)$$

By applying (3.4), we get

$$\begin{aligned} \mathcal{V}(T_g v_i) &= \mathcal{V}(v_i + t) = \sum_{j \in \mathcal{N}(i)} w_{ij} (\|(v_i + t)' - (v_j + t)'\| b_{ij} + (v_j + t - (v_j + t)') \cdot n_i) n_i = \\ &= \sum_{j \in \mathcal{N}(i)} w_{ij} (\|(v_i' - v_j')\| b_{ij} + (v_j - v_j') \cdot n_i) n_i = \mathcal{V}(v_i) = \delta_M. \end{aligned}$$

For  $T_g$  being rotation, due to the orthogonality of  $T_g$  and the fact that  $n_i(T_g v_i) = T_g n_i$ , we have

$$d(T_g v_i) = -\frac{1}{|\mathcal{N}(i)|} \sum_{j \in \mathcal{N}(i)} \langle T_g n_i, T_g v_j \rangle = -\frac{1}{|\mathcal{N}(i)|} \sum_{j \in \mathcal{N}(i)} n_i^T T_g^T T_g v_j = d(v_i)$$

and the projection of the rotated vertices is given by

$$(T_g v_i)' = T_g v_i - (d(v_i) + \langle T_g v_i, T_g n_i \rangle) T_g n_i = T_g (v_i - (d(v_i) + \langle v_i, n_i \rangle) n_i) = T_g v_i'.$$

Then

$$\begin{aligned}
\mathcal{V}(T_g v_i) &= \sum_{j \in \mathcal{N}(i)} w_{ij} (\|T_g v'_i - T_g v'_j\| b_{ij} + \langle T_g v_j - T_g v'_j, T_g n_i \rangle) T_g n_i = \\
&= \sum_{j \in \mathcal{N}(i)} w_{ij} (\|T_g(v'_i - v'_j)\| b_{ij} + \langle T_g(v_j - v'_j), T_g n_i \rangle) T_g n_i = \\
&= T_g \left( \sum_{j \in \mathcal{N}(i)} w_{ij} (\|v'_i - v'_j\| b_{ij} + \langle v_j - v'_j, n_i \rangle) n_i \right) = T_g \mathcal{V}(v_i).
\end{aligned}$$

The isotropic scaling can be expressed as  $T_g = sI_3$ , where  $s = a_1 = b_2 = c_3 \in \mathbb{R}$  in  $T_g$ , and it affects the normal vector  $n_i$ . However,

$$d(T_g v_i) = -\frac{1}{|\mathcal{N}(i)|} \sum_{j \in \mathcal{N}(i)} \langle n_i, T_g v_j \rangle = -\frac{1}{|\mathcal{N}(i)|} \sum_{j \in \mathcal{N}(i)} n_i^T (sI_3) v_j = sd(v_i)$$

and

$$(T_g v_i)' = T_g v_i - (sd(v_i) + \langle T_g v_i, n_i \rangle) n_i = T_g(v_i - (d(v_i) + \langle v_i, n_i \rangle) n_i) = T_g v'_i.$$

Therefore

$$\begin{aligned}
\mathcal{V}(T_g v_i) &= \sum_{j \in \mathcal{N}(i)} w_{ij} (\|T_g(v'_i - v'_j)\| b_{ij} + \langle T_g(v_j - v'_j), n_i \rangle) n_i = \\
&= \sum_{j \in \mathcal{N}(i)} w_{ij} (s\|v'_i - v'_j\| b_{ij} + s\langle v_j - v'_j, n_i \rangle) n_i \\
&= T_g \left( \sum_{j \in \mathcal{N}(i)} w_{ij} (\|v'_i - v'_j\| b_{ij} + \langle v_j - v'_j, n_i \rangle) n_i \right) = T_g \mathcal{V}(v_i).
\end{aligned}$$

□

In a general geometric processing setup, the transformations  $T_g$  that have acted on an object are not known. Hence descriptors that are G-invariant are trivially preferable to G-equivariant geometric descriptors which rely on the knowledge of the  $T_g$  applied in order to be used on the transformed mesh. This highlights one of the advantages of the MVE coordinates over the LAP and NCC. In particular, for the investigated task of GTT, the benefit of using MVE coordinates in the form of the non-linear map  $F$ , which allows to exploit the G-invariance property, is illustrated in Fig. 3.4 where a chocolate tablet geometric texture  $\mathcal{M}_S$  is applied to a face  $P$  of a cube object  $\mathcal{M}_I$ . The placing of  $\mathcal{M}_S$  with respect to  $P$  requires a rotation in the first row and an isotropic scaling in the second row of Fig. 3.4. The use of LAP and NCC, being not invariant under rotation and scaling, introduces visible artifacts when the deformation contains rotation or scaling, while the use of MVE allows to preserve the orientation and the dimension of the structural details.



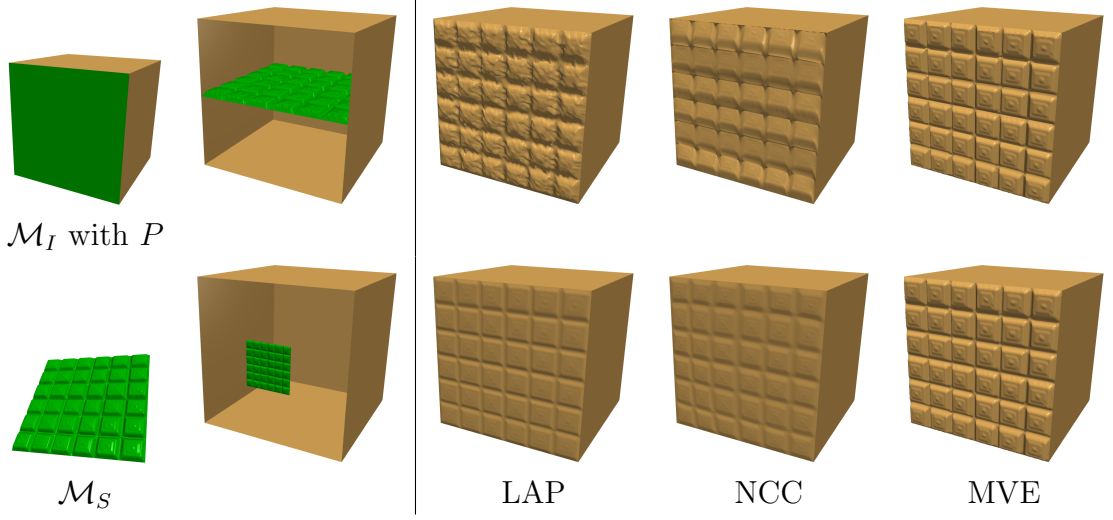


Figure 3.4: Invariance w.r.t geometric texture transfer task. The use of LAP and NCC coordinates, being not invariant under rotation and scaling, introduces visible artifacts when the deformation contains rotation (first row) or scaling (second row).

### 3.1.1 Inexact invariance under free-form deformation

The invariance properties introduced capture a geometry processing context where the texture-transferring task only requires affine transformations. In more realistic GTT we are more interested in a far larger set of transformations where global, exact invariance is replaced by a local, inexact one. We need in fact to deal with 3D geometry textures that undergo non-rigid deformations when applied to free-form objects. Such deformations can be modeled as transformations that preserve as much as possible the intrinsic structure of the underlying (Riemannian) manifold, while transferring at best the descriptors captured by the geometric texture.

Let  $D$  denote the space of Riemannian manifolds, and  $\mathcal{M}_S, \mathcal{M}_I \in D$ . Then the deformation of  $\mathcal{M}_S$  onto  $\mathcal{M}_I$  is a family of smooth and invertible maps  $\eta : D \rightarrow D$  that is modelled by the combination of an affine-transformation  $T_g$  that ‘aligns’ the domains in a way that the corresponding structures are best preserved, and a non-rigid deformation  $T_\tau$  to mold  $\mathcal{M}_S$  on the structure of  $\mathcal{M}_I$ :

$$\eta = T_\tau \circ T_g. \quad (3.5)$$

We can now extend the previous definitions of exact invariance and equivariance under group actions with a ‘softer’ notion of deformation stability (or inexact invariance):

$$\|f(T_\eta x) - f(x)\| \leq Cc(\tau)\|x\|, \quad \forall x \in D \quad (3.6)$$

where  $C$  is some constant independent of  $x$ , and the cost measure  $c(\tau)$  is such that  $c(\tau) = 0$  whenever  $\tau$  is an affine transformation, that is  $\tau \in G$ , thus generalising the  $G$ -

invariance property. A function  $f$  satisfying the above equation is said to be geometrically stable. Although  $c(\tau)$  is difficult to measure, we will incorporate (3.6) in the variational formulation proposed for the GTT solution in Section 3.2.

In the context of geometric texture transfer, a practical deformation cost can be computed when a reference ground truth textured surface  $\mathcal{M}_{GT}$  is available (for example as the result of displacement mapping onto a parametric surface), as the distance to the generated GTT textured surface  $\mathcal{M}_T$ . In general, a quantitative evaluation of the discrepancy between a manifold  $\Omega \in D$  and another manifold  $\bar{\Omega} \in D$  is the Hausdorff distance  $d_H(\Omega, \bar{\Omega})$  between the two meshes, defined as

$$d_H(\Omega, \bar{\Omega}) = \max \left\{ \sup_{x \in \Omega} \inf_{y \in \bar{\Omega}} d(x, y), \sup_{y \in \bar{\Omega}} \inf_{x \in \Omega} d(y, x) \right\}, \quad (3.7)$$

with  $d(x, y)$  being the Euclidean distance between vertices  $x$  and  $y$ .

Let  $\mathcal{M}_T$  be generated by GTT exploiting the geometric descriptors  $f$ , named GTT-f, then GTT-f is stable if  $d_H(\mathcal{M}_{GT}, \mathcal{M}_T)$  is small, being  $d_H(\mathcal{M}_{GT}, \mathcal{M}_T) = 0$  when the two manifolds are isometric.

## 3.2 Variational models for Geometric Texture Transfer

The geometric texture transfer is obtained by replacing the coarse patch  $P \subset \mathcal{M}_I$  with the finer mesh  $\mathcal{M}_S$  whose  $n_S$  vertices must be appropriately deformed into new positions  $V^*$  by an unknown deformation  $\eta$  defined in (3.5), thus preserving the shape of  $\mathcal{M}_I$  while adding the geometric textured details of  $\mathcal{M}_S$ . The resulting textured mesh  $\mathcal{M}_T$  has vertices  $V_T = (V_I \setminus P) \cup V^*$ . We propose to place the vertex positions  $V^*$  through a variational approach, taking inspiration from the deformation models (1.62, 1.65) defined in Sec. 1.5. In particular, we minimize the sum of a reconstruction term  $R(V)$  that aims at preserving the details of the texture, with a soft constraint term  $g(V)$  that forces  $V^*$  to recover the global shape of the patch  $P$ , which reads as

$$V^* = \arg \min_V \{R(V) + g(V; \lambda_C)\}, \quad (3.8)$$

where  $\lambda_C := \lambda \cdot 1_C(V)$ ,  $1_C(V)$  is the indicator function of a subset  $C$  of vertices of  $V_S$  with value 1 at points of  $C$  and 0 at points of  $V_S \setminus C$ ,  $\lambda \in \mathbb{R}$ , is a positive parameter. The set  $V^*$  has the same number of vertices  $n_S$  as  $\mathcal{M}_S$  and inherits its topological structure, and thus it represents the details with the appropriate fine mesh resolution. The term  $R(V)$  depends on the particular geometric descriptor involved, then  $R(V)$  can be formulated as the inverse reconstruction problems (1.29), (1.33) or (1.39), depending on whether the

transfer is obtained via LAP, NCC or MVE descriptors, respectively. This choice leads to the following three alternative variational problems

$$V^* = \arg \min_{V \in \mathbb{R}^{n_S \times 3}} \{ \mathcal{J}_1(V) := \frac{1}{2} \|L_w V - \delta_L\|_2^2 + g(V; \lambda_C) \} \quad (3.9)$$

$$V^* = \arg \min_{V \in \mathbb{R}^{n_S \times 3}} \{ \mathcal{J}_2(V) := \frac{1}{2} \|N_w V - \delta_N\|_2^2 + g(V; \lambda_C) \} \quad (3.10)$$

$$V^* = \arg \min_{V \in \mathbb{R}^{n_S \times 3}} \{ \mathcal{J}_3(V) := \frac{1}{2} \|V - F(V; w, b)\|_2^2 + g(V; \lambda_C) \}; \quad (3.11)$$

where  $w, b$  refer to the source mesh  $\mathcal{M}_S$ . The three variational models (3.9), (3.10), and (3.11) share the same soft constraint term  $g(V)$ , and Dirichlet boundary conditions in order to correctly align with the boundary  $b_P$  of the replaced patch  $P$  and to obtain a unique set of vertices  $V^*$ .

Details on boundary setting are given in Section 3.2.1, the discussion on the soft constraints will be given in Section 3.2.2, the overall geometric texture transfer algorithm will be finally described in Section 3.3.4.

### 3.2.1 Boundary setting

The boundary setting involves both the boundary alignment between the boundary  $b_S$  of the geometric texture  $\mathcal{M}_S$  and the boundary  $b_P$  of the patch  $P$ , and the boundary conditions for the operators in (3.8). The mild assumption formulated for a correct GTT is that  $b_P$  and  $b_S$  are polygonal approximants of counterclockwise oriented, closed, simple curves in  $\mathbb{R}^3$  with the same number of vertices. If this is not the case, a simple refinement pre-processing is applied by adding  $n_{b_S} - n_{b_P}$  new vertices to  $b_P$ , uniformly distributed among its edges. Hence a bijection between  $b_P$  and  $b_S$  is easily established by imposing the correspondence of a couple of adjacent vertices between  $b_S$  and  $b_P$ , either by user interaction - in case the geometric texture must pursue a predetermined orientation - or automatically, via random assignment. This allows to define the starting point between the two polygonals, since they already follow a common orientation.

The common shared boundary is then imposed as Dirichlet boundary conditions in  $V$  to obtain full-rank matrices  $L_w$  in (1.29) and  $N_w$  in (1.33), and a unique solution for (1.39).

Moreover, in order to avoid artifacts due to a different proportional ratio between the geometric texture  $\mathcal{M}_S$  and the target patch  $P$ , we uniformly scale  $\mathcal{M}_S$  by the factor  $|b_P|/|b_S|$ , with  $|b_P|$  and  $|b_S|$  the length of the boundaries.

### 3.2.2 Soft Constraints

The reconstruction process in (3.9), (3.10), and (3.11) is driven by soft constraints on the internal vertices of  $\mathcal{M}_S$  to recover at best the underlying shape of the original patch  $P$ .

At this aim, a conformal parametrization is applied to both the patch  $P \subset \mathcal{M}_I$  and the mesh  $\mathcal{M}_S$  onto a common circular parametric domain  $\Omega$  bounded by  $b_S \equiv b_P$ .

Given a rate  $r \in (0, 100)$ , eventually definable by a potential user, we aim to establish a bijection between  $m = \lfloor n_P \cdot r/100 \rfloor \ll n_V$  vertices of  $P$ , randomly chosen, and  $m$  vertices of  $\mathcal{M}_S$ . Let

$$C = \{(k_i, j_i) : k_i \text{ is vertex index of } P, j_i \text{ is vertex index of } \mathcal{M}_S, i = 1, \dots, m\}$$

be the subset of indices of vertices of  $P$  and associated vertices of  $\mathcal{M}_S$  closest to them in the parametric domain  $\Omega$ . Each  $v_{j_i} \in \mathcal{M}_S$  is associated with a vertical displacement  $\epsilon_{j_i}$ . Then, for each selected vertex  $p_{k_i} \in P$ , with associated normal  $\tilde{n}_{k_i}$ , the position of  $\bar{v}_{j_i}$  is given by

$$\bar{v}_{j_i} = p_{k_i} + \epsilon_{j_i} \tilde{n}_{k_i}. \quad (3.12)$$

The set of the new handles  $\bar{V} = \{\bar{v}_{j_i}\}_{i=1}^m$  defines the soft constraint term  $g(V; \lambda_C)$  which aims to preserve the shape of  $P$  while adding the texture displacement.

The role of the indicator function  $\lambda_C$  in (3.9), (3.10), and (3.11) is played upon discretization by a mask matrix  $\Lambda \in \{0, \lambda\}^{m \times n_S}$  non-zeros only in correspondence to constraints  $C$ , explicitly defined as

$$\Lambda_{ij} = \begin{cases} \lambda & \text{if } j = j_i \\ 0 & \text{otherwise.} \end{cases} \quad (3.13)$$

The displacement energy term thus reads as follows:

$$\lambda_C g(V) = \sum_{i=1}^m \lambda (v_{j_i} - \bar{v}_{j_i})^2 = \|\Lambda(V - \bar{V})\|_2^2, \quad (3.14)$$

which forces the vertices  $v_{j_i} \in \mathcal{M}_S$  towards the positions of the corresponding vertices  $p_{k_i}$  of the patch  $P$ , with a displacement in its normal direction  $\tilde{n}_{k_i}$  of value  $\epsilon_{j_i}$ .

Whenever the geometric texture  $\mathcal{M}_S$  is a 3D mesh that cannot be expressed as a height field, to compute the handle vertices in  $\bar{V}$  in the soft constraints (3.14), formula (3.12) is replaced by the following

$$\bar{v}_{j_i} = p_{k_i} + d_x^{j_i} t_{k_i} + d_y^{j_i} b_{k_i} + d_z^{j_i} \tilde{n}_{k_i} \quad (3.15)$$

where  $d^{j_i} = (d_x^{j_i}, d_y^{j_i}, d_z^{j_i})$  is the local displacement vector on  $\mathcal{M}_S$ , and  $(t_{k_i}, b_{k_i}, \tilde{n}_{k_i})$  is the local frame at  $p_{k_i}$  on the patch  $P$ .

We refer the reader to the example in Fig.3.5 and Tab.3.1 which validates the shape-preserving reconstruction property of the variational models (3.10) and (3.11) when applied to the solution of the GTT task with varying  $r$  factors. For the overall considered examples a very low  $r$  value proved to be sufficient for a good GTT.

### 3.3 Numerical optimization of the variational GTT models

In this section, we illustrate in detail efficient optimization algorithms used to numerically solve the Laplacian Model (3.9), the NCC Model (3.10), and the MVE Model (3.11). Finally in Section 3.3.4 we outline the main steps of the GTT algorithm.

#### 3.3.1 Solution of the Laplacian model (3.9)

The optimization problem (3.9) can be rewritten as a linear least squares problem

$$V^* \in \arg \min_V \mathcal{J}_1(V) = \frac{1}{2} \|A V - B\|_2^2 \quad (3.16)$$

with

$$A := \begin{bmatrix} L_w \\ \Lambda \end{bmatrix} \in \mathbb{R}^{(n_S+m) \times n_S}, \quad B := \begin{bmatrix} \delta_L \\ \Lambda \bar{V} \end{bmatrix} \in \mathbb{R}^{(n_S+m) \times 3}, \quad (3.17)$$

$L_w \in \mathbb{R}^{n_S \times n_S}$  defined in (1.21), and  $\Lambda$  is defined in (3.13). In matrix  $B$ ,  $\delta_L \in \mathbb{R}^{n_S \times 3}$  contains the Laplacian coordinates of the geometric texture  $\mathcal{M}_S$  and  $\bar{V} \in \mathbb{R}^{m \times 3}$  represents the constraints positions set defined in (3.12) or (3.15).

Under the Dirichlet boundary conditions on  $b_S$ , matrix  $A$  is full column rank,  $A^T A$  is non-singular, and thus the unique minimizer of (3.9) can be computed by solving the standard normal equations  $A^T A V = A^T B$ .

The computational cost for the solution of (3.16)-(3.17) corresponds to the solution of three linear systems for each coordinate vector  $V = (V_x, V_y, V_z)$ , with the same coefficient matrix and right-hand-side defined by the corresponding three columns of  $B$ .

#### 3.3.2 Solution of the NCC model (3.10)

The solution of the NCC-based variational problem (3.10) is analogous to the solution for (3.9) by rewriting the linear least squares problem as

$$V^* \in \arg \min_V \mathcal{J}_2(V) = \frac{1}{2} \|A V - B\|_2^2, \quad A = \begin{bmatrix} N_w \\ \Lambda \end{bmatrix}, \quad B = \begin{bmatrix} \delta_N \\ \Lambda \bar{V} \end{bmatrix} \quad (3.18)$$

with  $A \in \mathbb{R}^{(n_S+m) \times n_S}$  and  $B \in \mathbb{R}^{(n_S+m) \times 3}$ ,  $\delta_N$  are the NCC coordinates of the geometric texture  $\mathcal{M}_S$  and  $N_w$  is the sparse NCC matrix defined in (1.32). Under Dirichlet boundary conditions on  $b_S$ , the matrix  $A$  is full column rank, then the unique minimizer is obtained by solving the system of normal equations  $A^T A V = A^T B$ .

Although the use of Laplacian (differential) LAP or NCC descriptors forces local detail preserving, the shape of such details appears, in general, deformed since these descriptors are not invariant with respect to rotation and scaling. At this aim, in [75] the

authors proposed a rotated Laplacian reconstruction approach which relies on the estimate of the local frame rotations matrix  $R$  in order to exploit the equivariance property proved in Prop. 3.1.1:

$$N_w(R(V)) = R(N_w(V)).$$

A simpler but effective idea was proposed in [127] where an iterative approach alternates the solution of the normal equations (3.18) with an update of the NCC coordinates (and, therefore, of the matrix  $B$ ). In particular, starting from  $\delta^{(0)} = \delta_N$ , by recalling that the coordinates NCC are parallel to the normals to the vertices,  $V^{(k)}$  and  $\delta^{(k)}$  are updated through the following iterative scheme:

$$\begin{cases} \text{Solve } A^T A V = A^T B^{(k)} \text{ for } V^{(k+1)}; \\ \delta_i^{(k+1)} = \text{sgn}((\delta_N)_i) \|(\delta_N)_i\| n_i, \text{ with } n_i \text{ normal to the vertex } v_i^{(k+1)}, \quad \forall i. \end{cases} \quad (3.19)$$

This correction to the naive reconstruction method makes the final result less dependent on the original orientation of the source mesh  $\mathcal{M}_S$ .

The computational cost of one iteration of the NCC updating scheme (3.19) consists of the solution of three linear systems, the estimation of the vertex normal and the update of the directions  $\delta^{(k+1)}$ ; all of which depend on the number of vertices  $n_S$ .

### 3.3.3 Solution of the MVE model (3.11)

The vertex set  $V^*$  of the target mesh  $\mathcal{M}_T$ , corresponding to the region of interest  $P \in \mathcal{M}_I$ , is obtained as solution of a non-linear least squares (NLLS) problem. In particular, let  $n = n_S - |b_S|$ , be the cardinality of the sought vertex set  $V^*$ , we define the differentiable nonlinear residual vector function  $r : \mathbb{R}^{3n} \rightarrow \mathbb{R}^{3n}$  as  $r(V) = V - F(V)$ , with components  $r_i(V) \in \mathbb{R}^3$ ,  $r_i(V) = (r_i^{(x)}(V), r_i^{(y)}(V), r_i^{(z)}(V))^T$  the nonlinear residual at vertex  $v_i$ , defined as

$$r_i(V) := v_i - F_i(V), \quad i = 1, \dots, n. \quad (3.20)$$

We want to solve the NLLS minimization problem

$$V^* = \arg \min_{V \in \mathbb{R}^{3n}} \mathcal{J}_3(V) = \frac{1}{2} \|r(V)\|_2^2 + \lambda_C \|V - \bar{V}\|_2^2. \quad (3.21)$$

For the purpose of developing a numerical solution of the NNLS problem, we define the Jacobian of the residual vector function  $r(V)$  as the  $3 \times 3$  block matrix

$$J(V) = \begin{bmatrix} \boxed{\frac{\partial r^{(x)}(V)}{\partial V^{(x)}}} & \boxed{\frac{\partial r^{(x)}(V)}{\partial V^{(y)}}} & \boxed{\frac{\partial r^{(x)}(V)}{\partial V^{(z)}}} \\ \boxed{\frac{\partial r^{(y)}(V)}{\partial V^{(x)}}} & \boxed{\frac{\partial r^{(y)}(V)}{\partial V^{(y)}}} & \boxed{\frac{\partial r^{(y)}(V)}{\partial V^{(z)}}} \\ \boxed{\frac{\partial r^{(z)}(V)}{\partial V^{(x)}}} & \boxed{\frac{\partial r^{(z)}(V)}{\partial V^{(y)}}} & \boxed{\frac{\partial r^{(z)}(V)}{\partial V^{(z)}}} \end{bmatrix} \in \mathbb{R}^{3n \times 3n} \quad (3.22)$$

with blocks  $[J(V)]^{kl} \in \mathbb{R}^{n \times n}$ ,  $k, l = 1, 2, 3$ , and we summarize the main properties of  $J(V)$ .

**Proposition 3.3.1.** *The square matrix  $J(V) \in \mathbb{R}^{3n \times 3n}$  in (3.22) is full rank, positive definite and it is highly sparse having at each row corresponding to  $i$ -th vertex at most  $3|\mathcal{N}(i)| + 1$  non-zero elements. The elements  $(i, j)$ ,  $i, j = 1, \dots, n$  for all blocks  $[J(V)]^{kl}$ ,  $k, l \in \{1, 2, 3\}$ , are defined as*

$$\begin{aligned} [J(V)]_{i,i}^{kl} &= \left[ \frac{\partial r_i(V)}{\partial v_i} \right]^{kl} = I_3, \\ [J(V)]_{i,j}^{kl} &= \left[ \frac{\partial r_i(V)}{\partial v_j} \right]^{kl} = -w_{ij}I_3 + \frac{w_{ij}b_{ij}}{\|z - Q_1v_j\|_2} n_i (Q_1^T(z - Q_1v_j))^T + \\ &\quad - \sum_{k \neq j} \frac{w_{ik}b_{ik}}{\|Q_2v_j + z - N_iv_k\|_2} n_i (Q_2^T(Q_2v_j + z - N_iv_k))^T, \end{aligned} \quad (3.23)$$

where

$$Q_1 = (1 - w_{ij})N_i, \quad Q_2 = w_{ij}N_i, \quad z = N_i \sum_{l \neq j} w_{il}v_l.$$

*Proof.* To derive an explicit expression for  $J(V)$ , we first observe that  $\frac{\partial r_i(V)}{\partial v_j} \in \mathbb{R}^{3 \times 3}$ , and  $\frac{\partial r_i(V)}{\partial v_j} = 0$  for  $j \notin \mathcal{N}(i)$ . Next, we can rewrite  $r_i(V)$  by separating the neighbor  $v_j$  from the other neighbors  $v_k \in \mathcal{N}(v_i)$ , as

$$r_i(V) = v_i - w_{ij}(v_j + \|z - Q_1v_j\|_2 b_{ij}n_i) - \sum_{k \neq j} w_{ik}(v_k + \|Q_2v_j + z - N_iv_k\|_2 b_{ik}n_i), \quad (3.24)$$

Then, by applying in (3.24) the chain rule  $\frac{\partial}{\partial x} \|Ax - b\|_2 = \frac{A^T(Ax - b)}{\|Ax - b\|_2}$  for  $x = v_j$ , we get (3.23).  $\square$

An approximate solution for the minimization problem (3.21) can be obtained by imposing the first-order optimality conditions:

$$J^T(V)r(V) + 2\lambda_C(V - \bar{V}) = 0, \quad (3.25)$$

and then using a traditional optimization method, such as the Newton-Raphson method to solve (3.25). Alternatively, a well-assessed numerical approach for addressing directly the minimization problem (3.21) is the Gauss-Newton method, which does not require the computation of second-order derivatives. Essentially, it is based on the approximation of the Hessian matrix of  $\mathcal{J}_3(V)$  in (3.21) with  $\nabla^2 \mathcal{J}_3(V) \approx J(V)^T J(V) + 2\lambda_C$ , by ignoring all the second order terms from  $\nabla^2 \mathcal{J}_3(V)$ . Gauss-Newton method iterates from an initial

guess  $V^{(0)}$  and performs a line search along the direction  $p_{GN}^{(k)}$  determined by solving the following linear system

$$(J^T(V^{(k)})J(V^{(k)}) + 2\lambda_C)p_{GN} = -J^T(V^{(k)})r(V^{(k)}) - 2\lambda_C(V^{(k)} - \bar{V}), \quad (\text{GN})$$

The coefficient matrix has size  $3n \times 3n$  and is symmetric positive definite. The new vertex positions is then updated as

$$V^{(k+1)} = V^{(k)} + \alpha^{(k)}p_{GN}. \quad (3.26)$$

with an adaptive step-size  $\alpha^{(k)}$  obtained via line search with backtracking, satisfying Armijo condition:

$$\mathcal{J}_3(V^{(k)} + \alpha p_{GN}) - \mathcal{J}_3(V^{(k)}) \leq -2\beta \alpha (p_{GN})^T \nabla \mathcal{J}_3(V^{(k)}) \quad (3.27)$$

with  $\beta \in [0, 1)$  fixed parameter.

Due to the nonlinear nature of this model, an initial guess  $V^{(0)}$  sufficiently close to the sought solution would allow for a fast convergence in a few iterations. Nevertheless, in our experiments, we noticed that the scheme is robust and converges towards the expected solution even for  $V^{(0)}$  being located far away, e.g.  $V^{(0)}$  being  $\mathcal{M}_S$  in its original position. Favourable choices for  $V^{(0)}$  can be either the minimal surface fit for given boundary  $b_P$  or the least squares solution of the GTT-NCC model (3.10), due to the similarity between weights in MVE and in NCC.

Assuming the contribution of derivatives by non-matching coordinates is negligible, i.e.  $[J(V)]^{kl} \approx 0$ , if  $k \neq l$ ,  $k, l = 1, 2, 3$ , we can further simplify  $J(V)$  into a block-diagonal matrix by neglecting the non-diagonal blocks, thus making the Jacobian matrix separable for each spatial coordinate. This gives rise to the solution of three highly sparse linear systems of the form (3.26), eventually parallelizable. The experiments support the above assumption.

### 3.3.4 Algorithm GTT

We synthesize in Algorithm 1 the main steps of the GTT procedure for the three different variational models proposed, which will be named GTT-LAP, GTT-NCC and GTT-MVE according to the respective choice of descriptors.

For height-map geometric texture, the mesh  $\mathcal{M}_S$  is generated from an image by setting the vertices in  $V_S$  to be pixel-centered.

The influence of parameters  $r$  and  $\lambda$  on the GTT results is discussed in the examples on shape preserving ability and parameter estimation, respectively, which suggest satisfactory results can be achieved by selecting  $\lambda = 10$  for GTT-LAP and GTT-NCC and  $\lambda = 1$  for GTT-MVE, while setting the rate  $r = 20$ .



---

**Algorithm 1** Geometric Texture Transferring
 

---

- Input:** · base mesh  $\mathcal{M}_I = (V_I, E_I, T_I)$ , with patch  $P \subset \mathcal{M}_I$ ,  
 · geometric texture mesh  $\mathcal{M}_S = (V_S, E_S, T_S)$ .
- Output:** · textured mesh  $\mathcal{M}_T = (V_T, E_T, T_T)$  with  $V_T = (V_I \setminus P) \cup V^*$ .
- Parameters:** ·  $r \in (0, 100)$  rate of vertices chosen as soft constraints;  
 ·  $\lambda > 0$  soft-constraint parameter;  
 · corresponding vertices in  $b_S$  and  $b_P$ , if manual alignment is required.

**Preliminary set up:**

- **Dirichlet boundary conditions:** refinement of  $b_P$  (if needed), bijection between  $b_S$  and  $b_P$  (manually or automatically);
- **soft constraint term  $g(V)$ :** parametrization of  $P$  and  $\mathcal{M}_S$ , random choice of  $\lfloor n_P \cdot r/100 \rfloor$  vertices of  $P$ , correspondence with vertices of  $\mathcal{M}_S$ , definition of  $\bar{V}$  through (3.15) or (3.12);
- **reconstruction term  $R(V)$ :** computation of the descriptors LAP, NCC or MVE from the source mesh  $\mathcal{M}_S$ , to define three different formulas for  $R(V)$  and three models: (3.9),(3.10),(3.11).

**Variational Models Solution:**

- |                       |                                                                                                                                                                                                                                                                   |
|-----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>GTT-LAP</b> (3.9)  | Construction of matrices $A$ and $B$ in (3.17)<br>solve normal equation $A^T A V = A^T B$ .                                                                                                                                                                       |
| <b>GTT-NCC</b> (3.10) | Construction of matrices $A$ and $B$ in (3.18);<br><i>Until stopping criterion in (3.28) is satisfied</i><br>· solve normal equations $A^T A V = A^T B$<br>· update $\delta_N$ and $B$ as in (3.19);                                                              |
| <b>GTT-MVE</b> (3.11) | Set an initial guess $V^{(0)} = V_S$<br><i>Until stopping criterion in (3.28) is satisfied</i><br>· compute $p_{GN}$ from (GN);<br>· compute step-size $\alpha^{(k)}$ via line-search as in (3.27)<br>(or use a constant $\alpha$ );<br>· update $V$ as in (3.26) |
-

We terminate the iterations of the GTT algorithm, in case iterative procedures are used for GTT-NCC and GTT-MVE, as soon as either of the two following stopping criteria is satisfied

$$\frac{\|V^{(k)} - V^{(k-1)}\|_2}{\|V^{(k)}\|_2} < 10^{-4}, \quad \frac{\|\mathcal{J}(V^{(k)}) - \mathcal{J}(V^{(k-1)})\|_2}{\|\mathcal{J}(V^{(k)})\|_2} < 10^{-4}. \quad (3.28)$$

### 3.4 Numerical GTT examples

In this section, we illustrate examples of GTT that validate the overall procedure sketched in Algorithm 1 and we compare the performance of the GTT-LAP, GTT-NCC and GTT-MVE variational models. The naive implementation of the algorithms has been written in MatLab R2021a and executed on a laptop with a 2.10 GHz AMD Ryzen 5 quad-core processor and 16 GB 2.4 MHz RAM.

GTT-NCC $r\%$	$d_H(\mathcal{M}_I, \mathcal{M}_T)$		
	cylinder+flat	sphere+flat	free-form+flat
1	$5.83 \times 10^{-3}$	$1.59 \times 10^{-2}$	$1.03 \times 10^{-1}$
5	$1.42 \times 10^{-3}$	$4.67 \times 10^{-3}$	$4.36 \times 10^{-2}$
20	$6.45 \times 10^{-4}$	$2.09 \times 10^{-3}$	$2.14 \times 10^{-2}$
50	$3.87 \times 10^{-4}$	$1.25 \times 10^{-3}$	$8.25 \times 10^{-3}$
80	$3.23 \times 10^{-4}$	$1.06 \times 10^{-3}$	$5.40 \times 10^{-3}$
95	$2.93 \times 10^{-4}$	$1.01 \times 10^{-3}$	$4.59 \times 10^{-3}$
99	$2.93 \times 10^{-4}$	$1.01 \times 10^{-3}$	$4.47 \times 10^{-3}$
GTT-MVE $r\%$	$d_H(\mathcal{M}_I, \mathcal{M}_T)$		
	cylinder+flat	sphere+flat	free-form+flat
1	$5.89 \times 10^{-3}$	$1.59 \times 10^{-2}$	$9.73 \times 10^{-2}$
5	$1.49 \times 10^{-3}$	$4.67 \times 10^{-3}$	$2.46 \times 10^{-2}$
20	$6.27 \times 10^{-4}$	$2.09 \times 10^{-3}$	$1.55 \times 10^{-2}$
50	$2.81 \times 10^{-4}$	$1.25 \times 10^{-3}$	$3.28 \times 10^{-3}$
80	$2.43 \times 10^{-4}$	$1.07 \times 10^{-3}$	$2.64 \times 10^{-3}$
95	$2.05 \times 10^{-4}$	$1.07 \times 10^{-3}$	$2.02 \times 10^{-3}$
99	$1.99 \times 10^{-4}$	$1.07 \times 10^{-3}$	$2.02 \times 10^{-3}$

Table 3.1: Example 1: Hausdorff distances to ground truth patch  $P$  using planar geometric texture  $\mathcal{M}_S$  defined on a uniform grid of dimensions  $150 \times 150$  vertices.

#### Shape preserving GTT.

To validate the shape-preserving property of the proposed variational models, we investigate qualitatively and quantitatively the effect of transferring a flat geometric

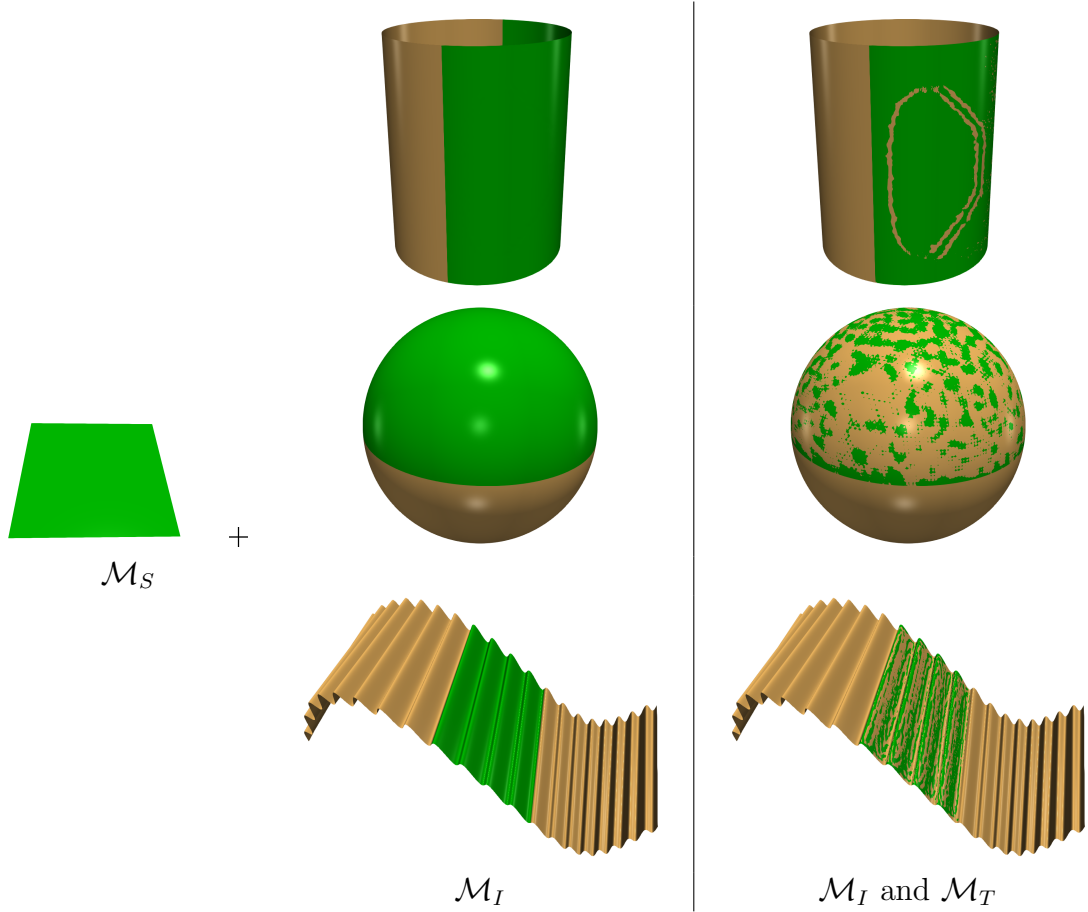


Figure 3.5: Example 1: shape preservation via GTT, using flat geometric texture  $\mathcal{M}_S$  for patch  $P \subset \mathcal{M}_I$  in green (middle column), with rate 20%. The resulting  $\mathcal{M}_T$  overlapped with base mesh  $\mathcal{M}_I$  (right column).

texture  $\mathcal{M}_S$ . The expected result is a mesh  $\mathcal{M}_T$  with the exact same shape as the base mesh  $\mathcal{M}_I$ , but with a different resolution in the patch region  $P$ , inherited from the geometric texture  $\mathcal{M}_S$ . From a qualitative point of view in Fig. 3.5 we show the results for three different meshes  $\mathcal{M}_I$ , and a set  $C$  of soft constraints  $g(V)$  obtained with a rate  $r = 20\%$  of vertices.

In Table 3.1 we reported the Hausdorff distances  $d_H(\mathcal{M}_I, \mathcal{M}_T)$ , defined in (3.7), for different values of the rate  $r$  of randomly chosen vertices in  $C$ . Although for sphere and cylinder shapes the results for GTT-NCC and GTT-MVE are quite similar, the measured distances for GTT-NCC double in the case of free-form surface  $\mathcal{M}_I$  as the one in Fig. 3.5, third row. As expected, in general, when the percentage  $r$  increases also the corresponding accuracy increases, or better, the Hausdorff distance decreases so that the accuracy of the models is reasonably high for high rates. However, the shape of the

patch  $P$  is well preserved even for low rates  $r$  of vertices in the soft constraints term  $g(V)$ .

#### **Influence of the $\lambda$ parameter.**

The second example analyses how the value of the soft constraint parameter  $\lambda$  in (3.8), where  $\lambda_C = \lambda 1_C$ , affects the final GTT results obtained from the three variational models. The rate  $r$  is fixed to 20% in accordance with the results shown in Example 1. At this aim an example of a geometric texture  $\mathcal{M}_S$  transferred to a dolphin mesh  $\mathcal{M}_I$  is illustrated in Fig. 3.6, first row. In the second and fourth rows (top of panels), the cross-section curves from GTT-LAP (in green, with  $\diamond$ ), GTT-NCC (in blue,  $\triangle$ ) and GTT-MVE (in red,  $\star$ ) results are shown for increasing values of  $\lambda$ , from left to right. Finally, in Fig. 3.6, third and fifth row (bottom of panels), we reported the corresponding plots of the terms  $R(V)$  and  $\lambda_C g(V)$  in (3.8), for the two iterative algorithms GTT-NCC and GTT-MVE in red and blue color, respectively.

For small  $\lambda$  values both GTT-LAP (3.9) and GTT-NCC (3.10) models perform poorly in reproducing the shape of the underlying patch  $P$ . This behavior, in particular for GTT-NCC, is confirmed by the blue-colored plots of its related terms  $R(V)$  and  $\lambda_C g(V)$ :  $R(V)$  value indicates perfect reconstruction, nevertheless the shape-preserving term  $\lambda_C g(V)$  remains at high values. As  $\lambda$  value increases, the results of GTT-LAP and GTT-NCC, illustrated by the green and blue cross-sections, respectively, in Fig. 3.6, improve.

However, only at a high value  $\lambda = 10^3$ , the GTT-NCC model (3.10) is forced to follow the shape of  $P$  as  $g(V)$  attains lower value with respect to  $R(V)$ . Nevertheless, under close inspection of the cross-sections, the high penalization to satisfy soft constraints restrains the reconstruction term, which results in a reconstruction that does not reproduce the original oscillations, as instead in the case of GTT-MVE reconstruction.

For what concerns the GTT-MVE (3.11) model, it proved to be extremely robust to variations of  $\lambda$  value, providing good reconstructions even for low  $\lambda$  values. This is qualitatively shown by the red cross-section in Fig. 3.6, and quantitatively by the convergence plots in the third row of Fig. 3.6 where the term  $\lambda_C g(V)$  attains always lower values with respect to the reconstruction term  $R(V)$ .

#### **Comparison with parametric displacement mapping**

It is quite easy to qualitatively assess whether the result of a GTT meets our expectations.

However, it is not entirely obvious what should be an ideal result on a free-form base surface. This may depend on subjective evaluations or on the context of the application. For the quantitative evaluation of the GTT results obtained by the three variational models, we considered three parametric surfaces as base meshes, in order to be able to perform an exact GTT, in terms of the standard displacement mapping formula (1.47), named GTT-DM in the following. This allowed us to evaluate a mea-

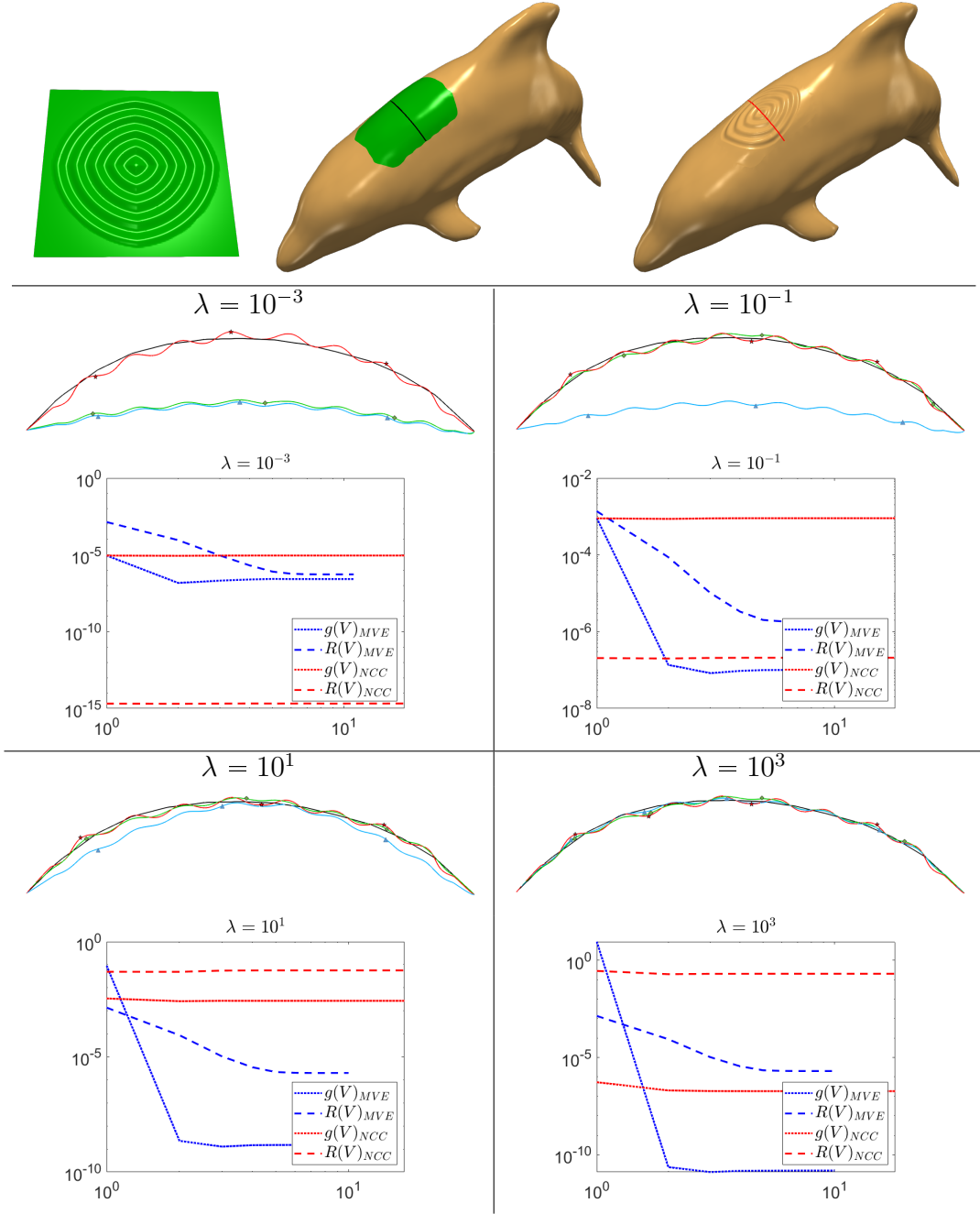


Figure 3.6: Example 2: influence of the  $\lambda$  parameter. First row: a GTT example with  $\mathcal{M}_S$ ,  $\mathcal{M}_I$  and  $\mathcal{M}_T$ . Second and third rows: cross-sections of the  $P$  patch (in black), of GTT-LAP (in green, marked  $\diamond$ ), of GTT-NCC (in blue, marked  $\triangle$ ), and of GTT-MVE (in red, marked  $\star$ ) and plots of the reconstruction term  $R(V)$  and the constraint term  $\lambda_C g(V)$  (related to GTT-NCC and GTT-MVE), for increasing values of  $\lambda$ .

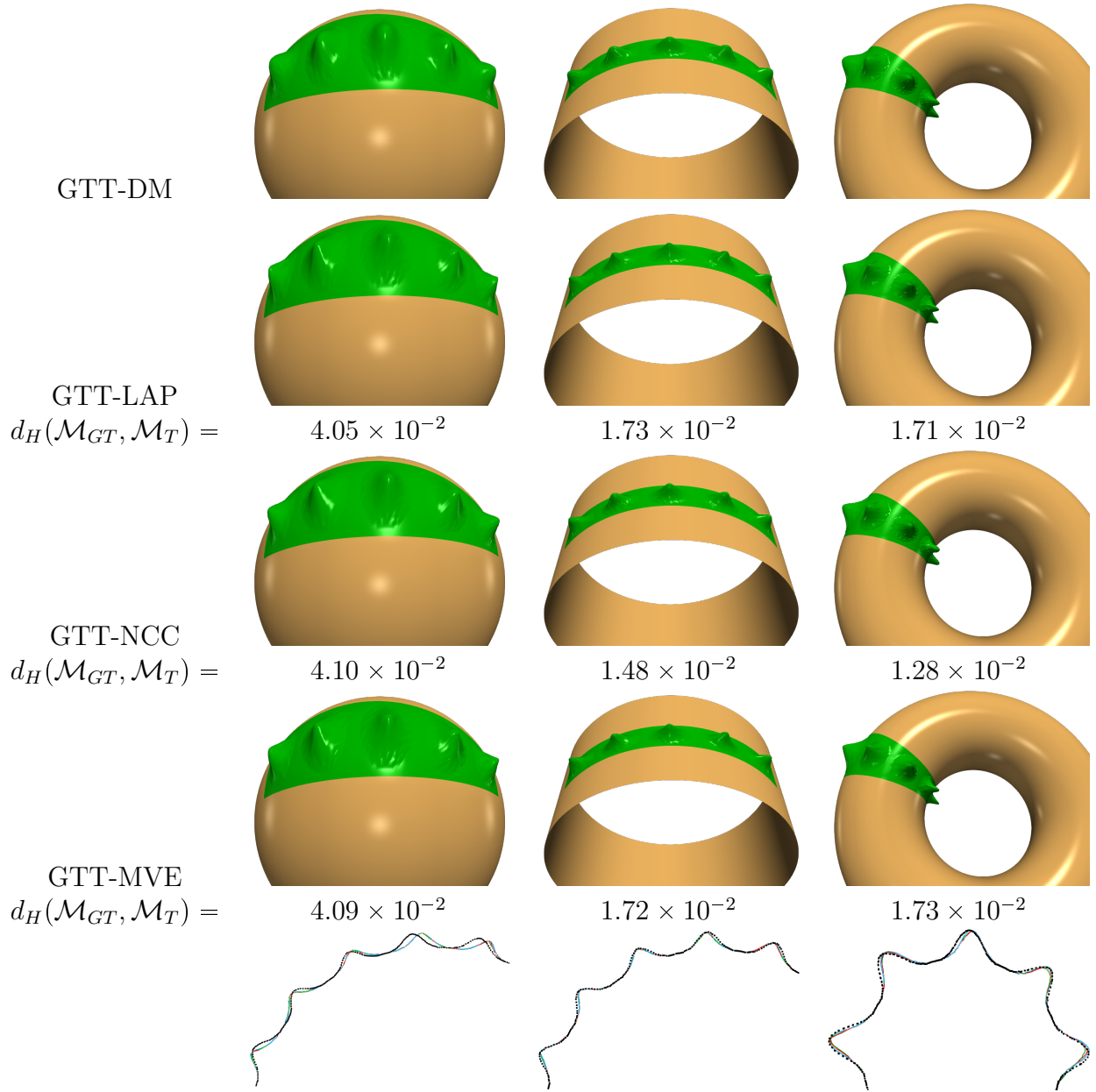


Figure 3.7: Example 3: Comparison with displacement mapping (DM) on parametric surfaces (sphere, cylinder, torus). In the last row, the corresponding slices: DM in dotted black, LAP in green, NCC in blue, MVE in red.

sure of discrepancy between the exact result  $\mathcal{M}_{DT}$  and  $\mathcal{M}_T$  by estimating the Hausdorff distance  $d_H(\mathcal{M}_{DT}, \mathcal{M}_T)$ .

Figure 3.7 illustrates the results obtained via displacement mapping (GTT-DM) and with the three models GTT-LAP (3.9), GTT-NCC (3.10), GTT-MVE (3.11) for the base surfaces  $\mathcal{M}_I$  sphere, cylinder and torus. We notice that the displacement mapping approach requires a bijective correspondence for all the parametric values of the height map that represents the geometric texture. In contrast, the proposed GTT models return the mesostructure  $\mathcal{M}_T$  computing only a relatively small number of corresponding points.

Resulting Hausdorff values  $d_H(\mathcal{M}_{DT}, \mathcal{M}_T)$ , reported below each result in Fig.3.7, confirm an overall good accuracy of the GTT-models GTT-LAP (3.9), GTT-NCC (3.10), GTT-MVE (3.11). However, in general, an exact GTT-DM cannot be applied to non-parametric surfaces, thus making it impossible to evaluate the GTT results quantitatively for arbitrary meshes.

### GTT performance

We finally evaluate GTT performance when applied to objects and geometric textures with several different shapes to qualitatively evaluate the adaptability of the GTT algorithm to different geometric features and details.

We first show in Fig. 3.9 the comparison results obtained by the GTT algorithms GTT-LAP, GTT-NCC and GTT-MVE which solve the variational problems in (3.9), (3.10) and (3.11), respectively. It is quite evident as the GTT-LAP - see the third column in Fig. 3.9 - produces the worst quality geometric transfer results for any kind of surface and geometric texture. The non-linear GTT-MVE optimization model (3.11) provides the best local detail-preservation and it avoids possible self-intersections, see the fifth column of Fig. 3.9. On the other hand, the GTT-NCC gives overall good results, but it is more prone to producing artifacts, see the teddy bear face and skull textures, in the third and the fourth row of Fig. 3.9, respectively. In the sixth row of Fig. 3.9, the GTT results have been produced using a 3D geometric texture  $\mathcal{M}_S$ , not representable by a height map. In this case, the soft constraints  $g(V)$  have been defined following (3.15). Even in this case, the GTT-MVE results preserve well the individual texture shape, while GTT-NCC fails to recover a few of the “petal” shapes located close as well as far left from the reader’s point of view.

Additional results are reported in Fig. 3.10 which illustrates the behavior of the GTT-MVE algorithm applied to various patches  $P \subset \mathcal{M}_I$ , for various geometric textures  $\mathcal{M}_S$ . Each GTT result is accompanied by the average execution time in seconds and the number of iterations of the Gauss-Newton method to emphasize its fast convergence for good-quality results. We observe column-wise in Fig. 3.10 that the efficiency depends strongly on the  $\mathcal{M}_S$  resolution, reported below each geometric texture, and on the level of detail contained in  $\mathcal{M}_S$ , while the shape of the path  $P$ , row-wise in Fig. 3.10, has much less influence on the execution time.

Certainly, there is room for improvement in efficiency by adopting code optimization

strategies, rather than the naive MATLAB implementation used.

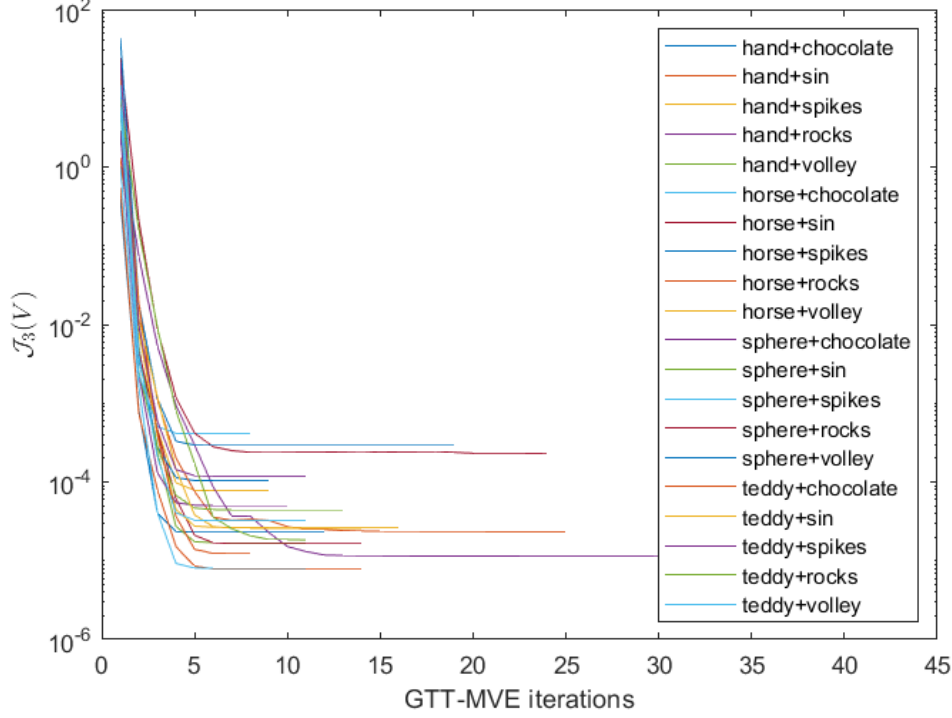


Figure 3.8: Numerical convergence of  $\mathcal{J}_3(V)$ , related to the MVE model (3.11), when minimized by the Gauss-Newton algorithm for the solution of some examples in Fig. 3.10.

We further investigated the empirical convergence of the Gauss-Newton iterative method for the GTT-MVE model (3.11). To that aim, we run the optimization algorithm with initial guess  $V^{(0)}$  set to be the original position of the texture mesh  $\mathcal{M}_S$  and we stopped the algorithm as soon as one of the stopping criterion in (3.28) is satisfied under the tolerance  $10^{-8}$ . We can observe the fast decreasing of the energy function  $\mathcal{J}_3(V)$  in Fig. 3.8 for every GTT example shown in Fig. 3.10.



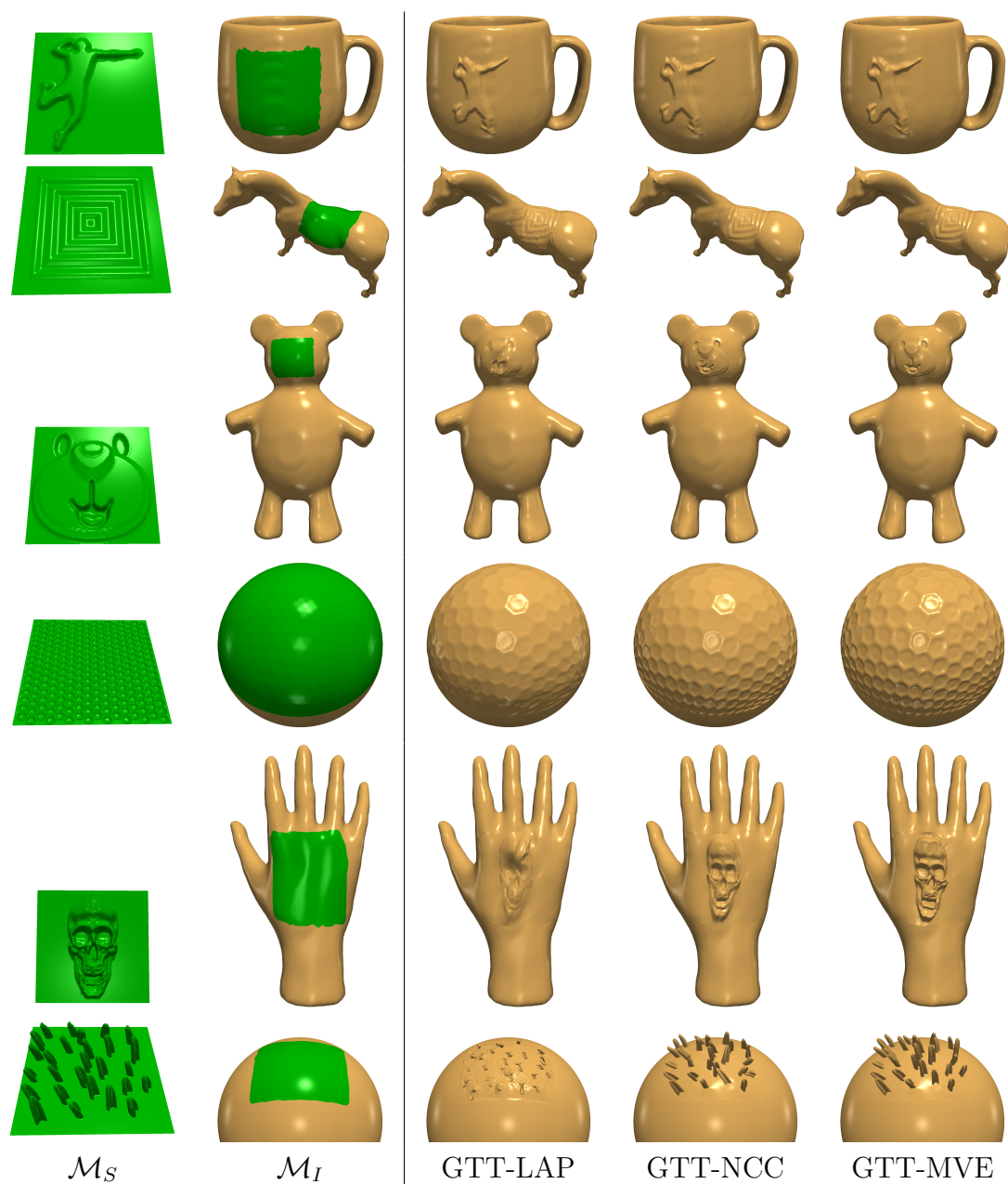


Figure 3.9: Example 4: comparison among GTT-LAP (3.9), GTT-NCC (3.10) and GTT-MVE (3.11) algorithms applied to different meshes and different geometric textures.

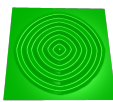


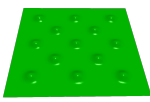



















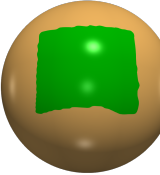
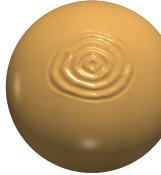
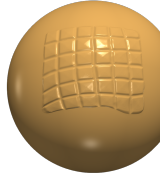
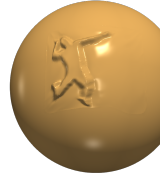
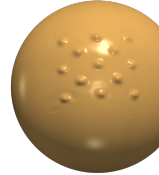
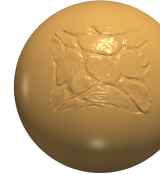




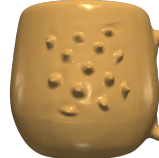
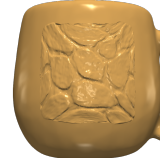
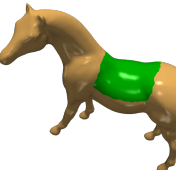





					
	140x140	150x150	106x116	140x140	150x150
	 (18.28, 5)	 (16.63, 4)	 (14.71, 7)	 (6.68, 6)	 (31.52, 8)
	 (18.01, 4)	 (15.89, 4)	 (10.75, 5)	 (5.53, 5)	 (24.73, 6)
	 (24.68, 5)	 (23.65, 6)	 (14.69, 7)	 (5.55, 5)	 (20.37, 5)
	 (17.94, 4)	 (15.80, 4)	 (10.53, 5)	 (4.51, 4)	 (30.29, 7)
	 (17.8, 4)	 (19.78, 5)	 (12.65, 6)	 (5.53, 5)	 (52.74, 13)
	 (18.49, 4)	 (16.21, 4)	 (18.26, 6)	 (5.57, 5)	 (39.35, 9)
Average	(19.2, 4)	(17.99, 4)	(13.59, 5)	(5.56, 5)	(33.16, 8)

Figure 3.10: Example 4: GTT-MVE results for meshes  $\mathcal{M}_I$  in rows, geometric textures  $\mathcal{M}_S$  in columns. In parenthesis, the average execution time in seconds and the number of iterations for each texture.



# Chapter 4

## Variational eigendecomposition of the graph $p$ -Laplacian

In this Chapter we address a variational approach to the  $p$ -Laplacian eigendecomposition, a generalization of the classical Laplacian spectral analysis. An application to spectral mesh segmentation is finally presented.

In the continuous setting (see [83]), the value of  $p$  influences the regularity of the solutions. In fact, as  $p \rightarrow \infty$  eigenfunctions have wider support and become smoother, tending to piecewise linear functions. On the other hand, for low values  $p \in (1, 2)$  the term  $|\nabla f|^{p-2}$  becomes singular as  $|\nabla f| \rightarrow 0$ , leading to singularities or discontinuities and producing sharper eigenfunctions, with concentrated support.

This behavior appears also in variational regularization models, where the  $L^p$  norm of the penalty term favors sparse solutions (for small  $p$ , as in Total Variation [31]) or smoother solutions without outliers (for large  $p$ ).

In this Chapter, the main goal is to investigate the discrete counterpart of the non-linear  $p$ -eigendecomposition problem and propose numerical approximations of the  $p$ -eigenpairs via a variational approach to this problem on graphs and meshes.

In many real-world problems where the data in practice is discrete, graphs constitute a natural structure suited to their representation. Each vertex of the graph corresponds to a datum, and the edges encode the pairwise relationships or similarities among the data. For the case of unorganized data such as point clouds, a graph can also be built by modeling neighborhood relationships between the data elements. Structured/unstructured meshes, which naturally approximate 2-manifold embedded in  $\mathbb{R}^3$ , have a graph representation where the edges represent geometric connections and nodes correspond to function evaluations on the mesh vertices.

Using this framework, the  $p$ -eigendecomposition problem is directly expressed and solved in a discrete setting. The problem has been faced in [35, 36], observing that eigenfunctions are steady-states of the  $p$ -flow which is the gradient flow with respect to the  $p$ -Dirichlet energy. However, their solutions apply to a simplified nonlinear  $p$ -Laplacian eigenvalue problem. Gradient flow extinction is also computed in [23] and

applied for  $p = 1$  to spectral clustering. To calculate the first eigenpair of the  $p$ -Laplacian operator, the authors in [14] propose an inverse power algorithm, whereas in [73] an adaptive finite element method is presented. A modified local minimax algorithm is proposed in [135] to find the critical points of a weighted  $p$ -Rayleigh quotient.

On the other hand, in [81], the authors build a sequence of variational models similar to (1.55), exploiting a non-linear generalization of the Rayleigh quotient, but keeping the linear orthogonality constraint.

In our proposal, we formulate the problem of estimating multiple eigenvectors of the graph  $p$ -Laplacian as a non-convex constrained optimization problem with cost function the  $p$ -Rayleigh quotient of the graph  $p$ -Laplacian, and constraints that force the  $p$ -orthogonality of the eigenvectors. The request for  $p$ -orthogonality renders the optimization problem extremely challenging.

To the best of our knowledge, there are no other numerical results in literature for the computation of more than two eigenpairs of the  $p$ -Laplacian which satisfy the original nonlinear eigendecomposition problem, i.e. the  $p$ -orthogonality constraint on the eigenfunctions.

In the following, we propose two different optimization algorithms to solve the  $p$ -eigendecomposition variational problem, while preserving the  $p$ -orthogonality constraints.

The key idea of our proposal is a preliminary reformulation of the challenging original variational problem into a simpler equivalent one by a suitable  $p$ -dependent change of variable which, for any  $p$ , transforms the  $p$ -orthogonality constraint into a simple linear constraint. We propose two numerical approaches for the solution of the reformulated problem which incrementally estimate a new eigenfunction,  $p$ -orthogonal to an already computed set of eigenfunctions. In particular, we present a simple projected gradient method on linear constraints as well as a second approach which, by leveraging scale invariance of the  $p$ -Rayleigh quotient, relies on an Alternating Direction Method of Multipliers (ADMM)-based algorithm with simple manifold constraints. Moreover, for each eigenfunction computed by one of the two algorithms, we propose a practical numerical estimate of the associated eigenvalue, which relies on the definition of the graph  $p$ -Rayleigh quotient and is based on the orthogonal least square fitting.

The rest of this chapter is organized as follows. In Section 4.1 we start by recalling some basic notations and some important preliminary notions on  $p$ -Laplacian eigendecomposition and the discretization of the  $p$ -Laplacian problem on graphs and meshes. Section 4.2 introduces the incremental optimization model with  $p$ -orthogonality constraints and Section 4.3 introduces the reformulation of the problem and details the two proposed algorithmic frameworks. In particular, Section 4.3.2 is devoted to the projected gradient descent approach, while in Section 4.3.3 we propose an ADMM-based optimization method on manifold. Finally, Section 4.4 illustrates with some experimental results the performance of the two algorithms.

## 4.1 Notations and Preliminaries

First, in Section 4.1.1 we briefly recall some essential definitions and results from the theory of nonlinear  $p$ -Laplacian eigendecomposition in the continuous setting, then in Section 4.1.2 we introduce the discrete counterpart on graphs and meshes. In the paper, we denote by  $\mathbb{R}_+$  and  $\mathbb{R}_+^*$  the sets of non-negative and positive real numbers, respectively, by  $0_n$  and  $1_n$  the  $n$ -dimensional vectors of all zeros and all ones, respectively.

### 4.1.1 The continuous setting: $p$ -Laplacian eigenproblem

The  $p$ -Laplacian operator  $\Delta_p$ , introduced in (1.15), with  $p \in (1, +\infty)$ , is a nonlinear generalization of the linear Laplace operator  $\Delta_2$  and is defined for smooth functions  $f$  on a bounded domain  $\Omega \subset \mathbb{R}^d$ ,  $d \in \mathbb{N}$ , as

$$\Delta_p f := \operatorname{div} ( |\nabla f|^{p-2} \nabla f ) , \quad (4.1)$$

**Definition 4.1.1 ( $p$ -Laplacian eigenproblem).** *For any  $p \in (1, +\infty)$ , a function  $f : \Omega \rightarrow \mathbb{R}$  is said to be a  $p$ -eigenfunction of the  $p$ -Laplacian operator  $\Delta_p$  defined in (4.1)-(1.16) if and only if there exists a real number  $\lambda \in \mathbb{R}$ , called the associated  $p$ -eigenvalue, such that*

$$(-\Delta_p f)(x) = \lambda |f(x)|^{p-2} f(x), \quad \forall x \in \Omega. \quad (4.2)$$

**Remark 2.** *We use the convention of studying the eigenpair of the negative  $p$ -Laplacian  $-\Delta_p$  in order to consider non-negative eigenvalues.*

Eq. (4.2) formalizes the continuous nonlinear  $p$ -Laplacian eigendecomposition problem, with  $p \in (1, +\infty)$ , and should be complemented with suitable boundary conditions, yielding different eigenpairs. A classical assumption is that  $\Omega \subset \mathbb{R}^d$  is a bounded domain and that boundary conditions of Dirichlet or Neumann type are imposed on the boundary  $\partial\Omega$  of  $\Omega$  - see, e.g., [83] for a detailed review of the most popular boundary conditions.

We remark that, according to definition (4.2), if  $f$  is a  $p$ -eigenfunction with associated eigenvalue  $\lambda$ , then every rescaling  $cf$ , with  $c \in \mathbb{R} \setminus \{0\}$ , is also a  $p$ -eigenfunction with the same associated eigenvalue. In fact, it follows from (1.17) and (4.2) that

$$(-\Delta_p(cf))(x) = \lambda |cf(x)|^{p-2} cf(x) \iff -|c|^{p-2} c \Delta_p(f) = \lambda |c|^{p-2} c |f(x)|^{p-2} f(x) \quad \forall x \in \Omega.$$

It is well-known [51] that the  $p$ -Laplace equation  $\Delta_p f \equiv 0$  is the Euler-Lagrange equation for the so-called  $p$ -Dirichlet energy functional  $I_p[f]$ , defined in terms of the  $p$ -norm of the gradient field  $\nabla f$  as

$$I_p[f] := \frac{1}{p} \|\nabla f\|_p^p = \frac{1}{p} \int_{\Omega} |\nabla f|^p d\Omega. \quad (4.3)$$

After defining the functional  $N_p[f]$  in terms of the  $p$ -norm of  $f$  as

$$N_p[f] := \frac{1}{p} \|f\|_p^p = \frac{1}{p} \int_{\Omega} |f|^p d\Omega, \quad (4.4)$$

it is also easy to demonstrate that

$$\partial_f I_p[f] = \lambda \partial_f N_p[f] \iff (\lambda, f) \text{ is a } p\text{-eigenpair}. \quad (4.5)$$

Generalizing the notion of Rayleigh quotient for the Laplace operator, the  $p$ -Rayleigh quotient for the  $p$ -Laplacian operator is naturally defined by

$$R_p[f] := \frac{I_p[f]}{N_p[f]} = \frac{\|\nabla f\|_p^p}{\|f\|_p^p}. \quad (4.6)$$

It follows easily from (4.5) and (4.6) the important property that

$$\partial_f R_p[f] = 0 \iff \partial_f I_p[f] = \frac{I_p[f]}{N_p[f]} \partial_f N_p[f] \implies (\lambda = R_p[f], f) \text{ is a } p\text{-eigenpair}, \quad (4.7)$$

that is, all stationary points  $f^*$  of the  $p$ -Rayleigh quotient functional  $R_p[f]$  in (4.6) are  $p$ -eigenfunctions with associated  $p$ -eigenvalue  $\lambda$  equal to  $R_p[f^*]$ .

Since  $R_p[f]$  in (4.6) is clearly scale-invariant - in fact,  $R_p[cf] = R_p[f]$  for any  $c \in \mathbb{R} \setminus \{0\}$  - if  $f^*$  is a stationary point of  $R_p[f]$  and, hence, a  $p$ -eigenfunction with associated  $\lambda = R_p[f^*]$ , then any scaled function  $cf^*$  is also a  $p$ -eigenfunction with the same associated  $\lambda$ . The  $p$ -eigenpairs can thus be equivalently sought among the stationary points of the numerator  $I_p[f]$  of the  $p$ -Rayleigh quotient  $R_p[f]$  in (4.6) under the constraint

$$f \in \mathcal{S}_p := \{ f : \|f\|_p^p = 1 \}.$$

It follows that the sufficient condition in (4.7) for  $(\lambda, f)$  being a  $p$ -eigenpair can be equivalently written as

$$(\partial_f I_p[f] = 0) \wedge (f \in \mathcal{S}_p) \implies (\lambda = I_p[f], f) \text{ is a } p\text{-eigenpair}.$$

In the following section, we extend the above theory, defined for Euclidean domains  $\Omega$ , to discrete domains such as graphs and manifold meshes.

### 4.1.2 The discrete setting: graph $p$ -Laplacian eigenproblem

In this section, we define the graph  $p$ -Laplacian eigenproblem, which represents the discrete counterpart of the continuous  $p$ -Laplacian eigenproblem (4.2) on graphs.

A graph  $G$  can be thought as a generalization of a mesh  $\mathcal{M}$ , defined in (1.4), where the notion of faces has been lost, while edges  $e_{ij}$  are now equipped with positive weights

$w_{ij} \in \mathbb{R}_+^*$ . In formulas, a weighted undirected graph is denoted as  $G = (V, E, W)$ , with  $V$  set of vertices,  $E$  set of edges with corresponding weights  $W$ .

In this chapter, we denote the cardinalities of the three sets as given by  $n := |V|$  and  $m := |E| = |W| \leq n^2$ .

Functions acting on a graph  $G$  are defined as in (1.18), denoting as  $\mathcal{F}_V$  and  $\mathcal{F}_E$  the sets of scalar real-valued functions  $f$  and  $g$  with domains the discrete sets of vertices and edges, respectively, with the corresponding standard scalar products.

In the following Defs. 4.1.2-4.1.4 we recall the standard definitions of graph gradient, graph divergence and graph  $p$ -Laplacian operators. In particular, we note that the graph  $p$ -Laplacian expression introduced in Def. 4.1.4 naturally follows from the definition (4.1) of the  $p$ -Laplacian operator in the continuous setting.

**Definition 4.1.2 (graph gradient).** *The graph gradient is the operator  $\nabla : \mathcal{F}_V \rightarrow \mathcal{F}_E$  defined component-wise by*

$$(\nabla f)_{i,j} := \sqrt{w_{i,j}} (f_j - f_i), \quad (i, j) \in E. \quad (4.8)$$

**Definition 4.1.3 (graph divergence).** *The graph divergence is the operator  $\text{div} : \mathcal{F}_E \rightarrow \mathcal{F}_V$  which satisfies  $\langle \nabla f, g \rangle_{\mathcal{F}_E} = \langle f, \text{div} g \rangle_{\mathcal{F}_V}$  for all  $f \in \mathcal{F}_V$ ,  $g \in \mathcal{F}_E$ , defined component-wise by*

$$(\text{div} g)_i := \frac{1}{2} \sum_{j \in \mathcal{N}(i)} \sqrt{w_{i,j}} (g_{i,j} - g_{j,i}), \quad i \in 1, \dots, n, \quad (4.9)$$

From the latest expression, we can easily observe that the divergence of a symmetric function  $g$  (i.e.  $g_{i,j} = g_{j,i}$  for all  $i, j$ ) is zero.

Combining graph gradient and divergence expressions, we derive the definition of graph  $p$ -Laplacian operator.

**Definition 4.1.4 (graph  $p$ -Laplacian).** *The graph  $p$ -Laplacian is the operator  $\Delta_p : \mathcal{F}_V \rightarrow \mathcal{F}_V$  defined by*

$$\Delta_p f := \text{div} (\|\nabla f\|^{p-2} \nabla f), \quad (4.10)$$

which, in component-wise form, reads

$$(\Delta_p f)_i = \sum_{j \in \mathcal{N}(i)} w_{i,j}^{p/2} \psi_p(f_j - f_i), \quad i \in 1, \dots, n, \quad (4.11)$$

with function  $\psi_p : \mathbb{R} \rightarrow \mathbb{R}$  defined by

$$\psi_p(x) := \frac{1}{p} (|x|^p)' := |x|^{p-1} \text{sign}(x) = |x|^{p-2} x. \quad (4.12)$$



We note that (4.11) follows easily by replacing in (4.10) the expressions of the discrete operators  $\nabla$  and  $\text{div}$  given in (4.8) and (4.9), respectively, for a generic vertex  $i = 1, \dots, n$ :

$$(\Delta_p f)_i = \sum_{j \in N(i)} \sqrt{w_{i,j}} |\sqrt{w_{i,j}} (f_j - f_i)|^{p-2} \sqrt{w_{i,j}} (f_j - f_i), \quad i = 1, \dots, n.$$

Analogously to the continuous case, since  $\psi_2$  is just the identity operator, the graph  $p$ -Laplacian in (4.10), (4.11) reduces to the standard linear graph Laplacian for  $p = 2$ , whereas for  $p \neq 2$  the operator  $\Delta_p$  is nonlinear, as we have

$$\Delta_p(cf) = |c|^{p-2} c \Delta_p f.$$

The  $p$ -Laplacian eigenproblem formalized in the continuous setting in Def. 4.1.1 has a natural counterpart in the discrete graph context, expressed in the following Def. 4.1.5.

**Definition 4.1.5 (graph  $p$ -Laplacian eigenproblem).** *For any  $p \in (1, +\infty)$ , a function  $f \in \mathcal{F}_V$  is said to be a  $p$ -eigenvector of the graph  $p$ -Laplacian operator  $\Delta_p$  defined in (4.10) if and only if there exists a real number  $\lambda \in \mathbb{R}$ , called the associated  $p$ -eigenvalue, such that*

$$(-\Delta_p f)_i = \lambda \psi_p(f_i), \quad \forall i = 1, \dots, n, \quad (4.13)$$

with function  $\psi_p$  defined in (4.12). Any pair  $(\lambda, f) \in \mathbb{R} \times \mathbb{R}^n$  satisfying (4.13) is called a  $p$ -eigenpair of the operator  $\Delta_p$ .

Eq. (4.13) formalizes the discrete nonlinear graph  $p$ -Laplacian eigendecomposition problem, with  $p \in (1, +\infty)$ . In this discrete setting, we assume Dirichlet boundary conditions for graphs/meshes with boundary, whereas no boundary conditions are required for graphs/meshes without boundary.

For graphs and meshes, the multiplicity of the first eigenvalue  $\lambda^{(1)} = 0$  of the  $p$ -Laplacian is equal to the number of connected components of the graph [40]. Moreover, any eigenvector  $f \in \mathbb{R}^n$  associated with a non-zero eigenvalue, called non-zero eigenvector, satisfies the following property [22]:

$$\sum_{i=1, \dots, n} |f_i|^{p-2} f_i = \sum_{i=1, \dots, n} \psi_p(f_i) = 0.$$

Analogously to the definitions (4.3), (4.4), (4.6) of functionals  $I_p[f]$ ,  $N_p[f]$ ,  $R_p[f]$  introduced in Section 4.1.1 for the continuous setting, we introduce here the discrete counterparts on graphs, namely the graph  $p$ -Dirichlet energy function  $I_p(f)$ , the function

$N_p(f)$  and the graph  $p$ -Rayleigh quotient function  $R_p(f)$ :

$$I_p(f) := \frac{1}{p} \|\nabla f\|_p^p = \frac{1}{p} \sum_{(i,j) \in E} \frac{1}{2} |\sqrt{w_{i,j}} (f_j - f_i)|^p = \frac{1}{2p} \sum_{(i,j) \in E} w_{i,j}^{p/2} |f_j - f_i|^p, \quad (4.14)$$

$$N_p(f) := \frac{1}{p} \|f\|_p^p = \frac{1}{p} \sum_{i=1}^n |f_i|^p, \quad (4.15)$$

$$R_p(f) := \frac{I_p(f)}{N_p(f)} = \frac{1}{2} \frac{\sum_{(i,j) \in E} w_{i,j}^{p/2} |f_j - f_i|^p}{\sum_{i=1}^n |f_i|^p}. \quad (4.16)$$

We note that  $\Delta_p(f)$ ,  $I_p(f)$ ,  $N_p(f)$ ,  $R_p(f)$  defined in (4.11), (4.14), (4.15), (4.16), respectively, which strictly speaking are operators acting on functions  $f \in \mathcal{F}_V$ , can all be regarded as functions from  $\mathbb{R}^n$  to  $\mathbb{R}$  once we (equivalently) identify functions  $f \in \mathcal{F}_V$  with the vectors  $f \in \mathbb{R}^n$  of their values at the mesh vertices.

In the following Lemma 4.1.1 and Proposition 4.1.1 we highlight some important properties of function  $\psi_p$  and, then, of functions  $\Delta_p(f)$ ,  $I_p(f)$ ,  $N_p(f)$ ,  $R_p(f)$ . These properties will be useful in the subsequent Proposition 4.1.2, where we state and prove the discrete counterpart of the crucial sufficient condition such that a pair  $(\lambda, f)$  is a  $p$ -eigenpair, previously formalized in (4.7) for the continuous setting. Results outlined in Proposition 4.1.1 will also be used in Section 4.2 - namely, in Proposition 4.2.1 - for the analysis of the two proposed variational eigendecomposition approaches. We do not give the proof of Lemma 4.1.1 as it is a matter of simple calculus, whereas we report the proof of Proposition 4.1.2 for its importance and for completeness of presentation, even if an analogous proof can be found, e.g., in [22].

**Lemma 4.1.1.** *The function  $\psi_p : \mathbb{R} \rightarrow \mathbb{R}$  defined in (4.12), with  $p \in (1, +\infty)$ , satisfies*

$$\psi_p \in C^\infty(\mathbb{R} \setminus \{0\}) \cap C^s(\mathbb{R}), \quad \text{with } s = \begin{cases} \infty & \text{if } p \in \mathbb{N} \\ \lfloor p-1 \rfloor & \text{if } p \notin \mathbb{N} \end{cases}, \quad (4.17)$$

with derivatives reading

$$\psi_p^{(z)} = \left( \prod_{i=1}^z (p-i) \right) |x|^{p-1-z} (\text{sign}(x))^{z-1}, \quad z = 1, 2, 3, \dots$$

**Proposition 4.1.1.** *For any  $p \in (1, +\infty)$ , the functions  $\Delta_p$ ,  $I_p$ ,  $N_p$ ,  $R_p$  defined in (4.11), (4.14), (4.15), (4.16) satisfy*

$$\begin{aligned} I_p &\in C^{s+1}(\mathbb{R}^n) \cap C^\infty(\mathbb{R}^n \setminus \mathcal{Z}_1), & -\Delta_p &= \nabla I_p \in C^s(\mathbb{R}^n) \cap C^\infty(\mathbb{R}^n \setminus \mathcal{Z}_1), \\ N_p &\in C^{s+1}(\mathbb{R}^n) \cap C^\infty(\mathbb{R}^n \setminus \mathcal{Z}_2), & R_p &\in C^{s+1}(\mathbb{R}^n \setminus \{0_n\}) \cap C^\infty(\mathbb{R}^n \setminus (\mathcal{Z}_1 \cup \mathcal{Z}_2)), \end{aligned} \quad (4.18)$$

with  $s$  defined in (4.17) and with sets  $\mathcal{Z}_1, \mathcal{Z}_2 \subset \mathbb{R}^n$  given by

$$\mathcal{Z}_1 = \{f \in \mathbb{R}^n : \exists (i, j) \in E : f_i = f_j\}, \quad \mathcal{Z}_2 = \{f \in \mathbb{R}^n : \exists i = 1, \dots, n : f_i = 0\}.$$

After defining with a little abuse of notation  $\psi_p(f) := (\psi_p(f_1), \dots, \psi_p(f_n))^T$ , we also have

$$\nabla I_p(f) = -\Delta_p f, \quad \nabla N_p(f) = \psi_p(f), \quad \nabla R_p(f) = \frac{1}{N_p(f)} (-\Delta_p f - R_p(f) \psi_p(f)). \quad (4.19)$$

*Proof.* Starting from the definitions of functions  $I_p(f)$ ,  $N_p(f)$ ,  $R_p(f)$  in (4.14), (4.15), (4.16), using the symmetry of  $w_{i,j}$  and recalling from definition (4.12) that  $(|x|^p)' = p \psi_p(x)$ , the partial derivatives of  $I_p(f)$ ,  $N_p(f)$ ,  $R_p(f)$  with respect to the  $i$ -th independent variable  $f_i$  read

$$\begin{aligned} \frac{\partial}{\partial f_i} I_p(f) &= \frac{1}{2p} \frac{\partial}{\partial f_i} \left( \sum_{j \in \mathcal{N}(i)} \left( w_{i,j}^{p/2} |f_i - f_j|^p + w_{j,i}^{p/2} |f_j - f_i|^p \right) \right) \\ &= \sum_{j \in \mathcal{N}(i)} \left( w_{i,j}^{p/2} \phi_p(f_i - f_j) \right) = -(\Delta_p f)_i, \quad \forall i = 1, \dots, n, \end{aligned} \quad (4.20)$$

$$\frac{\partial}{\partial f_i} N_p(f) = \frac{1}{p} \frac{\partial}{\partial f_i} (\|f\|_p^p) = \frac{1}{p} \frac{\partial}{\partial f_i} \left( \sum_{i=1}^n |f_i|^p \right) = \psi_p(f_i), \quad \forall i = 1, \dots, n, \quad (4.21)$$

$$\begin{aligned} \frac{\partial}{\partial f_i} R_p(f) &= \frac{\partial}{\partial f_i} \left( \frac{I_p(f)}{N_p(f)} \right) = \frac{\left( \frac{\partial}{\partial f_i} I_p(f) \right) N_p(f) - I_p(f) \left( \frac{\partial}{\partial f_i} N_p(f) \right)}{N_p^2(f)} \\ &= \frac{(-\Delta_p f)_i - R_p(f) \psi_p(f_i)}{N_p(f)}, \quad \forall i = 1, \dots, n. \end{aligned} \quad (4.22)$$

where the last equality in (4.20) comes immediately from the definition in (4.11). The gradient expressions in (4.19) follow from the partial derivative expressions in (4.20)-(4.22). The smoothness properties in (4.18) are easy to prove starting from the definitions of functions  $I_p$ ,  $N_p$ ,  $R_p$  in (4.14), (4.15), (4.16) and recalling the smoothness property of function  $\psi_p$  reported in (4.17). The smoothness of function  $\Delta_p$  follows from that of the function  $I_p$ , as we proved that  $-\Delta_p = \nabla I_p$ .  $\square$

**Proposition 4.1.2.** *For any  $p \in (1, +\infty)$ , if a function  $f \in \mathcal{F}_V$  is a critical (i.e. stationary) point of the graph  $p$ -Rayleigh quotient  $R_p$  in (4.16), then  $f$  is a  $p$ -eigenvector of the graph  $p$ -Laplacian  $\Delta_p$  in (4.11) and the associated  $p$ -eigenvalue is given by  $\lambda = R_p(f)$ .*

*Proof.* It follows immediately from the expression of the gradient of  $R_p$  given in (4.19) that

$$\nabla R_p(f) = 0_n \iff -\Delta_p f = R_p(f) \psi_p(f).$$

Hence, any critical point  $f$  of the graph  $p$ -Rayleigh quotient  $R_p$  satisfies the graph  $p$ -Laplacian eigendecomposition equation in (4.13) with eigenvalue  $\lambda = R_p(f)$ .  $\square$

Being the Rayleigh quotient  $R_p(f)$  scale-invariant, namely

$$R_p(\alpha f) = R_p(f), \quad \forall f \in \mathbb{R}^n, \alpha \in \mathbb{R} \neq 0, \quad (4.23)$$

then we can restrict the study of its critical points to the special case of  $\|f\|_p^p = 1$  in (4.16). Moreover, as in the linear eigendecomposition, also the eigenvectors are scale-invariant, i.e. if  $f$  is a  $p$ -eigenvector, then, for all  $c \in \mathbb{R}$ ,  $cf$  is a  $p$ -eigenvector, [81].

For the standard linear Laplace operator, it is well-known that the eigenvectors form an orthogonal basis. Unfortunately, this property is lost when we consider the nonlinear  $p$ -Laplacian eigendecomposition. However, it is possible to generalize the orthogonality property, starting from the following definition.

**Definition 4.1.6 ( $p$ -orthogonality).** *Two functions  $f, g \in \mathcal{F}_V$  are  $p$ -orthogonal if*

$$\sum_{i=1}^n \psi_p(f_i) \psi_p(g_i) = \sum_{i=1}^n (|f_i|^{p-2} f_i |g_i|^{p-2} g_i) = \sum_{i=1}^n (f_i g_i |f_i g_i|^{p-2}) = 0. \quad (4.24)$$

With a little abuse of notation, by indicating as  $\psi_p(f)$  the component-wise application of function  $\psi_p$  in (4.12) to vector  $f$ , Eq. (4.24) can be rewritten in compact form as

$$\langle \psi_p(f), \psi_p(g) \rangle = \psi_p(f)^T \psi_p(g) = 0. \quad (4.25)$$

We notice that, for  $p = 2$ , the  $p$ -orthogonality condition in (4.24) or (4.25) reduces to the standard orthogonality condition, since  $\psi_2(f) = f$ .

Finally, we recall an interesting result given in [81] on  $p$ -orthogonality of the eigenvectors of the graph  $p$ -Laplacian operator.

**Theorem 4.1.1** (Theorem 3 from [81]). *Let  $p \in (1, +\infty)$  and let  $f$  and  $g$  be two  $p$ -eigenvectors of the graph  $p$ -Laplacian operator with associated eigenvalues  $\lambda_f \neq \lambda_g$ . Then,  $f$  and  $g$  are  $p$ -orthogonal up to the second order Taylor expansion.*

After having introduced a rigorous definition of  $p$ -eigenpairs, in the next section we focus on how to compute them numerically.

## 4.2 The variational $p$ -eigendecomposition model, with $p$ -orthogonality constraint

In this section, we address the problem of computing the graph  $p$ -Laplacian eigenpairs, i.e. the solutions of the nonlinear  $p$ -eigenproblem (4.13) by variational approaches.

Proposition 4.1.2 states the relationship between critical points of the  $p$ -Rayleigh quotient  $R_p(f)$  in (4.16) and  $p$ -eigenpairs of the graph  $p$ -Laplacian operator  $\Delta_p$ : critical points are always eigenfunctions and critical values of  $R_p(f)$  are their eigenvalues, but the converse is not true in general. Then, a variational approach to compute a  $p$ -eigenpairs  $(\lambda^*, f^*) \in \mathbb{R} \times \mathbb{R}^n$  is through finding a local (minimum, minimizer) of  $R_p(f)$ , by solving

$$f^* \in \arg \min_{f \in \mathbb{R}^n} R_p(f). \quad (4.26)$$

The scale invariance property (4.23) allows us to reformulate the optimization problem (4.26) as the following constrained minimization

$$f^* \in \arg \min_{f \in \mathbb{R}^n} I_p(f) \quad \text{subject to} \quad \|f\|_p^p = 1.$$

By introducing the Lagrangian

$$\mathcal{L}(f; \lambda) = \langle |\nabla f|^p, 1 \rangle - \lambda (\|f\|_p^p - 1),$$

with  $\lambda \in \mathbb{R}$  the Lagrange multiplier, equating the derivative of Lagrangian to zero gives

$$\mathbb{R}^n \ni \frac{\partial \mathcal{L}}{\partial f} = -p\Delta_p(f) - \lambda p\psi_p(f) = 0_n$$

which corresponds to the  $p$ -eigendecomposition problem defined in (4.13), and by definition,  $f$  is an eigenvector of  $\Delta_p$ .

Solving the optimization problem (4.26) allows to find only a single  $p$ -eigenfunction of the graph  $p$ -Laplacian. In particular, since  $R_p(f) \geq 0$  for all  $f$ , with  $R_p(f) = 0$  for constant functions, the first candidate for critical points is the trivial solution  $f^{(1)} \equiv (1/n)^{1/p} \mathbf{1}_n$ . To determine non-trivial solutions, we can exploit the  $p$ -orthogonality property of the eigenfunctions.

The problem of simultaneously estimating multiple eigenvectors  $f^{(1)}, \dots, f^{(k)}$  of the graph  $p$ -Laplacian can be formulated as an optimization problem under nonlinear  $p$ -orthogonality constraints, defining a matrix  $\Phi^{(k)} \in \mathbb{R}^{n \times k}$  whose columns are estimates of the first  $k$  eigenfunctions  $f^{(l)}$ ,  $l = 1, \dots, k$ , associated with the  $k$  smallest eigenvalues of the graph  $p$ -Laplacian operator  $\Delta_p$ ; in formula,

$$\Phi^{(k)} = (f^{(1)}; f^{(2)}; \dots; f^{(k)}). \quad (4.27)$$

Then, the problem is defined as follows:

$$\Phi^{(k)} \in \arg \min_{\Phi \in \mathbb{R}^{n \times k}} R_p(\Phi) = \sum_{i=1}^k \frac{I_p(f^{(i)})}{\|f^{(i)}\|_p^p} \quad \text{s.t.} \quad \langle \psi_p(f^{(i)}), \psi_p(f^{(j)}) \rangle = 0 \quad \forall i, j = 1, \dots, k, i \neq j \quad (4.28)$$

It is worth noting that, in the case  $p = 2$ , the constraint in (4.28) reduces to  $\Phi^{(k)} \in \mathcal{O}_2$ , the standard linear orthogonal manifold which corresponds to imposing standard orthogonality between all computed eigenfunctions in the domain.

The numerical difficulty of solving (4.28) can be partially reduced by an incremental scheme which computes a new eigenfunction  $f^{(k+1)}$  - i.e., add a new,  $(k+1)$ -th column to matrix  $\Phi^{(k)}$  - given a set of already computed  $p$ -orthogonal eigenfunctions  $f^{(1)}, \dots, f^{(k)}$  - i.e., given matrix  $\Phi^{(k)}$ . Starting from the first, known constant eigenfunction  $f^{(1)} = (1/n)^{1/p} \mathbf{1}_n$ , problem (4.28) is thus transformed into the following sequence of simpler problems:

$$f^{(k+1)} \in \arg \min_{f \in \mathbb{R}^n} R_p(f) \quad \text{subject to} \quad f \in \mathcal{O}_p^{(k)}, \quad k = 1, 2, \dots, \quad (4.29)$$

$$\text{with} \quad \mathcal{O}_p^{(k)} := \{ f \in \mathcal{F}_V : \langle \psi_p(f), \psi_p(f^{(l)}) \rangle = 0, \quad \forall l = 1, \dots, k \}, \quad (4.30)$$

where the manifold constraint represents the  $k$ -th  $p$ -orthogonality constraint manifold.

Applying the scaling invariance property (4.23) we can reformulate problem (4.29)-(4.30) as follows

$$f^{(k+1)} \in \arg \min_{f \in \mathbb{R}^n} I_p(f) \quad \text{subject to} \quad f \in \mathcal{O}_p^{(k)} \cap \mathcal{S}_p, \quad k = 1, 2, \dots, \quad (4.31)$$

$$\text{with} \quad \mathcal{S}_p := \{ f \in \mathcal{F}_V : \|f\|_p^p = 1 \}, \quad (4.32)$$

and  $\mathcal{O}_p^{(k)}$  defined in (4.30). We notice that the cost function  $I_p$  in (4.31) is the numerator of the graph  $p$ -Rayleigh quotient in (4.16) and the manifold  $\mathcal{S}_p$  in (4.32) represents the  $p$ -hypersphere with center the origin and unitary radius.

In the following Proposition 4.2.1, partly relying on the results derived in Proposition 4.1.1, we analyze the two proposed (incremental) optimization problems (4.29)-(4.30) and (4.31)-(4.32).

**Proposition 4.2.1.** *For any  $p \in (1, +\infty)$  and any  $k \in \{1, 2, \dots\}$ , both the optimization problems in (4.29)-(4.30) and (4.31)-(4.32) admit solutions. However, due to the non-convexity of the cost functions and/or of the constraints, the uniqueness of the solution is not guaranteed and even non-connected sets of global constrained minimizers may exist.*

*Proof.* For any  $p \in (1, +\infty)$  the cost functions  $R_p$  in (4.29) and  $I_p$  in (4.31) are both bounded below by zero and non-coercive,  $I_p$  is convex (not strictly convex) and at least  $C^1(\mathbb{R}^n)$ ,  $R_p$  is not defined in  $f = 0_n$ , non-convex and at least  $C^1(\mathbb{R}^n \setminus \{0_n\})$ . The constraint set  $\mathcal{O}_p^{(k)}$  of the first optimization problem in (4.29)-(4.30) is an unbounded manifold of dimension  $n - k$ , whereas the constraint set  $\mathcal{O}_p^{(k)} \cap \mathcal{S}_p$  of the second optimization problem in (4.31)-(4.32) is compact, as it is the intersection of  $\mathcal{O}_p^{(k)}$  and the  $p$ -hypersphere  $\mathcal{S}_p$  in (4.32), which is compact. It follows that the second problem (4.31)-(4.32) admits solutions, but convexity of the cost function  $I_p$  is not sufficient to guarantee the uniqueness of the solution, as the constraint set is a nonlinear manifold, hence a nonconvex

set. The cost function  $R_p$  of the first problem is noncoercive and nonconvex, but it is a radial function - i.e.,  $R_p(cf) = R_p(f)$  for any  $f \in \mathbb{R}^n \setminus \{0_n\}$  and any  $c \in \mathbb{R} \setminus \{0\}$ . It follows that, even if the constraint set is unbounded,  $R_p$  can not admit a global infimizer for  $\|f\|_2$  tending to  $+\infty$  without admitting a global minimizer for some  $f \in \mathbb{R}^n$ . Hence, also the first optimization problem admits solutions but, like the second one, uniqueness is not guaranteed.  $\square$

### 4.3 Iterative optimization to solve the $p$ -eigendecomposition problem

In this section, we address the numerical solution of the two (incremental) optimization problems (4.29)-(4.30) and (4.31)-(4.32). The most complicated ingredient to deal with numerically in both the two problems is the nonlinear  $p$ -orthogonality manifold constraint defined in (4.30). To tackle this issue, first in Section 4.3.1 we reformulate the two problems into equivalent ones by a suitable  $p$ -dependent change of variable which, for any  $p \in (0, +\infty)$ , transforms the  $p$ -orthogonality constraint into a simple, linear constraint, independently of  $p$ . Then, in Section 4.3.2 and Section 4.3.3 we propose two numerical approaches for the solution of the reformulated versions of problems (4.29)-(4.30) and (4.31)-(4.32), respectively. Thanks to the reformulation, the first approach can estimate each new eigenfunction  $f^{(k+1)}$  through a simple gradient descent method projected on linear constraints, while the second approach relies on an ADMM-based algorithm with simple manifold constraints. We would like to stress that both the two algorithmic approaches, referred to as M-PGD and M-ADMM, respectively, guarantee to provide output eigenfunctions that satisfy exactly the  $p$ -orthogonality constraints, with  $p \in (1, 2)$ .

For both approaches, once a set of estimated eigenfunctions  $f^{(2)}, \dots, f^{(k)}$  has been obtained, we want to estimate also the set of associated eigenvalues  $\lambda^{(2)}, \dots, \lambda^{(k)}$  - note that the first constant eigenfunction  $f^{(1)} = (1/n)^{1/p} \mathbf{1}_n$  has associated eigenvalue  $\lambda^{(1)} = 0$ . The first method to compute estimates of the eigenvalues is based on Proposition 4.1.2 which states that, if an estimated eigenfunction  $f^{(k)}$  is an unconstrained stationary point of the  $p$ -Rayleigh quotient  $R_p$ , then the associated eigenvalue is given by

$$\lambda_{R_p}^{(k)} := R_p(f^{(k)}) , \quad k = 1, 2, \dots \quad (4.33)$$

However, Prop. 4.1.2 do not ensure that the eigenvalues are local maximal/minimal values of the  $p$ -Rayleigh quotient on the linear subspaces spanned by the corresponding eigenfunction, but only the vice-versa.

For this reason, we propose a second method which estimates  $\lambda^{(k)}$  directly from the original nonlinear eigenvalue equation (4.13), based on noting that the eigenvalue  $\lambda$  associated to any true eigenfunction  $f$  is (clearly) the angular coefficient of the homogeneous line  $y = \lambda x$  passing through the  $n$  points  $(x_i, y_i) \in \mathbb{R}^2$ ,  $i = 1, \dots, n$ , with  $x_i = (\psi_p(f))_i$

and  $y_i = (-\Delta_p(f))_i$ . Hence, by introducing for any computed eigenfunction  $f^{(k)}$  the  $n$  points

$$\left(x_i^{(k)}, y_i^{(k)}\right), \quad x_i^{(k)} = \left(\psi_p(f^{(k)})\right)_i, \quad y_i^{(k)} = \left(-\Delta_p(f^{(k)})\right)_i, \quad i = 1, \dots, n, \quad (4.34)$$

we estimate the associated eigenvalue  $\lambda_{\text{OLS}}^{(k)}$  by Orthogonal Least-Squares (OLS) [103] fitting of data in (4.34) by line  $y = \lambda x$ . It is well-known that such a fitting admits the explicit solution

$$\lambda_{\text{OLS}}^{(k)} = \frac{S_{yy}^{(k)} - S_{xx}^{(k)} + \sqrt{\left(S_{yy}^{(k)} - S_{xx}^{(k)}\right)^2 + 4\left(S_{xy}^{(k)}\right)^2}}{2 S_{xy}^{(k)}}, \quad (4.35)$$

with data second-order moments defined by

$$S_{xx}^{(k)} = \sum_{i=1}^n \left(x_i^{(k)}\right)^2, \quad S_{yy}^{(k)} = \sum_{i=1}^n \left(y_i^{(k)}\right)^2, \quad S_{xy}^{(k)} = \sum_{i=1}^n x_i^{(k)} y_i^{(k)}. \quad (4.36)$$

We chose OLS fitting - and not, e.g., ordinary (vertical) LS - since both the  $x_i^{(k)}$  and  $y_i^{(k)}$  data are computed as functions of the estimated  $f^{(k)}$  and, hence, can both be affected by errors. We did not use a more general (and, potentially, more accurate) total LS fitting as we have no information about the characteristics (covariance matrix) of the errors.

### 4.3.1 Optimization problems reformulation by change of variable

The key ingredient shared by the two optimization approaches is the following preliminary one-to-one (invertible),  $p$ -dependent, component-wise change of variable,

$$\tilde{f} = \psi_p(f) = |f|^{p-1} \text{sign}(f) \iff f = \psi_p^{-1}(\tilde{f}) = |\tilde{f}|^r \text{sign}(\tilde{f}), \quad r = \frac{1}{p-1} \in (1, +\infty), \quad (4.37)$$

where all functions are intended component-wise and where the expression of the inverse  $\psi_p^{-1}$  comes easily from that of  $\psi_p$ . In the top row of Figure 4.1 we show the component-wise change of variable and its inverse (i.e.,  $f, \tilde{f} \in \mathbb{R}$ ,  $\psi_p, \psi_p^{-1} : \mathbb{R} \rightarrow \mathbb{R}$ ) for  $p \in \{2, 1.8, 1.5, 1.2\}$ . This change of variable is mainly aimed to transform the complicated  $p$ -orthogonality manifold constraints  $f \in \mathcal{O}_p^{(k)} \subset \mathbb{R}^n$  present in both the proposed (incremental) optimization problems (4.29)-(4.30) and (4.31)-(4.32) into (equivalent) very simple, linear ones. In fact, it is immediate to verify that the effect of change of variable (4.37) on the manifold constraints  $f \in \mathcal{O}_p^{(k)}$  defined in (4.30) is as follows

$$f \in \mathcal{O}_p^{(k)} \iff \tilde{f} \in \tilde{\mathcal{O}}_p^{(k)} = \left\{ \tilde{f} \in \mathcal{F}_V : \left(\tilde{\Phi}^{(k)}\right)^T \tilde{f} = 0_k \right\} = \mathcal{O}_2^{(k)}, \quad (4.38)$$



where, analogously to  $\Phi^{(k)}$  in (4.27), the introduced matrix  $\tilde{\Phi}^{(k)} \in \mathbb{R}^{n \times k}$  contains (in its columns) the first  $k$  estimated eigenfunctions in the transformed domain; in formula,

$$\tilde{\Phi}^{(k)} = \left( \tilde{f}^{(1)}; \tilde{f}^{(2)}; \dots; \tilde{f}^{(k)} \right) = \left( \psi_p(f^{(1)}); \psi_p(f^{(2)}); \dots; \psi_p(f^{(k)}) \right).$$

We remark that the transformed  $p$ -orthogonality manifold in (4.38), which we have called  $\mathcal{O}_2^{(k)}$ , is not only linear, but also corresponds to imposing standard orthogonality between all computed eigenfunctions in the transformed domain (like for the case  $p = 2$  of standard Laplace operator), independently of the original value of  $p$ . To get a visual insight into geometry of the  $p$ -orthogonality constraints, in the bottom row of Figure 4.1 we show the original  $p$ -orthogonality constraint manifold  $\mathcal{O}_p^{(1)}$  and its transformed  $\tilde{\mathcal{O}}_p^{(1)}$  in  $\mathbb{R}^3$  (that is,  $n = 3$  and the manifolds are surfaces) for  $p \in \{2, 1.8, 1.5, 1.2\}$ , in the particular case of a first eigenfunction given by  $f^{(1)} = \bar{f}^{(1)} / \|\bar{f}^{(1)}\|_p$ , with  $\bar{f}^{(1)} = (1; 2; 1.5)$ . It can be seen how, already starting from a value of  $p = 1.5$  the original manifold  $\mathcal{O}_p^{(k)}$  (in blue) is rather complicated, to the point of becoming almost intractable (in the sense of calculating the orthogonal projection of a vector onto it) for  $p = 1.2$ .

For what concerns the manifold constraint  $f \in \mathcal{S}_p$  (which is present only in the latter problem), we have

$$f \in \mathcal{S}_p \iff \tilde{f} \in \tilde{\mathcal{S}}_p = \left\{ \tilde{f} \in \mathcal{F}_V : \|\tilde{f}\|_q^q = 1 \right\} = \mathcal{S}_q, \quad q = \frac{p}{p-1}. \quad (4.39)$$

That is, the original  $p$ -hypersphere  $\mathcal{S}_p$  is transformed into a  $q$ -hypersphere  $\mathcal{S}_q$ , with  $p, q$  Hölder conjugates,  $p \in (1, 2)$ ,  $q \in (2, \infty)$ . The transformed constrained problem is not numerically harder to solve than the original, because in both case we have non-linear unitary-norm constraints.

Finally, the cost function  $I_p(f)$  in the second model (4.31)-(4.32) turns into the new function  $\tilde{I}_p(\tilde{f})$  given by

$$\tilde{I}_p(\tilde{f}) = I_p(\phi_p^{-1}(\tilde{f})) = \frac{1}{2} \sum_{i,j=1}^n w_{i,j}^{p/2} H_p(\tilde{f}_i, \tilde{f}_j), \quad (4.40)$$

with function  $H_p : \mathbb{R}^2 \rightarrow \mathbb{R}_+$  defined by

$$H_p(x_1, x_2) = |\psi_p^{-1}(x_1) - \psi_p^{-1}(x_2)|^p,$$

whereas the cost function  $R_p(f)$  in the first model (4.29)-(4.30) is transformed into  $\tilde{R}_p(\tilde{f})$  reading

$$\tilde{R}_p(\tilde{f}) = R_p(\psi_p^{-1}(\tilde{f})) = \frac{I_p(\psi_p^{-1}(\tilde{f}))}{\|\psi_p^{-1}(\tilde{f})\|_p^p} = \frac{\tilde{I}_p(\tilde{f})}{\|\tilde{f}\|_q^q}. \quad (4.41)$$

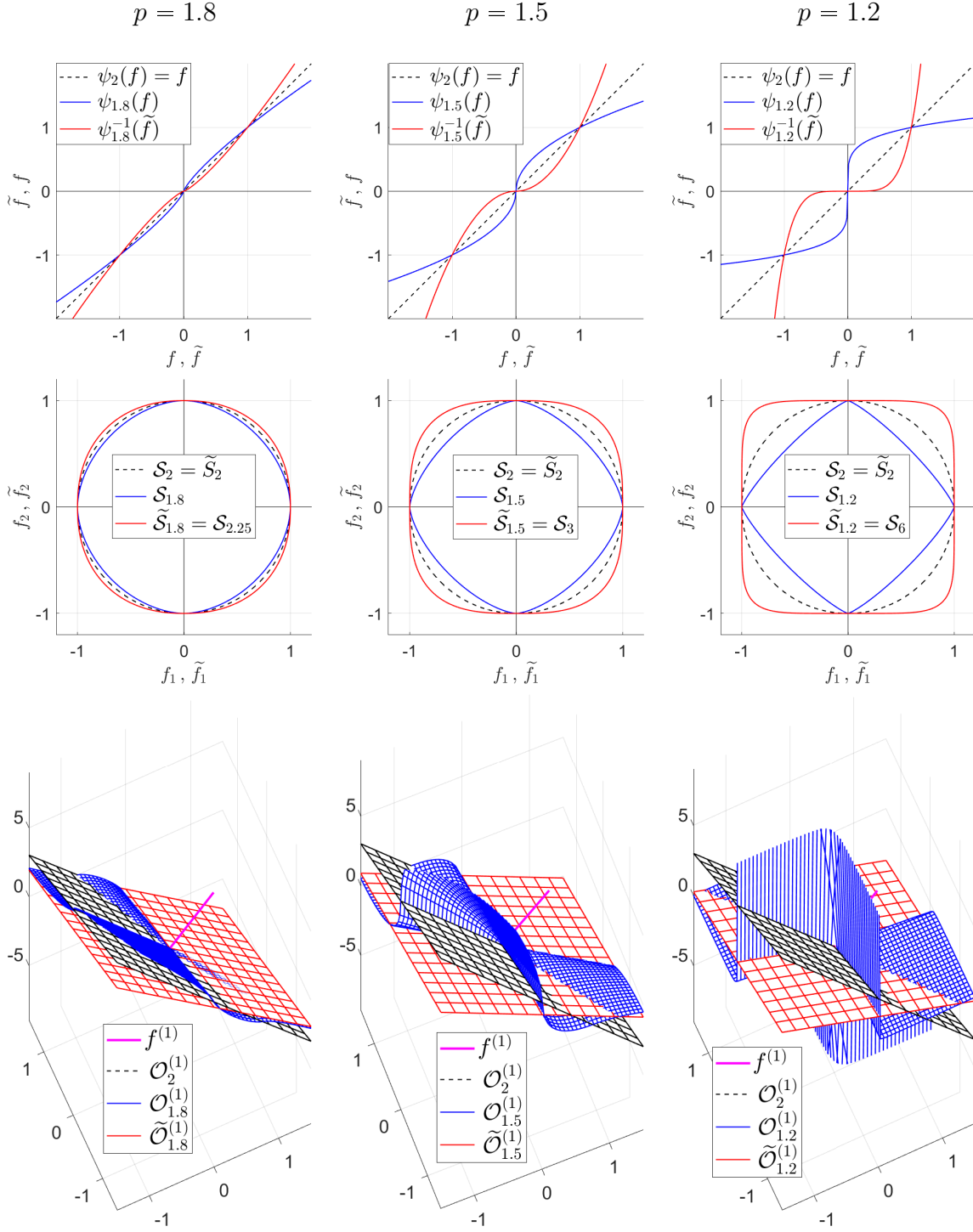


Figure 4.1: Visual analysis of the one-to-one change of variable introduced in (4.37) for  $p \in \{2, 1.8, 1.5, 1.2\}$ . Blue and red colors indicate - independently of  $p$  - quantities in the original and transformed domains, respectively, whereas the black color is associated with the particular case  $p = 2$ , for which original and transformed quantities coincide. Top row: direct  $\tilde{f} = \psi_p(f)$  and inverse  $f = \psi_p^{-1}(\tilde{f})$  scalar change of variable; middle row: unitary  $p$ -norm constraint manifold  $\mathcal{S}_p$  and its transformed  $\tilde{\mathcal{S}}_p$  in  $\mathbb{R}^2$ ; bottom row:  $p$ -orthogonality constraint manifold  $\mathcal{O}_p^{(1)}$  and its transformed  $\tilde{\mathcal{O}}_p^{(1)}$  in  $\mathbb{R}^3$  for  $f^{(1)} = \bar{f}^{(1)} / \|\bar{f}^{(1)}\|_p$ , with  $\bar{f}^{(1)} = (1; 2; 1.5)$ .

Therefore, the two original optimization problems (4.29)-(4.30) and (4.31)-(4.32) are transformed by (4.37) into the equivalent ones:

$$\tilde{f}^{(k+1)} \in \arg \min_{\tilde{f} \in \mathbb{R}^n} \tilde{R}_p(\tilde{f}) \quad \text{subject to} \quad \tilde{f} \in \mathcal{O}_2^{(k)}, \quad k = 1, 2, \dots, \quad (4.42)$$

$$\tilde{f}^{(k+1)} \in \arg \min_{\tilde{f} \in \mathbb{R}^n} \tilde{I}_p(\tilde{f}) \quad \text{subject to} \quad \tilde{f} \in \mathcal{O}_2^{(k)} \cap \mathcal{S}_q, \quad k = 1, 2, \dots, \quad (4.43)$$

respectively, with  $\tilde{R}_p$  defined in (4.41),  $\mathcal{O}_2^{(k)}$  in (4.38),  $\tilde{I}_p$  in (4.40) and  $\mathcal{S}_q$  in (4.39). For both approaches, the last step is to compute the eigenfunctions in the original domain by simply applying the inverse change of variable, that is

$$f^{(k)} = \psi_p^{-1}(\tilde{f}^{(k)}) = \left| \tilde{f}^{(k)} \right|^{\frac{1}{p-1}} \text{sign}(\tilde{f}^{(k)}), \quad k = 2, 3, \dots,$$

with all operations intended component-wise.

Finally, since both the two optimization problems in (4.42) and (4.43) include the linear constraint  $\tilde{f} \in \mathcal{O}_2^{(k)}$ , in Proposition 4.3.1 below we recall a well-known result (no proof is given, [104]) concerning orthogonal projection onto linear subspaces, which will be used in the next sections.

**Proposition 4.3.1.** *Let  $z \in \mathbb{R}^n$  and let  $Z \in \mathbb{R}^{n \times k}$  be a tall matrix ( $k < n$ ) with full column rank. Then, the orthogonal projection  $\bar{z}$  of  $z$  onto the linear subspace  $Z^T x = 0$  is given by*

$$\bar{z} = \text{proj}_{Z^T z=0}(z) = P z, \quad \text{with} \quad P = I_n - Z (Z^T Z)^{-1} Z^T \in \mathbb{R}^{n \times n}. \quad (4.44)$$

If all columns  $Z_i \in \mathbb{R}^n$ ,  $i = 1, \dots, k$ , of  $Z$  are mutually orthogonal, (4.44) simplifies to

$$\bar{z} = P z, \quad \text{with} \quad P = I_n - \Psi \Psi^T \in \mathbb{R}^{n \times n}, \quad \Psi = \left( \frac{Z_1}{\|Z_1\|_2}, \dots, \frac{Z_k}{\|Z_k\|_2} \right) \in \mathbb{R}^{n \times k},$$

so that, denoting by  $\Psi_i \in \mathbb{R}^n$  the  $i$ -th column of matrix  $\Psi$ ,  $\bar{z}$  is efficiently computed by

$$\bar{z} = \text{proj}_{Z^T z=0}(z) = z - \sum_{i=1}^k \Psi_i \left( \Psi_i^T z \right). \quad (4.45)$$

### 4.3.2 Solving (4.42) via Projected Gradient Descent method on Manifold (M-PGD)

We compute approximate solutions to any  $k$ -th transformed optimization problem in (4.42) by means of the M-PGD iterative algorithm with constraint manifold  $\mathcal{O}_2^{(k)}$  defined in (4.38). Generalizing the projected gradient descent method from Euclidean spaces to Riemannian manifolds requires us to use the Riemannian gradient as the search direction and the projection to move between points on the manifold.

For any  $p \in (1, 2)$ , the function  $\tilde{R}_p$  in (4.41) is continuously differentiable on  $\mathbb{R}^n$  and bounded below by zero, and its gradient reads

$$\nabla \tilde{R}_p(\tilde{f}) = \frac{1}{\|\psi_p^{-1}(\tilde{f})\|_p^p} \left( p\Delta_p(\psi_p^{-1}(\tilde{f})) - \frac{p\tilde{f}}{\|\psi_p^{-1}(\tilde{f})\|_p^p} I_p(\psi_p^{-1}(\tilde{f})) \right) \frac{|\tilde{f}|^{\frac{2-p}{p-1}}}{p-1}. \quad (4.46)$$

In fact, we recall that  $\tilde{R}_p(\tilde{f}) = R_p(\psi_p^{-1}(\tilde{f}))$ . Recovering from equation (4.19) the expression for  $\nabla R_p(f)$  and observing that

$$(\psi_p^{-1})'(x) = \frac{|x|^{\frac{2-p}{p-1}}}{p-1},$$

we can apply the chain rule and get the expression

$$\nabla R_p(\psi_p^{-1}(\tilde{f})) = \frac{1}{\|\psi_p^{-1}(\tilde{f})\|_p^p} \left( p\Delta_p(\psi_p^{-1}(\tilde{f})) - \frac{p\psi_p(\psi_p^{-1}(\tilde{f}))}{\|\psi_p^{-1}(\tilde{f})\|_p^p} I_p(\psi_p^{-1}(\tilde{f})) \right) \frac{|\tilde{f}|^{\frac{2-p}{p-1}}}{p-1},$$

from which we derive (4.46).

The M-PGD algorithm is trivial. We denote by  $\ell = 0, 1, \dots$  the index of the M-PGD iterations, which we refer as inner iterations of the overall incremental approach proposed in (4.42) to distinguish them from the outer ones, with index  $k = 1, 2, \dots$ , over the different eigenfunctions to compute. For any  $k = 1, 2, \dots$ , we compute the  $(k+1)$ -th eigenfunction  $\tilde{f}^{(k+1)}$  by applying the M-PGD algorithm to the solution of problem (4.42). At each  $l$ -th M-PGD iteration, first we compute the gradient  $g^{(l)} \in \mathbb{R}^n$  of the cost function  $\tilde{R}_p$  by means of the formula in (4.46). Then, we compute the orthogonal projection of vector  $g^{(l)}$  onto the linear subspace  $\mathcal{O}_2^{(k)}$  defined in (4.38), indicated by  $d^{(l)} = \text{proj}_{\mathcal{O}_2^{(k)}}(g^{(l)})$ . Since the columns of matrix  $\tilde{\Phi}^{(k)}$  in (4.38) are orthogonal by construction (note, however, that their Euclidean norm is not unitary), by applying Prop. 4.3.1, in particular formula (4.45), we have

$$d^{(\ell)} = \text{proj}_{\mathcal{O}_2^{(k)}}(g^{(\ell)}) = g^{(\ell)} - \sum_{i=1}^k \tilde{\Psi}_i^{(k)} \left( \left( \tilde{\Psi}_i^{(k)} \right)^T g^{(\ell)} \right), \quad (4.47)$$

---

**Algorithm 2** M-PGD algorithm solving any  $k$ -th optimization problem in (4.42)

---

1. **Input:**  $\tilde{f}^{(1)}, \dots, \tilde{f}^{(k)} \in \mathbb{R}^n$  computed (orthogonal) eigenfunctions, step-size  $\alpha > 0$
  2. **Output:**  $\tilde{f}^{(k+1)}$  new (orthogonal) eigenfunction, approximate solution to (4.42)
  3. **Initialize:**  $\ell = 0, t^{(0)} = t_0$ , compute  $\tilde{\Psi}_i^{(k)} \in \mathbb{R}^n, i = 1, \dots, k$ , by (4.48)
  4. **While the stopping criterion is not satisfied**
    5.     • compute the gradient  $g^{(\ell)}$  by (4.46)
    6.     • project the gradient  $d^{(\ell)} = \text{proj}_{\mathcal{O}_2^{(k)}}(g^{(\ell)})$  by (4.47)
    7.     • update  $t^{(\ell+1)} = t^{(\ell)} - \alpha d^{(\ell)}$
    8.     • update  $\ell = \ell + 1$
  9. **end while**
  10.  $\tilde{f}^{(k+1)} = t^{(\ell)}$
- 

with column vectors  $\tilde{\Psi}_i^{(k)} \in \mathbb{R}^n$  given by

$$\tilde{\Psi}_i^{(k)} = \frac{\tilde{f}^{(i)}}{\|\tilde{f}^{(i)}\|_2}, \quad i = 1, 2, \dots, k. \quad (4.48)$$

We note that all vectors  $\tilde{\Psi}_i^{(k)}$  in (4.48) do not change along the M-PGD iterations, hence they can be computed once for all before starting to iterate. It follows that, thanks to the change of variable introduced in Section 4.3.1, the orthogonal projection of any vector in  $\mathbb{R}^n$  on the transformed, linear  $p$ -orthogonality manifold constraint  $\mathcal{O}_2^{(k)}$  in (4.38) not only admits an explicit solution, but costs only  $2n(k+1)$  FLOPS - or, in other terms, has computational complexity  $\mathcal{O}(kn)$ , where commonly  $k \ll n$ . We highlight how, without the change of variable, calculating the orthogonal projection onto the nonlinear  $p$ -orthogonality manifold constraint  $\mathcal{O}_p^{(k)}$  in (4.30) would certainly require using an iterative method with an incomparable computational cost - see also the visual representations of the non-linear manifold  $\mathcal{O}_p^{(k)}$  in the bottom row of Figure 4.1, in blue.

The main computational steps of the M-PGD approach are summarized in Algorithm 2. The M-PGD iterations are stopped as soon as one of the two following conditions is fulfilled:

$$\frac{\|t^{(\ell+1)} - t^{(\ell)}\|_2}{\|t^{(\ell)}\|_2} < \text{Tol}, \quad \ell > \text{Nits}. \quad (4.49)$$

Being the cost function  $\tilde{R}_p$  continuously differentiable and lower bounded, the convergence of the M-PGD algorithm to critical points of the problem is guaranteed by standard arguments when  $\alpha$  belongs to a range of step-sizes that lead to sufficient decrease [19].

### 4.3.3 Solving (4.43) via ADMM on Manifold (M-ADMM)

We solve any  $k$ -th transformed optimization problem in (4.43) by means of a two-block Manifold ADMM approach [66]. Like for M-PGD, we use  $\ell = 0, 1, \dots$  to denote the index of the M-ADMM iterations and  $t \in \mathbb{R}^n$  to indicate the inner M-ADMM optimization variable corresponding to the variable  $\tilde{f}$  in (4.43).

First, we introduce the auxiliary variable  $g \in \mathbb{R}^{n+2m}$  defined by

$$g = \begin{pmatrix} g_1 \\ g_2 \end{pmatrix}, \quad g_1 = t \in \mathbb{R}^n, \quad g_2 = S t \in \mathbb{R}^{2m}, \quad (4.50)$$

where  $S \in \mathbb{R}^{2m \times n}$  is a binary matrix which selects, in the order, the  $t$ -values in correspondence of the first and second vertices of all the  $m \leq n^2$  graph edges with associated positive weights, that we indicate by  $w_1, \dots, w_m \in \mathbb{R}_+^*$ . Matrix  $S$  is clearly very sparse (1 non-zero entry on each row) and satisfies

$$S^T S = \text{diag}(v_1, v_2, \dots, v_n), \quad (4.51)$$

with  $v_i \in \mathbb{N}$  denoting the valence (number of incident edges) of vertex  $i$  in the graph. It is then easy to verify that, based on (4.50), the optimization problem in (4.43) can be equivalently re-written as

$$\{\tilde{f}^{(k+1)}, g^{(k+1)}\} \in \arg \min_{t, g} \{F(t) + G(g)\} \quad \text{subject to} \quad A t + B g = 0_{n+2m}, \quad (4.52)$$

where the two cost functions  $F: \mathbb{R}^n \rightarrow \mathbb{R}$  and  $G: \mathbb{R}^{n+2m} \rightarrow \mathbb{R}$  are defined by

$$F(t) = \iota_{\mathcal{O}_2^{(k)}}(t), \quad (4.53)$$

$$G(g) = \underbrace{\iota_{\mathcal{S}_q}(g_1)}_{G_1(g_1)} + \underbrace{\sum_i^m w_i^{p/2} \left| |g_{2,i}|^r \text{sign}(g_{2,i}) - |g_{2,i+m}|^r \text{sign}(g_{2,i+m}) \right|^p}_{G_2(g_2)}, \quad (4.54)$$

and the two matrices  $A \in \mathbb{R}^{(n+2m) \times n}$  and  $B \in \mathbb{R}^{(n+2m) \times (n+2m)}$  read

$$A = \begin{pmatrix} I_n \\ S \end{pmatrix}, \quad B = -I_{n+2m}. \quad (4.55)$$

The augmented Lagrangian function associated to the reformulated problem (4.52)-(4.55) reads

$$\mathcal{L}_\beta(t, g, \rho) = F(t) + G(g) + \langle \rho, A t + B g \rangle + \frac{\beta}{2} \|A t + B g\|_2^2, \quad (4.56)$$

where  $\rho = (\rho_1; \rho_2) \in \mathbb{R}^{n+2m}$ ,  $\rho_1 \in \mathbb{R}^n$ ,  $\rho_2 \in \mathbb{R}^{2m}$ , is the vector of Lagrange multipliers associated to the linear constraint in (4.52) and  $\beta \in \mathbb{R}_+^*$  is the ADMM penalty parameter.

Solving problem (4.52) amounts to seek the saddle point(s) of the augmented Lagrangian function  $\mathcal{L}_\beta$  in (4.56), namely points which are simultaneously minimizers in the primal (joint) variable  $(t, g)$  and maximizers in the dual variable  $\rho$ . According to the standard two-blocks ADMM approach, saddle points of  $\mathcal{L}_\beta$  can be computed as limit points of the following iterative procedure:

$$t^{(\ell+1)} \in \arg \min_{t \in \mathbb{R}^n} \mathcal{L}_\beta(t, g^{(\ell)}, \rho^{(\ell)}) , \quad (4.57)$$

$$g^{(\ell+1)} \in \arg \min_{g \in \mathbb{R}^{n+2m}} \mathcal{L}_\beta(t^{(\ell+1)}, g, \rho^{(\ell)}) , \quad (4.58)$$

$$\rho^{(\ell+1)} = \rho^{(\ell)} + \beta (A t^{(\ell+1)} + B g^{(\ell+1)}) . \quad (4.59)$$

In the following Subsections 4.3.3.1, 4.3.3.2 and 4.3.3.3 we detail how to solve the subproblem for variable  $t$  in (4.57) and, separately, the subproblems for variables  $g_1$  and  $g_2$  in (4.58), respectively. In fact, as it will be shown at the beginning of Subsection 4.3.3.2, the optimization problem for variable  $g = (g_1; g_2)$  in (4.58) is separable in the sub-variables  $g_1$  and  $g_2$ .

#### 4.3.3.1 Solving ADMM subproblem for primal variable $t$

Recalling definition (4.56) of the augmented Lagrangian function  $\mathcal{L}_\beta$  and definitions of function  $F$  in (4.53) and of matrix  $B$  in (4.55), after dropping constant terms the  $t$ -subproblem in (4.57) reads

$$\begin{aligned} t^{(\ell+1)} &= \arg \min_{t \in \mathbb{R}^n} \left\{ \iota_{\mathcal{O}_2^{(k)}}(t) + \frac{\beta}{2} \left\| A t + \left( B g^{(\ell)} + \frac{1}{\beta} \rho^{(\ell)} \right) \right\|_2^2 \right\} \\ &= \arg \min_{t \in \mathcal{O}_2^{(k)}} \left\{ f^T A^T A t - 2 t^T A^T \underbrace{\left( g^{(\ell)} - \frac{1}{\beta} \rho^{(\ell)} \right)}_{q^{(\ell)}} \right\} . \end{aligned} \quad (4.60)$$

Then, recalling the definition of matrix  $A$  in (4.55), we have

$$D := A^T A = I_n + S^T S = \text{diag}(1 + v_1, 1 + v_2, \dots, 1 + v_n) ,$$

where the last equality follows easily from property (4.51) of the binary selection matrix  $S$ . Matrix  $D$  is diagonal and positive definite, hence any power  $D^p$  with  $p \in \mathbb{R}$  is diagonal and positive definite. Problem (4.60) is thus equivalent to

$$t^{(\ell+1)} = \arg \min_{t \in \mathcal{O}_2^{(k)}} \left\{ t^T D t - 2 t^T D^{1/2} \overbrace{\left( D^{-1/2} A^T q^{(\ell)} \right)}^{z^{(\ell)}} \right\} \quad (4.61)$$

$$= \arg \min_{t \in \mathcal{O}_2^{(k)}} \left\| D^{1/2} t - z^{(\ell)} \right\|_2^2 . \quad (4.62)$$

We now introduce the one-to-one change of variable

$$u = D^{1/2}t \iff t = D^{-1/2}u,$$

so that problem (4.62) can be equivalently re-written as

$$t^{(\ell+1)} = D^{-1/2}u^{(\ell+1)}, \quad \text{with } u^{(\ell+1)} = \arg \min_{u \in \mathcal{U}^{(k)}} \|u - z^{(\ell)}\|_2^2, \quad (4.63)$$

where the transformed version  $\mathcal{U}^{(k)}$  (i.e., for the new variable  $u$ ) of the linear orthogonality constraint manifold  $\mathcal{O}_2^{(k)}$  is also linear and reads

$$\mathcal{U}^{(k)} = \left\{ u \in \mathcal{F}_V : (Z^{(k)})^T u = 0, \quad Z^{(k)} = D^{-1/2} \tilde{\Phi}^{(k)} \right\}. \quad (4.64)$$

Hence, solution vector  $u^{(\ell+1)}$  in (4.63) is nothing other than the orthogonal projection of vector  $z^{(\ell)}$  onto the linear set in (4.64). Recalling Proposition 4.3.1, in particular equation (4.44), we have

$$u^{(\ell+1)} = \text{proj}_{\mathcal{U}^{(k)}}(z^{(\ell)}) = P^{(k)} z^{(\ell)} = \left( I_n - Z^{(k)} \left( (Z^{(k)})^T Z^{(k)} \right)^{-1} (Z^{(k)})^T \right) z^{(\ell)}.$$

Finally, recalling that  $t^{(\ell+1)} = D^{-1/2}u^{(\ell+1)}$  and the definition of vector  $z^{(\ell)}$  in (4.61), we have

$$\begin{aligned} t^{(\ell+1)} &= D^{-1/2} \left( I_n - Z^{(k)} \left( (Z^{(k)})^T Z^{(k)} \right)^{-1} (Z^{(k)})^T \right) D^{-1/2} h^{(l)} \\ &\quad \underbrace{\hspace{10em}}_{+n = 2n(k+1) + k \times k} \\ &\quad \underbrace{\hspace{10em}}_{+n} \\ &\quad \underbrace{\hspace{10em}}_{+n \times k} \\ &\quad \underbrace{\hspace{10em}}_{+k \times k} \\ &\quad \underbrace{\hspace{10em}}_{n \times k} \\ &= D^{-1} \left\{ h^{(l)} - \tilde{\Phi}^{(k)} \left[ \left( (Z^{(k)})^T Z^{(k)} \right)^{-1} \left( \overbrace{(\tilde{\Phi}^{(k)})^T h^{(l)}}^{n \times k} \right) \right] \right\}, \end{aligned} \quad (4.65)$$

with vector  $h^{(l)}$  given by

$$h^{(l)} = D^{1/2} z^{(l)} = A^T q^{(\ell)} = g_1^{(\ell)} - \frac{1}{\beta} \rho_1^{(\ell)} + S^T \left( g_2^{(\ell)} - \frac{1}{\beta} \rho_2^{(\ell)} \right). \quad (4.66)$$

In (4.65) we have inserted curly brackets in order to better highlight the sequence of macro-operations to be performed, from right to left, to calculate the solution  $t^{(\ell+1)}$



as efficiently as possible. Moreover, above the horizontal braces we reported the number of elementary floating point operations (FLOPS) necessary to execute each macro-operation. Therefore, as written above the last curly bracket, thanks to the change of variable introduced in Section 4.3.1, the orthogonal projection of any vector in  $\mathbb{R}^n$  on the transformed, linear  $p$ -orthogonality manifold constraint  $\tilde{O}_2^{(k)}$  in (4.38) not only admits an explicit solution, but costs only  $2n(k+1) + k^2$  FLOPS - or, in other terms, has computational complexity  $O(kn)$ , where commonly  $k \ll n$ .

In the particular but not rare case of a regular valence graph, for which the valence of all vertices is the same  $\bar{v} = v_i \ \forall i = 1, \dots, n$ , the explicit solution formula in (4.65) simplifies considerably. In fact, we have

$$D = (\bar{v}+1)I_n \implies Z^{(k)} = \frac{1}{\sqrt{\bar{v}+1}}\tilde{\Phi}^{(k)} \implies (Z^{(k)})^T Z^{(k)} = \frac{1}{\bar{v}+1} \text{diag} \left( \|\tilde{f}^{(1)}\|_2^2, \dots, \|\tilde{f}^{(k)}\|_2^2 \right),$$

which leads (after some simple manipulations) to the solution formula

$$t^{(\ell+1)} = \frac{1}{\bar{v}+1} \left( h^{(l)} - \left( \Psi^{(k)} \left( \overbrace{\left( \overbrace{\left( \Psi^{(k)} \right)^T h^{(l)}}^{n \times k} \right)^T}^{n \times k} \right) \right) \right), \quad (4.67)$$

with vector  $h^{(l)}$  defined in (4.66) and matrix  $\Psi^{(k)} \in \mathbb{R}^{n \times k}$  defined by

$$\Psi^{(k)} = \tilde{\Phi}^{(k)} \text{diag} \left( \|\tilde{f}^{(1)}\|_2^{-1}, \dots, \|\tilde{f}^{(k)}\|_2^{-1} \right) = \left( \frac{\tilde{f}^{(1)}}{\|\tilde{f}^{(1)}\|_2} ; \dots ; \frac{\tilde{f}^{(k)}}{\|\tilde{f}^{(k)}\|_2} \right).$$

Like all matrices in the general projection formula (4.65), also matrix  $\Psi^{(k)}$  in (4.67) can be calculated before starting the ADMM iterations. Hence, the computational cost of the simplified projection in (4.67) amounts to  $2n(k+1)$  FLOPS.

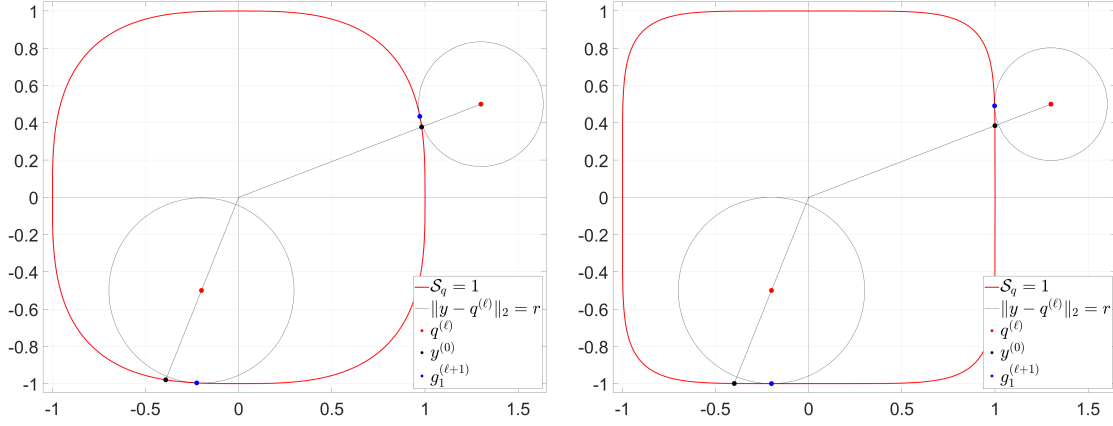


Figure 4.2: 2D representation of  $g_1$ -subproblem in (4.70), for  $q = 3$  (left) and  $q = 6$  (right). In each figure, the problem is visualized simultaneously for vector  $q^{(l)}$  placed inside and outside the  $q$ -hypersphere.

#### 4.3.3.2 Solving ADMM subproblem for the primal variable $g_1$

Recalling the definition of  $\mathcal{L}_\beta$  in (4.56), dropping the constant terms and carrying out some simple algebraic manipulations, the subproblem for variable  $g$  in (4.58) reads

$$\begin{aligned} g^{(\ell+1)} &\in \arg \min_{g \in \mathbb{R}^{n+2m}} \left\{ G(g) + \frac{\beta}{2} \left\| Bg + \left( At^{(\ell+1)} + \frac{1}{\beta} \rho^{(\ell)} \right) \right\|_2^2 \right\} \\ &= \arg \min_{g \in \mathbb{R}^{n+2m}} \left\{ G_1(g_1) + \frac{\beta}{2} \left\| g_1 - \left( t^{(\ell+1)} + \frac{1}{\beta} \rho_1^{(\ell)} \right) \right\|_2^2 \right. \end{aligned} \quad (4.68)$$

$$\left. + G_2(g_2) + \frac{\beta}{2} \left\| g_2 - \left( St^{(\ell+1)} + \frac{1}{\beta} \rho_2^{(\ell)} \right) \right\|_2^2 \right\}, \quad (4.69)$$

with functions  $G_1$  and  $G_2$  defined in (4.54). It follows from (4.68)-(4.69) that the subproblem for variable  $g$  is separable into two independent subproblems for  $g_1$  and  $g_2$ . In the remaining part of this section, we deal with the  $g_1$  subproblem, while in the next section with the  $g_2$  subproblem.

Starting from (4.68), the  $g_1$  subproblem can be compactly written as the projection

$$g_1^{(\ell+1)} \in \arg \min_{g_1 \in S_q} \frac{1}{2} \|g_1 - q^{(\ell)}\|_2^2, \quad \text{with } q^{(\ell)} = t^{(\ell+1)} + \frac{1}{\beta} \rho_1^{(\ell)}. \quad (4.70)$$

Being  $p, q$  Hölder conjugates, with  $q = p/(p-1)$ , if  $p \in (1, 2]$ , then  $q \geq 2$ . When  $q = 2$ , problem (4.70) reduces to the standard Euclidean projection onto the 2-hypersphere, which admits explicit solution  $g_1^{(\ell+1)} = q^{(\ell)} / \|q^{(\ell)}\|_2$  [123]. For  $q > 2$ , the problem consists

---

**Algorithm 3** Proposed  $\mathcal{S}_q$ -GPM algorithm for solving (4.70) by projecting on  $\mathcal{S}_q$ 


---

1. **Input:**  $q^{(\ell)} \in \mathbb{R}^n$  vector to project, step-size  $\alpha > 0$
  2. **Output:**  $g_1^{(\ell+1)} \in \mathbb{R}^n$  approximate solution to (4.70)
  3. **Initialize:**  $\kappa = 0$ ,  $y^{(0)} = q^{(\ell)} / \|q^{(\ell)}\|_q \in \mathcal{S}_q$
  4. **While the stopping criterion is not satisfied**
    5.     • **compute gradient**  $g^{(\kappa)} = \nabla \left( \frac{1}{2} \|y - q^{(\ell)}\|_2^2 \right) (y^{(\kappa)}) = y^{(\kappa)} - q^{(\ell)} \in \mathbb{R}^n$
    6.     • **project**  $d^{(\kappa)} = \text{proj}_{T_{y^{(\kappa)}} \mathcal{S}_q} (g^{(\kappa)}) = \left( I_n - \frac{\psi_p(y^{(\kappa)}) \psi_p(y^{(\kappa)})^T}{\|\psi_p(y^{(\kappa)})\|_2^2} \right) g^{(\kappa)} \in T_{y^{(\kappa)}} \mathcal{S}_q$
    7.     • **descend**  $y^{(\kappa+1/2)} = y^{(\kappa)} - \alpha d^{(\kappa)} \in T_{y^{(\kappa)}} \mathcal{S}_q$
    8.     • **retract**  $y^{(\kappa+1)} = \text{Retr}_{y^{(\kappa)}} (y^{(\kappa+1/2)}) = y^{(\kappa+1/2)} / \|y^{(\kappa+1/2)}\|_q \in \mathcal{S}_q$
    9.     • **update**  $\kappa = \kappa + 1$
  10. **end while**
  11.  $g_1^{(\ell+1)} = y^{(\kappa)}$
- 

of a projection onto a generic  $q$ -hypersphere, as shown in Fig. 4.2 by means of a simplified 2D representation. In [132] the projection is found by solving a nonlinear equation, but the method is limited to the case  $\|q^{(\ell)}\|_q > 1$  (the vector to project is outside the  $q$ -hypersphere).

For the general case  $q > 2$  with generic vector  $q^{(\ell)} \in \mathbb{R}^n$ , we propose to find an approximate solution  $g_1^{(\ell+1)}$  of (4.70) by applying a Projected Gradient Method on the  $q$ -hypersphere manifold  $\mathcal{S}_q$ , that we refer as  $\mathcal{S}_q$ -PGM algorithm, as outlined in Algorithm 3 (note that the iteration index of the algorithm is indicated by  $\kappa$ , to distinguish it from the ADMM iteration index  $\ell$ , and the iterated variable is  $y^{(\kappa)}$ ). This method requires only suitable definitions of the projection operator onto the tangent space  $T_{y^{(\kappa)}} \mathcal{S}_q$  and the retraction operator onto  $\mathcal{S}_q$ , both detailed in Algorithm 3. We highlight that the explicit formula of the projection operator in line 6 of Alg. 2 relies on the fact that the normal vector to  $\mathcal{S}_q$  at the point  $y$  is simply given by  $\psi_p(y)$ . The algorithm iterates until one of the two following conditions is met:

$$\frac{\|y^{(\kappa+1)} - y^{(\kappa)}\|_2}{\|y^{(\kappa)}\|_2} < \text{Tol}, \quad \kappa > \text{Nits}.$$

Results on the linear convergence of the GPM are proved for arbitrary closed sets in [8].

### 4.3.3.3 Solving ADMM subproblem for the primal variable $g_2$

Recalling (4.68)-(4.69) and the definition of function  $G_2$  in (4.54), the subproblem for  $g_2$  reads

$$\begin{aligned} g_2^{(\ell+1)} &\in \arg \min_{g_2 \in \mathbb{R}^{2m}} \left\{ \sum_i^m w_i^{p/2} \left| |g_{2,i}|^r \text{sign}(g_{2,i}) - |g_{2,i+m}|^r \text{sign}(g_{2,i+m}) \right|^p + \frac{\beta}{2} \left\| g_2 - s^{(\ell)} \right\|_2^2 \right\}, \\ &= \arg \min_{g_2 \in \mathbb{R}^{2m}} \sum_i^m \left\{ w_i^{p/2} \left| |g_{2,i}|^r \text{sign}(g_{2,i}) - |g_{2,i+m}|^r \text{sign}(g_{2,i+m}) \right|^p \right. \\ &\quad \left. + \frac{\beta}{2} \left[ \left( g_{2,i} - s_i^{(\ell)} \right)^2 + \left( g_{2,i+m} - s_{i+m}^{(\ell)} \right)^2 \right] \right\}, \end{aligned} \quad (4.71)$$

with

$$p \in (1, 2), \quad r = \frac{1}{p-1} \in (2, +\infty), \quad s^{(\ell)} = St^{(\ell+1)} + \frac{1}{\beta} \rho_2^{(\ell)} \in \mathbb{R}^{2m}. \quad (4.72)$$

The  $2m$ -dimensional optimization problem (4.71)-(4.72) is clearly equivalent to  $m$  independent 2-dimensional problems all having the same structure and reading

$$\begin{aligned} \{ g_{2,i}^{(\ell+1)}, g_{2,i+m}^{(\ell+1)} \} &\in \arg \min_{(x_1, x_2) \in \mathbb{R}^2} \left\{ \overbrace{\left| |x_1|^r \text{sign}(x_1) - |x_2|^r \text{sign}(x_2) \right|^p}^{H_p(x_1, x_2)} \right. \\ &\quad \left. + \frac{\gamma_i}{2} \left[ \left( x_1 - s_i^{(\ell)} \right)^2 + \left( x_2 - s_{i+m}^{(\ell)} \right)^2 \right] \right\} \end{aligned} \quad (4.73)$$

$$= \text{prox}_{H_p}^{\gamma_i} \left( s_i^{(\ell)}, s_{i+m}^{(\ell)} \right), \quad \gamma_i = \frac{\beta}{w_i^{p/2}}, \quad i = 1, \dots, m, \quad (4.74)$$

where  $\text{prox}_{H_p}^{\gamma_i}$  denotes the bivariate proximal operator of function  $H_p$  with proximity parameter  $\gamma$ . To start analyzing the solution to a generic bivariate problem in (4.74), first we introduce the following partition of  $\mathbb{R}^2$ ,

$$\mathbb{R}^2 = O \cup \left( \cup_{i=1}^8 h_i \right) \cup \left( \cup_{i=1}^8 \mathcal{H}_i \right),$$

with  $O$  the origin,  $h_1, \dots, h_8$  the 8 open half-lines with origin  $O$  delimiting the 8 open half-quadrants  $\mathcal{H}_1, \dots, \mathcal{H}_8$ , as represented in Figure 4.3(a). We also introduce the projection

$$T : \mathbb{R}^2 \rightarrow \overline{\mathcal{H}}_{81}, \quad T(v) = \begin{cases} v & \text{for } v \in \overline{H}_{81} := O \cup h_8 \cup h_1 \cup h_2 \cup \mathcal{H}_8 \cup \mathcal{H}_1 \\ -v & \text{for } v \in \tilde{H}_{45} := h_4 \cup h_5 \cup h_6 \cup \mathcal{H}_4 \cup \mathcal{H}_5 \\ B_2 v & \text{for } v \in H_{23} := h_3 \cup \mathcal{H}_2 \cup \mathcal{H}_3 \\ -B_2 v & \text{for } v \in H_{67} := h_7 \cup \mathcal{H}_6 \cup \mathcal{H}_7 \end{cases}, \quad (4.75)$$

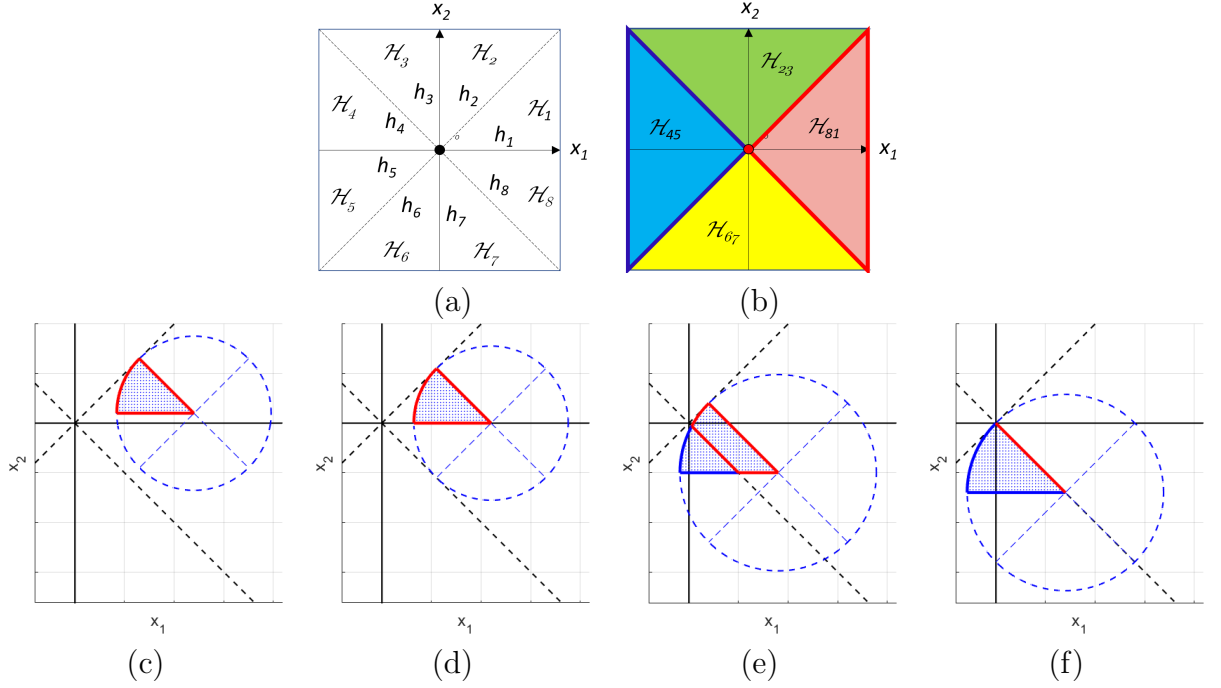


Figure 4.3: Useful partition of  $\mathbb{R}^2$  (a) and the four subsets considered by the projection  $T$  in (4.75) (b). Closed subsets  $\overline{\mathcal{H}}_{81} \cap \Lambda(\tilde{v})$  (red-bordered) to which the proximal values must belong depending only on the position of the vector  $\tilde{v}$  to approximate (c)-(f), which corresponds to the center of the depicted circles.

with  $B_2 \in \mathbb{R}^{2 \times 2}$  the permutation (backward identity) matrix exchanging the entries of  $v$  and where the four subsets  $\overline{\mathcal{H}}_{81}, \overline{\mathcal{H}}_{23}, \overline{\mathcal{H}}_{45}, \overline{\mathcal{H}}_{67}$  are visualized with different colors in Figure 4.3(b). It is easy to verify that, thanks to symmetries of function  $H_p$  - in particular,  $H_p$  is symmetric with respect to the origin and to the two bisectors of the quadrants - the solution to a problem in (4.74) can be restricted to the case  $(s_i^{(\ell)}, s_{i+m}^{(\ell)}) \in \overline{\mathcal{H}}_{81}$ .

In the following Proposition 4.3.2 we show that, after preliminarily applying the projection in (4.75), the solution(s) of the proximal map in (4.74) must belong to well-defined 1-dimensional arcs belonging to the subset  $\overline{\mathcal{H}}_{81}$ .

**Proposition 4.3.2.** *Let  $p \in (1, 2)$  and let  $H_p : \mathbb{R}^2 \rightarrow \mathbb{R}_+$  be the function defined in (4.73). Then, the proximity operator of function  $H_p$  with proximity parameter  $\gamma \in \mathbb{R}_+^*$  calculated at vector  $v \in \mathbb{R}^2$  is the (possibly) set-valued function  $\text{prox}_{H_p}^\gamma : \mathbb{R}^2 \rightrightarrows \mathbb{R}^2$  defined by*

$$\hat{z}(v; p, \gamma) \in \text{prox}_{H_p}^\gamma(v) := \arg \min_{z \in \mathbb{R}^2} \left\{ E_p^\gamma(z; v) := H_p(z) + \frac{\gamma}{2} \|z - v\|_2^2 \right\}. \quad (4.76)$$

Problem (4.76) admits solution(s) given by

$$\hat{z}(v; p, \gamma) = T(\hat{z}(\tilde{v}; p, \gamma)) , \quad \tilde{v} = T(v) \in \overline{H}_{18} , \quad (4.77)$$

with the map  $T : \mathbb{R}^2 \rightarrow \mathbb{R}^2$  and the closed set  $\overline{H}_{18} \subset \mathbb{R}^2$  defined in (4.75) and with

$$\hat{z}(\tilde{v}; p, \gamma) \in \arg \min_{z \in \mathcal{A}_p(\tilde{v})} E_p^\gamma(\tilde{z}; p, \gamma) ,$$

where

$$\mathcal{A}_p(\tilde{v}) = \begin{cases} \tilde{v} & \text{if } \tilde{v} \in O \cup h_2 \\ O\tilde{v} & \text{if } \tilde{v} \in h_8 \\ \mathcal{H}_{81} \cap \Lambda(\tilde{v}) \cap \mathcal{C}_p(\tilde{v}) & \text{if } \tilde{v} \in \overline{\mathcal{H}}_{81} \setminus (O \cup h_2 \cup h_8) \end{cases} ,$$

with

$$\Lambda(\tilde{v}) = \{z \in \mathbb{R}^2 : \|z - \tilde{v}\|_2 \leq d(\tilde{v}, h_1) \wedge \tilde{v}_2 \leq z_2 \leq \tilde{v}_1 + \tilde{v}_2 - z_1\} , \quad (4.78)$$

$$\mathcal{C}_p(\tilde{v}) = \{z \in \mathbb{R}^2 : z_1^{r-1}(z_2 - \tilde{v}_2) = |z_2|^{r-1}(\tilde{v}_1 - z_1)\} . \quad (4.79)$$

*Proof.* The function  $H_p$  in (4.73) is clearly symmetric with respect to the origin and to both the bisectors of the quadrants, in formula,

$$H_p(z) = H_p(-z) = H_p(B_2 z) = H_p(-B_2 z) , \quad \forall z \in \mathbb{R}^2 , \quad (4.80)$$

with  $B_2 \in \mathbb{R}^{2 \times 2}$  the backward identity matrix.

Then, it follows easily from its definition in (4.76) that, for any  $p \in (1, 2)$ , any  $\gamma \in \mathbb{R}_+^*$  and any  $v \in \mathbb{R}^2$ , the function  $E_p^\gamma$  is continuous on  $\mathbb{R}$ , bounded below by zero and also coercive since it is the sum of the bounded below function  $H_p$  and a quadratic coercive function. Hence,  $E_p^\gamma$  admits global minimizers and the proximity operator in (4.76) is always defined. Moreover, based on symmetries in (4.80), function  $E_p^\gamma$  also satisfies

$$E_p^\gamma(z; v) = E_p^\gamma(-z; -v) = E_p^\gamma(B_2 z; B_2 v) = E_p^\gamma(-B_2 z; -B_2 v) , \quad \forall z, v \in \mathbb{R}^2 .$$

The last three equalities imply as many properties of the proximity operator in (4.76),

$$\hat{z}(v; p, \gamma) = -\hat{z}(-v; p, \gamma) = B_2 \hat{z}(B_2 v; p, \gamma) = -B_2 \hat{z}(-B_2 v; p, \gamma) , \quad \forall z, v \in \mathbb{R}^2 .$$

It follows from the last expression that the proximal map in (4.76) can be equivalently re-written in terms of an analogous proximal map but with input vector  $v = \tilde{v}$  suitably restricted to the set  $\overline{H}_{81}$ , closure of the union of half-quadrants 1 and 8 (see Figure 4.3(b)), as formalized in (4.77).

Then, it is quite easy to demonstrate (we omit the proof for shortness) that, for any vector  $\tilde{v} \in \overline{\mathcal{H}}_{81}$ , the corresponding proximal value(s)  $\hat{z}(\tilde{v}; p, \gamma)$  must belong to the closed set  $\overline{\mathcal{H}}_{81} \cap \Lambda(\tilde{v})$ , given by the intersection between  $\overline{\mathcal{H}}_{81}$  and the eighth of a closed circle  $\Lambda(\tilde{v})$  defined in (4.78). The circle has its center in  $\tilde{v}$  and is tangent to the bisector of

the first quadrant  $h_1$ . This intersection set is shown in Figure 4.3(c)-(f) for four different positions of vector  $\tilde{v}$  (red-bordered regions).

In order to reduce the candidate solutions set from dimension 2 to dimension 1 or 0 - i.e., to curves/arcs or points - we analyze the gradient of the cost function  $E_p^\gamma(z; \tilde{v})$  in (4.76) for  $z \in H_{81}$ . First, we note that for  $z \in H_{81}$  - where  $z_1 > 0$  and  $|z_2| < z_1$  - the function  $H_p(z)$  in (4.73) simplifies to

$$H_p(z) = (z_1^r - z_2 |z_2|^{r-1})^p.$$

Then, the gradient of  $E_p^\gamma$  clearly reads

$$\nabla E_p^\gamma(z; \tilde{v}) = \nabla H_p(z) + \gamma(z - \tilde{v}), \quad (4.81)$$

with

$$\nabla H_p(z) = \begin{bmatrix} \frac{\partial H_p(z)}{\partial z_1} \\ \frac{\partial H_p(z)}{\partial z_2} \end{bmatrix} = p r (z_1^r - z_2 |z_2|^{r-1})^{p-1} \begin{bmatrix} z_1^{r-1} \\ -|z_2|^{r-1} \end{bmatrix}.$$

The two components of  $\nabla H_p$  are both continuous (note that  $r - 1 > 0$  and  $p - 1 > 0$ ) for any  $z \in H_{81}$ , hence  $\nabla E_p^\gamma$  in (4.81) is also continuous. It follows that a necessary condition for a point  $z \in H_{81}$  to be a proximal value of  $\tilde{z} \in H_{81}$  is that it is a stationary point of  $E_p^\gamma$ , that is

$$\nabla E_p^\gamma(z; \tilde{v}) = 0_2 \iff \begin{cases} \delta (z_1^r - z_2 |z_2|^{r-1})^{p-1} z_1^{r-1} = \tilde{v}_1 - z_1, \\ \delta (z_1^r - z_2 |z_2|^{r-1})^{p-1} |z_2|^{r-1} = z_2 - \tilde{v}_2, \end{cases} \quad \delta = \frac{p r}{\gamma}. \quad (4.82)$$

After noting that, for  $\tilde{v} \in H_{81} \implies \tilde{v}_1 > 0$ , the first equation in (4.82) can not be satisfied for  $z_1 = 0$ , we can divide the equation by  $z_1$  and then substitute the left-hand side into the second equation:

$$\begin{cases} \delta (z_1^r - z_2 |z_2|^{r-1})^{p-1} = \frac{\tilde{v}_1 - z_1}{z_1^{r-1}} \\ \delta (z_1^r - z_2 |z_2|^{r-1})^{p-1} |z_2|^{r-1} = z_2 - \tilde{v}_2 \end{cases} \implies z_1^{r-1} (z_2 - \tilde{v}_2) = |z_2|^{r-1} (\tilde{v}_1 - z_1),$$

which corresponds to the curve  $\mathcal{C}_p(\tilde{v})$  in (4.79). This concludes the proof.  $\square$

Finally, in order to compute the proximal values, we suitably parameterize the arc  $\mathcal{A}_p(\tilde{v})$  and, then, we solve iteratively the associated 1-dimensional optimization problem.

The main computational steps of the proposed M-ADMM approach are summarized in Algorithm 4. The M-ADMM iterations are stopped in the same way as the M-PGD algorithm, i.e., as soon as one of the two criteria in (4.49) is satisfied. We observe experimentally that the method converges globally independently of the initialization. However, the theoretical rate of convergence for M-ADMM algorithms is a problem that has yet to be addressed.

---

**Algorithm 4** M-ADMM algorithm for solving any  $k$ -th optimization problem in (4.43)

---

1. **Input:**  $\tilde{f}^{(1)}, \dots, \tilde{f}^{(k)} \in \mathbb{R}^n$  computed (orthogonal) eigenfunctions, penalty  $\beta > 0$
  2. **Output:**  $\tilde{f}^{(k+1)}$  new (orthogonal) eigenfunction, approximate solution to (4.43)
  3. **Initialize:**  $\ell = 0$ ,  $t^{(0)} = t_0$ , compute  $\tilde{\Psi}_i^{(k)} \in \mathbb{R}^n$ ,  $i = 1, \dots, k$ , by (4.48)
  4. **While** the stopping criterion is not satisfied
  5.     • compute  $t^{(\ell+1)}$  by (4.65)-(4.66) or (4.66)-(4.67)
  6.     • compute  $g_1^{(\ell+1)}$  by Algorithm 3
  7.     • compute  $g_2^{(\ell+1)}$  by solving (4.74), based on Proposition 4.3.2
  8.     • compute  $\rho^{(\ell+1)}$  by (4.59)
  9.     • update  $\ell = \ell + 1$
  10. **end while**
  11.  $\tilde{f}^{(k+1)} = t^{(\ell)}$
- 

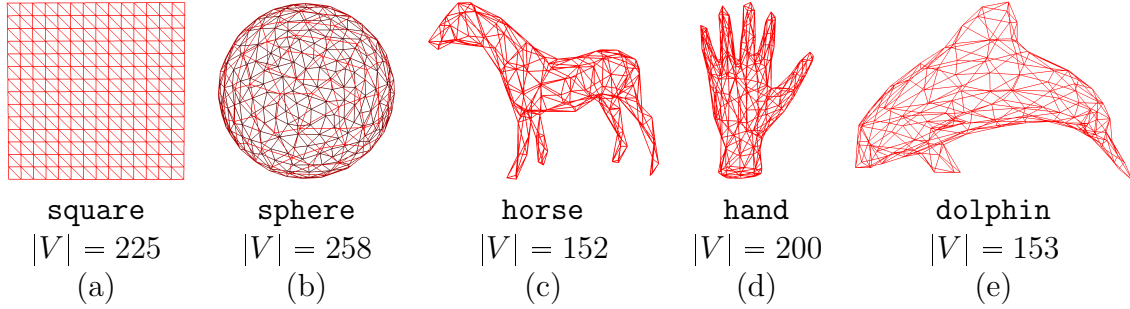


Figure 4.4: Test meshes and associated number of vertices  $n = |V|$ .

## 4.4 Numerical Results

In this section, we numerically test the performance of the proposed algorithms M-PGD and M-ADMM for the computation of a few eigenpairs of the graph  $p$ -Laplacian, with the main focus on the case  $p \in (1, 2]$ . We address both the case where the domain is a regular triangulated grid in  $\mathbb{R}^2$ , as illustrated in Fig.4.4(a), and the case involving more irregular mesh domains in  $\mathbb{R}^3$ , such as those depicted in Fig.4.4(b)-(e).

To the best of our knowledge, there are no other numerical algorithms in literature for the computation of more than two eigenpairs of the  $p$ -Laplacian which satisfy the original nonlinear eigendecomposition problem (4.13), i.e. the  $p$ -orthogonality constraint on the eigenfunctions. This motivated us, in the first example described in Section 4.4.1, to restrict the computation to the 2-Laplacian eigendecomposition in order to validate both the goodness of the proposed algorithms, and the computation of the estimated eigenfunctions  $f^{(k)}$ ,  $k = 2, 3, \dots$ . This allowed us to compare the obtained results with those obtained by standard eigendecomposition linear algebra methods. The second



example, illustrated in Section 4.4.2, extends the validation of the  $p$ -Laplacian eigendecomposition to different  $p$  values and compares the performance of the two proposed approaches. The third example in Section 4.4.3 is devoted to a possible application of the  $p$ -Laplacian eigendecomposition which aims to cluster a 3D mesh into meaningful parts.

For all the reported tests, given the computed eigenfunction  $f^{(k)}$ , the estimation of the associated eigenvalue  $\lambda_{\text{OLS}}^{(k)}$  is determined by the procedure of orthogonal linear regression explicitly derived in (4.35)-(4.36). Therefore, the validation of the goodness in eigenvalue computation can be evaluated by the Root Mean Square Residual (RMSR), which represents a goodness-of-fit measure, and reads as follows

$$\text{RMSR}^{(k)} := \sqrt{\frac{1}{n} \frac{S_{xx}^{(k)} (\lambda_{\text{OLS}}^{(k)})^2 - 2 S_{xy}^{(k)} \lambda_{\text{OLS}}^{(k)} + S_{yy}^{(k)}}{(\lambda_{\text{OLS}}^{(k)})^2 + 1}},$$

where  $S_{xx}^{(k)}, S_{yy}^{(k)}, S_{xy}^{(k)}$  are defined as in (4.36). Low values of RMSR indicate that the linear relationship is verified with a good approximation, that is  $(f^{(k)}, \lambda_{\text{OLS}}^{(k)})$  is an actual eigenpair of the  $p$ -Laplacian.

We will denote by  $\lambda_{R_p}^{(k)}$  the estimation of the eigenvalue associated to a computed eigenfunction  $f^{(k)}$  according to the definition of Rayleigh quotient  $\lambda_{R_p} = R_p(f^{(k)})$ , see Prop. 4.1.2 and (4.33).

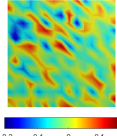
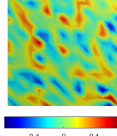
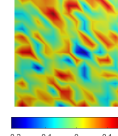
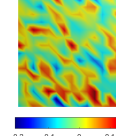
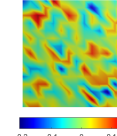
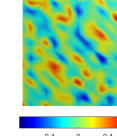
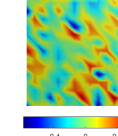
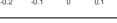
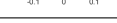
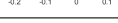
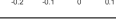
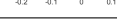
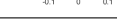
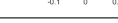
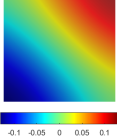
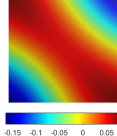
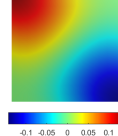
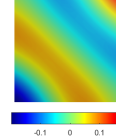
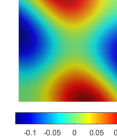
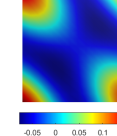
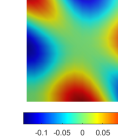
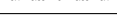
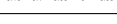
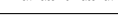
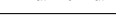
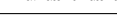
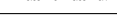
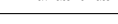
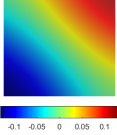
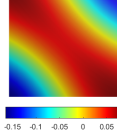
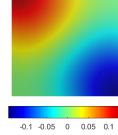
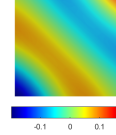
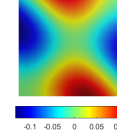
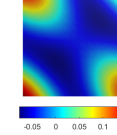
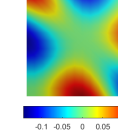







In order to measure how much, at each step  $k$  of our approaches, corresponding to the eigenfunction  $f^{(k)}$ , the  $p$ -orthogonality constraint is satisfied globally by the first  $k$  estimated eigenfunctions  $f^{(1)}, \dots, f^{(k)}$ , we introduce and compute the following scalar measure of  $p$ -orthogonality,

$$\text{pOrth}^{(k)} := \max_{i,j=1,\dots,k, j < i} |\langle \psi_p(f^{(i)}), \psi_p(f^{(j)}) \rangle|.$$

In the implementation of the M-PGD method we set the step-length in the range  $\alpha \in [10^{-4}, 10^{-1}]$ , manually tuned according to the problem, in order to achieve convergence. The maximum number of iterations is set to be  $\text{Nits} = 5 \times 10^5$ . As the numerical convergence has been always satisfied, we avoided the backtracking line search. In the M-ADMM algorithm, we set  $\beta = 5$  and  $\text{Nits} = 10^4$ .

#### 4.4.1 Example 1: computing 2-Laplacian eigenpairs

The two proposed algorithmic frameworks for the  $p$ -Laplacian eigendecomposition are subject to a preliminary sanity test to ensure that the proposed algorithms work as expected. This is carried out on the computation of the first eight eigenpairs of the 2-Laplacian on the grid domain **square** (see Fig.4.4(a)), by comparing the results from M-PGD and M-ADMM with the eigenpairs obtained by the standard Lancsoz bidiagonalization iterative method [72] - using the MatLab **eigs** function.

		$f^{(2)}$	$f^{(3)}$	$f^{(4)}$	$f^{(5)}$	$f^{(6)}$	$f^{(7)}$	$f^{(8)}$
INIT.								
								
GT								
								
OUR								
								
M-PGD	$\lambda_{GT}$	0.0463	0.1114	0.1143	0.2361	0.3067	0.3294	0.4410
	$\lambda_{R_p}$	0.0463	0.1114	0.1143	0.2361	0.3067	0.3294	0.4410
	$\lambda_{OLS}$	0.0463	0.1114	0.1143	0.2361	0.3067	0.3294	0.4410
	RMSR	0	1.51e-10	1.23e-10	2.41e-10	0	0	0
	pOrth	6.84e-16	4.54e-16	3.83e-16	8.67e-17	1.26e-16	3.68e-16	1.57e-16
M-ADMM	$\lambda_{R_p}$	0.0463	0.1114	0.1143	0.2361	0.3067	0.3294	0.4410
	$\lambda_{OLS}$	0.0463	0.1114	0.1143	0.2361	0.3067	0.3294	0.4410
	RMSR	0	0	0	1.70e-10	3.35e-10	5.27e-10	7.18e-10
	pOrth	1.79e-16	1.00e-16	4.67e-14	6.15e-17	1.27e-16	1.22e-16	8.53e-16

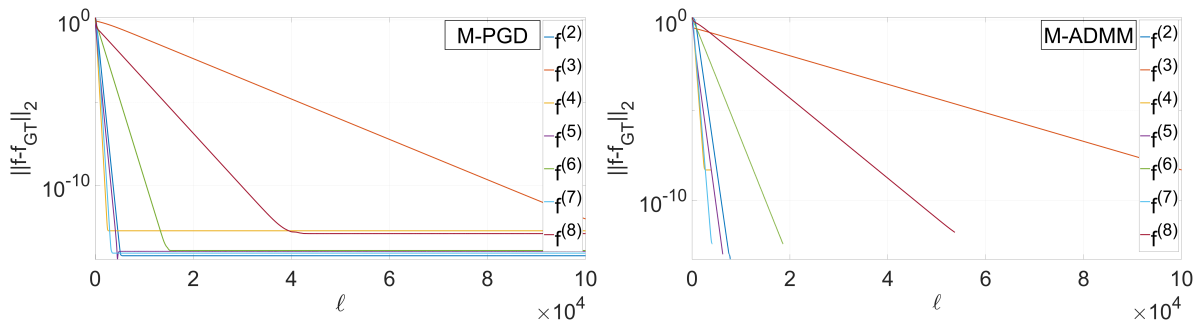


Figure 4.5: Example 1 - 2-Laplacian eigenpairs. Top: random initializations (INIT.), Ground truth eigenfunctions (GT), and our results (OUR). Middle: eigenvalues associated with the computed eigenfunctions using M-PGD and M-ADMM, and error metrics. Bottom: plot of  $\ell_2$ -norm error for  $\ell$  iterations of the M-PGD (left) and M-ADMM (right) approaches.

In Fig. 4.5, column-wise, we show the random initializations, the expected ground truth (GT) eigenfunctions, and our results (OUR). We reported only the M-PGD eigenfunctions as the M-ADMM results are indistinguishable. Visually, we note that, as expected, these eigenfunctions behaves like the Laplacian eigenfunctions in the continuous setting, i.e. like trigonometric functions, with regular and increasing number of oscillations. The bottom part of Fig. 4.5 reports the plots of the  $\ell_2$ -norm error  $\|f^{(k)} - f_{GT}^{(k)}\|_2$  of the eigenfunctions on the  $n$  vertices of the domain, for M-PGD (left) and M-ADMM (right). The rapid decay of the error along the iterations  $\ell$  confirms that the models correctly compute the eigenfunctions. This is further confirmed by observing the results reported in the middle part in Fig. 4.5. The computed eigenvalues  $\lambda_{R_p}, \lambda_{OLS}$  perfectly match the ground truth  $\lambda_{GT}$  value, computed via the `eig` function. Moreover, the two metrics RMSR and  $p$ -orthogonality `p Orth`, both vanishing for all the eigenfunctions and for both methods. As expected,  $p$ -orthogonality is satisfied and the tests validated both the proposed M-PGD and M-ADMM which correctly find the known eigenpairs of the 2-Laplacian.

#### 4.4.2 Example 2: computing $p$ -Laplacian eigenpairs

In the second example, we demonstrate how the proposed algorithms perform in computing eigenpairs of  $p$ -Laplacian with  $p \in \{1.8, 1.5, 1.2\}$ .

In Fig. 4.6 (top) we show the eigenfunctions of the  $p$ -Laplacian on the planar **square** domain obtained via M-PGD (indistinguishable results are obtained using M-ADMM). As  $p$  decreases, the shape of the eigenfunctions tends to be sharper with piecewise-constant regions, while reducing the transition parts. Similar behavior is visually observable in Fig. 4.7 for the **sphere** domain. As  $p$  decreases, the eigenfunctions more clearly partition the domain into constant regions. This behavior resembles the effects of Total Variation regularization (which involves an  $L^1$  penalization term), providing sparse restoration and piecewise-constant functions.

To get a quantitative insight into the ability of the two methods to achieve their results efficiently, in Fig. 4.6(bottom) we plot the evolution along the iterations of the  $\text{RMSR}^{(k)}$  metric associated with the three eigenfunctions computed. We recall that low values of  $\text{RMSR}^{(k)}$  indicate good satisfaction of the eigenvalue equation. Note that the scale of the horizontal (iterations) axis is quite different in the two graphs, and the M-PGD algorithm, as expected, requires more iterations to get a stationary evolution (and, hence, to fulfill the common stopping criterion). It can be seen from the plots that M-ADMM not only is able to reach lower RMSR values, but also a smaller number of iterations is required to achieve a given level of the RMSR metric.

These visual/qualitative observations are confirmed by the quantitative results shown in Table 4.1(top) for the **square** domain and, analogously, in Table 4.2(top) for the **sphere** domain, concerning the behavior of the two methods at convergence (according to the corresponding stopping criteria). In particular, in these sub-tables we report, for

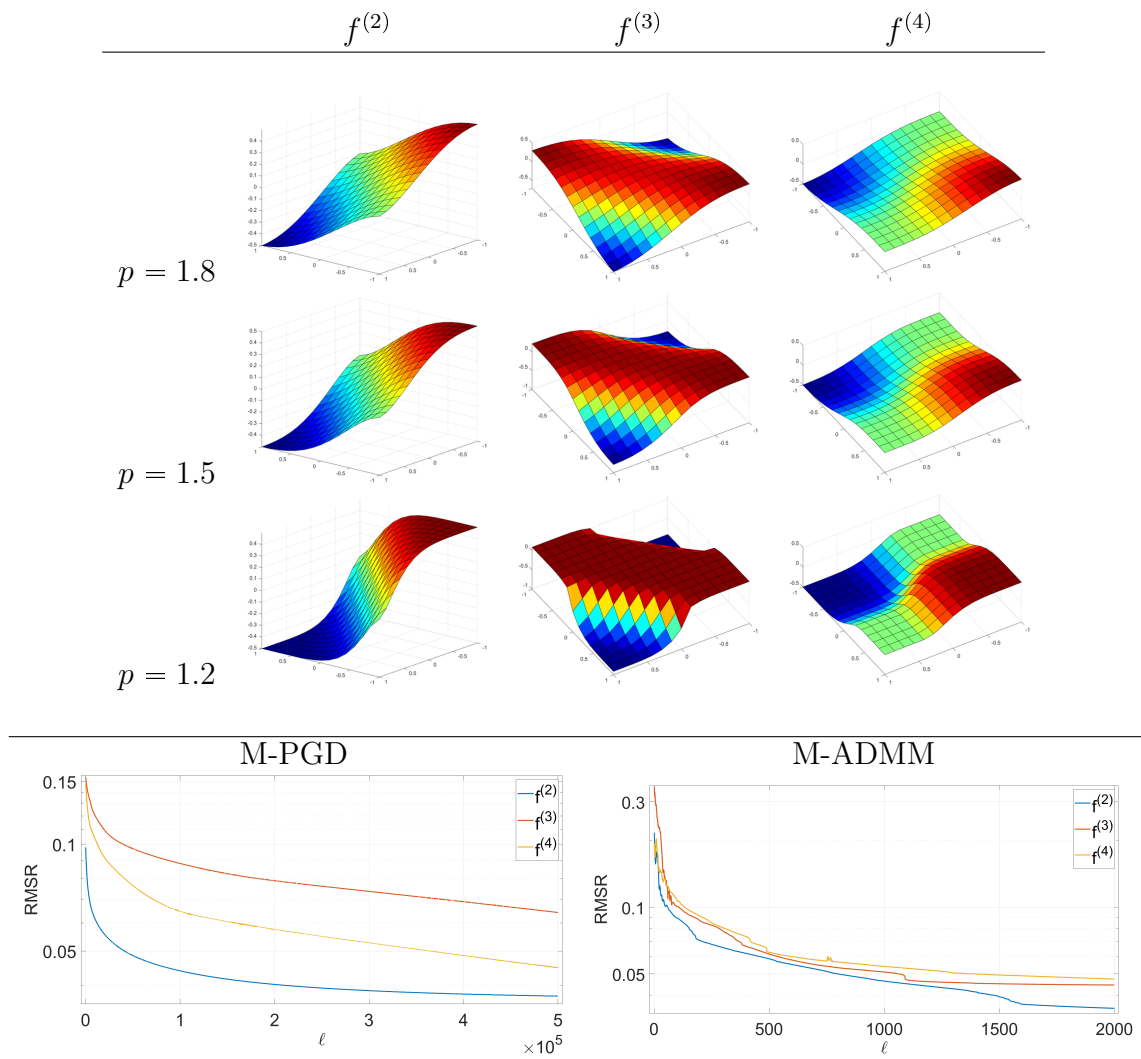


Figure 4.6: Example 2 - square domain. Top: visualization of three  $p$ -Laplacian eigenfunctions, for  $p = \{1.8, 1.5, 1.2\}$ . Bottom: evolution of the RMSR values for  $p = 1.2$ , applying the two methods, in terms of the number  $\ell$  of iterations.

	M-PGD			M-ADMM		
	$f^{(2)}$	$f^{(3)}$	$f^{(4)}$	$f^{(2)}$	$f^{(3)}$	$f^{(4)}$
$p = 1.8$						
$\lambda_{\text{OLS}}$	0.0702935	0.152202	0.15915	0.0706232	0.152895	0.159909
$\lambda_{R_p}$	0.0707049	0.153215	0.160195	0.0706232	0.152901	0.159909
RMSR	4.62e-4	1.04e-3	1.11e-3	<b>8.27e-8</b>	<b>7.89e-5</b>	<b>1.12e-6</b>
pOrth	8.86e-16	2.04e-16	9.33e-17	1.45e-16	1.86e-15	1.92e-09
$\ell$	1e5	1e5	1e5	1571	1447	1374
$p = 1.5$						
$\lambda_{\text{OLS}}$	0.125676	0.232494	0.250887	0.129046	0.238016	0.321386
$\lambda_{R_p}$	0.129981	0.241249	0.261726	0.129046	0.238181	0.272558
RMSR	3.57e-3	6.39e-3	1.02e-2	<b>1.31e-4</b>	<b>5.50e-6</b>	<b>9.46e-4</b>
pOrth	9.37e-17	1.47e-16	5.16e-17	1.84e-16	1.60e-11	1.84e-07
$\ell$	1e5	1e5	30681	1398	5e3	1802
$p = 1.2$						
$\lambda_{\text{OLS}}$	0.214728	0.351384	0.375077	0.220319	0.34469	0.396112
$\lambda_{R_p}$	0.220989	0.359384	0.398689	0.220311	0.356279	0.393743
RMSR	3.73e-2	6.42e-2	4.47e-2	<b>2.28e-3</b>	<b>3.94e-2</b>	<b>2.62e-2</b>
pOrth	1.02e-15	4.31e-16	1.75e-16	2.44e-16	2.23e-11	1.17e-08
$\ell$	5e5	5e5	5e5	1793	5e3	5e3

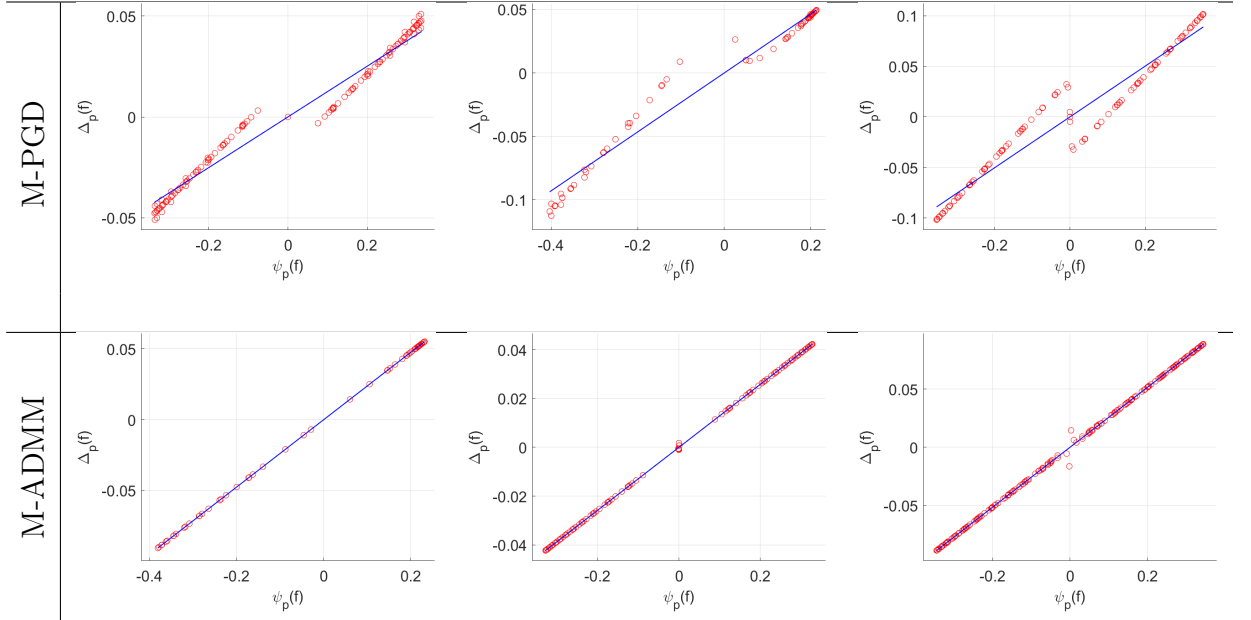


Table 4.1: Example 2 - square domain. Top: results for different  $p$  values with M-PGD and M-ADMM. Bottom: orthogonal regression fitting line (solid blue) and data points (red circles) defined in (4.34), for  $p = 1.5$ .

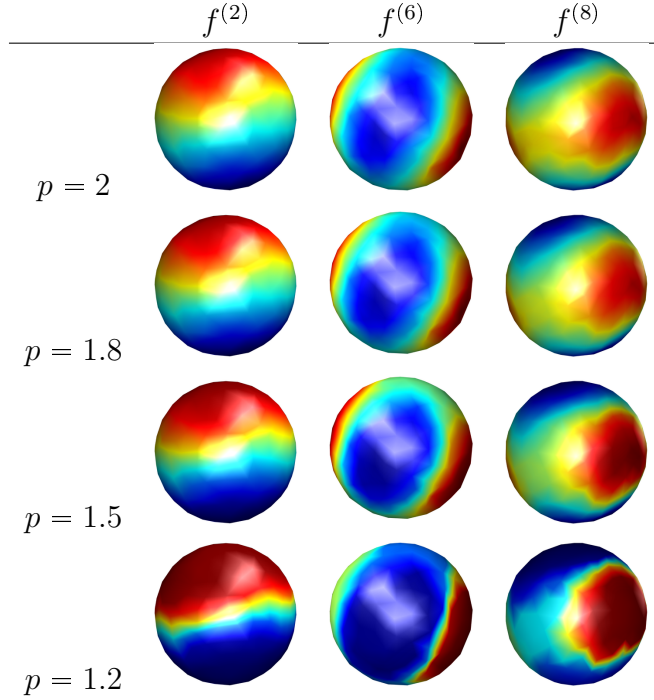


Figure 4.7: Example 2 - **sphere** domain. Visualization of three  $p$ -Laplacian eigenfunctions, for  $p = \{2, 1.8, 1.5, 1.2\}$ .

each eigenpair computed, from top to bottom, the two estimates of the eigenvalues  $\lambda_{\text{OLS}}^{(k)}$  and  $\lambda_{R_p}^{(k)}$ , the  $\text{RMSR}^{(k)}$  values - in boldface the lowest among the two values - and the  $\text{pOrth}^{(k)}$  values (in the table we drop the  $^{(k)}$  superscript for shortness of notation). First, we note that the two estimates  $\lambda_{\text{OLS}}^{(k)}$  and  $\lambda_{R_p}^{(k)}$  provide very similar results and that the  $p$ -orthogonality metric  $\text{pOrth}^{(k)}$  is always very small, namely less than  $10^{-6}$  and often in the order of  $10^{-15} - 10^{-16}$ . Then, the  $\text{RMSR}^{(k)}$  values seem to indicate clearly that the M-ADMM approach has the potential to provide higher-quality estimates of the  $p$ -eigenfunctions, quite uniformly with respect to the value of  $p$ .

Finally, in the plots reported in Table 4.1(bottom) for the **square** domain and, analogously, in Table 4.2(bottom) for the **sphere** domain, we visualize, for both approaches and for the case  $p = 1.5$ , the Orthogonal Least Squares fitting line (solid blue line) together with the associated data points  $(\psi_p(f^{(k)}), \Delta_p(f^{(k)}))$  (red circles) for the same three eigenfunctions considered in the top part of the tables (namely,  $k = 2, 3, 4$  for **square**,  $k = 2, 6, 8$  for the **sphere** domain). Note that the angular coefficients of the fitting lines shown coincide with the values  $\lambda_{\text{OLS}}^{(k)}$  reported in the top part of the tables for  $p = 1.5$ . The better fit obtained by M-ADMM with respect to the M-PGD is thus well assessed not only from the RMSR values in the top tables but also from these line-fitting graphs in the bottom tables.

	M-PGD			M-ADMM		
	$f^{(2)}$	$f^{(6)}$	$f^{(8)}$	$f^{(2)}$	$f^{(6)}$	$f^{(8)}$
$p = 1.8$						
$\lambda_{\text{OLS}}$	0.215205	0.565198	0.625266	0.215883	0.567494	0.627531
$\lambda_{R_p}$	0.21605	0.567618	0.62815	0.215879	0.567275	0.62683
RMSR	1.38e-3	3.74e-3	4.28e-3	<b>4.22e-4</b>	<b>2.24e-3</b>	<b>2.87e-3</b>
$p\text{Orth}$	1.77e-15	1.54e-15	1.66e-15	3.88e-16	2.91e-7	5.21e-7
$p = 1.5$						
$\lambda_{\text{OLS}}$	0.311952	0.736322	0.790814	0.315339	0.718229	0.785175
$\lambda_{R_p}$	0.317441	0.734007	0.802354	0.315302	0.716472	0.784318
RMSR	1.46e-2	4.14e-2	3.21e-2	<b>1.59e-3</b>	<b>6.89e-3</b>	<b>6.52e-3</b>
$p\text{Orth}$	1.16e-15	2.34e-15	3.13e-15	2.77e-16	2.72e-7	6.10e-7
$p = 1.2$						
$\lambda_{\text{OLS}}$	0.415395	0.903748	1.05252	0.41965	0.848454	0.955792
$\lambda_{R_p}$	0.420532	0.860773	0.950283	0.41854	0.844079	0.922103
RMSR	5.11e-2	1.32e-1	1.62e-1	<b>2.77e-2</b>	<b>4.88e-2</b>	<b>8.67e-2</b>
$p\text{Orth}$	1.83e-15	8.59e-16	1.36e-15	6.10e-16	6.64e-7	6.79e-7

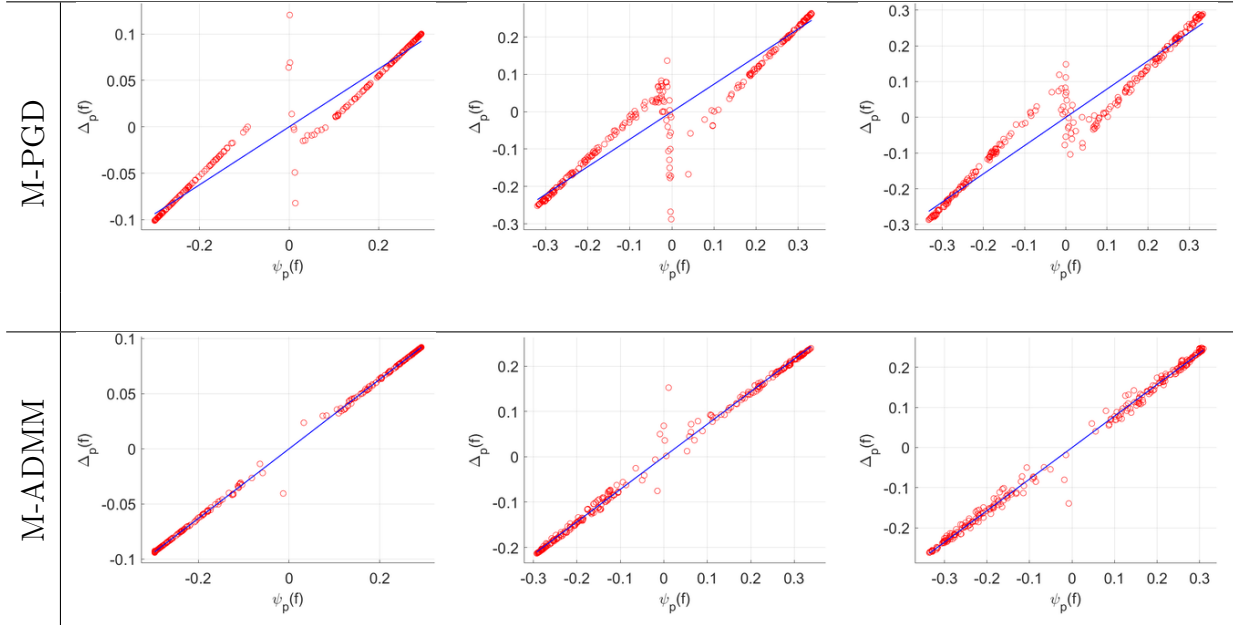


Table 4.2: Example 2 - **sphere** domain. Top: results for different  $p$  values with M-PGD (after 5e5 iterations) and M-ADMM (after 1e3 iterations). Bottom: orthogonal regression fitting line (solid blue) and data points (red circles) defined in (4.34), for  $p = 1.5$ .

### 4.4.3 Example 3: approach to the spectral clustering

As the proposed method does not depend on the discretization of the domain, it can be easily adapted to free-form domains, as the 3D meshes `horse`, `dolphin` and `hand`, see Fig.4.4(c)-(e). We demonstrate it with the following eigendecomposition examples, computing the  $p$ -Laplacian eigenpairs, in the classical case  $p = 2$ , in the case  $p = 1.2$  (as a partial approach to the limit case  $p = 1$ ) and in the opposite direction of  $p = 3$ . Some eigenfunctions, among the first 14, are depicted in Fig.4.8; for each 3D mesh, the three rows report, from top to bottom, the eigenfunctions of the graph  $p$ -Laplacian for  $p = 3$ ,  $p = 2$  and  $p = 1.2$ . From left to right, the eigenfunctions are associated with eigenvalues of increasing magnitude. In analogy with the behavior of the 2-eigenvalues which represent graph frequencies, small values are associated with low frequencies and large values with higher frequencies. Low frequencies correspond to smooth and slowly varying functions and represent macroscopic shape information; while the high frequencies correspond to eigenfunctions with rapid oscillations, and generally describe microscopic (details) behavior. These preliminary results highlight how the non-linearity with  $p \in (1, 2)$ , in particular  $p$  near 1, better captures the underlying geometry of the data. In particular, eigenfunctions not only emphasize the different curvatures areas, but also they divide them (see the legs of the `horse` mesh in Fig.4.8). This deserves a more in-depth analysis in a future work.

A possible application of the computed  $p$ -Laplacian eigendecomposition comes naturally from the visual analysis of the geometric properties of these eigenfunctions: each should correspond to object parts and protrusions [30, 60]. This could easily be exploited for spectral clustering or mesh segmentation algorithms based on the Principal Component Analysis interpretation of the  $p$ -Laplacian eigenvectors of the mesh.



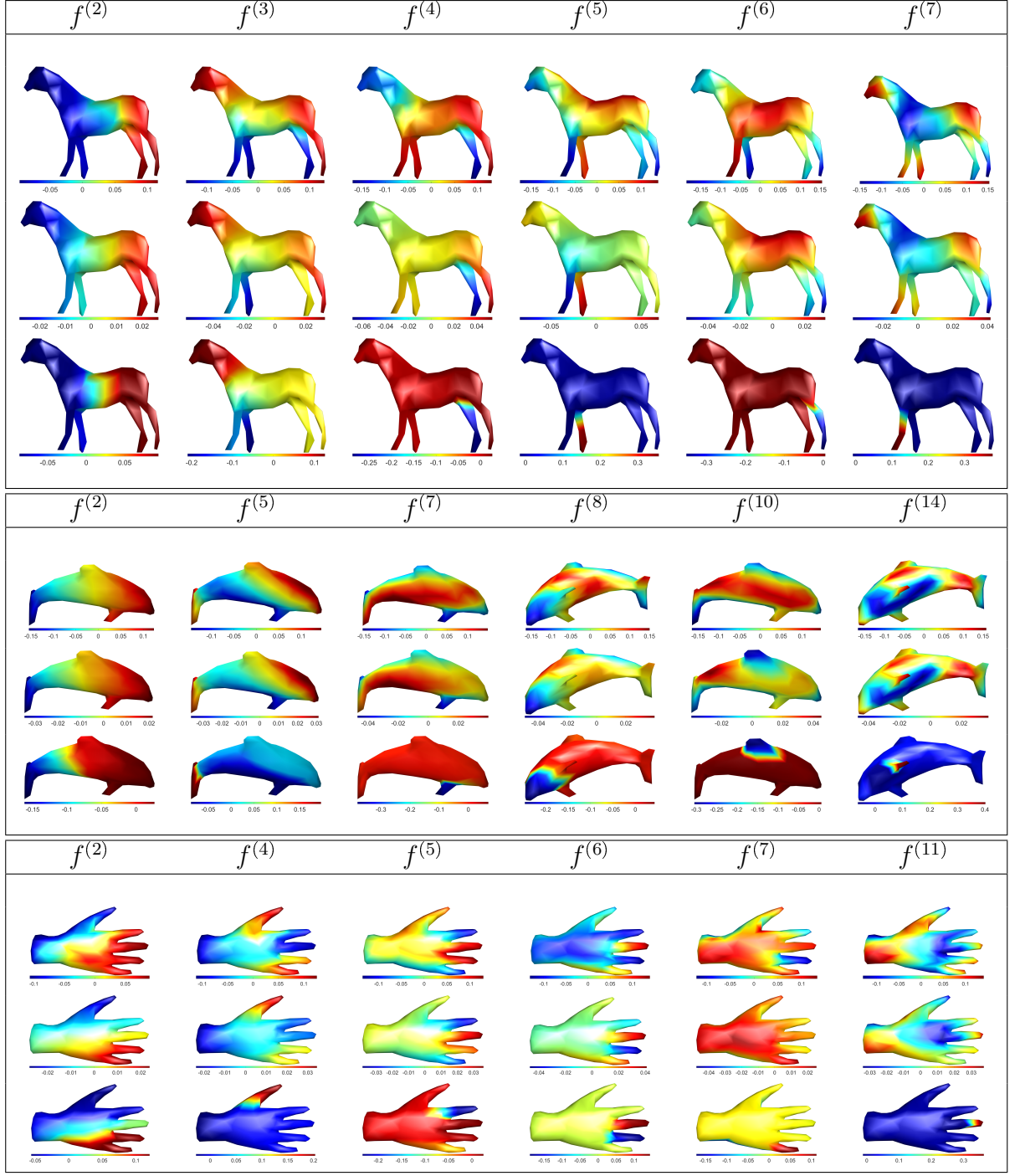


Figure 4.8: Example 3 - Selected  $p$ -Laplacian eigenfunctions on 3D meshes horse, dolphin, hand. Rows 1,4,7:  $p = 3$ . Rows 2,5,8:  $p = 2$ . Rows 3,6,9:  $p = 1.2$ .

# Chapter 5

## Linear PDE models for graph osmotic flow

The osmotic flow, introduced in Sec.1.4, can be formulated as the following linear isotropic diffusion-transport PDE

$$\partial_t u(x, t) = \Delta u(x, t) - \operatorname{div}(\mathbf{d}(x)u(x, t)). \quad (5.1)$$

Starting from initial data  $u_0$ , the evolution of a point  $x \in \Omega$  driven by the model in Eq. (5.1), depends mainly on the local differential properties of the data near the point. This behavior is not always accurate enough to model physical phenomena or to perform data processing tasks.

For this reason, we are interested in defining and studying a non-local version of the osmosis model. Non-local models capture interactions or dependencies that occur over larger distances and are better suited to describe phenomena in fields such as material science, geophysics and network analysis. For example, in diffusion processes happening in biology or population dynamics, changes at one point can quickly affect others non-locally. Furthermore, non-local models are often more robust when dealing with discontinuities or singularities and numerically more stable.

A natural way to consider long-range interactions in the model definition consists in replacing the (local) differential operators with corresponding (non-local) integral operators, integrating over a wider domain, specified by a kernel function  $W$ . Such kernel functions identify not only the 'active' region that affects function behavior on a point  $x$ , but also the specific weight or influence of its points  $y$ . For this reason, the kernel  $W$  is often defined in terms of the distance  $\|x - y\|_2^2$  between the two points, for example as a Gaussian kernel  $W(d) = \exp(-d^2/\sigma^2)$  with  $\sigma > 0$  a positive parameter.

As the kernel support decreases, it is expected that the data evolution using the non-local model gets similar to the one obtained via the local formulation. This can only happen if the integral operators are accurately defined as approximations of the differential operators.

For a detailed overview and analysis of non-local diffusion problems, we refer to [3],

where the authors consider models such as the diffusion flow (1.50) or the  $p$ -Laplacian flow (1.56), in  $\mathbb{R}^n$  as a Cauchy problem or in bounded domain  $\Omega \subset \mathbb{R}^n$  with Dirichlet or Neumann boundary condition. Furthermore, they study a particular case of the osmosis equation (5.1), with  $\mathbf{d}$  constant in space.

The definition of the non-local model provides useful suggestions on how to derive an appropriate discretization of the corresponding local PDE model on graph domains  $G$ . In fact, the edge weights  $w_{ij}$  take the role of the kernel functions  $W$ , measuring the influence of a neighbor vertex  $v_j$  in the evolution of the data on the vertex  $v_i$ .

In this Chapter, we provide a precise formalization of these observations. In particular, in Sec.5.1 we report the original definition of the osmosis model [129], along with its main properties and discretization on image domains, used to perform a variety of image editing tasks. In Sec.5.2, we analyze theoretically the linear local model, in terms of wellposedness and regularity of the solutions. The same inquiry is described in Sec.5.3 for our proposed non-local model. In Sec.5.4, we report results and conjectures about the non-local model consistency, i.e. the convergence of non-local solutions to local solutions. Then, in Sec.5.5, we derive a graph discretization of the model, tested in Sec.5.6 to verify the behavior of the model as an editing tool for color functions defined on meshes.

## 5.1 Linear model: properties and application in image editing

The osmosis model was first studied in [129] for rectangular domains  $\Omega \subset \mathbb{R}^2$ . The original formulation, involving an initial data  $u_0$  and homogeneous Neumann boundary condition, reads as follows:

$$\begin{cases} \partial_t u = \Delta u - \operatorname{div}(\mathbf{d}u) & \text{on } \Omega \times (0, T], \\ \langle \nabla u - \mathbf{d}u, \mathbf{n} \rangle = 0 & \text{on } \partial\Omega \times (0, T]; \\ u(x, 0) = u_0(x) > 0 & \text{on } \Omega, \end{cases} \quad (5.2)$$

with  $\mathbf{d} : \Omega \rightarrow \mathbb{R}^2$  and  $\mathbf{n}$  is the outer normal vector to the image boundary  $\partial\Omega$ .

In [129], the authors proved that the evolution process (5.2) preserves the average intensity (grey value) of the image  $f$  as well as its non-negativity, thanks to its divergence form. Moreover, if the drift  $\mathbf{d}$  is defined in a canonical form, i.e. in terms of a given reference image  $v : \Omega \rightarrow \mathbb{R}_+^*$  as:

$$\mathbf{d} := \nabla \log v, \quad (5.3)$$

then the steady-state solution  $w : \Omega \rightarrow \mathbb{R}_+$  is a multiplicative rescaling of  $v$ , that is  $w = cv$ , with  $c > 0$  [129]. Furthermore, analogous properties hold upon suitable finite-difference discretization [124] or operator splitting [24]. The osmosis convergence

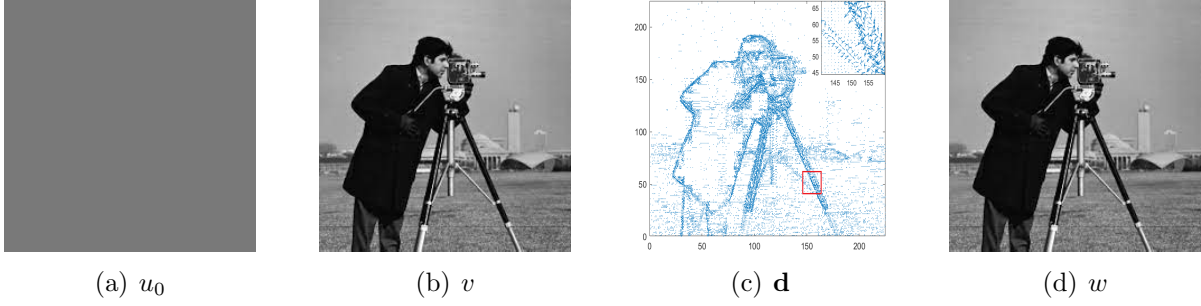


Figure 5.1: Evolution of the osmosis model (5.2). (a) (constant) initial image  $u_0$ , (b) reference image  $v$ , (c) vector representation of drift  $\mathbf{d}$  (with zoom), (d) steady state  $w$ .

property is shown in Fig. 5.1, where the PDE model, defined by a drift term  $\mathbf{d}$  expressed in canonical form from a reference image  $v$ , evolves from a constant initial data  $u_0$  to the steady-state  $w$ .

There is of course no clear interest in observing convergence to a rescaled version  $w$  of an image which is needed in advance to define the drift  $\mathbf{d}$  required for the modeling. However, setting different definitions of  $\mathbf{d}$  in a disjoint partition  $\Omega_{in} \cup \Omega_{out} \cup \Omega_b$  of  $\Omega$ , with  $\Omega_b$  suitably thick, osmosis can be applied to many different imaging tasks such as shadow/light-spot removal, compact data representation and image cloning.

**Shadow/light-spot removal.** For shadow/light-spot removal problems, we assume that the given image  $f_0$  is characterized by the presence of a region  $S \subset \Omega$  with different lighting, see Fig. 5.2(a). The mask  $M$  is thus associated with a preliminary detection of an appropriate region boundary  $\Omega_b$  (for instance, by using [33, 7, 32]), see Fig.5.2(b). Setting

$$\mathbf{d}(x) := \left( \nabla(\log v) \chi_{\Omega_b^c} \right) (x) = \begin{cases} \frac{\nabla v(x)}{v(x)} & \text{if } x \in \Omega_{in} \cup \Omega_{out}, \\ 0 & \text{if } x \in \Omega_b, \end{cases}, \quad (5.4)$$

with  $v = u_0$  as reference image and  $\chi_D : \Omega \rightarrow \{0, 1\}$  the characteristic function of the subset  $D$ , the evolution described by the model (5.2) leads to the result in Fig.5.2(c).

Due to its intrinsic transport-diffusion properties, linear osmosis filtering does correctly perform shadow removal by balancing the intensity between the differently lit regions of the image. However, due to the choice (5.4), the drift term  $\mathbf{d}$  vanishes on  $\Omega_b$ , where pure Laplace diffusion is enforced, possibly causing oversmoothing.

**Compact image representation** The task of compact image (or, generally, data) representation consists in representing a given image with as little information as possible using some limited, but significant, image content (see, e.g. [28], [85]). Osmosis is able to reconstruct a greyscale image by initializing the model using the intensity values on

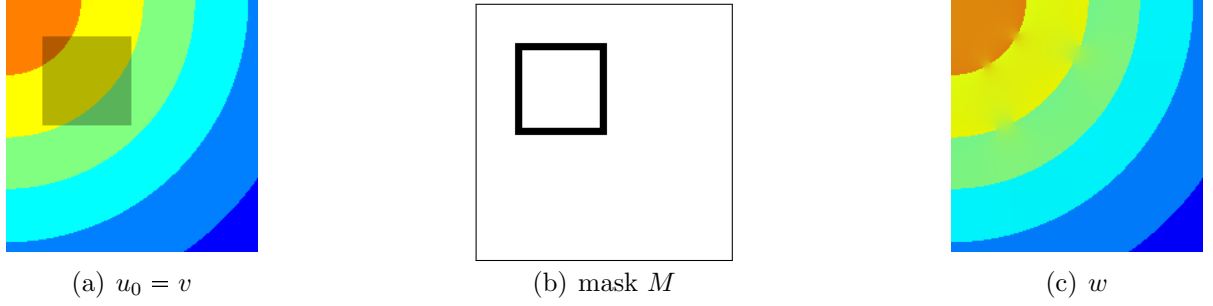


Figure 5.2: Shadow removal using linear (5.2) osmosis filtering.

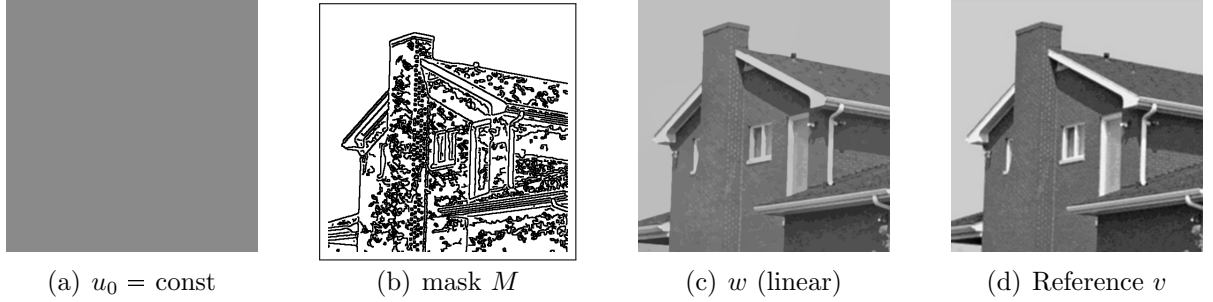


Figure 5.3: Compact data representation using the linear osmosis model.

pre-detected edges which are encoded in the definition of the drift-term  $\mathbf{d}$  so that:

$$\mathbf{d} := \left( \frac{\nabla v}{v} \chi_{\Omega_b} \right) (x) = \begin{cases} \frac{\nabla v(x)}{v(x)} & \text{if } x \in \Omega_b, \\ 0 & \text{if } x \in \Omega \setminus \Omega_b, \end{cases} \quad (5.5)$$

where  $\Omega_b$  denotes here the set of the edges. The task is thus a sort of interpolation process where starting from  $u_0$  (Fig. 5.3(a)) and using only the data assigned on the mask  $M$  (Fig. 5.3(b)) to define the drift term as in (5.5), the osmosis evolution produces the result in Fig. 5.3(c). Fig. 5.3(d) is the ground-truth image, reported here for comparison.

**Image cloning** Given two reference images  $v_1, v_2$  defined on the same domain  $\Omega$ , cloning consists in computing an image  $w$  that merges the information of the two images in a seamless way, around a region  $S \subset \Omega$ , as shown in Fig.5.4, where the result is obtained through osmosis model, setting

$$\mathbf{d} := \begin{cases} \frac{\nabla v_1(x)}{v_1(x)} & \text{if } x \in \Omega_{out}, \\ \frac{\nabla v_2(x)}{v_2(x)} & \text{if } x \in \Omega_{in}, \\ \left( \frac{\nabla v_1(x)}{v_1(x)} + \frac{\nabla v_2(x)}{v_2(x)} \right) / 2 & \text{if } x \in \Omega_b. \end{cases} \quad (5.6)$$

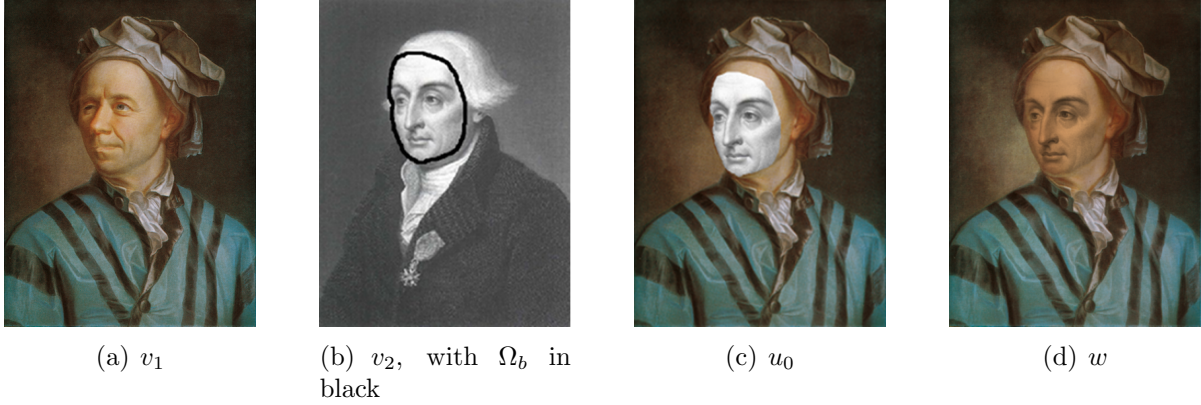


Figure 5.4: Image cloning using linear osmosis model.

With respect to pure diffusion, the osmosis model can integrate mismatching image data in a seamless way (see [92] for a variational osmosis-based model for image fusion showing the advantage with respect to state-of-the-art methods).

## 5.2 Local osmosis model

The osmosis model equation  $u_t = \Delta u - \operatorname{div}(\mathbf{d} u)$  is a particular case of the general second-order parabolic equation. When parabolic equations are associated with homogeneous boundary value constraints, results about existence, uniqueness and regularity properties of weak solutions are well-known. In the following, we report from [44, Chapter 7] those results and we observe how to apply them for the osmosis PDE.

Let  $T > 0$  and  $\Omega \subset \mathbb{R}^n$  open and bounded. We consider the problem

$$\begin{cases} \partial_t u = \Delta u - \operatorname{div}(\mathbf{d} u), & \text{on } \Omega \times [0, T], \\ u \equiv 0, & \text{on } \partial\Omega \times [0, T], \\ u \equiv u_0, & \text{on } \Omega \times \{0\}. \end{cases} \quad (\text{L-Osm})$$

The PDE can be seen as  $\partial_t u = -Lu + h$ , with

$$Lu = - \sum_{i,j=1}^n a^{ij}(x,t) u_{x_i x_j} + \sum_{i=1}^n b_i(x,t) u_{x_i} + c(x,t) u \quad (5.7)$$

$$\text{with } a^{ij}(x,t) = \delta_{ij}, \quad b(x,t) = \mathbf{d}(x), \quad c(x,t) = (\operatorname{div} \mathbf{d})(x), \quad h \equiv 0 \quad (5.8)$$

where the operator  $L$  does not change over time.

The general parabolic equations theory imposes the following assumptions:

$$\begin{cases} a^{ij}, b^i, c \in L^\infty(\Omega \times [0, T]) \\ h \in L^2(\Omega \times [0, T]) \\ u_0 \in L^2(\Omega) \end{cases} \quad \text{i.e., in our case,} \quad \begin{cases} \mathbf{d}, \operatorname{div} \mathbf{d} \in L^\infty(\Omega) \\ u_0 \in L^2(\Omega) \end{cases} \quad (5.9)$$

We observe the existence and uniqueness of a weak solution.

**Notation.** Let  $W_0^{1,2}(\Omega)$  be the closure of  $C_0^\infty(\Omega)$  onto the Sobolev space  $W^{1,2}(\Omega)$  of locally summable functions with first derivatives (in the weak sense) in  $L^2(\Omega)$ . Let  $H^{-1}(\Omega)$  be the dual space of  $W_0^{1,2}(\Omega)$ . We denote as  $H^2(\Omega) = \{u \in W^{2,2}(\Omega), \text{ s.t. } u' \in W^{1,2}(\Omega)\}$ .

**Definition 5.2.1.** A weak solution of (L-Osm) is a function  $u \in L^2(0, T; W_0^{1,2}(\Omega))$ , with  $u' \in L^2(0, T; W^{-1}(\Omega))$  such that

- $\langle u', v \rangle + B[u, v; t] = (h, v)$  for each  $v \in W_0^{1,2}(\Omega)$  and a.e. time  $t \in [0, T]$ ;
- $u(0) = u_0$ .

$$\text{with } B[u, v; t] = \int_{\Omega} \left[ \sum_{i,j=1}^n a^{ij}(\cdot, t) u_{x_i} v_{x_j} + \sum_{i=1}^n b^i(\cdot, t) u_{x_i} v + c(\cdot, t) uv \right] dx.$$

In osmosis case,  $B[u, v; t] = \int_{\Omega} [\sum_{i=1}^n u_{x_i} v_{x_i} + \sum_{i=1}^n \mathbf{d}_i u_{x_i} v + (\operatorname{div} \mathbf{d}) uv] dx$ .

Using Galerkin approximation as in Theorem 3 of [44, Chapter 7], we have:

**Theorem 5.2.1.** Given the assumptions in (5.9), there exists a unique weak solution of (L-Osm).

Recapping, local osmosis PDE with homogeneous Dirichlet boundary conditions is well-posed, if the initial data  $f \in L^2(\Omega)$  and if the drift term and its divergence are bounded.

If we further assume that the drift term  $\mathbf{d}$  and its divergence  $\operatorname{div} \mathbf{d}$  are smooth on  $\overline{\Omega}$  and that  $\Omega$  has a smooth boundary, then we have the following regularity results, as derived from Theorem 5 and Theorem 7 in [44, Chapter 7]:

**Theorem 5.2.2** (Improved regularity). Assume  $u_0 \in W_0^{1,2}(\Omega)$ . If  $u \in L^2(0, T; W_0^{1,2}(\Omega))$ , with  $u' \in L^2(0, T; H^{-1}(\Omega))$  is the weak solution of (L-Osm), then

$$\begin{cases} u \in L^2(0, T; H^2(\Omega)) \cap L^\infty(0, T; W_0^{1,2}(\Omega)), \\ u' \in L^2(0, T; L^2(\Omega)) \end{cases} \quad (5.10)$$

and we have the estimate

$$\operatorname{ess\,sup}_{t \in [0, T]} \|u(t)\|_{W_0^{1,2}(\Omega)} + \|u\|_{L^2(0, T; H^2(\Omega))} + \|u'\|_{L^2(0, T; L^2(\Omega))} \leq C \|f\|_{W_0^{1,2}(\Omega)}.$$

If, in addition,  $u_0 \in H^2(\Omega)$  and  $u'_0 \in L^2(0, T; L^2(\Omega))$ , then

$$\begin{cases} u \in L^\infty(0, T; H^2(\Omega)), \\ u' \in L^\infty(0, T; L^2(\Omega)) \cap L^2(0, T; W_0^{1,2}(\Omega)), \\ u'' \in L^2(0, T; H^{-1}(\Omega)) \end{cases} \quad (5.11)$$

and we have the estimate

$$\operatorname{ess\,sup}_{t \in [0, T]} (\|u(t)\|_{H^2(\Omega)} + \|u'(t)\|_{L^2(\Omega)}) + \|u'\|_{L^2(0, T; W_0^{1,2}(\Omega))} + \|u''\|_{L^2(0, T; H^{-1}(\Omega))} \leq C \|u_0\|_{H^2(\Omega)}.$$

Moreover, further regularity of the initial data leads to further regularity of the solution, as in the following theorem.

**Theorem 5.2.3.** *Assume  $u_0 \in C^\infty(\overline{\Omega})$  and that for all  $m = 1, 2, \dots$  the  $m$ -th order compatibility holds:*

$$\begin{cases} f_0 := u_0 \in W_0^{1,2}(\Omega) \\ f_1 := u(0) - L f_0 \in W_0^{1,2}(\Omega), \\ f_m := \frac{\partial^{m-1} u}{\partial t^{m-1}}(0) - L f_{m-1} \in W_0^{1,2}(\Omega) \end{cases}$$

Then the problem (L-Osm) has a unique solution  $u \in C^\infty(\overline{\Omega} \times [0, T])$ .

### 5.3 Non-local osmosis model

To derive a non-local formulation of the osmosis model, first we need to which points of the domain  $\Omega$  influence the data evolution on a point  $x$ , defining a kernel function  $W$ .

**Assumption 5.3.1.** *We consider  $W : \mathbb{R}^N \rightarrow \mathbb{R}_+ \cup \{0\}$  to be a symmetric, radial, smooth kernel function, with compact support  $D$ . Furthermore, for all  $i, j = 1, \dots, N$  we assume*

$$\int_{\mathbb{R}^N} W^2(z) z_i z_j dz = \delta_{ij} \quad (5.12)$$

The integral condition in (5.12) ensures that  $W(z)$  is isotropic and that its contributions balance properly in all directions, ensuring that the derived integral operators correctly produce diffusion effects. Furthermore, in stochastic or probabilistic interpretations of non-local operators, this condition can arise if  $W(z)$  is related to a covariance function in a Gaussian process.



Now we can define a non-local version of the gradient operator. While the local gradient of a function  $u : \mathbb{R}^N \rightarrow \mathbb{R}$  is a vector field  $\nabla u : \mathbb{R}^N \rightarrow \mathbb{R}^N$ , in the non-local setting the gradient is defined on a pair of points  $(x, y) \in \mathbb{R}^N \times \mathbb{R}^N$  and represents a weighted measure of the change of the function  $u$  between the two points. This concept is formalized as follows:

**Definition 5.3.1.** *The **non-local gradient**  $\nabla^{NL}$  with respect to a kernel function  $W$  of a function  $u : \mathbb{R}^N \rightarrow \mathbb{R}$  is the scalar function  $\nabla^{NL}u : \mathbb{R}^N \times \mathbb{R}^N \rightarrow \mathbb{R}$  defined as*

$$(\nabla^{NL}u)(x, y) = W(x - y)(u(y) - u(x))$$

From the non-local gradient we can derive the definition of the non-local divergence, using the concepts of inner product between scalar functions and of the adjoint property.

**Definition 5.3.2.** *Given two functions  $u, v : \Omega \rightarrow \mathbb{R}$ , their inner product over the domain  $\Omega$  is*

$$\langle u, v \rangle = \int_{\Omega} u(x)v(x) \, dx$$

In the local setting, the differential operators gradient  $\nabla$  and divergence  $\text{div}$  are related by the adjoint property:

$$\langle \nabla u, g \rangle = -\langle u, \text{div} g \rangle$$

for any given scalar field  $u$  and vector field  $g$ . Interpreting this equivalence in non-local sense, one can derive the following definition of non-local divergence for an antisymmetric scalar function.

**Definition 5.3.3.** *The non-local divergence  $\text{div}^{NL}$  with respect to a kernel function  $W$  of an antisymmetric function  $g : \mathbb{R}^N \times \mathbb{R}^N \rightarrow \mathbb{R}$  is the function  $\text{div}^{NL}g : \mathbb{R}^N \rightarrow \mathbb{R}$  defined as*

$$(\text{div}^{NL}g)(x) = 2 \, p.v. \int_{\mathbb{R}^N} W(x - y)g(x, y) \, dy$$

Finally, the Laplacian operator is the composition  $\Delta = \text{div} \circ \nabla$ , interpretable in local or non-local sense.

**Definition 5.3.4.** *The **non-local Laplacian**  $\Delta^{NL}$  of a function  $u : \mathbb{R}^N \rightarrow \mathbb{R}$  is the function  $\Delta^{NL}u : \mathbb{R}^N \rightarrow \mathbb{R}$  defined as*

$$\Delta^{NL}u(x) = 2 \int_{\mathbb{R}^N} W^2(x - y)(u(y) - u(x)) \, dy$$

In order to define the non-local version of the osmosis PDE  $u_t = \Delta u - \text{div}(\mathbf{d}u)$ , we need to understand what relation lies between the local vector field  $\mathbf{d} : \Omega \rightarrow \mathbb{R}^N$  and its non-local counterpart  $d$ . First, we observe that, as for the gradient, in the non-local setting vector fields are transformed into scalar fields, defined over a pair  $(x, y)$ . Furthermore,  $\mathbf{d}$  and  $d$  must have an integral relation that takes into account the interactions between the two points  $x$  and  $y$ . This relation is expressed as follows:

**Definition 5.3.5.** The **non-local drift** function  $d : \mathbb{R}^N \times \mathbb{R}^N \rightarrow \mathbb{R}$  corresponding to a local drift  $\mathbf{d} : \mathbb{R}^N \rightarrow \mathbb{R}^N$  is

$$d(x, y) = W(x - y)\bar{d}(x, y) = W(x - y) \int_0^1 \langle \mathbf{d}(x + t(y - x)), y - x \rangle dt \quad (5.13)$$

The assumption  $\mathbf{d} \in L^\infty(\Omega)$  implies  $d \in L^\infty(\mathbb{R}^N \times \mathbb{R}^N)$ , since  $W \in C_0^\infty(\Omega)$ . Furthermore, the non-local drift is antisymmetric and therefore its non-local divergence is well-defined.

In Sec.5.4, the reason behind this specific definition (5.13) will be more clear. For the moment, we just observe that in the simple 1-dimensional case  $N = 1$ , if  $\mathbf{d}$  is the derivative of some function  $\delta : \mathbb{R} \rightarrow \mathbb{R}$ , then the right-hand-side reduces to  $W(x - y)(\delta(y) - \delta(x))$ , corresponding to its non-local derivative.

We now have all the ingredients to define the non-local version of the osmosis equation.

**Definition 5.3.6.** The **non-local osmosis equation** is

$$\begin{cases} \partial_t u + (A^{NL} + B^{NL})u = 0 & \text{on } \Omega \times [0, T], \\ u \equiv 0 & \text{on } (\mathbb{R}^N \setminus \Omega) \times [0, T], \end{cases} \quad (\text{NL-Osm})$$

with  $A^{NL} = -\Delta^{NL}$  and  $B^{NL}$  representing the divergence term  $B : u \mapsto \text{div}(\mathbf{d}u) = \text{div}(d)u + \langle d, \nabla u \rangle$  defined as

$$\begin{aligned} B^{NL}u &= \left( 2 \int_{\mathbb{R}^N} W(x - y)d(x, y) dy \right) u(x) + \int_{\mathbb{R}^N} W(x - y)(u(y) - u(x))d(x, y)dy = \\ &= \int_{\mathbb{R}^N} W(x - y)d(x, y)(u(y) + u(x)) dy \end{aligned}$$

In Sec.5.2, we have observed that the local model has weak solutions, with  $u(t) \in W^{1,2}(\Omega)$  for all  $t \in [0, T]$ . The non-local model involves integral operators and thus a solution does not need to have weak derivatives in space. For this reason, we first observe how the operators behave when applied to functions in the Hilbert space  $L^2(\Omega)$ .

**Proposition 5.3.1.**

1.  $B^{NL} \in \mathcal{L}(L^2(\Omega))$ , i.e. it is a linear bounded operator, therefore also a Lipschitz operator;
2.  $A^{NL} \in \mathcal{L}(L^2(\Omega))$ , i.e. it is a linear bounded operator, therefore also a Lipschitz operator;
3.  $A^{NL}$  is a maximally monotone operator.

*Proof.* 1) Since  $W \in C_0^\infty(\mathbb{R}^N)$  and  $d \in L^\infty(\mathbb{R}^N \times \mathbb{R}^N)$ , for all  $u \in L^2(\Omega)$  we have:

$$\|B^{NL}u\|_{L^2(\Omega)} \leq \|W\|_{C_0^\infty} 2\|d\|_{L^\infty} \cdot \|u\|_{L^2(\Omega)} \leq C\|u\|_{L^2(\Omega)}$$

i.e.  $B^{NL}$  is bounded. Moreover, since it is clearly linear, it is a Lipschitz operator.

2)  $A^{NL}$  is linear and since  $W \in C_0^\infty(\mathbb{R}^N)$ , for all  $u \in L^2(\Omega)$  we have

$$\|A^{NL}u\|_{L^2(\Omega)} \leq \|W\|_{C_0^\infty} 2\|u\|_{L^2(\Omega)} \leq C\|u\|_{L^2(\Omega)}$$

Therefore, it is also a Lipschitz operator.

3) Proven in [3, Th. 6.7] for the non-local  $p$ -Laplacian. The case  $p = 2$  is straightforward.  $\square$

These properties of the integral operators  $A^{NL}$  and  $B^{NL}$  are useful to prove the wellposedness of the non-local osmosis model.

### 5.3.1 Well-posedness

To show the existence of a solution for the non-local osmosis, we exploit the properties of the operators  $A^{NL} + B^{NL}$ , together with the Cauchy-Lipschitz-Picard theorem [21, Theorem 7.3].

**Proposition 5.3.2** (Existence and Uniqueness in  $C^1$ ). *Let  $A^{NL}$ ,  $B^{NL}$  defined as in Def.5.3.6. Then, given  $u_0 \in L^2(\Omega)$ , the problem (NL-Osm) has a unique solution  $u \in C^1([0, \infty), L^2(\Omega)) \cap C([0, \infty), D(A^{NL} + B^{NL}))$ , where  $D$  is the domain of definition of the operators.*

*Proof.* We begin by proving the existence result, which amounts to finding some  $u \in C([0, \infty), L^2(\Omega))$  satisfying the integral equation

$$u(t) = u_0 + \int_0^t F^{NL}(u(s))ds. \quad (5.14)$$

with  $F^{NL} = -A^{NL} - B^{NL}$ . Given  $k > 0$ , to be fixed later, set

$$E = \left\{ u \in C([0, +\infty); L^2(\Omega)); \sup_{t \geq 0} e^{-kt} \|u(t)\|_{L^2(\Omega)} < \infty \right\}.$$

It is easy to check that  $E$  is a Banach space for the norm

$$\|u\|_E = \sup_{t \geq 0} e^{-kt} \|u(t)\|_{L^2(\Omega)}.$$

In fact, it suffices to establish that  $E$  is complete. To do so, let  $u_p \in E$  be a Cauchy sequence, in other words, for any  $\varepsilon > 0$ , there exists an  $N \in \mathbb{N}$  such that for all  $p, q \geq N$ ,

$$\|u_p - u_q\|_E = \sup_{t \geq 0} e^{-kt} \|u_p(t) - u_q(t)\|_{L^2} < \varepsilon.$$

This implies that for each fixed  $t \geq 0$ , the sequence  $\{u_p(t)\}$  is a Cauchy sequence in  $L^2$ , since  $e^{-kt} > 0$ . Since  $L^2$  is a Banach space, any Cauchy sequence in  $L^2$  converges to a limit in  $L^2$ . Thus, for each fixed  $t \geq 0$ , there exists  $u(t) \in L^2$  such that

$$\|u_p(t) - u(t)\|_{L^2} \rightarrow 0 \quad \text{as } n \rightarrow \infty.$$

For all  $t \geq 0$ , for  $p, q \geq N$ , we have

$$e^{-kt} \|u_p(t) - u_q(t)\|_{L^2} < \varepsilon.$$

Taking the limit as  $q \rightarrow \infty$ , and using the pointwise convergence  $\|u_q(t) - u(t)\|_{L^2} \rightarrow 0$ , we deduce

$$e^{-kt} \|u_p(t) - u(t)\|_{L^2} \rightarrow 0, \quad \forall t \geq 0.$$

Since  $\|u_p - u\|_E = \sup_{t \geq 0} \|u_p(t) - u(t)\|_{L^2}$ , the uniform convergence implies

$$\|u_p - u\|_E \rightarrow 0 \quad \text{as } p \rightarrow \infty.$$

The sequence  $u_p$  converges to  $u$  in the  $E$ -norm, and  $u \in E$ . This confirms that  $E$  is a Banach space.

Now, for every  $u \in E$ , the function  $\phi u$  defined by

$$(\phi u)(t) = u_0 + \int_0^t F^{NL}(u(s)) ds$$

also belongs to  $E$ . Moreover, we have

$$\begin{aligned} \|\phi u - \phi v\|_E &= \sup_{t \geq 0} e^{-kt} \|\phi u(t) - \phi v(t)\|_{L^2(\Omega)} \\ &= \sup_{t \geq 0} e^{-kt} \left\| \int_0^t (F^{NL}(u(s)) - F^{NL}(v(s))) ds \right\|_{L^2(\Omega)}. \end{aligned}$$

However, we have

$$\begin{aligned} \|F^{NL}(u) - F^{NL}(v)\|_{L^2(\Omega)} &\leq \|-\Delta^{NL}u + \Delta^{NL}v\|_{L^2(\Omega)} + \|B^{NL}u - B^{NL}v\|_{L^2(\Omega)} \\ &\leq 2 \|J\|_{C_0^\infty} \|u - v\|_{L^2(\Omega)} + 4 \|W\|_{C_0^\infty} \|d\|_{L^\infty} \|u - v\|_{L^2(\Omega)} \\ &\leq L \|u - v\|_{L^2(\Omega)}, \end{aligned}$$

where  $L = 2 \|J\|_{C_0^\infty} + 4 \|W\|_{C_0^\infty} \|d\|_{L^\infty}$ . Therefore, we have that

$$\|\phi u - \phi v\|_E \leq \frac{L}{K} \|u - v\|_E.$$

Fixing any  $K > L$ , we find that  $\phi$  has a fixed point  $u \in E$ , which is a solution of (5.14).

Now we turn to proving uniqueness. Let  $u$  and  $\bar{u}$  two solutions of (NL-Osm) and set

$$\varphi(t) = \|u(t) - \bar{u}(t)\|_{L^2(\Omega)}.$$

From (5.14), we deduce that

$$\begin{aligned} \varphi(t) &= \left\| \int_0^t F^{NL}(u(s)) - F^{NL}(\bar{u}(s)) ds \right\|_{L^2(\Omega)} \\ &\leq \int_0^t \|F^{NL}(u(s)) - F^{NL}(\bar{u}(s))\|_{L^2(\Omega)} ds \\ &\leq L \int_0^t \|u(s) - \bar{u}(s)\|_{L^2(\Omega)} ds \\ &\leq L \int_0^t \varphi(s) ds, \end{aligned}$$

for all  $t \geq 0$  and consequently, using the Grönwall's inequality, we get  $\varphi(t) \leq 0$ . By the definition of  $\varphi(t)$  as a non-negative function, it follows that  $\varphi \equiv 0$ .  $\square$

Now that we know that a unique solution exists, we want to prove that it respects the same conservation properties valid in the local case.

**Proposition 5.3.3** (Average preservation). *Let  $u(x, t)$  be a solution of the non-local osmosis model. Then for all  $t \in [0, T]$  we have*

$$\int_{\Omega} u(x, t) dx = \int_{\Omega} u_0(x) dx$$

*Proof.* Using antisymmetry of  $d(x, y)$  and of  $(u(y) - u(x))$ , the time derivative of the average value is

$$\frac{\partial}{\partial t} \int_{\Omega} u(x, t) dx = \int_{\Omega} \frac{\partial}{\partial t} u(x, t) dx = - \int_{\Omega} ((A^{NL} + B^{NL})u)(x, t) dx = 0$$

$\square$

### 5.3.2 Regularity in time

The proven continuity and differentiability in time of the solution allows to study its evolution as  $t$  grows. An estimation is computed in the following Proposition.

**Proposition 5.3.4.** *Let  $A^{NL}$ ,  $B^{NL}$  defined as in Def.5.3.6. If  $u$  and  $v$  are two solutions of (NL-Osm) associated to initial data  $u_0$  and  $v_0$  respectively, then*

$$\|u(t) - v(t)\|_{L^2(\Omega)} \leq e^{Kt} (\|u_0 - v_0\|_{L^2(\Omega)}), \quad \forall t \in [0, T],$$

where  $K = 4 \|W\|_{C_0^\infty} \|d\|_{L^\infty}$  is the Lipschitz constant of  $B^{NL}$ .

*Proof.* Since  $\|u(t) - v(t)\|_{L^2(\Omega)}^2$  is absolutely continuous on every compact of  $]0, T[$  and continuous on  $[0, T]$  we get, using the monotonicity of  $\Delta^{NL}$  that

$$\begin{aligned} \frac{1}{2} \partial_t \|u(t) - v(t)\|_{L^2(\Omega)}^2 &= \left\langle \frac{d}{dt} u(t) - \frac{d}{dt} v(t), u(t) - v(t) \right\rangle \\ &= \langle \Delta^{NL} u(t) - \Delta^{NL} v(t), u(t) - v(t) \rangle + \\ &\quad + \langle -B^{NL} u(t) + B^{NL} v(t), u(t) - v(t) \rangle \\ &\leq \langle -B^{NL} u(t) + B^{NL} v(t), u(t) - v(t) \rangle. \end{aligned}$$

Now, we integrate over  $]s, t[$  and we obtain

$$\frac{1}{2} \|u(t) - v(t)\|_{L^2(\Omega)}^2 - \frac{1}{2} \|u(s) - v(s)\|_{L^2(\Omega)}^2 \leq \int_s^t \langle -B^{NL} u(\tau) + B^{NL} v(\tau), u(\tau) - v(\tau) \rangle d\tau.$$

We then deduce using [20, Lemma A.5]

$$\begin{aligned} \|u(t) - v(t)\|_{L^2(\Omega)} &\leq \|u(s) - v(s)\|_{L^2(\Omega)} + \int_s^t \|B^{NL} u(\tau) - B^{NL} v(\tau)\|_{L^2(\Omega)} d\tau \\ &\leq \|u(s) - v(s)\|_{L^2(\Omega)} + K \int_s^t \|u(\tau) - v(\tau)\|_{L^2(\Omega)} d\tau \end{aligned}$$

for all  $0 \leq s \leq t \leq T$ . Now using Grönwall's inequality, we deduce that

$$\|u(t) - v(t)\|_{L^2(\Omega)} \leq e^{Kt} \|u_0 - v_0\|_{L^2(\Omega)}.$$

□

## 5.4 Consistency

We have defined the integral operators  $A^{NL}$  and  $B^{NL}$  taking inspiration from the corresponding differential operators  $A$  and  $B$ . In order to verify that our proposal is a correct approximation of the osmosis model, we need to define a sequence of non-local models and observe if they generate a sequence of solutions that converge to the local solution.

Since the integral operators are defined in terms of a kernel  $W$  whose support identifies the region of influence for points  $y$  with respect to  $x$ , a natural way to study consistency is to generate a sequence of kernels  $W_\varepsilon$ , with  $\varepsilon > 0$ , with decreasing support as  $\varepsilon \rightarrow 0$ .

**Definition 5.4.1.** *Let  $\varepsilon > 0$ . Then we define a **rescaled kernel function**  $W_\varepsilon(x)$  as*

$$W_\varepsilon(x) = \frac{W(x/\varepsilon)}{\varepsilon^{N/2+1}}$$

We note that  $W_\varepsilon$ , being a simple rescaling, respects the same properties as the original kernel  $W$  of Assumption 5.3.1, i.e. it is radial, symmetric, smooth, with compact support. In particular, we observe that  $\text{supp}(W_\varepsilon) = \varepsilon \text{supp}(W)$ , converging monotonically to the origin  $\{0\}$ .

The rescaled kernel function  $W_\varepsilon$  defines new sequences of integral operators  $A_\varepsilon$ ,  $B_\varepsilon$  and drift terms  $d_\varepsilon(x, y) = W_\varepsilon(x - y)\bar{d}(x, y)$ .

The resulting sequence of non-local osmosis models reads as

$$\begin{cases} \partial_t u_\varepsilon + (A_\varepsilon + B_\varepsilon)u_\varepsilon = 0 \\ u_\varepsilon(0) = u_0 \end{cases} \quad (5.15)$$

and produces a corresponding sequence of solutions  $u_\varepsilon$ . As we have seen in Prop. 5.3.2, for any  $\varepsilon > 0$ , a function  $u_\varepsilon \in C^1([0, \infty), L^2(\Omega)) \cap C([0, \infty), D(A^{NL} + B^{NL}))$ .

Fixed a time  $t \in [0, T]$ , our goal is to show that the sequence  $(u_\varepsilon(t))_\varepsilon \subset L^2(\Omega)$  converges to a function  $v(t) \in W^{1,2}(\Omega)$  that is the unique solution of the local model.

**Conjecture 5.4.1.** *Let  $u_0 \in L^2(\Omega)$ . Let  $u_\varepsilon$  be the unique non-local solution of (5.15) and  $v$  the unique local solution of (L-Osm). Then*

$$\lim_{\varepsilon \rightarrow 0} \sup_{t \in [0, T]} \|u_\varepsilon(\cdot, t) - v(\cdot, t)\|_{L^2(\Omega)} = 0$$

In [3, Ch.4], the above convergence is proven in the particular case of a constant vector field  $\mathbf{d}$ , with zero divergence, exploiting the semigroup formulation of the two solutions, involving the fundamental solutions  $G_t$  and  $S_\varepsilon(t)$  of the local and non-local diffusion equation, respectively. In our case,

$$v(x, t) = G_t^2 * u_0 + \int_0^t G_{t-s}^2 * Bv(s) ds \quad (5.16)$$

$$u_\varepsilon(t) = S_\varepsilon(t) * u_0 + \int_0^t S_\varepsilon(t-s) * B_\varepsilon u_\varepsilon(s) ds \quad (5.17)$$

Using triangular inequality, for all  $t \in [0, T]$  we have:

$$\begin{aligned}
& \|u_\varepsilon(t) - v(t)\|_{L^2(\Omega)} \leq \\
& \leq \|S_\varepsilon(t) * u_0 - G(t)^2 * u_0\|_{L^2(\Omega)} + \left\| \int_0^t S_\varepsilon(t-s) * B_\varepsilon u_\varepsilon(s) ds - \int_0^t G^2(t-s) * Bv(s) ds \right\|_{L^2(\Omega)} = \\
& = \|S_\varepsilon(t) * u_0 - G(t)^2 * u_0\|_{L^2(\Omega)} + \\
& \quad + \left\| \int_0^t S_\varepsilon(t-s) * (B_\varepsilon(u_\varepsilon(s) - v(s))) ds + \int_0^t (S_\varepsilon(t-s) * B_\varepsilon v(s) - G^2(t-s) * Bv(s)) ds \right\|_{L^2(\Omega)} \leq \\
& \leq \underbrace{\|S_\varepsilon(t) * u_0 - G(t)^2 * u_0\|_{L^2(\Omega)}}_{I_0(t)} + \underbrace{\int_0^t \|S_\varepsilon(t-s) * (B_\varepsilon(u_\varepsilon(s) - v(s)))\|_{L^2(\Omega)} ds}_{I_1(t)} + \\
& \quad + \underbrace{\int_0^t \|(S_\varepsilon(t-s) * B_\varepsilon v(s) - G^2(t-s) * Bv(s))\|_{L^2(\Omega)} ds}_{I_2(t)}
\end{aligned}$$

The truthfulness of the Conjecture is therefore related to the estimation of the terms  $I_0, I_1, I_2$ , involving the convergence of the non-local operators  $A_\varepsilon, B_\varepsilon$  to the corresponding local operators  $A, B$ . In the following, we report partial results about this property.

We consider a sequence of functions  $u_\varepsilon \in L^2(\Omega)$  weakly converging in  $L^2$  norm to a function  $v \in W^{1,2}(\Omega)$ . With this assumption, we observe in the next two Propositions that

$$(A_\varepsilon + B_\varepsilon)u_\varepsilon \longrightarrow (A + B)v \quad \text{weakly in } L_2(\Omega)$$

**Proposition 5.4.1.** *Let  $u_\varepsilon$  be a sequence of functions in  $L^2(\Omega)$ , converging to  $v \in W^{1,2}(\Omega)$  weakly in  $L^2$  norm. Then  $A_\varepsilon u_\varepsilon \rightarrow Av$  weakly in  $L^2$  norm.*

*Proof.* Let  $\xi \in C_0^\infty(\Omega)$  be a smooth test function. We observe that

$$\begin{aligned}
& \int_{\mathbb{R}^N} (A_\varepsilon u_\varepsilon)(x) \xi(x) dx = \quad (\text{via change of variables } z = (x - y)/\varepsilon) \\
& = \int_{\mathbb{R}^N} \int_{\mathbb{R}^N} \frac{2W^2(z)}{\varepsilon^2} [\xi(x + \varepsilon z) - \xi(x)] dz u_\varepsilon(x) dx \\
& = \int_{\mathbb{R}^N} \int_{\mathbb{R}^N} \frac{2W^2(z)}{\varepsilon^2} \left[ \langle \nabla \xi(x), \varepsilon z \rangle + \int_0^1 \int_0^1 \langle D^2 \xi(x + st\varepsilon z) \varepsilon z, s\varepsilon z \rangle dt ds \right] dz u_\varepsilon(x) dx \\
& = 0 + \int_{\mathbb{R}^N} \int_{\mathbb{R}^N} 2W^2(z) \left[ \int_0^1 \int_0^1 s \langle D^2 \xi(x + st\varepsilon z) z, z \rangle dt ds \right] dz u_\varepsilon(x) dx
\end{aligned}$$

where we used the symmetry of  $W(z)$  and the regularity of  $\xi$ . Since  $\xi \in C_0^\infty(\Omega)$ ,



$u_\varepsilon \in L^2(\Omega)$  and  $W \in C_0^\infty(\Omega)$ , we can pass to the limit as  $\varepsilon \rightarrow 0$ .

$$\begin{aligned} \int_{\mathbb{R}^N} (A_\varepsilon u_\varepsilon)(x) \xi(x) dx &\longrightarrow \int_{\mathbb{R}^N} \int_{\mathbb{R}^N} 2W^2(z) \left[ \int_0^1 \int_0^1 s \langle D^2 \xi(x) z, z \rangle dt ds \right] dz v(x) dx = \\ &= \int_{\mathbb{R}^N} \sum_{ij} D_{ij}^2 \xi(x) \int_{\mathbb{R}^N} W^2(z) z_i z_j dz v(x) dx = \\ &= \int_{\mathbb{R}^N} \Delta \xi(x) v(x) dx = - \int_{\mathbb{R}^N} \Delta v(x) \xi(x) dx = \int_{\mathbb{R}^N} A v(x) \xi(x) dx . \end{aligned}$$

□

**Proposition 5.4.2.** *Let  $u_\varepsilon$  be a sequence of functions  $L^2(\Omega)$ , converging to  $v \in W^{1,2}(\Omega)$  weakly in  $L^2$  norm. Then  $B_\varepsilon u_\varepsilon \rightarrow Bv$  weakly in  $L^2$  norm.*

*Proof.* Let  $\xi \in C_0^\infty(\Omega)$  be a smooth test function. We observe that

$$\begin{aligned} \int_{\mathbb{R}^N} (B_\varepsilon u_\varepsilon)(x) \xi(x) dx &= \quad (\text{via change of variables } z = (x - y)/\varepsilon) \\ &= - \int_{\mathbb{R}^N} \sum_{ij} \int_{\mathbb{R}^N} W^2(z) z_i z_j \int_0^1 \int_0^1 \partial_i \xi(x + s\varepsilon z) \mathbf{d}_j(x + t\varepsilon z) ds dt dz u_\varepsilon(x) dx \end{aligned}$$

Since  $\xi \in C_0^\infty(\Omega)$ ,  $\mathbf{d} \in L^\infty(\Omega)$  and  $u_\varepsilon \in L^2(\Omega)$ , we can pass to the limit as  $\varepsilon \rightarrow 0$  and get

$$\begin{aligned} \int_{\mathbb{R}^N} (B_\varepsilon u_\varepsilon)(x) \xi(x) dx &\longrightarrow - \int_{\mathbb{R}^N} \sum_{ij} \int_{\mathbb{R}^N} W^2(z) z_i z_j \partial_i \xi(x) \mathbf{d}_j(x) dz v(x) dx = \\ &= - \int_{\mathbb{R}^N} \langle \nabla \xi(x), \mathbf{d}(x) \rangle v(x) dx = \\ &= \int_{\mathbb{R}^N} \operatorname{div}(\mathbf{d}v)(x) \xi(x) dx = \int_{\mathbb{R}^N} (Bv)(x) \xi(x) dx \end{aligned}$$

□

To prove the Conjecture, it is therefore necessary to show that:

1. the non-local equation defines a converging sequence of solutions  $u_\varepsilon \in L^2(\Omega)$ ,
2. the corresponding limit is a function  $v \in W^{1,2}(\Omega)$ .

We expect these properties to be related to a space regularity of the non-local solutions  $u_\varepsilon$ , stronger than  $u_\varepsilon \in L^2(\Omega)$ , but weaker than  $u_\varepsilon \in W^{1,2}(\Omega)$ . The ongoing research is focused on these relations.

## 5.5 Graph discretization and applications

We consider a connected undirected weighted graph  $G = (V, E, W)$  with vertices  $V = \{v_i\}$ , edges  $E = \{e_{ij}\}$  and symmetric weights  $w_{ij} \geq 0$ . Let  $\mathcal{F}_V$  and  $\mathcal{F}_E$ , as defined in (1.18), the set of scalar functions acting on vertices and edges, respectively. We aim to discretize on the graph  $G$  the osmosis PDE with positive initial value  $u_0$ :

$$\begin{cases} u_t = \Delta u - \operatorname{div}(du) \\ u(0) = u_0 > 0 \end{cases} \quad (5.18)$$

First, we focus on the spatial discretization of the right-hand-side of the equation, by defining the differential operators gradient, divergence and Laplacian.

### 5.5.1 Space Discretization

The non-local model construction in Section 3 can easily be translated into the discrete graph domain. In fact, the function  $u$  is sampled on the graph vertices, obtaining a vector  $u = (u_1, \dots, u_{n_V}) \in \mathcal{F}_V$ , while the non-local drift  $d(x, y)$  in (5.13) is discretized as a function  $d \in \mathcal{F}_E$ , with value  $d_{[i,j]}$  on the edge connecting vertices  $v_i$  and  $v_j$ . Finally, the weights  $w_{ij}$  have the role of the kernel  $W^2(x)$  and, on a vertex  $v_i$ , the kernel involves only the vertices in the first ring  $\mathcal{N}(i)$ .

With this analogy, the definitions of graph gradient and divergence correspond to the ones already given in Def. 4.1.2-4.1.3. Consequently, the graph Laplacian is defined as follows:

**Definition 5.5.1.** *The **graph Laplacian** is the operator  $\Delta : \mathcal{F}_V \rightarrow \mathcal{F}_V$  defined as  $\Delta u := \operatorname{div}(\nabla u)$ . At vertex  $i$ , its value is:*

$$(\Delta u)_i = 2 \sum_{j \in \mathcal{N}(i)} w_{ij} (u_j - u_i) \quad (5.19)$$

For the transport term, we use again the definitions of graph gradient and divergence and compute  $\operatorname{div}(du) = \operatorname{div}(d)u + \langle d, \nabla u \rangle$ , with  $u \in \mathcal{F}_V$  and  $d \in \mathcal{F}_E$ , as

$$\operatorname{div}(du)_i = 2 \sum_{j \in \mathcal{N}(i)} \sqrt{w_{ij}} d_{[i,j]} u_i + \sum_{j \in \mathcal{N}(i)} \sqrt{w_{ij}} d_{[i,j]} (u_j - u_i) = \sum_{j \in \mathcal{N}(i)} \sqrt{w_{ij}} d_{[i,j]} (u_i + u_j) \quad (5.20)$$

Using (5.19) and (5.20), we get the final spatial discretization of the osmosis PDE

$$(\Delta u - \operatorname{div}(du))_i = \sum_{j \in \mathcal{N}(i)} [2 w_{ij} (u_j - u_i) - \sqrt{w_{ij}} (u_i + u_j) d_{[i,j]}] \quad (5.21)$$

Separating the terms related to  $u_i$  and  $u_j$ , we can express the differential operator  $\Delta u - \operatorname{div}(du)$  as a matrix-vector product

$$\begin{aligned} \Delta u - \operatorname{div}(du) &= Au, \quad \text{with } A \in \mathbb{R}^n, \\ A_{ik} &= \begin{cases} 2w_{ik} - \sqrt{w_{ik}}d_{[i,k]} & \text{if } i \neq k; \\ \sum_{j \in \mathcal{N}(i)} [-2w_{ij} - \sqrt{w_{ij}}d_{[i,j]}] & \text{if } i = k. \end{cases} \end{aligned} \quad (5.22)$$

The matrix  $A$  respects some important properties, that we analyze in the following proposition.

**Proposition 5.5.1** (Properties of  $A$ ). *Consider  $A$  defined in (5.22). Then:*

1.  $A$  is not symmetric;
2.  $A$  has column sums zero;
3.  $A$  is irreducible.

*Proof.* 1.  $A$  is not symmetric because  $d$  is antisymmetric.

2. Since  $w_{ik} = 0$  if  $k \notin \mathcal{N}(i)$ ,  $W$  is symmetric and  $d$  is antisymmetric, then we have

$$\sum_{i=1}^n A_{ik} = \sum_{i \neq k} A_{ik} + A_{kk} = \sum_{i \in \mathcal{N}(k)} [2w_{ik} - \sqrt{w_{ik}}d_{[i,k]}] + \sum_{j \in \mathcal{N}(k)} [-2w_{kj} - \sqrt{w_{kj}}d_{[k,j]}] = 0$$

3. The digraph associated with  $A$  is simply the graph  $G$ , which is connected, and therefore the matrix is irreducible (see [13, Ch.2]).  $\square$

### 5.5.2 Time Discretization

In the previous section, we have discretized the right-hand-side of the PDE (5.18), obtaining  $u_t = Au$ . For the time discretization on the left-hand-side, we define a time-step  $\tau > 0$  and denote as  $u^k$  the value of the function  $u$  at time  $k\tau$ . We consider two options:

- **Explicit scheme:**

$$\frac{u^{k+1} - u^k}{\tau} = Au^k \quad \implies \quad u^{k+1} = (I + \tau A)u^k \quad (\text{EXP})$$

- **Implicit scheme:**

$$\frac{u^{k+1} - u^k}{\tau} = Au^{k+1} \quad \implies \quad u^{k+1} = (I - \tau A)^{-1}u^k \quad (\text{IMP})$$

Both the evolution schemes can be written as  $u^{k+1} = Pu^k$ , with the appropriate definition of  $P$ . In [124], the authors proved the following proposition, that analyzes the evolution of  $u$  depending on some properties of  $A$ .

**Proposition 5.5.2** (Prop.2 in [124]). *Consider the process  $u'(t) = Au(t)$ , starting from  $u(0) = u_0 \in \mathbb{R}_+^n$ . Suppose that  $A \in \mathbb{R}^{n \times n}$  satisfies the following properties:*

- P1) All column sums of  $A$  are 0;*
- P2)  $A$  has non-negative off-diagonal entries;*
- P3)  $A$  is irreducible;*

*Consider the schemes:*

- 1. explicit:  $u^{k+1} = Pu^k = (I + \tau A)u^k$ , with  $\tau < 1/a_{ii}$  for all  $i = 1, \dots, n$ ;*
- 2. implicit:  $u^{k+1} = Pu^k = (I - \tau A)^{-1}u^k$ , with  $\tau > 0$ ;*

*Then, in both cases, the following results hold:*

- a) the average grey value is preserved:*

$$\mu(u) := \frac{1}{n} \sum_{i=1}^n u_i^k = \frac{1}{n} \sum_{i=1}^n (u_0)_i^k \quad \forall k > 0$$

- b) The evolution preserves positivity:  $u_i^k > 0$  for all  $i = 1, \dots, n$ ,  $k > 0$ ;*
- c) There exists a unique steady-state for  $k \rightarrow \infty$ , which is the simple eigenvector  $w \in \mathbb{R}_+^n$  of  $P$  to the eigenvalue 1. It has the same average grey value as  $u_0$ .*

As seen in Prop. 5.5.1, conditions  $P1$  and  $P3$  are verified by construction, while condition  $P2$  depends on the drift term  $d$ .

We consider a specific expression for  $d$ , defining the canonical drift term as follows:

**Definition 5.5.2.** *Let  $v \in \mathbb{R}_+^n$  be a **reference function** defined on the vertices  $V$  of the graph  $G$ . The **canonical drift term** is  $\mathbf{d} = \nabla \ln v = \nabla v/v$ , discretized on each half-edge  $[i, j]$  as*

$$d_{[i,j]} = \frac{2\sqrt{w_{ij}}(v_j - v_i)}{v_j + v_i}. \quad (5.23)$$

We can easily test that the definition of the non-local drift in (5.13) is coherent. In fact, for  $\mathbf{d} = \nabla \ln v$ , we have

$$\begin{aligned} d(x, y) &= W_\varepsilon(x - y) \int_0^1 \langle \nabla \ln v(x + t(y - x)), y - x \rangle dt = W_\varepsilon(x - y) [\ln v(x + t(y - x))]_0^1 = \\ &= W_\varepsilon(x - y) (\ln v(y) - \ln v(x)) = (\nabla^{NL} \ln v)(x, y) \end{aligned} \quad (5.24)$$

The next Proposition shows that such  $d$  fulfills condition  $P2$  too and, furthermore, allows to explicitly know the steady-state of the evolution process.

**Proposition 5.5.3.** *Let  $v \in \mathbb{R}_+^n$  be a **reference function** defined on the vertices  $V$  of the graph  $G$ . Consider  $d$  in canonical form (5.23). Then  $A$  defined in (5.22) verifies condition  $P2$ . Moreover, the steady-state of the evolution process  $u_t = Au$  is  $w = \frac{\mu(u_0)}{\mu(v)}v$ , where  $\mu(\cdot)$  is the average grey value of Prop. 5.5.2.*

*Proof.* We just need to observe that  $d_{[i,k]} \leq 2w_{[i,k]}$ .

$$d_{[i,k]} = \frac{2w_{i,k}(v_k - v_i)}{v_k + v_i} \leq 2w_{i,k} \iff \frac{(v_k - v_i)}{v_k + v_i} \leq 1$$

which is true since  $v$  is strictly positive. Hence,  $A$  verifies all the conditions  $P1, P2, P3$  from Prop. 5.5.1. In order to find the steady-state eigenvector, we first observe from simple computations that  $Av = 0$ . Therefore:

- in the explicit case,  $Pv = (I + \tau A)v = v$ ;
- in the implicit case,  $P^{-1}v = (I - \tau A)v = v \implies Pv = v$ .

Hence,  $v$  is the simple eigenvector of  $P$  to the eigenvalue 1. Since the evolution preserves the average grey value, the steady-state is  $w = cv$ , with  $c = \mu(u_0)/\mu(v)$ .  $\square$

## 5.6 Numerical results

We test the correctness of the proposed graph discretization of the osmosis model with some simple examples <sup>1</sup>, using as  $G$  the graph structure inherited by the mesh approximation of a surface  $\mathcal{S} \subset \mathbb{R}^3$  and as functions  $u_0, u, v : V \rightarrow \mathbb{R}^3$  color functions defined on the vertices, acting on each RGB channel separately.

In the first test, depicted in Fig.5.5, we set  $v$  as reference function,  $d = \nabla v/v$  the canonical drift and  $u_0 \equiv \mu(v)$  a constant initial data. As expected, the osmosis drives the process towards a steady-state  $w$  that corresponds to the reference function  $v$ .

Same result we obtain setting  $u_0 = (\mu(v)/\mu(\bar{u}_0))\bar{u}_0$ , with  $\bar{u}_0 \in \mathbb{R}^{n \times 3}$  generated on each component by a random uniform distribution in  $[0, 1]$ .



Figure 5.5: From the left:  $u_0$  constant,  $w = v$  result and reference,  $u_0$  random,  $w = v$  result and reference.



Figure 5.6: Left: mask  $V_b$ . Centre: Initial data  $u_0 = v = f$ , with shadowed region  $V_S$ . Right: result  $w$ .

A further example is inspired by the applications of the osmosis model in image processing, described in Sec. 1.2. Given a function  $f$  representing colors on a mesh  $G$ , with a shadow region  $S \subset \mathcal{S}$ , identified by a corresponding subset of vertices  $V_S$ . We set both the initial data  $u_0$  and the reference data  $v$  to coincide with  $f$ . We identify a set  $V_b$  of vertices around the boundary of  $V_S$  and set the drift term as prescribed in (5.4), which, in discrete graph setting, corresponds to

$$d_{[i,j]} = \begin{cases} \frac{2w_{i,j}(v_j - v_i)}{v_j + v_i} & \text{if } v_i, v_j \notin V_b \text{ and } j \in \mathcal{N}(i) \\ 0 & \text{else} \end{cases}$$

Figure 5.6 shows that, as in the image context, the osmosis is able to balance the color intensity outside and inside the shadowed region. However, around the shadow boundary the diffusion effect produces oversmoothing, losing information about the color feature of the surface.

<sup>1</sup>courtesy of Anass Nouri, Christophe Charrier, Olivier Lezoray - Greyc 3D Colored Mesh Database, Technical report, 2017. <https://nouri.users.greyc.fr/ColoredMeshDatabase.html>.

To avoid this problem, in Chapter 6 we propose a non-linear version of the osmosis PDE model, able to vary the diffusion intensity in the different regions of the domain. We analyze results only in the case of image shadow removal, but the discretization provided in Sec.5.5.1 allows to extend the application to 3D surfaces.

With the same argument, we observe that the tasks of compact data representation and image cloning are easily adapted in the case of color functions defined on a graph  $G$  discretely representing a surface in  $\mathbb{R}^3$ .

Since osmosis can be applied to any kind of positive data defined on a graph, it is potentially an extremely effective tool for surface processing tasks. In Chapter 7, we propose preliminary work on the use for surface inpainting and cloning, where osmosis acts directly on the Euclidean or differential coordinate functions defined on the mesh vertices.

# Chapter 6

## Non-linear PDE models for image osmotic flow

In Sec.5.1, we have observed how the linear osmosis model (5.2) can be employed in image processing for applications such as shadow/light-spot removal, compact data representation and cloning.

This formulation, despite correctly performing these data-integrating tasks, may suffer from some reconstruction artifacts (typically, over-smoothing) due to the underlying linear smoothing enforced by the diffusion terms. In particular, in shadow/light-spot removal the drift term is defined as in (5.4), causing pure diffusion in the boundary between two differently lid regions of the image, with consequent possible loss of important features.

This reason justifies the research for different non-linear variants of the models. For example, in [93] the model is anisotropic, thanks to the use of a positive symmetric matrix field  $D : \Omega \rightarrow \mathbb{R}^2$  encoding local directional information. It provides better results but with a high computational cost due to the computation of  $D$  through a tensor voting strategy.

In this Chapter we present a non-linear version of the osmosis PDE model that exploits a scalar positive diffusivity function  $g : \Omega \rightarrow \mathbb{R}$ , thus promoting isotropic edge-stopping diffusion.

Analytically, we prove that the proposed evolution model maintains the same conservation properties of the original linear model and that it can be interpreted as the gradient flow of suitable non-smooth energy for specific choices of the drift term. We then show that analogous properties hold also at a discrete level, whenever appropriate finite-difference discretization stencils and explicit and semi-implicit schemes are used. For those schemes, we prove conditional and unconditional stability, respectively. Unconditional stability is particularly interesting from a computational viewpoint as it allows for the computation of the desired steady-state solution. The proposed model is fully automatic as no hyperparameters need to be tuned. The algorithm<sup>1</sup> results are compared

---

<sup>1</sup>codes freely available at <https://github.com/giusepperecupero/non-linear-osmosis.git>



with the ones obtained with the standard linear osmosis model, with the anisotropic version [93] and with other state-of-the-art methods.

For example, a sophisticated approach has been proposed in [10] where a non-local version of the osmosis operator along with an anisotropic regularization term has been integrated into a variational formulation. This model achieves good performance in shadow removal, at the price of a high computational cost. Alternative approaches specific to the shadow removal task are described in [47, 46, 33, 52, 71], based, for instance, on suitable mapping into appropriate color spaces and/or statistical learning.

The Chapter is structured as follows. In Section 6.1 the proposed continuous non-linear osmosis model is described and its conservation and variational properties are proved. In Section 6.2 spatial and temporal discretization schemes are studied, in particular from the point of view of consistency with the analogous properties in the continuous setting and in terms of stability and convergence. In Section 6.3 several results for three different imaging applications (shadow removal, light removal and compact data representation) are shown, confirming that the use of the proposed non-linear model improves upon linear and state-of-the-art methods.

**Notation** In Section 6.1 we will use the **bold** notation to denote vector fields in  $\mathbb{R}^N$ . To avoid unnecessary heavy notation in the discretized setting introduced starting from Section 6.2, we will use standard (unbold) notation to denote both vectors in  $\mathbb{R}^N$  and matrices in  $\mathbb{R}^{N \times N}$ . We analyze the model properties in the simple case of greyscale images. In case of color images, the functions  $u = (u^R, u^G, u^B)$ ,  $v = (v^R, v^G, v^B)$  and  $u_0 = (u_0^R, u_0^G, u_0^B)$  have values in  $(\mathbb{R}_+^*)^3$  and the straightforward extension of the model treats separately each R, G, B channel.

## 6.1 The continuous non-linear model

Given a rectangular image domain  $\Omega \subset \mathbb{R}^2$  with boundary  $\partial\Omega$  and a finite time  $T > 0$ , let  $\mathbf{d} : \Omega \rightarrow \mathbb{R}^2$  be a given drift vector field,  $u_0 \in L^\infty(\Omega; \mathbb{R}_+^*)$  a positive greyscale image (the extension to the RGB case is straightforward) and  $g : \Omega \times (0, T] \rightarrow \mathbb{R}_+^*$  be a positive (non-linear) diffusivity function driving the evolution process. The proposed non-linear osmosis model is given by the following drift-diffusion PDEs:

$$\partial_t u(x, t) = \operatorname{div} (g(x, t) (\nabla u(x, t) - \mathbf{d}(x)u(x, t))) \quad \text{for } (x, t) \in \Omega \times (0, T], \quad (6.1)$$

which we endow with homogenous Neumann boundary conditions and initial condition

$$\begin{cases} \langle \nabla u - \mathbf{d}u, \mathbf{n} \rangle = 0 & \text{on } \partial\Omega \times (0, T]; \\ u(x, 0) = u_0(x) & \text{on } \Omega. \end{cases} \quad (6.2)$$

with  $g$  defined as

$$g(x, t) := g(|\mathbf{s}(x, t)|) := \frac{1}{|\mathbf{s}(x, t)|}, \quad \text{with } \mathbf{s}(x, t) := \nabla u(x, t) - \mathbf{d}(x)u(x, t) \quad (6.3)$$

in order to normalize the argument of the divergence, balancing the osmosis flow evolution, consisting of both diffusion and transport effects, in the different regions of the domain.

The following result shows that, in general, for any positive diffusivity function  $g$  the model (6.1) enjoys interesting conservation properties.

**Proposition 6.1.1** (Conservation properties). *Any solution  $u : \Omega \times (0, T] \rightarrow \mathbb{R}$  of the non-linear osmosis model (6.1) satisfies the following properties:*

1. *The average grey value is preserved:*

$$\frac{1}{|\Omega|} \int_{\Omega} u(x, t) dx = \frac{1}{|\Omega|} \int_{\Omega} u_0(x) dx =: \mu(u_0) \quad \forall t > 0; \quad (6.4)$$

2. *The evolution preserves non-negativity:*

$$u(x, t) \geq 0 \quad \forall x \in \Omega, \forall t > 0. \quad (6.5)$$

*Proof.* 1) Define, for all  $t \in (0, T]$ ,  $m(t) := (\int_{\Omega} u(x, t) dx) / |\Omega|$ . Then, by (6.1),

$$\begin{aligned} \frac{dm}{dt} &= \frac{1}{|\Omega|} \int_{\Omega} \partial_t u(x, t) dx = \frac{1}{|\Omega|} \int_{\Omega} \operatorname{div} (g(x, t)(\nabla u(x, t) - \mathbf{d}(x)u(x, t))) dx = \\ &= \frac{1}{|\Omega|} \int_{\partial\Omega} \langle g(s, t)(\nabla u(s, t) - \mathbf{d}(s)u(s, t)), \mathbf{n}(s) \rangle ds = 0, \quad \forall t > 0 \end{aligned}$$

thanks to the divergence theorem and by imposing Neumann boundary conditions.

2) Since  $u(x, 0) = u_0(x) > 0$  a.e., let now  $\tau > 0$  be the smallest time such that  $\min_x u(x, t) = 0$ . Suppose that this minimum is obtained at a  $\xi \in \operatorname{int}(\Omega)$ . Then  $u(\xi, \tau) = 0$  and  $\nabla u(\xi, \tau) = 0$ . Computing the time derivative of  $u$  at point  $(\xi, \tau)$  gives:

$$\begin{aligned} \partial_t u(\xi, \tau) &= (\operatorname{div}(g(\cdot, \cdot)(\nabla u(\cdot, \cdot) - \mathbf{d}(\cdot)u(\cdot, \cdot))) (\xi, \tau) = \\ &= g(\xi, \tau)\Delta u(\xi, \tau) + \langle \nabla u(\xi, \tau), \nabla g(\xi, \tau) \rangle - \operatorname{div}(\mathbf{d}(\cdot)g(\cdot, \cdot)u(\cdot, \cdot))(\xi, \tau) = \\ &= g(\xi, \tau)\Delta u(\xi, \tau) - (\operatorname{div} \mathbf{d}(\cdot))(\xi, \tau)u(\xi, \tau)g(\xi, \tau) - \langle \mathbf{d}(\cdot), \nabla((gu)(\cdot, \cdot)) \rangle (\xi, \tau) = \\ &= g(\xi, \tau)\Delta u(\xi, \tau) - \langle \mathbf{d}(\cdot), \nabla u(\cdot, \cdot) \rangle g(\cdot, \cdot) + u(\cdot, \cdot)\nabla g(\cdot, \cdot) (\xi, \tau) = \\ &= g(\xi, \tau)\Delta u(\xi, \tau) = g(\xi, \tau)\Delta u(\xi, \tau), \end{aligned}$$

by standard properties of the divergence operator. Then, at  $(\xi, \tau)$  the evolution behaves as the diffusion equation  $u_t = g\Delta u$ , where  $g$  is a positive function. Hence, the operator  $\mathcal{L} := g\Delta$  is elliptic so the standard minimum/maximum principle can be applied. This tells us that for any  $t \geq \tau$  the solution of the non-linear model remains non-negative, as desired.  $\square$

While the last results hold for any positive  $g$ , the specific formulation in (6.3) is used to characterize the steady states of (6.1) as minimizers of a suitable energy functional. To show that, we first need the following lemma.

**Lemma 6.1.1.** *Let  $\Omega$  be an open rectangular domain in  $\mathbb{R}^2$  and  $\mathbf{w} \in C^1(\Omega, \mathbb{R}^2) \cap C^0(\overline{\Omega}, \mathbb{R}^2)$ . Then:*

$$\int_{\partial\Omega} \langle \mathbf{w}, \mathbf{n} \rangle z \, ds - \int_{\Omega} \operatorname{div}(\mathbf{w}) z \, dx = 0 \quad \forall z \in C^0(\overline{\Omega}), \quad (6.6)$$

if and only if

$$\begin{cases} \langle \mathbf{w}, \mathbf{n} \rangle = 0 & \text{on } \partial\Omega \\ \operatorname{div}(\mathbf{w}) = 0 & \text{on } \Omega. \end{cases} \quad (6.7)$$

*Proof.* First, we remark that the function  $z$  is the same for both the two integrals (otherwise the Theorem would be trivial). Now, eq. (6.7) implies eq. (6.6), obviously. For the inverse implication, consider the two alternative cases: either  $\operatorname{div}(\mathbf{w})(x) = 0$  for all  $x \in \Omega$ , or there exists a point  $x_0 \in \Omega$  s.t.  $\operatorname{div}(\mathbf{w})(x_0) \neq 0$ . We prove by contradiction that the second case is not possible. Suppose that  $\operatorname{div}(\mathbf{w})(x_0) > 0$ . Since  $\mathbf{w} \in C^1(\Omega; \mathbb{R}^2)$ , then there exists a neighbourhood  $B(x_0, r) \subset \Omega$  s.t.  $\operatorname{div}(\mathbf{w})(x) > 0$  for all  $x \in B(x_0, r)$ . Now, let  $z \in C^0(\overline{\Omega})$  be a function satisfying (6.6). Now define

$$z^* := z + \Phi(\chi_{B(x_0, r)}) \in C^0(\overline{\Omega})$$

where  $\Phi(\chi_{B(x_0, r)})$  is a non-negative smooth regularization of  $\chi_{B(x_0, r)}$ , with support in  $B(x_0, r)$ . Then, we have

$$\begin{aligned} \int_{\partial\Omega} \langle \mathbf{w}, \mathbf{n} \rangle z^* \, ds - \int_{\Omega} \operatorname{div}(\mathbf{w}) z^* \, dx &= \\ &= \int_{\partial\Omega} \langle \mathbf{w}, \mathbf{n} \rangle z \, ds - \int_{\Omega} \operatorname{div}(\mathbf{w}) z \, dx - \int_{\Omega} \operatorname{div}(\mathbf{w}) \Phi(\chi_{B(x_0, r)}) \, dx = \\ &= 0 - \int_{\Omega} \operatorname{div}(\mathbf{w}) \Phi(\chi_{B(x_0, r)}) \, dx = \\ &= - \int_{B(x_0, r)} \operatorname{div}(\mathbf{w}) \Phi(\chi_{B(x_0, r)}) \, dx < 0 \end{aligned}$$

because  $\operatorname{supp}(\Phi(\chi_{B(x_0, r)})) \subset B(x_0, r)$ ,  $\Phi(x) \geq 0$  and  $\operatorname{div}(\mathbf{w}) > 0$  in  $B(x_0, r)$ . We have thus reached a contradiction, as for  $z^* \in C^0(\overline{\Omega})$  equation (6.6) is not verified. Therefore, it must be  $\operatorname{div}(\mathbf{w})(x) = 0$  for all  $x \in \Omega$ . We claim that this implies also that

$$\int_{\partial\Omega} \langle \mathbf{w}, \mathbf{n} \rangle z \, ds = 0 \quad \forall z \in C^0(\overline{\Omega}, \mathbb{R}). \quad (6.8)$$

By contradiction, consider a point  $x_1 \in \partial\Omega$  s.t.  $\langle \mathbf{w}, \mathbf{n} \rangle(x_1) \neq 0$  (suppose positive). Since  $\mathbf{w} \in C^0(\overline{\Omega}; \mathbb{R}^2)$ , there exists  $B(x_1, r)$  s.t.  $\langle \mathbf{w}, \mathbf{n} \rangle(x) > 0$  for all  $x \in B(x_1, r) \cap \partial\Omega$ . Let now  $z \in C^0(\Omega)$  be a function satisfying (6.8) and define  $z^* := z + \tilde{\Phi}(\chi_{B(x_1, r)})$ , where  $\tilde{\Phi}(\cdot)$  is as a smooth regularisation of  $\chi_{B(x_1, r)}$  with support in  $B(x_1, r)$ . Then,

$$\begin{aligned} \int_{\partial\Omega} \langle \mathbf{w}, \mathbf{n} \rangle z^* ds &= \int_{\partial\Omega} \langle \mathbf{w}, \mathbf{n} \rangle z ds + \int_{\partial\Omega} \langle \mathbf{w}, \mathbf{n} \rangle \Phi(\chi_{B(x_1, r)}) ds = \\ &= 0 + \int_{\partial\Omega} \langle \mathbf{w}, \mathbf{n} \rangle \Phi(\chi_{B(x_1, r)}) ds = \\ &= \int_{B(x_1, r) \cap \partial\Omega} \langle \mathbf{w}, \mathbf{n} \rangle \Phi(\chi_{B(x_1, r)}) ds > 0 \end{aligned}$$

because  $\text{supp}(\Phi(\chi_{B(x_1, r)})) \cap \partial\Omega \subset B(x_1, r) \cap \partial\Omega$ ,  $\Phi(x) \geq 0$  and  $\langle \mathbf{w}, \mathbf{n} \rangle(x) > 0$  in  $B(x_1, r) \cap \partial\Omega$ . We thus built a function  $z^*$  that does not satisfy (6.8), which is of course a contradiction. Therefore,  $\langle \mathbf{w}, \mathbf{n} \rangle(x) = 0$  for all  $x \in \partial\Omega$ .  $\square$

Thanks to this lemma, we are now able to interpret the non-linear PDE model (6.1) as the gradient flow of an energy functional  $E(u)$  in the special case when the diffusivity function  $g$  is defined as in (6.3).

**Proposition 6.1.2.** *Let  $v \in H^1(\Omega; \mathbb{R}_+^*)$  be a given reference image,  $\mathbf{d} = \nabla(\log v)$  the canonical drift vector field associated to  $v$ , and  $g$  be defined as in (6.3). Then, a function  $u^* : \Omega \times (0, T] \rightarrow \mathbb{R}_+$  satisfies the steady state equation*

$$\text{div}(g(|\nabla u^* - \mathbf{d}u^*|)(\nabla u^* - \mathbf{d}u^*)) = 0 \quad (6.9)$$

*with boundary condition (6.2) associated to the osmosis model (6.1), if and only if  $u^*$  is a stationary point of the energy functional*

$$E(u) := \int_{\Omega} \left| \nabla \left( \frac{u}{v} \right) (x) \right| dx. \quad (6.10)$$

*Moreover,  $u^*$  is a multiplicative rescaling of  $v$  where the rescaling constant is the ratio between the average gray value of  $u_0 = u(\cdot, 0)$  and  $v$ , i.e.:*

$$u^*(x) = \frac{\mu(u_0)}{\mu(v)} v(x), \quad \forall x \in \Omega.$$

*Proof.* Conventionally, for  $x \in \Omega$  and  $t \in (0, T]$  such that  $\nabla u(x, t) - \mathbf{d}(x)u(x, t) = 0$ , we set

$$\frac{\nabla u(x, t) - \mathbf{d}(x)u(x, t)}{|\nabla u(x, t) - \mathbf{d}(x)u(x, t)|} = 0.$$

Let us now consider any test function  $z \in C_c^\infty(\Omega; \mathbb{R}_+^*)$ . We have

$$\begin{aligned}
\left( \frac{d}{d\eta} E(u + \eta z) \right) \Big|_{\eta=0} &= \left( \int_{\Omega} \frac{d}{d\eta} \left| \nabla \left( \frac{u + \eta z}{v} \right) \right| dx \right) \Big|_{\eta=0} = \\
&= \left( \int_{\Omega} \left\langle \frac{\nabla \left( \frac{u + \eta z}{v} \right)}{\left| \nabla \left( \frac{u + \eta z}{v} \right) \right|}, \nabla \left( \frac{z}{v} \right) \right\rangle dx \right) \Big|_{\eta=0} = \\
&= \int_{\Omega} \left\langle \frac{\nabla \left( \frac{u}{v} \right)}{\left| \nabla \left( \frac{u}{v} \right) \right|}, \nabla \left( \frac{z}{v} \right) \right\rangle dx = \\
&= \int_{\Omega} \left\langle \frac{\nabla u - \mathbf{d}u}{|\nabla u - \mathbf{d}u|}, \nabla \left( \frac{z}{v} \right) \right\rangle dx = \\
&= \int_{\Omega} \left[ \operatorname{div} \left( \frac{\nabla u - \mathbf{d}u}{|\nabla u - \mathbf{d}u|} \cdot \frac{z}{v} \right) - \frac{z}{v} \operatorname{div} \left( \frac{\nabla u - \mathbf{d}u}{|\nabla u - \mathbf{d}u|} \right) \right] dx = \\
&= \int_{\partial\Omega} \left\langle \frac{\nabla u - \mathbf{d}u}{|\nabla u - \mathbf{d}u|} \cdot \frac{z}{v}, \mathbf{n} \right\rangle ds - \int_{\Omega} \frac{z}{v} \operatorname{div} \left( \frac{\nabla u - \mathbf{d}u}{|\nabla u - \mathbf{d}u|} \right) dx = \\
&= \int_{\partial\Omega} \left\langle g(|\nabla u - \mathbf{d}u|)(\nabla u - \mathbf{d}u) \cdot \frac{z}{v}, \mathbf{n} \right\rangle ds - \int_{\Omega} \frac{z}{v} \operatorname{div} (g(|\nabla u - \mathbf{d}u|)(\nabla u - \mathbf{d}u)) dx
\end{aligned} \tag{6.11}$$

If a function  $u^*$  satisfies the steady-state equation (6.9) and the homogeneous Neumann boundary conditions (6.2), then the two integrals in (6.11) are zeros and  $u^*$  is a stationary point of the energy  $E(u)$ . The reverse holds thanks to Lemma 6.1.1.

To conclude the proof, we observe that every image  $u^* = cv$ , with  $c \in \mathbb{R}_+^*$ , is a stationary point of the energy  $E(\cdot)$ . Hence, such functions are solutions of the steady-state equation (6.9). Since the osmosis evolution preserves the average grey value by Proposition 6.1.1, one can easily show that:

$$\mu(u_0) = \mu(u^*) = \frac{1}{|\Omega|} \int_{\Omega} cv(x) dx = c\mu(v),$$

whence  $c = \mu(u_0)/\mu(v) > 0$ . □

An analogous result can be proved as a corollary of the previous proposition by allowing further dependence on a smoothing parameter  $p \in [1, 2)$ . In Section 6.3, we test a non-linear osmosis model for different values of  $p$ , evaluating the practical effects of its choice on exemplar tests.

**Corollary 6.1.1.** *Under the same assumptions of Proposition 6.1.2 and defining for  $p \in [1, 2)$  the diffusivity term  $g_p$  as*

$$g_p(x, t) := g_p(|\mathbf{s}(x, t)|) := \frac{1}{|\mathbf{s}(x, t)|^p}, \quad \forall (x, t) \in \Omega \times (0, T], \tag{6.12}$$

then the steady-state of the energy functional

$$E_p(u) := \int_{\Omega} \frac{v^{1-p}}{2-p} \left| \nabla \left( \frac{u}{v} \right) (x) \right|^{2-p} dx \quad (6.13)$$

satisfies:

$$\operatorname{div} (g_p(|\nabla u^* - \mathbf{d}u^*|) (\nabla u^* - \mathbf{d}u^*)) = 0. \quad (6.14)$$

A possible direction of future work consists in investigating the  $p$ -osmosis model (6.14) so as to extend the osmosis filtering to spatially adaptive filtering, as proposed for image denoising, e.g., in [9].

## 6.2 Model discretisation using finite difference schemes

In this section, we introduce fast and stable computational methods for model (6.1) by rewriting it using the linearity of the divergence operator as follows

$$\partial_t u = \operatorname{div} (g(|\nabla u - \mathbf{d}u|) \nabla u) - \operatorname{div} (g(|\nabla u - \mathbf{d}u|) \mathbf{d}u), \quad (6.15)$$

along with the boundary conditions and initial condition  $u_0 \in L^\infty(\Omega; \mathbb{R}_+^*)$  in (6.2).

We approximate the solution of (6.15) on a regular grid  $\Omega_h \subset \Omega$  by approximating the space domain  $\Omega$  of the PDE, where  $h > 0$  denotes the spatial grid size along both horizontal and vertical directions.

We first apply a semi-discretization in space on  $\Omega_h$ , and then a full discretization in time on  $(0, T]$  using both explicit and semi-implicit strategies, with the final objective of providing a numerical scheme enjoying stability to any fixed time-step discretization parameter.

The image can be interpreted as a special type of graph, where the vertices coincide with pixels, while edges connect two adjacent pixels and have constant weights equal to  $1/h$ . In this setting, the semi-discretization in space of the PDE model is easily derived from the graph gradient and divergence operator defined in 4.1.2-4.1.3.

However, we proceed maintaining the image setting, which provides an intuitive underlying structure for the data, computing an analog discretization via Finite Difference Method (FDM).

In the following, we thus assume  $u \in \mathbb{R}^{m_x \times m_y}$  to be a discretised image defined on  $\Omega_h$  of size  $N := m_x m_y$  and we denote by  $u_{i,j}$  the value of  $u$  at pixel  $(i, j)$  for  $i = 1, \dots, m_x$ ,  $j = 1, \dots, m_y$ . To avoid singularities appearing due to the special choice of the diffusivity function  $g$  in (6.3), we use an  $\epsilon$ -regularised version defined in terms of  $0 < \epsilon \ll 1$  as  $g(\mathbf{s}) = 1/|\mathbf{s}|_\epsilon$  with  $\mathbf{s} = \nabla u - \mathbf{d}u$ , with

$$|\mathbf{s}|_\epsilon := \sqrt{(s^x)^2 + (s^y)^2 + \epsilon}$$

where  $s^x$  and  $s^y$  denote the horizontal and vertical components of  $\mathbf{s}$ , respectively. Considering the canonical drift term  $\mathbf{d} = \nabla \log v = \nabla v / v$ , the value of  $g$  at  $(i, j)$  is approximated using central finite difference schemes for the discretization of the partial first-order derivatives by

$$g_{i,j} = g(s_{ij}) \quad (6.16)$$

$$s_{ij} = \left( \frac{u_{i+1,j} - u_{i-1,j}}{2h} - \frac{v_{i+1,j} - v_{i-1,j}}{2h} \cdot \frac{u_{i,j}}{v_{i,j}}, \frac{u_{i,j+1} - u_{i,j-1}}{2h} - \frac{v_{i,j+1} - v_{i,j-1}}{2h} \cdot \frac{u_{i,j}}{v_{i,j}} \right).$$

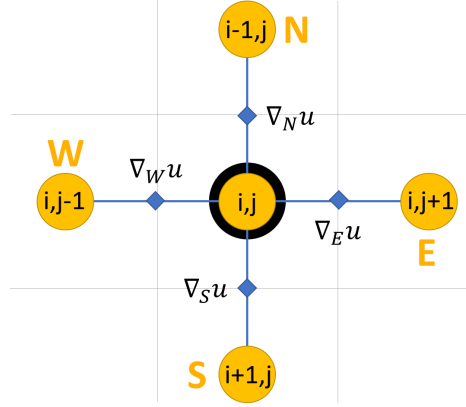


Figure 6.1: Finite difference stencil: the pixel neighborhood of  $(i, j)$  are denoted by south (S), north (N), east (E), west (W).

### 6.2.1 Semi-discretisation in space

The spatial differential operators in (6.15), will be discretized by locating the values of  $\nabla u$ ,  $\mathbf{d} = (d^x, d^y)$ ,  $u$  and the  $\epsilon$ -regularised diffusivity function  $g$  on the edges between two adjacent pixels, following the stencil illustrated in Fig. 6.1. Hence, the semi-discretisation in space of (6.15) at  $(i, j)$  applying second-order central finite differences reads:

$$(\partial_t u)_{ij} = \frac{1}{h} \left[ \left[ \nabla_N u \cdot g_{N_{i,j}} + \nabla_S u \cdot g_{S_{i,j}} + \nabla_E u \cdot g_{E_{i,j}} + \nabla_W u \cdot g_{W_{i,j}} \right] + \right. \\ \left. - \left[ d_N^y \cdot g_{N_{i,j}} \cdot u_{N_{i,j}} + d_S^y \cdot g_{S_{i,j}} \cdot u_{S_{i,j}} + d_E^x \cdot g_{E_{i,j}} \cdot u_{E_{i,j}} + d_W^x \cdot g_{W_{i,j}} \cdot u_{W_{i,j}} \right] \right] \quad (6.17)$$

where each term is defined as in Table 6.1.

Note that the discretization (6.17) also holds on boundary pixels, having preliminarily set  $\mathbf{d} = 0$  on the boundary edges and mirroring the image therein in order to satisfy the Neumann homogeneous boundary condition (6.1).

$\nabla_N u = \frac{u_{i+1,j} - u_{i,j}}{h},$	$\nabla_S u = \frac{u_{i-1,j} - u_{i,j}}{h},$	$\nabla_E u = \frac{u_{i,j+1} - u_{i,j}}{h},$	$\nabla_W u = \frac{u_{i,j-1} - u_{i,j}}{h},$
$g_N = \frac{g_{i+1,j} + g_{i,j}}{2},$	$g_S = \frac{g_{i-1,j} + g_{i,j}}{2},$	$g_E = \frac{g_{i,j+1} + g_{i,j}}{2},$	$g_W = \frac{g_{i,j-1} + g_{i,j}}{2},$
$u_N = \frac{u_{i+1,j} + u_{i,j}}{2},$	$u_S = \frac{u_{i-1,j} + u_{i,j}}{2},$	$u_E = \frac{u_{i,j+1} + u_{i,j}}{2},$	$u_W = \frac{u_{i,j-1} + u_{i,j}}{2},$
$d_N^y = \frac{2(v_{i+1,j} - v_{i,j})}{h(v_{i+1,j} + v_{i,j})},$	$d_S^y = \frac{2(v_{i-1,j} - v_{i,j})}{h(v_{i-1,j} + v_{i,j})},$	$d_E^x = \frac{2(v_{i,j+1} - v_{i,j})}{h(v_{i,j+1} + v_{i,j})},$	$d_W^x = \frac{2(v_{i,j-1} - v_{i,j})}{h(v_{i,j-1} + v_{i,j})}.$

Table 6.1: FD discretization schemes.

We thus rewrite (6.17) by incorporating the discretisations reported in Table 6.1 and collecting the coefficients of the pixel  $(i, j)$  and its neighbours:

$$\begin{aligned}
(\partial_t u)_{i,j} = & u_{i-1,j} \frac{g_S}{h} \left( \frac{1}{h} - \frac{d_S^y}{2} \right) + u_{i+1,j} \frac{g_N}{h} \left( \frac{1}{h} - \frac{d_N^y}{2} \right) + \\
& + u_{i,j-1} \frac{g_W}{h} \left( \frac{1}{h} - \frac{d_W^x}{2} \right) + u_{i,j+1} \frac{g_E}{h} \left( \frac{1}{h} - \frac{d_E^x}{2} \right) + \\
& + u_{i,j} \left[ \frac{g_S}{h} \left( -\frac{1}{h} - \frac{d_S^y}{2} \right) + \frac{g_N}{h} \left( -\frac{1}{h} - \frac{d_N^y}{2} \right) + \frac{g_W}{h} \left( -\frac{1}{h} - \frac{d_W^x}{2} \right) + \frac{g_E}{h} \left( -\frac{1}{h} - \frac{d_E^x}{2} \right) \right].
\end{aligned} \tag{6.18}$$

Note that, when  $d = 0$  and  $g = 1$ , (6.18) reduces to the standard discretization of the Laplacian, with stencil valued  $-4/h^2$  on the central  $(i, j)$  pixel and  $1/h^2$  on its neighbors.

In order to represent (6.18) in matrix-vector form, we first rearrange into a single vector  $u(t) = (u_1(t), \dots, u_N(t))^T \in \mathbb{R}^N$  the values of the evolving image  $u(T)$  at each pixel  $i = 1, \dots, N$ . Then, we write the semi-discretized model as the Cauchy problem:

$$\begin{cases} \frac{du}{dt} = A(u) u(t), \\ u(0) = u_0, \end{cases} \tag{6.19}$$

where  $A(u) \in \mathbb{R}^{N \times N}$  is a non-symmetric sparse matrix, with only five non-zero diagonals and entries given by

$$a_{ij} := \begin{cases} \sum_{k \in \mathcal{N}(i)} \frac{g_k + g_i}{2h^2} \left( -1 - \frac{v_k - v_i}{v_k + v_i} \right) & \text{if } i = j, \\ \frac{g_j + g_i}{2h^2} \left( 1 - \frac{v_j - v_i}{v_j + v_i} \right) & \text{if } i \neq j, j \in \mathcal{N}(i), \\ 0 & \text{if } i \neq j, j \notin \mathcal{N}(i). \end{cases} \tag{6.20}$$



where  $\mathcal{N}(i)$  represents the set of the neighbours in directions  $\{N, E, O, W\}$  of the pixel  $i$  and  $g_i$  is evaluated at pixel  $i$  according to formula (6.16).

The next proposition shows some useful properties of the matrix  $A(u)$ .

**Proposition 6.2.1.** *For every vector  $u \in \mathbb{R}^N$ , let  $A(u) \in \mathbb{R}^{N \times N}$  be the matrix defined by (6.20). Then, the following properties hold:*

1.  $A(u)$  has non-negative off-diagonals,
2. all column sums of  $A(u)$  are 0,
3.  $A(u)$  is irreducible,
4.  $A(u)v = 0$ .

*Proof.* 1) According to (6.20), if pixel  $i$  and pixel  $j$  are not adjacent, then the entry  $a_{ij}$  is zero. On the other hand, as  $g_i, g_j, v_i, v_j > 0$ , then  $a_{ij} > 0$ ,  $\forall i, j$ .

2) Considering an arbitrary column  $l$  of  $A$ , we have:

$$\begin{aligned} \sum_{i=1}^N a_{i,l} &= a_{l,l} + \sum_{j \in \mathcal{N}(l)} a_{j,l} = \\ &= \sum_{j \in \mathcal{N}(l)} \frac{g_j + g_l}{2h^2} \left( -1 - \frac{v_j - v_l}{v_j + v_l} \right) + \sum_{j \in \mathcal{N}(l)} \frac{g_l + g_j}{2h^2} \left( 1 - \frac{v_l - v_j}{v_l + v_j} \right) = 0. \end{aligned}$$

3) Since  $a_{i,j} > 0$  for all adjacent  $i, j$ , so the directed graph  $G_A$  associated to  $A$  is strongly connected and  $A$  is irreducible.

4) The value of the  $ij$  coordinate of the vector  $A(u)v$  is obtained from the right hand side of the expression (6.17), substituting  $u$  with  $v$ . Writing explicitly every term as described in Table (6.1), it is easy to see that they cancel out. Therefore  $A(u)v = 0$ .  $\square$

The properties shown in the lemma above are crucial for the following. Typically, they are proved in the context of symmetric space-discretisation matrices (see, e.g., [130]), but have been shown to hold also in the case of non-symmetric diffusion-transport operators such as the osmosis one, see, e.g., [93, 124].

### 6.2.2 Time discretisation

We present two possible fully discretized approximations of (6.19), obtained by applying Euler integration methods in time. Let  $\tau > 0$  denote a constant time-step and for  $k = 0, 1, \dots$  let  $u^k$  be the approximation of  $u$  at discrete time  $k\tau \in (0, T]$ .

- **Explicit discretisation:** We first consider the simplest space discretisation at the  $k$ th discrete time step that leads to the following full explicit scheme

$$\frac{u^{k+1} - u^k}{\tau} = A(u^k)u^k \quad \implies \quad u^{k+1} = (I + \tau A(u^k))u^k. \quad (\text{E})$$

- **Semi-implicit discretization:** Alternatively, we can consider a semi-implicit scheme where the non-linear terms of the equation are treated from the previous time step, while the linear ones are considered at the current time step; this leads to the semi-implicit scheme

$$\frac{u^{k+1} - u^k}{\tau} = A(u^k)u^{k+1} \quad \implies \quad u^{k+1} = (I - \tau A(u^k))^{-1}u^k. \quad (\text{S.I.})$$

In both cases, the time-stepping process approximating (6.19) can be represented in terms of a matrix  $P \in \mathbb{R}^{N \times N}$  as

$$\begin{cases} u^{k+1} = P(u^k)u^k, \\ u^0 = u_0. \end{cases} \quad (6.21)$$

where

$$P = I + \tau A, \quad \text{in (E),} \quad P = (I - \tau A)^{-1} \quad \text{in (S.I.).} \quad (6.22)$$

We have already encountered the evolution process (6.21) in Sec. 5.5.2, for linear osmosis. In that case, Prop. 5.5.2 shows preservation and convergence results for the explicit and purely implicit schemes, using solely assumptions on the matrix  $A$ , namely: zero column sums, non-negative off-diagonals entries and irreducibility.

As seen in Prop. 6.2.1, the matrix  $A(u)$  defined in (6.20) respects such conditions for all  $u \in \mathbb{R}^n$  and thus we can directly apply Prop. 5.5.2 in our case, for explicit and semi-implicit schemes, obtaining the following Theorem.

**Theorem 6.2.1.** *Let  $f \in (\mathbb{R}_+^*)^N$ . Then, the solution  $u^k$  computed at discrete time  $k\tau$  with  $g$  defined as in (6.16) preserves its mass and non-negativity for any  $k \geq 1$ . Moreover, it converges to the steady-state solution  $u^*$  defined by*

$$u^* = \frac{\mu(u_0)}{\mu(v)}v, \quad (6.23)$$

if

1.  $0 < \tau \leq 1/\max\{|a_{1,1}|, \dots, |a_{N,N}|\}$ , when the explicit scheme (E) is applied
2.  $\tau > 0$ , when the semi-implicit scheme (S.I.) is applied.

Numerically, the proposed semi-implicit scheme (S.I.) can be solved at each iteration by means of any efficient (preconditioned) linear iterative solver suitable for sparse, diagonally dominant M-matrices, such as e.g., SOR (successive over-relaxation) method. Further inquiries on the asymptotic spectral distribution of the sequence of the  $P(u^k)$  matrices may suggest how to construct effective preconditioners.

## 6.3 Numerical results

As described in Section 1.4, upon the specific choice of the drift term  $\mathbf{d}$  in (6.1), the non-linear osmosis model can be applied to the shadow removal (see Section 6.3.1), light-spot removal (see Section 6.3.2) and the compact image representation (see Section 6.3.3) tasks. We will compare qualitatively and quantitatively the results of our approach with the ones obtained by other osmosis-based methods as well as with state-of-the-art approaches. Furthermore, some considerations on the computational efficiency of the proposed models will be made. Reconstruction quality will be measured in terms of the well-known Structural Similarity Index (SSIM  $\in [0, 1]$ ) function between the original non-shadowed image and the reconstructed image, due to its intrinsic dependence on luminance, contrast and structure features. Note that the explicit iterative scheme (E) is practically unusable for most of the examples considered, due to the constraint  $\tau \leq 1/\max\{|a_{1,1}|, \dots, |a_{N,N}|\}$  which for the following examples forced constraints of the order  $\tau < 10^{-8}$ . On the contrary, the unconditional stability of the semi-implicit scheme (S.I.) allows for the use of possibly large values of  $\tau$ . In the examples, we used  $\tau = 10^3$ , which provides convergence and a good level of accuracy only in a few iterations. For the processing of color images, the proposed model is run on each R, G, B channel independently. All algorithms are tested on an AMD Ryzen 5 3450U processor (2.10GHz) with 16GB RAM using MATLAB R2021a.

### 6.3.1 Shadow removal

Recalling the shadow removal problem described in Section 1.4, we test the model by setting the starting point of the osmosis evolution and the reference image as  $u^0 = v = u_0$ , the given image. We remember that for this problem, the drift term  $\mathbf{d}$  is defined as in (5.4). This choice corresponds to considering two different evolutions in the regions composing the image domain which explicitly read

$$\begin{cases} \partial_t u = \operatorname{div} \left( \frac{\nabla u}{|\nabla u|} \right) & \text{on } \Omega_b \times (0, T] \\ \partial_t u = \operatorname{div} \left( \frac{\nabla u - \frac{\nabla u_0}{u_0} u}{\left| \nabla u - \frac{\nabla u_0}{u_0} u \right|} \right) & \text{on } (\Omega_{in} \cup \Omega_{out}) \times (0, T] \end{cases} \quad (6.24)$$

with boundary conditions defined as in (6.2). Note that choosing  $\mathbf{d}$  in a canonical form according (5.4) allows preserving the image content outside  $\Omega_b$ , while diffusing non-linearly, in the form of a Total Variation flow, on  $\Omega_b$  itself. The intensity balancing effect in the regions  $\Omega_{in}$  and  $\Omega_{out}$  depends solely on the evolution equation on  $\Omega_b$ , the region where the drift term is set to zero. Moreover, the diffusivity function  $g$  allows tuning

the diffusion intensity on the pixels in  $\Omega_b$  depending on the value of  $\nabla u$ . In particular,  $g$  slows down the diffusion on the pixels with high gradient values, which correspond to edges that need to be preserved.

We test the performance of the proposed non-linear shadow removal model both in the case of hard shadows and for more realistic soft-edged shadows.

In the case of hard shadows, synthetic examples  $f$  can be easily generated by a constant multiplicative rescaling of a ground-truth image  $f^*$  only in correspondence of a bounded region  $S \subset \Omega$ . Denoting by  $c \in (0, 1)$  the (unknown) constant loss of luminosity on  $S$ , we can construct first the shadow image by setting:

$$s = \begin{cases} c & \text{on } S \\ 1 & \text{on } \Omega \setminus S, \end{cases} \quad (6.25)$$

and obtain a shadowed version  $f$  of  $f^*$  by Hadamard (point-wise) multiplication

$$f = \begin{cases} f^* \odot s & \text{(hard shadow)} \\ f^* \odot (G_\sigma * s) & \text{(soft shadow)} \end{cases} \quad (6.26)$$

where ‘ $*$ ’ denotes the standard convolution product and  $G_\sigma$  is a Gaussian convolution kernel with standard deviation  $\sigma \geq 0$ . Such convolution allows to generate an area of penumbra around shadow boundaries, which is often the case in real-world images where the shadow edges can not be precisely distinguished.

Fig. 6.2 shows the results of the proposed non-linear osmosis model solved by means of the semi-implicit scheme on two hard-shadowed ( $\sigma = 0$ ) images. For such images, it is easy to distinguish whether a pixel belongs to the shadowed or to the un-shadowed region. Hence, we can use a thin (2-pixel wide) mask covering one pixel from each side of the boundary between the two regions. This choice is sufficient to accurately remove the shadow while preserving all features on the shadow boundary.

Fig. 6.3 shows the results obtained for the soft-edged shadow removal problem in correspondence of two different masks (wide and thin) and three input images where shadows are characterized by penumbra boundaries of increasing dimension (from left to right) corresponding to different increasing values of  $\sigma \in \{0, 1, 2\}$  in (6.26). As seen before, the thin mask is a good choice for the case of hard shadows (i.e.  $\sigma = 0$ ), but it becomes less effective as  $\sigma$  increases, i.e. as the penumbra region becomes wider, as it cannot cover appropriately the boundary between the two regions  $\Omega_{in}$  and  $\Omega_{out}$ . In these cases, it is necessary to enlarge the shadow mask. Using a wider one, we observe that the shadows are properly removed and features are well-preserved, in particular near the pixels with limited intensity variation.

Note, however, that using a large mask may result in some drawbacks, as shown in Fig. 6.4, where a wide (6-pixel) mask is used. As this choice enlarges the region where non-linear TV-type diffusion is induced, typical TV drawbacks are observed. For instance,

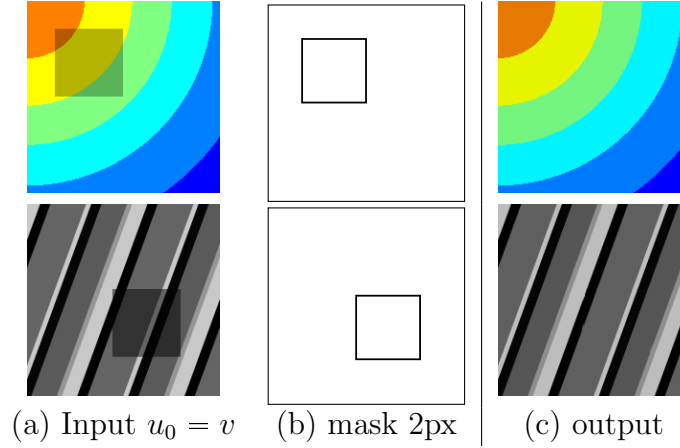


Figure 6.2: Hard-shadow removal.

we observe in Fig. 6.4(b) that some artifacts appear in the attempt to connect circular structures (see the boundary between the yellow and the green region). Furthermore, as it can be observed in Fig. 6.4(d), when the shadow mask is too large some undesired connectivity effect is observed. As it is well-known, the use of TV as a regulariser in inpainting problems forces the minimization of the length of level lines causing artifacts, a property that can be seen through the use of the standard coarea formula [108].

A possible remedy for these two drawbacks consists in employing a different diffusivity function  $g$ . As we remarked in Corollary 6.1.1, a slight variation of the non-linear model depending on a diffusivity function  $g_p$  defined in terms of a constant parameter  $p \in [1, 2)$  as in (6.12) can be considered. The use of higher values of  $p$  helps to better identify the regions of the shadow boundary that are crossed by the underlying edges of the image. In fact, since the edges represents regions with high gradient values, the diffusivity functions  $g_p(u) = 1/|\nabla u|^p$  slows down the diffusion effect, thus favoring edge preservation. On the other hand, diffusion is accelerated in constant or smooth regions.

In Figure 6.5 we show some results with  $p = 1$  (second row) and  $p = 1.9$  (third row) on the images of Fig. 6.4 and on two other examples. The use of a possible space-variant strategy adapting the value of  $p$  to the local image content would mitigate the previously observed drawbacks and, as such, is an interesting direction for future research.

We now compare qualitatively and quantitatively our results with the ones obtained by means of the standard linear osmosis model (5.2) and its anisotropic variant proposed in [93].

In Fig. 6.6, some examples of hard shadow removal are shown. We notice that the linear model is effective in removing the shadow, but produces visible blurring on the shadow boundary. The anisotropic model is able to connect well the underlying features of the image, but does not well balance the image intensity between the inside and the outside of the shadow. Our model works well for both tasks, achieving good SSIM values

in comparison with the original ground truth images.

In Fig. 6.7 we compare the results obtained by the three different models on soft-shadowed images and real images. We observe that around the mask region our model performs as well as the anisotropic one. However, while the anisotropic model tends to make the whole image slightly blurred (due to the smoothing kernels required to define the diffusion tensor  $\mathbf{W}$ ), the non-linear model preserves the sharpness of the image everywhere.

We now compare the computational efficiency required to compute the numerical solution of the non-linear model by using the semi-implicit scheme (S.I.) with the linear [129] (solved with a fully implicit scheme with  $\tau = 10^3$ ), and the anisotropic model [93] (solved by exponential integration).

In this respect, we report in Table 6.2 the execution time (in seconds) and the number of iterations (in brackets) for the three models when applied to the images of Figure 6.6

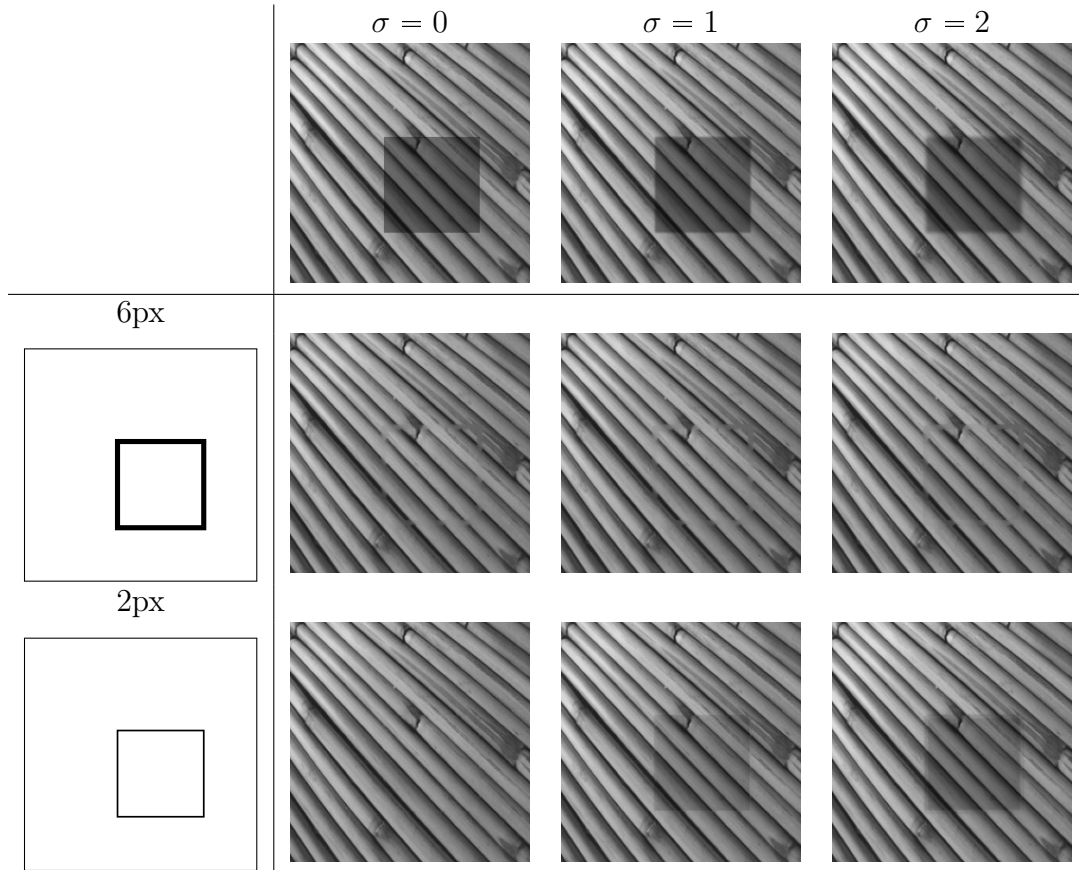


Figure 6.3: Shadow removal on input images  $f$  corrupted by a constant shadow convolved with convolution kernel  $G_\sigma$  with increasing standard deviation  $\sigma$ . Masks have different thicknesses, 6px and 2px.

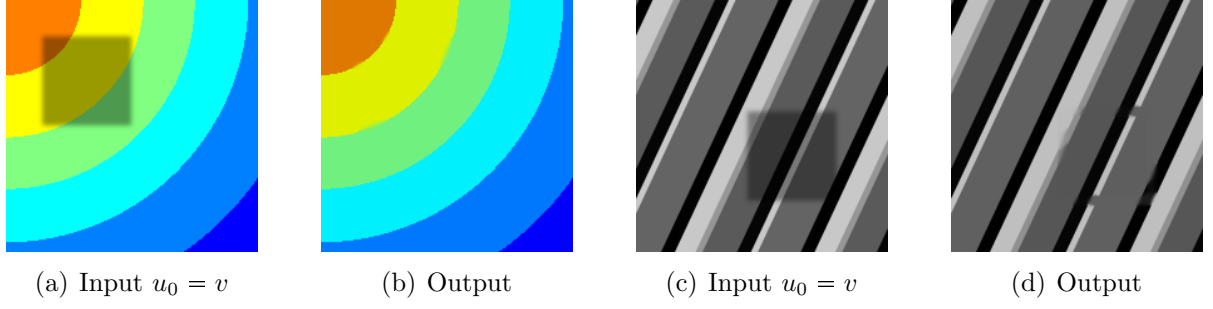


Figure 6.4: Drawbacks of soft-shadow removal using a too thick (6 pixels wide) mask.

and 6.7, using as stopping criteria a mean squared error w.r.t. to the ground truth smaller than a given tolerance for artificially corrupted images (a,b,c,d), and a sufficiently small ( $10^{-3}$ ) relative change for real-world images (e,f,g). From a visual inspection of Fig. 6.7, we can observe that the results obtained by the anisotropic model are qualitatively comparable with the non-linear ones, although the former require a much more significant

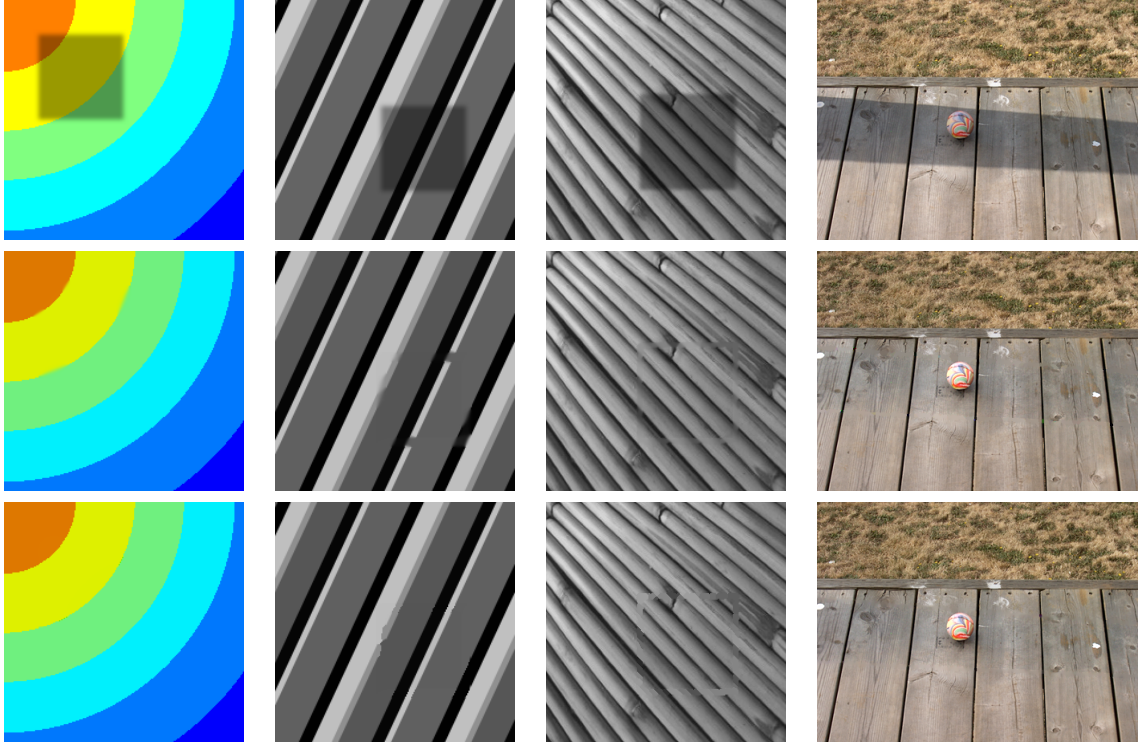


Figure 6.5: Soft-shadow removal using the diffusivity function  $g_p$  with  $p = 1$  (second row) and  $p = 1.9$  (third row).

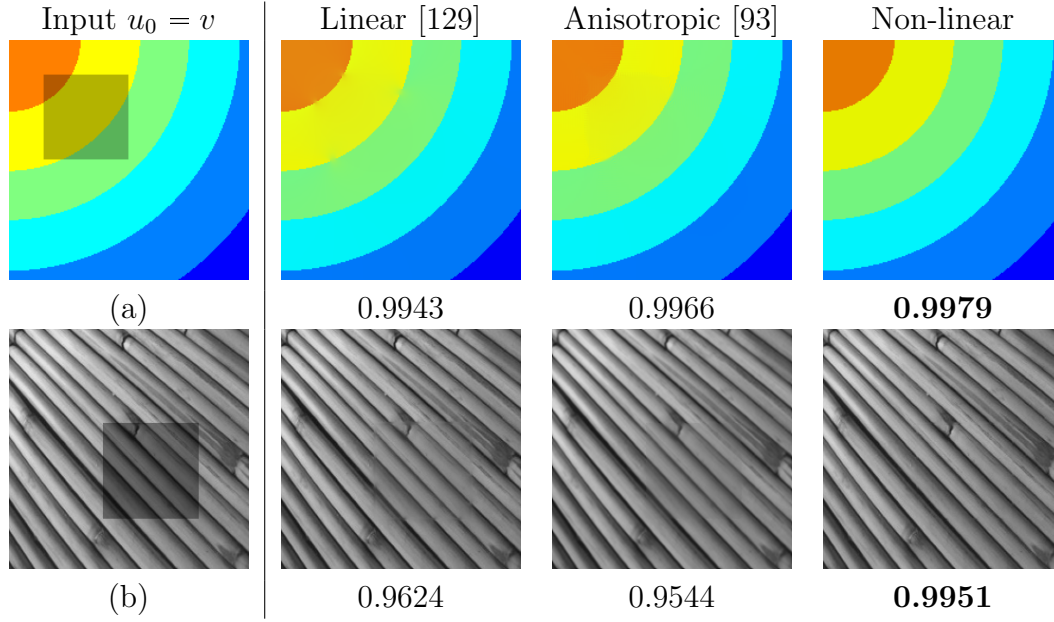


Figure 6.6: Comparison between linear, anisotropic and non-linear osmosis models on hard-shadowed images, with associated SSIM values.

computational effort, see Table 6.2. This is due to the more sophisticated nature of the anisotropic model which requires the computation of the matrix field  $\mathbf{W}$  and its possible update at every iteration. Interestingly, the proposed non-linear model appears faster than the linear model in reaching convergence. This is due to the action of the diffusivity term  $g$ , which accelerates the osmosis process on  $\Omega_{in} \cup \Omega_{out}$ .

Finally, we provide an extensive list of comparisons with alternative approaches proposed for shadow removal, all relying on a separate treatment of the three color (R,G,B) channels. Illuminant invariant approaches can be used (as, e.g., in [47, 46]) to extract an "intrinsic" image from the input image so that shadow removal is treated as a reintegration problem using this image as a guide to derive gradient information or similarity measures between shadowed and lighted regions. In the context of image decomposition, a variational approach to shadow and spot-light removal has been proposed in [57]. Color transfer techniques, such as [76, 112], make use of gradient information to recover the shadow-free image by solving a Poisson equation. Other methods consist in scaling the shadow region by a suitable factor, modeling the shadow as uniform [49] or even non-uniform [4]. Another class of solvers treats the task as a matting problem, considering the shadow pixels as foreground and the lit pixels as background [133, 134].

In Fig. 6.8 we report a comparison between the results obtained by the proposed non-linear osmosis model and three of the most recent and successful methods for shadow



	Linear [129]		Anisotropic [93]		Non-linear	
(a)	3.98	(7)	27.39	(4)	<b>1.25</b>	<b>(3)</b>
(b)	6.44	(19)	29.71	(5)	<b>1.45</b>	<b>(4)</b>
(c)	11.54	(20)	20.77	<b>(2)</b>	<b>1.80</b>	(3)
(d)	8.66	(26)	18.70	<b>(3)</b>	<b>3.53</b>	(8)
(e)	31.63	(9)	1307.06	(10)	<b>10.37</b>	<b>(3)</b>
(f)	105.55	(30)	2592.00	(18)	<b>27.08</b>	<b>(7)</b>
(g)	19.14	(9)	723.48	(10)	<b>8.62</b>	<b>(4)</b>

Table 6.2: Execution time (in seconds) and number of iterations till convergence (in brackets) for results in Fig. 6.6 (a,b) and Fig. 6.7 (c,d,e,f,g).

removal<sup>2</sup>: a deep learning approach, [71], a region-based shadow detection and removal [52] and, finally, a variational approach based on a non-local version of the osmosis model, [10] followed by a further contrast correction step as described by the authors. We observe that our method produces visually pleasant results compared to state-of-the-art methods and more sophisticated variants of the standard osmosis model [10], which further requires a post-processing contrast correction step. Quantitatively, it outperforms all other approaches in terms of SSIM values.

### 6.3.2 Spot-light removal

Recalling Section 1.4, an analogous use of the osmosis model (6.24) can be done for the spot-light removal problem. Here, a synthetic spot-lighted image  $f$  is obtained by Hadamard product  $f^* \odot (G_\sigma * \ell)$  between a light- spot-free image  $f^*$  and a light image (within a bounded lighted region  $L \subset \Omega$ )  $\ell$  defined as

$$\ell = \begin{cases} c & \text{on } L \\ 1 & \text{on } \Omega \setminus L, \end{cases} \quad (6.27)$$

with  $c > 1$ . Positive values of the standard deviation  $\sigma$  of the convolution kernel  $G_\sigma$  produce soft-lighted images, which require a larger mask. Fig. 6.9 shows the results for four increasing levels of  $\sigma$ . Analogous comments to the ones made in the case of shadow removal applications can be done in this case, too.

### 6.3.3 Compact data representation

As recalled in Section 1.4, image osmosis can also be applied to the problem of compact data representation, by means of a proper definition of the drift term. Given an image  $v$

<sup>2</sup>The test images have been downloaded from the ISTD dataset, available at <https://github.com/DeepInsight-PCALab/ST-CGAN>.

with average grey value  $\mu_v$ , we thus consider  $M = \Omega_b$  to be a pre-computed edge-mask via standard segmentation algorithms (Canny, Sobel, ...) and define the drift term as in (5.5). Explicitly, the two equations guiding the osmosis evolution in this case then read:

$$\begin{cases} \partial_t u = \operatorname{div} \left( \frac{\nabla u}{|\nabla u|} \right) & \text{on } (\Omega \setminus \Omega_b) \times [0, T] \\ \partial_t u = \operatorname{div} \left( \frac{\nabla u - \frac{\nabla v}{v} u}{|\nabla u - \frac{\nabla v}{v} u|} \right) & \text{on } \Omega_b \times [0, T] \end{cases} \quad (6.28)$$

with initial and boundary conditions defined as in (6.2). By setting  $u_0 \equiv \mu_v$  on  $\Omega$  and making use of the actual values of the reference image  $v$  only on  $\Omega_b$  to define the drift term  $\mathbf{d}$  therein, by evolving (6.28) the information is diffused on  $\Omega \setminus \Omega_b$ , which can be approximated as the union of piecewise-constant regions in the image. The process thus converges to an approximation  $w$  of the reference image  $v$ .

Fig. 6.10 compares the results obtained via linear and non-linear osmosis. In general, the reconstructed images seem to have a lower contrast than the original ones, but the non-linear model favors a better approximation, as confirmed by the SSIM values, in particular on strongly piecewise-constant images.

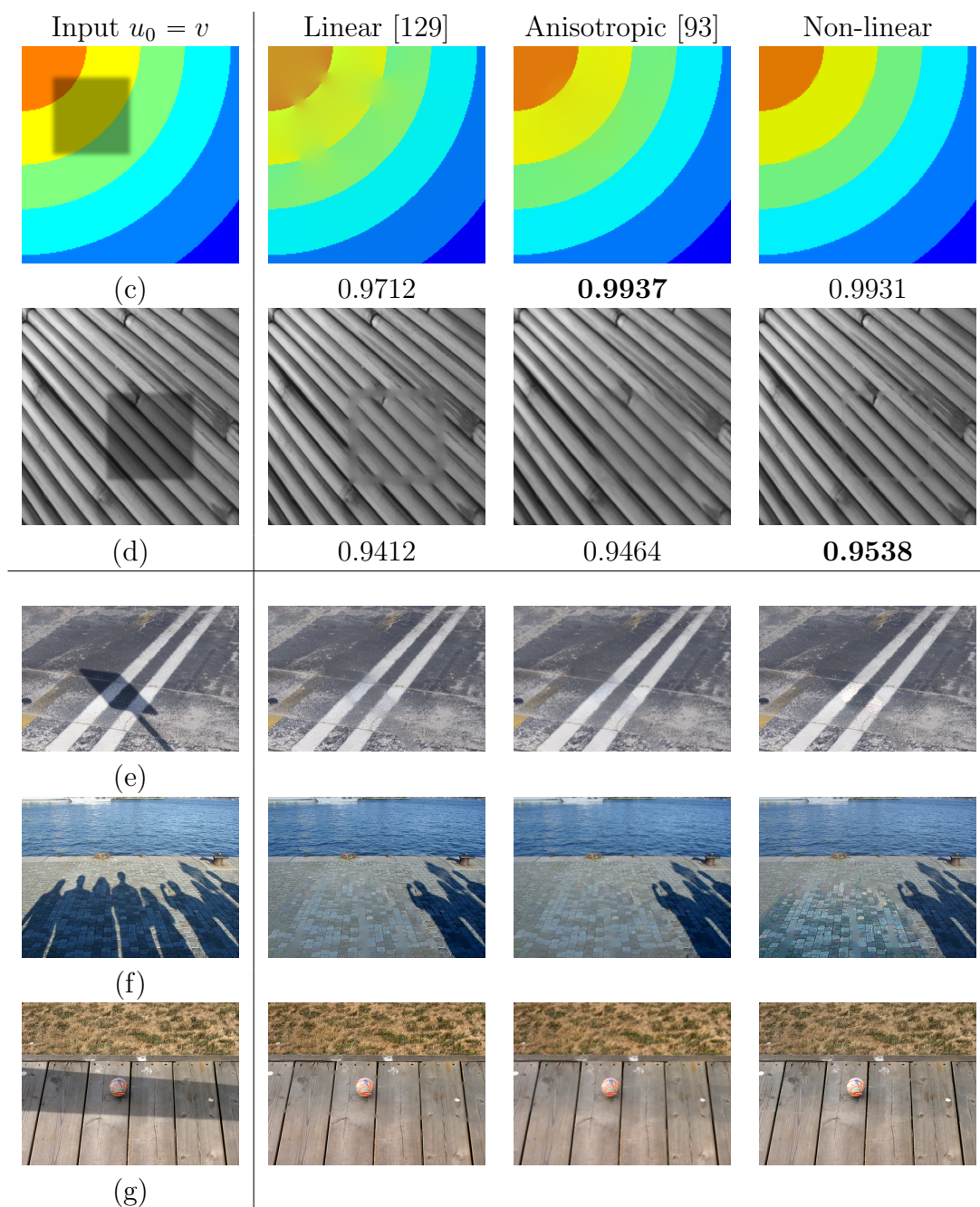


Figure 6.7: Results of osmosis models on artificially soft-shadowed images (with associated SSIM computed from the ground-truth unshadowed image) and natural images.






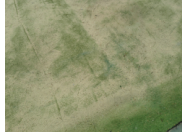


























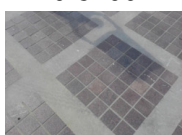
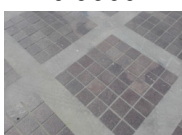
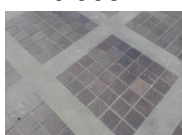
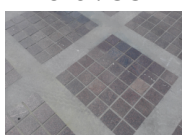
					
		0.9172	0.9603	0.9667	<b>0.9676</b>
					
		0.7816	0.9482	0.9557	<b>0.9726</b>
					
		0.8288	0.9092	0.9350	<b>0.9488</b>
					
		0.8335	0.9312	0.9265	<b>0.9650</b>
					
		0.8106	0.9500	0.9531	<b>0.9733</b>
					
		0.6657	0.8174	0.8134	<b>0.9008</b>
Original	Input $u_0 = v$	[52]	[71]	[10]	ours

Figure 6.8: Comparison between shadow removal approaches, with associated SSIM values.

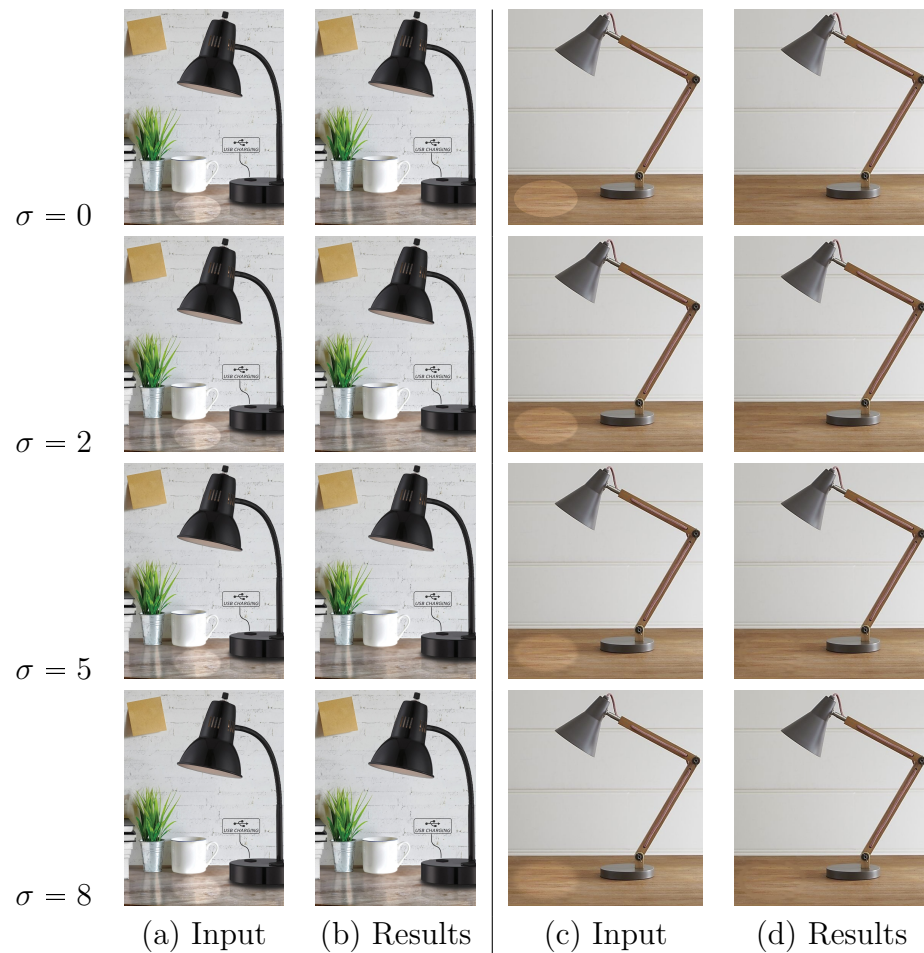


Figure 6.9: Spot-light removal results. Input images were obtained by convolving the light-free image with a Gaussian kernel with variance  $\sigma$ , whose values are reported in the left column.



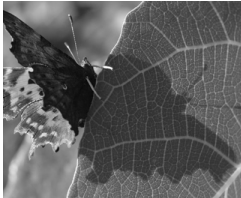

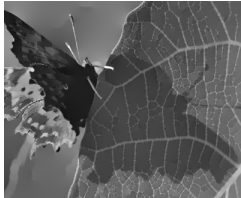
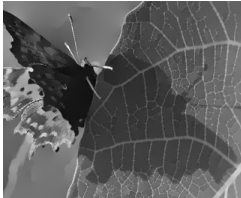













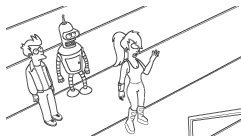


			
		0.8581	<b>0.8632</b>
			
		0.9432	<b>0.9626</b>
			
		0.9058	<b>0.9453</b>
			
		0.8827	<b>0.9160</b>
			
		0.9671	<b>0.9814</b>
Input $v$	Edge mask	Linear [124]	Non-linear

Figure 6.10: Compact data representation results with relative SSIM values. Edge mask has been computed by standard Canny segmentation.



# Chapter 7

## Linear PDE models for 3D mesh osmotic flow

In Chapter 5, we have described how to apply osmotic PDE flow to process functions defined on graphs  $G$ . The spatial domain (the graph  $G$  associated with a mesh  $\mathcal{M}$  representing a surface  $\mathcal{S} \subset \mathbb{R}^3$ ) does not vary during the evolution.

In this Chapter, instead, we aim at applying osmotic flow (5.18) directly to the Euclidean or differential coordinate functions that encode the structure of the spatial domain (the mesh). As a consequence of this, the shape of the mesh itself changes during the evolution.

This effect can be exploited to perform mesh processing tasks. In particular, we propose two frameworks exploiting osmotic flow for mesh cloning and inpainting tasks.

### 7.1 Cloning via osmosis in Euclidean coordinates

Cloning task has been introduced in 1.3.4, as the operation that replaces a region of a given surface  $\mathcal{S}_1$  with a region of another surface  $\mathcal{S}_2$ , with no visible discontinuities.

Starting from the given meshes  $\mathcal{M}_1 = (V_1, E_1, T_1)$  and  $\mathcal{M}_2 = (V_2, E_2, T_2)$ , the cloning task is formulated in (1.45) and illustrated in Fig. 7.1.

We denote the region of interest  $ROI$  and the patch-to-be-cloned  $P$  respectively as

$$ROI \subset \mathcal{M}_1, ROI = (V_{ROI}, E_{ROI}, T_{ROI}); \quad P \subseteq \mathcal{M}_2, P = (V_P, E_P, T_P) \quad (7.1)$$

On the surface  $\mathcal{M}_1$ , see Fig. 7.1 (a), the region of interest  $ROI$  is highlighted in yellow color. The surface  $\mathcal{M}_2$  is reported in Fig. 7.1 (b), with the patch  $P$  yellow-colored. The resulting mesh, illustrated in Fig. 7.1 (c), is defined as

$$\mathcal{M}^* = (\mathcal{M}_1 \setminus ROI) \cup ROI^* \quad (7.2)$$

where  $ROI^* = (V^*, E^*, T^*)$  is obtained by cloning the geometric information of  $P$  into  $ROI$ .



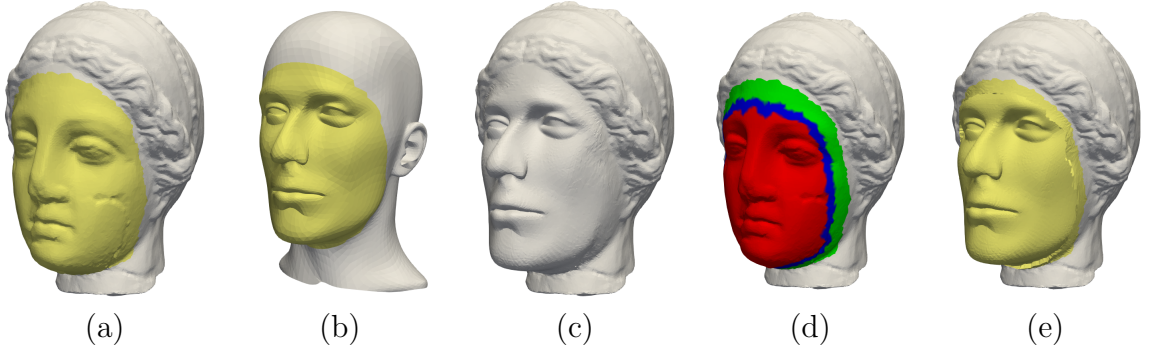


Figure 7.1: Osmosis process for mesh cloning. (a)  $\mathcal{M}_1$  with  $ROI$  (in yellow); (b)  $\mathcal{M}_2$  and  $P$  (in yellow); (c) cloned mesh  $\mathcal{M}^*$ ; (d) Partitions  $\Omega_{Ext}$  (green),  $\Omega_{Tr}$  (blue) and  $\Omega_{Int}$  (red) mapped onto  $ROI$ ; (e) osmosis initialization  $u^0$ .

Usually, for 2D cloning of images, the osmosis process acts between two intensity functions defined on the same structured image domain, as described in 5.1.

Unlike the image case, which is based on a single 2D-regular grid domain, in the mesh case the two distinct triangular meshes  $ROI$  and  $P$  are defined on different parametric domains. In particular, the Euclidean coordinate functions representing  $ROI$  and  $P$  are

$$\begin{aligned} f_{ROI} : V_{ROI} &\rightarrow \mathbb{R}^3, & f_{ROI}(v_i) &= v_i \quad \forall v_i \in V_{ROI}, \\ f_P : V_P &\rightarrow \mathbb{R}^3, & f_P(v_i) &= v_i \quad \forall v_i \in V_P. \end{aligned} \quad (7.3)$$

In order to perform the osmotic flow, we need to construct a common triangulation that will serve as the initialization for the process. At this aim, we propose a three-step procedure, consisting of: S1) the alignment of the meshes  $ROI$  and  $P$ ; S2) their parametrization on the same domain; S3) the construction of the fused triangulation.

S1) *ROI and P alignment and scaling.* The cloning task requires that  $ROI$  and  $P$  have similar sizes and respect a specific relative orientation. This repositioning is mainly driven by the user's desired result. Thus the patch  $P$  can be manually rotated, translated and scaled by the user.

However, a semi-automatic repositioning and scaling procedure can facilitate the process. First, the patch  $P$  is scaled by a factor  $|\partial ROI|/|\partial P|$ , with  $|\cdot|$  being the length of the boundary. Then, the patch  $P$  is rotated to align two pairs of user-selected adjacent vertices  $(v_1^{ROI}, v_2^{ROI})$ ,  $(v_1^P, v_2^P)$ , respectively on  $\partial ROI$  and on  $\partial P$  and finally translated to the center of mass of  $ROI$ .

S2) *Construction of  $X_{ROI}$  and  $X_P$  parametrizations.* Both submeshes  $P$  and  $ROI$  are conformally parameterized on the same parametric domain  $\Omega = [0, 1]^2$ ,

$$X_{ROI} : \Omega \rightarrow V_{ROI}, \quad X_P : \Omega \rightarrow V_P. \quad (7.4)$$

Vertices on the boundaries of  $\partial P$  and  $\partial ROI$  are associated with corresponding points on the boundary  $\partial\Omega$  of the parametric domain. We denote the points on the domain  $\Omega$ , corresponding to the vertices of  $ROI$  and  $P$ , as the sets

$$\bar{V}_{ROI} = \{\bar{v}_i^{ROI} \in \Omega, i = 1, \dots, n_{ROI}\}, \quad \bar{V}_P = \{\bar{v}_i^P \in \Omega, i = 1, \dots, n_P\}, \quad (7.5)$$

respectively. The parametric points  $\bar{V}_{ROI}$  and  $\bar{V}_P$  define two planar meshes contained in  $\Omega$ :

$$ROI^\Omega = (\bar{V}_{ROI}, E_{ROI}, T_{ROI}); \quad P^\Omega = (\bar{V}_P, E_P, T_P). \quad (7.6)$$

S3) *Construction of the common triangulation.* The common triangulation  $\overline{ROI}^\Omega = (\bar{V}, E, T)$  is built directly in  $\mathbb{R}^2$  on the parametric domain  $\Omega$  as an enrichment of the triangulation  $P^\Omega$  with the new set of vertices  $\bar{V}_{ROI}$ , such that  $\bar{V} = \bar{V}_P \cup \bar{V}_{ROI}$ . The connectivity  $T$  is initialized with  $T_P$ , and enriched as follows.

For each  $\bar{v}_i^{ROI} \in \bar{V}_{ROI}$  located in a triangle  $\tau = (\bar{v}_j^P, \bar{v}_k^P, \bar{v}_\ell^P) \in T$ ,  $\tau$  is replaced by three triangles, each characterized by an edge of  $\tau$  and the new vertex  $\bar{v}_i^{ROI}$ .

Before applying the linear osmosis model (5.18), we need to define the initialization function  $u_0$  and the drift vector field  $\mathbf{d}$ . At this aim, we define the two reference functions  $u_{ROI}$  and  $u_P$ , which interpolate the Euclidean coordinates of the vertices  $\bar{V}$ .

In particular,  $u_{ROI} : \bar{V} \subset \Omega \rightarrow \mathbb{R}^3$  is defined as

$$u_{ROI}(\bar{v}) = \begin{cases} X_{ROI}(\bar{v}) & \text{if } \bar{v} \in \bar{V}_{ROI} \\ \alpha X_{ROI}(\bar{v}_j^{ROI}) + \beta X_{ROI}(\bar{v}_k^{ROI}) + \gamma X_{ROI}(\bar{v}_\ell^{ROI}) & \text{if } \bar{v} \in \bar{V}_P \end{cases} \quad (7.7)$$

where  $(\alpha, \beta, \gamma)$  are the barycentric coordinates of  $\bar{v}$  in  $ROI^\Omega$ . Analogously, we define  $u_P : \bar{V} \subset \Omega \rightarrow \mathbb{R}^3$  as

$$u_P(\bar{v}) = \begin{cases} X_P(\bar{v}) & \text{if } \bar{v} \in \bar{V}_P \\ \alpha X_P(\bar{v}_j^P) + \beta X_P(\bar{v}_k^P) + \gamma X_P(\bar{v}_\ell^P) & \text{if } \bar{v} \in \bar{V}_{ROI} \end{cases} \quad (7.8)$$

with  $(\alpha, \beta, \gamma)$  the barycentric coordinates of  $\bar{v}$  in  $P^\Omega$ .

In the image context, the drift term  $\mathbf{d}$  is set differently in three distinct regions of the domain, see (5.6). These regions identify the external part, where the original image is preserved, the internal part, where the information is replaced, and a transition region, where a smooth connection is expected. Following the same principle, the sub-mesh  $\overline{ROI}^\Omega$  can be partitioned into

$$\overline{ROI}^\Omega = \Omega_{Ext} \sqcup \Omega_{Tr} \sqcup \Omega_{Int}, \quad (7.9)$$

where  $\Omega_{Ext}$  is the external part,  $\Omega_{Tr}$ , the transition part and  $\Omega_{Int}$ , the internal part. The partitioning can be obtained by considering from  $\partial\Omega$  the first  $n_{Ext}$  inner triangle layers to be included in  $\Omega_{Ext}$ , the next  $n_{Tr}$  triangle layers in  $\Omega_{Tr}$  and the remaining triangles to be in  $\Omega_{Int}$ . The partition is illustrated in Fig. 7.1 (d), where the mapping of the partitions in  $\mathbb{R}^3$  is shown in green, blue and red colors for  $\Omega_{Ext}$ ,  $\Omega_{Tr}$ ,  $\Omega_{Int}$  respectively.

The reference functions  $u_P$  and  $u_{ROI}$ , defined on the same domain  $\overline{ROI}^\Omega$ , encode the geometric information to be processed.

Since the osmosis model requires strict positivity of the involved data functions, if  $u_{ROI}$  and  $u_P$  are not positive, a simple translation by a constant offset vector  $m \in \mathbb{R}^3$  is preliminarily applied, as

$$u_{ROI} \leftarrow u_{ROI} - m, \quad u_P \leftarrow u_P - m, \quad \text{with } m = \min_{V \in \mathbb{R}^3} \{V_{ROI}, V_P\} - \varepsilon, \quad \varepsilon > 0. \quad (7.10)$$

The drift term is then defined for each distinct partition as

$$d_{[i,j]} := \begin{cases} \frac{(\nabla u_{ROI})_{[i,j]}}{(u_{ROI})_{[i,j]}} & \text{if } \bar{v}_i \wedge \bar{v}_j \in \Omega_{Ext} \\ \frac{(\nabla u_P)_{[i,j]}}{(u_P)_{[i,j]}} & \text{if } \bar{v}_i \wedge \bar{v}_j \in \Omega_{Int} , \\ \left( \frac{(\nabla u_{ROI})_{[i,j]}}{(u_{ROI})_{[i,j]}} + \frac{(\nabla u_P)_{[i,j]}}{(u_P)_{[i,j]}} \right) / 2 & \text{if } \bar{v}_i \vee \bar{v}_j \in \Omega_{Tr} \end{cases} \quad (7.11)$$

where  $(u_P)_{[i,j]} = (u_P(\bar{v}_i) + u_P(\bar{v}_j))/2$  and  $(u_{ROI})_{[i,j]} = (u_{ROI}(\bar{v}_i) + u_{ROI}(\bar{v}_j))/2$ .

Denoting by  $\chi_{\Omega_{Ext}}$  the characteristic function of the set  $\Omega_{Ext}$ , we define the initial function  $u^0 : \bar{V} \rightarrow \mathbb{R}^3$  as

$$u^0 = u_{ROI} \chi_{\Omega_{Ext}} + u_P (1 - \chi_{\Omega_{Ext}}). \quad (7.12)$$

The evolution of the osmotic flow converges to a vector steady-state function  $u^* : \bar{V} \rightarrow \mathbb{R}^3$ . Finally, the cloned submesh  $ROI^* = (V^*, E^*, T^*)$  which replaces the initial  $ROI$  as in (1.45) is obtained by setting

$$V^* := u^* - m, \quad E^* = E, \quad T^* = T. \quad (7.13)$$

### 7.1.1 Numerical Results

In Algorithm 5, we summarize the proposed method for mesh cloning, which includes the three-step pre-processing phase.

Two parameters  $tol > 0$  and  $k_{max} \in \mathbb{N}$  are used to stop the osmotic process as soon as one of the two following conditions is met:

$$\frac{\|u^{k+1} - u^k\|_2}{\|u^k\|_2} < tol, \quad k > k_{max} \quad (7.14)$$

**Algorithm 5** Mesh Cloning via Euclidean Osmosis

---

**Input:** · mesh  $\mathcal{M}_1$ , patch  $ROI \subset \mathcal{M}_1$ ,  
· mesh  $\mathcal{M}_2$ , patch  $P \subseteq \mathcal{M}_2$ ,  
**Output:** · cloned mesh  $\mathcal{M}^*$ .  
**Parameters:** ·  $tol > 0$ ,  $k_{max} \in \mathbb{N}$ ,  $\tau > 0$

**Preliminary set up:**  
S1) Alignment and scaling of  $ROI$  and  $P$ ,  
S2) Construct  $X_{ROI}$  and  $X_P$  for  $ROI$  and  $P$  on  $\Omega = [0, 1]^2$   
S3) Construct the triangulation  $\overline{ROI}^\Omega = (\bar{V}, E, T)$

**Osmotic flow between  $u_{ROI}$  and  $u_P$ :**  
- Compute  $u_{ROI}$  and  $u_P$  as in (7.7-7.8)  
- Compute partition of  $\overline{ROI}^\Omega$  as in (7.9)  
- Compute and apply offset  $m$ , if necessary, as in (7.10)  
- Set drift term  $\mathbf{d}$  as in (7.11) and initialize  $u^0$  as in (7.12)  
while stopping criterion in (7.14) is not satisfied  
     $u^{k+1} \leftarrow Pu^k$  via (EXP) or (IMP) scheme  
end  
- Set  $ROI^* = (u^{k+1} - m, E, T)$   
- Set  $\mathcal{M}^* = (\mathcal{M}_1 \setminus ROI) \cup ROI^*$

---

where  $u^{k+1}$  and  $u^k$  are consecutive time steps of  $u$  in the explicit or implicit scheme. The implicit scheme is superior thanks to its unconditional stability and convergence properties, shown in Prop.5.5.2. The choice of a high value for the time-step  $\tau > 0$  (the third and last parameter) allows for fast convergence in a few time steps.

Figure 7.1 shows the cloning process applied to replace the face region of the **igea** mesh (a) with a patch from a **mannequin** mesh (b). The cloned surface, shown in (c), does not present any discontinuities or fractures, having smooth behavior in the transition region  $\Omega_{Tr}$ .

In the second example, illustrated in Fig.7.2, we show how the partition of  $\overline{ROI}^\Omega$  defined in (7.9) affects the final result by selecting different widths of  $\Omega_{Ext}$  (in green) and  $\Omega_{Tr}$  (in blue), see (c), (e). As expected, enlarging  $\Omega_{Ext}$  allows to preserve geometric information coming from  $ROI$  (f), while a larger  $\Omega_{Tr}$  facilitates a smoother transition.

Another example of mesh cloning is shown in Fig. 7.3. The **human** torso (b) in yellow is cloned onto the **horse** mesh (a) to obtain a **centaur** mesh (d), with a smooth connection in the transition region in blue in (c).

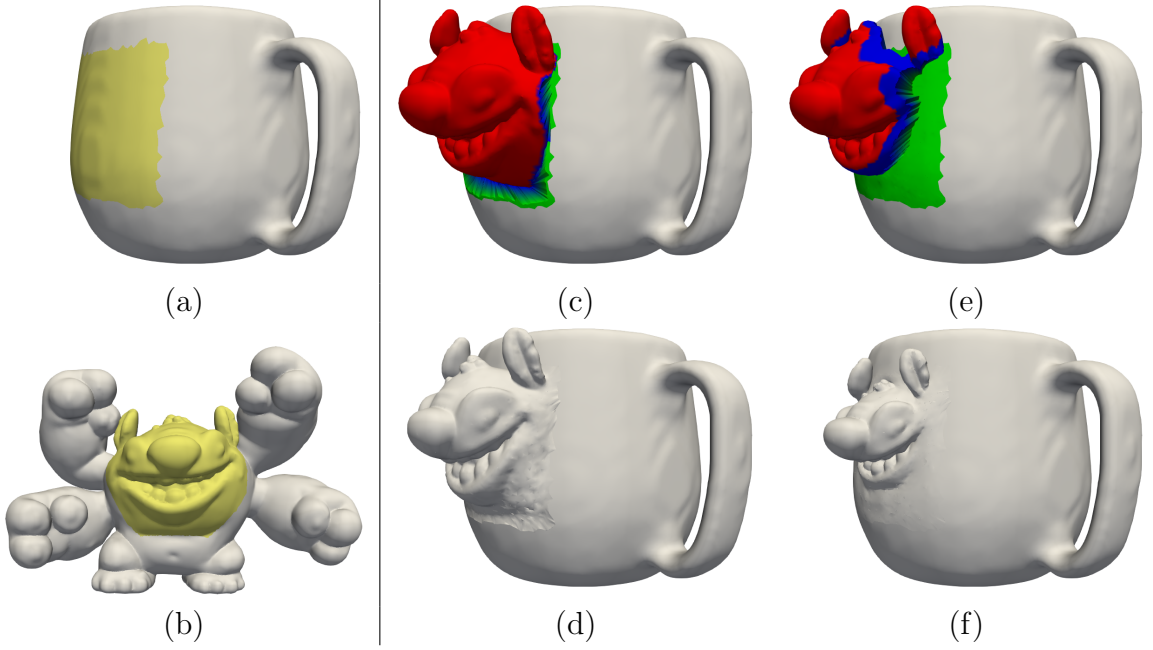


Figure 7.2: Mesh cloning. Influence of  $\Omega_{Ext}$  and  $\Omega_{Tr}$  width : (a)  $\mathcal{M}_1$  with  $ROI$  (yellow); (b)  $\mathcal{M}_2$  with  $P$  (yellow); (c) initialization  $u^0$ , with thin  $\Omega_{Ext}$  (green) and  $\Omega_{Tr}$  (blue) and (d) final result  $\mathcal{M}^*$ ; (e) initialization  $u^0$  with wide  $\Omega_{Ext}$  (green) and  $\Omega_{Tr}$  (blue) and (f) final result  $\mathcal{M}^*$ .

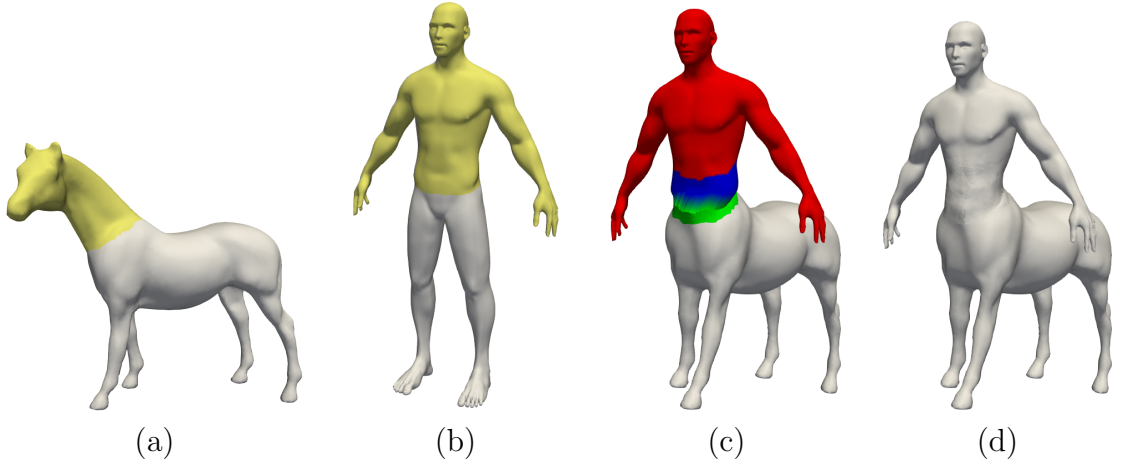


Figure 7.3: Mesh cloning: (a)  $\mathcal{M}_1$  with  $ROI$  (yellow); (b)  $\mathcal{M}_2$  with  $P$  (yellow); (c) initialization  $u^0$ , with  $\Omega_{Ext}$  (green) and  $\Omega_{Tr}$  (blue) and (d) final result  $\mathcal{M}^*$

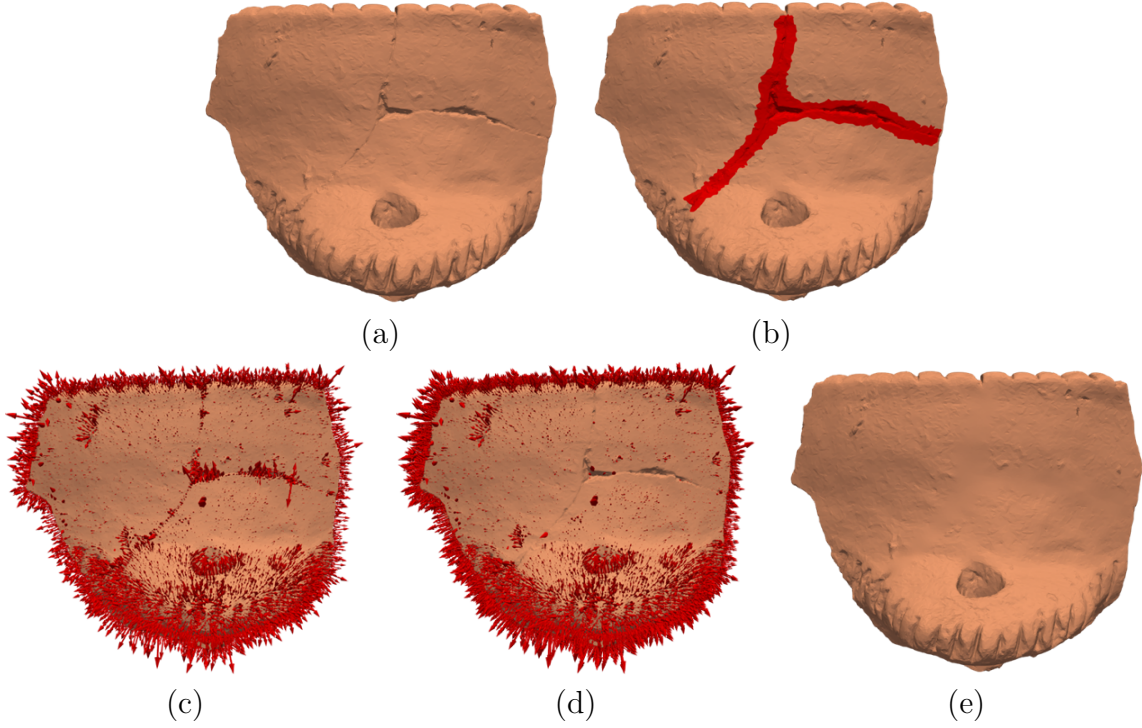


Figure 7.4: Inpainting osmotic process. (a) damaged mesh  $\mathcal{M}_1$  ; (b) mesh  $\overline{\mathcal{M}} = \mathcal{M}_1$ , with sub-mesh  $ROI$  in red; (c) starting NCC  $\delta_N$  extracted from  $\overline{\mathcal{M}}$ ; (d) NCC  $\delta_N^*$  resulting from osmotic flow; (e) repaired mesh  $\mathcal{M}^*$ .

## 7.2 Inpainting via osmosis in differential coordinates

The mesh inpainting task, described in Sec.1.3.1, can be viewed as a specific case of hole-filling/repairing task where the missing mesh part must be smoothly reconstructed to seamlessly blend with the surrounding regions.

In Chapter 2 we have proposed an algorithm that modifies the shape of the surface around the damaged areas by means of a variational model acting on the Euclidean vertex coordinates, exploiting a smoothing-promoting penalization term.

In the following, we formulate the inpainting process as an osmotic flow exploiting the Normal-Controlled Coordinates as descriptors of the mesh.

We denote by  $\mathcal{M}_1$  the mesh which presents holes and/or fractures. Fig. 7.4(a) shows an example of damaged mesh  $\mathcal{M}_1$ , and in Fig. 7.4(b) the damaged part ( $ROI$ ) is red colored. Normal-Controlled Coordinates are able to encode the local geometrical details since they represent approximations of the normal curvature. In Fig. 7.4(c) the NCC are shown as a red vector field, they are small (not visible) in the flattest regions of the surface and large where the surface presents details and features, such as in the fracture highlighted in Fig.7.4(b).

The preliminary setup for the inpainting via NCC consists in the definition of an initial mesh  $\overline{\mathcal{M}}$  and the computation of its NCC. Specifically:

S1) *Generate the initial mesh  $\overline{\mathcal{M}}$ .*

In case of hole filling, a coarse triangulated patch  $ROI$  is added to close the hole, thus obtaining a mesh  $\overline{\mathcal{M}} = (V, E, T)$  with added vertices, edges and faces. In case of mesh repairing,  $ROI$  is the existent part of the mesh  $\mathcal{M}_1$  which appears corrupted, and  $\overline{\mathcal{M}}$  coincides with  $\mathcal{M}_1$ .

S2) *Compute the Normal Controlled Coordinates.*

As described in Sec. 1.2.3, we compute the NCC coordinates of the mesh  $\overline{\mathcal{M}}$  as the vector field  $\delta_N = N_w V$ , with  $N_w \in \mathbb{R}^{n_V \times n_V}$  being the weight matrix defined in (1.31) and  $n_V$  the number of vertices of  $\overline{\mathcal{M}}$ .

The goal of the linear osmotic flow (5.18), given an appropriate definition of the drift term  $\mathbf{d}$ , is to obtain a modified NCC vector field  $\delta_N^*$  which differs from the initial one  $\delta_N$  only over the damaged part  $ROI$ , preserving the undamaged geometric features in  $\overline{\mathcal{M}} \setminus ROI$  and smoothly changing in the transition regions.

Before applying the osmotic flow, we need to carefully set the initialization function  $u^0$  and the drift vector field  $\mathbf{d}$ , which both depend on the reference function  $u_R$  simply defined as

$$u_R = \delta_N - m, \quad m = \min_{\delta_N \in \mathbb{R}^3} \{\delta_N\} - \varepsilon, \quad (7.15)$$

where the vector  $m \in \mathbb{R}^3$  and the small offset  $\varepsilon > 0$ , similarly to Sec. 7.1, aim to guarantee the strict positivity of the initial function.

The drift term  $\mathbf{d}$  needs to distinguish between the damaged region  $ROI$  and the rest of the mesh  $\overline{\mathcal{M}}$ , therefore, it is defined as

$$d_{[i,j]} := \begin{cases} \frac{(\nabla u_R)_{[i,j]}}{(u_R)_{[i,j]}} & \text{if } v_i \wedge v_j \in \overline{\mathcal{M}} \setminus ROI, \\ 0 & \text{otherwise} \end{cases} \quad (7.16)$$

with  $(u_R)_{[i,j]} = (u_R(v_i) + u_R(v_j))/2$ . The definition of  $\mathbf{d}$  in (7.16) allows to preserve the NCC data outside  $ROI$ , while diffusing linearly in  $ROI$ , since the transport term vanishes. The osmotic evolution process is then applied to the initial function

$$u^0 := u_R \quad (7.17)$$

and performs separately on each of the three channels using individual drift vector fields  $\mathbf{d}$ , according to either explicit (EXP) or implicit (IMP) discretization scheme. It converges to a resulting function  $u^*$ , from which the repaired NCC vector field  $\delta_N^*$  is obtained as

$$\delta_N^* = u^* + m. \quad (7.18)$$

**Algorithm 6** Mesh Inpainting via NCC Osmosis

---

**Input:** · damaged mesh  $\mathcal{M}_1$ ,  
**Output:** · repaired mesh  $\mathcal{M}^* = (V^*, E^*, T^*)$   
**Parameters:** ·  $tol > 0$ ,  $k_{max} \in \mathbb{N}$ ,  $\tau > 0$

**Preliminary set up:**  
S1) Generate  $\overline{\mathcal{M}} = (V, E, T)$ , with corrupted region  $ROI \subset \overline{\mathcal{M}}$ , closing holes in  $\mathcal{M}_1$  (if necessary)  
S2) Compute  $\delta_N$ , the NCC of the mesh  $\overline{\mathcal{M}}$

**Osmotic flow:**  
- Set reference function  $u_R$  as in (7.15), with offset  $m$  if necessary  
- Set the drift term  $\mathbf{d}$  as in (7.16)  
- Set the initial function  $u^0$  as in (7.17)  
while  $(\|u^{k+1} - u^k\|_2 / \|u^k\|_2 > tol)$  and  $(k < k_{max})$   
     $u^{k+1} \leftarrow Pu^k$  via (EXP) or (IMP) scheme;  
end  
- Set  $\delta^*$  as in (7.18)  
- Solve (1.33) for  $V^*$   
- Set  $\mathcal{M}^* = (V^*, E, T)$ .

---

Figure 7.4 (d) shows the vector field  $\delta_N^*$  obtained from the osmotic process. We notice the underlying mesh has not yet been modified, however, the NCC vectors in the  $ROI$  region are not visible anymore due to their small norm. This is a desired natural consequence of the diffusion process over the  $ROI$  region.

The repaired mesh  $\mathcal{M}^* = (V^*, E^*, T^*)$ , defined in (1.40), keeps the connectivity of  $\overline{\mathcal{M}}$ , i.e.  $E^* = E$  and  $T^* = T$ , while its vertices  $V^*$  are obtained from  $\delta_N^*$  by solving the inverse reconstruction problem (1.33).

The unique least square solution of (1.33) is computed via three linear, sparse, symmetric, semi-definite positive normal equations systems, with a Dirichlet constraint imposed onto a single vertex coordinates  $v_k \in V$ , with  $v_k$  chosen away from  $ROI$ .

The resulting repaired mesh  $\mathcal{M}^*$  is shown in Fig.7.4 (e), it shows a perfect reconstruction of the region  $ROI$ .

### 7.2.1 Numerical Results

The proposed inpainting algorithm via osmotic flow is summarized in Algorithm 6. As in Sec.7.1, required parameters are  $tol > 0$  and  $k_{max} \in \mathbb{N}$  for the stopping criterion in (7.14) and  $\tau > 0$  as the time-step for the explicit or implicit resolution scheme, with the former preferred for its fast convergence.

The computational cost can be further reduced, taking advantage of the requirement that the resulting mesh  $\mathcal{M}$  is to be preserved almost exactly in the regions far away



from  $ROI$ . Therefore, the whole inpainting process can be applied to a smaller sub-mesh  $\overline{\mathcal{M}}_S \subset \overline{\mathcal{M}}$ , such that  $ROI \subset \overline{\mathcal{M}}_S$  and  $\overline{\mathcal{M}}_S$  includes a few triangle layers away from  $ROI$ , thus reducing the number of variable vertices to  $n \ll n_V$ . Then the final mesh  $\mathcal{M}^*$  is defined as  $\mathcal{M}^* = (\overline{\mathcal{M}} \setminus \overline{\mathcal{M}}_S) \cup \overline{\mathcal{M}}_S^*$ . The only needed modifications in the inpainting process consist of adding the Dirichlet boundary conditions at  $\partial\overline{\mathcal{M}}_S$  both in the osmosis PDE (5.18) and in the reconstruction (1.33).

To validate the accuracy of the proposed repairing tool, we show in Fig.7.5 the results of applying the inpainting osmotic flow to three different damaged meshes  $\mathcal{M}_1$ , shown in Fig.7.5 (c,f,i). In particular, in Fig.7.5 (a,d,g) we report the damaged meshes  $\overline{\mathcal{M}}$  with the  $ROI$  part in red, while the reconstruction is reported in Fig.7.5 (b,e,h). We notice that in Fig.7.5 (d) not all the fractures in the damaged object are marked as  $ROI$  regions, to highlight the effect of the inpainting on the repaired mesh reported in Fig.7.5 (e).

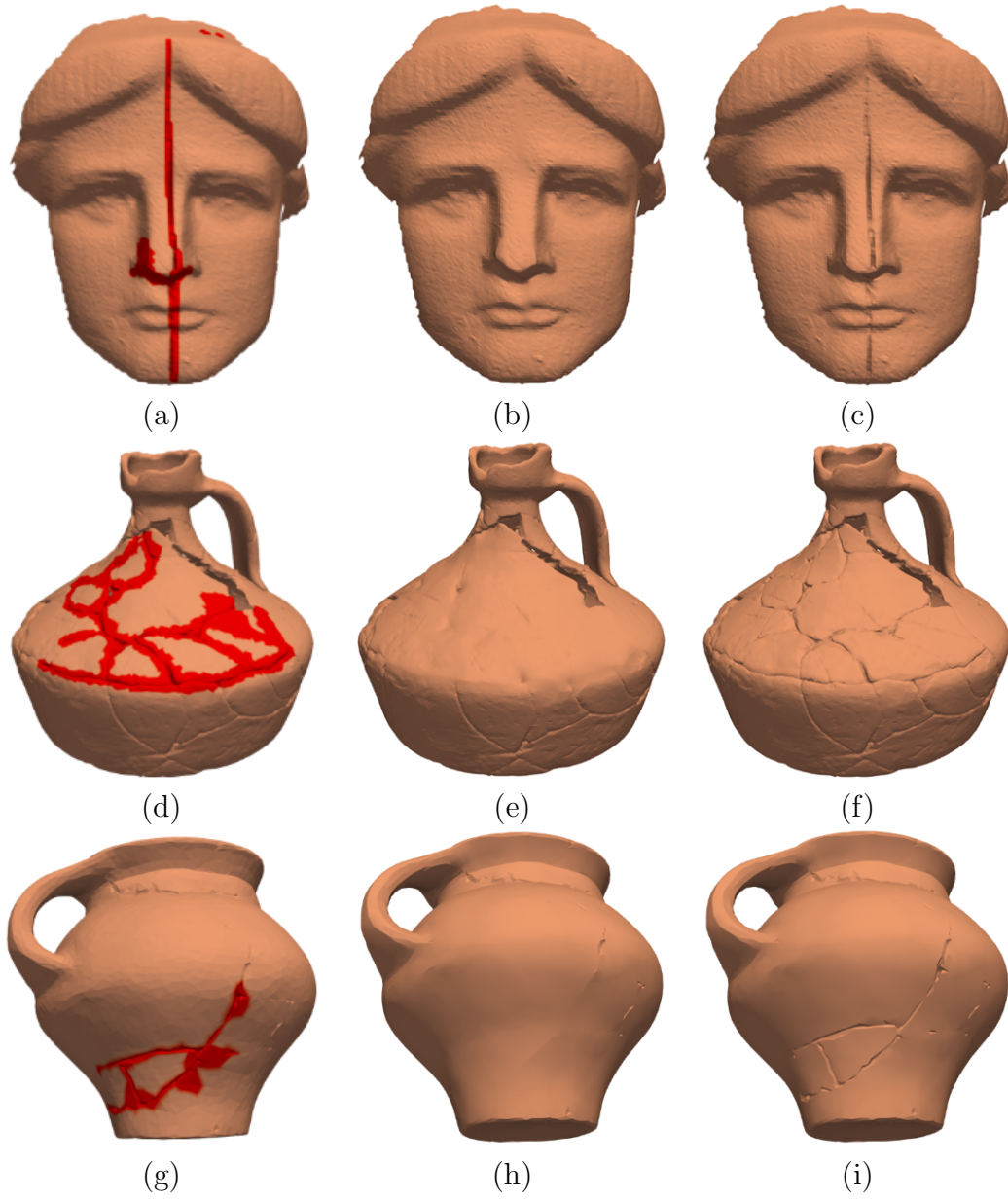


Figure 7.5: Inpainting task:  $\overline{\mathcal{M}}$  with  $ROI$  in red (a,d,g); repaired mesh  $\mathcal{M}^*$  (b,e,h); damaged mesh  $\mathcal{M}_1$  (c,f,i).



# Conclusions

In this thesis, we addressed a variety of surface processing tasks, proposing novel approaches and frameworks, based on advanced mathematical formulations.

In particular, we principally focused on two types of tasks: repairing and deforming. Repairing tasks consist in recovering a complete and precise 3D numerical approximation of a possibly damaged 3D object, given a noisy or incomplete representation obtained, for example, through a scanning process. Deforming tasks aim to obtain a new surface with a particular global shape or specified local details, given one or more reference surfaces.

Solving these tasks requires representing the involved surfaces through the most adapt shape descriptor function. We described and compared the classical Euclidean coordinates, the Laplacian coordinates, the Normal Controlled Coordinates and the Mean Value Encoding, analyzing their theoretical properties and their efficacy for each specific task.

The proposed frameworks rely on two types of mathematical tools for data processing: Partial Differential Equations and variational models. In both cases, the model definition involves differential operators, because of their ability to encode information about the geometrical properties and details of a surface.

For each task, we provide validation practically, through many numerical results.

The variational approach is used in Chapters 2 and 3.

In particular, in Chapter 2 we presented a novel geometric framework for denoising, inpainting and context-based completion for the recovery of damaged and incomplete meshes resulting from range scanning as well as for all modeling operations aiming at replacing damaged or missing parts of the surface.

The framework is based on a single variational problem, parametrization-free and normal consistent, which minimizes a functional that is spatially variant and characterized by a convex-non-convex structure: it varies spatially from being convex (due to the presence of the Willmore energy) to non-convex (Minimax-Concave penalty) according to a mask operator.

Future investigations will focus on the study of the theoretical convergence of the proposed numerical algorithm, which nevertheless demonstrates empirical convergence and very satisfying practical performance. The encouraging results can further be extended

to the completion of missing parts of objects and template patches with boundaries of different topologies in order to validate the process in more realistic cases. Finally, a future direction will be to couple the proposed approach with image inpainting models favoring the completion of texture-like regions.

In Chapter 3 we deal with Geometric Texture Transfer (GTT), aimed to equip a region  $P$  of a base surface with finer details extracted from a textured surface  $\mathcal{M}_T$ . We first investigate properties of local shape descriptors appropriated for this task, to better capture the shape morphology, and then formulate the geometric texture transfer task as constrained variational linear/nonlinear optimization problems based on these descriptors. We analyzed suitable numerical algorithms and several critical aspects in the solution of these minimization problems, such as the influence of the parameters or the invariance/equivariance under affine and non-affine deformations.

The proposed GTT variational approaches provide consistent values across boundaries between  $P$  and  $\mathcal{M}_T$  under the mild assumption of the same number of vertices in the common boundary. In case boundary subdivisions are required to satisfy this condition, this may produce an unpleasant wrinkle along the boundary. A smoother joint can be obtained by considering suitable boundary overlap strategies, and boundary remeshing, which can be investigated in future work. Additional study direction represents integrating this GTT framework with deep learning techniques for texture synthesis which generate 3D geometric textures.

Chapters 4-7 involve differential models.

In Chapter 4, we investigated the eigenproblem for graph  $p$ -Laplacian identified by the PDE  $\Delta_p f = \lambda f$ . The resolution, based on a variational approach, iteratively computes eigenfunctions through a sequence of minimization problems, involving a non-linear generalization of the Rayleigh quotient and of the orthogonality constraint, for cases  $p \neq 2$ .

We reformulated each nonlinear problem into an equivalent formulation by a suitable  $p$ -dependent change of variable, transforming the  $p$ -orthogonality constraint into a simple linear constraint. Then, we proposed two resolution algorithms, based on the projected gradient descent method, called M-PGD, and an Alternating Direction Method of Multipliers strategy, called M-ADMM.

As a further novelty, we also described a practical method based on orthogonal least square fitting for the estimation of the associated eigenvalues, which returns also a metric that measures whether the eigenpair equation is satisfied.

When using the M-ADMM approach we observed a faster convergence for arbitrary  $p$  values, but the slower M-PGD algorithm can be sped up through standard preconditioning and backtracking line search strategies. The consistency of the obtained results is proven numerically and a visualization of eigenfunctions computed on meshes suggests the potential usage for mesh segmentation tasks. A further possible application of the method consists of graph clustering problems.

In Chapters 5-6 we focus on the diffusion-transport osmosis PDE model.

In particular, in Chapter 5, after describing the formulation of the original linear model and its applications to image editing tasks, we propose a non-local model, where the differential operators are replaced by corresponding integral operators, in order to formalize an evolution process that takes into account also long-range interactions between the points of the domain, weighted by a kernel function. The theoretical study of wellposedness and regularity of the solutions of local and non-local models is followed by a consistency analysis, where we conjecture the convergence of non-local solutions, given an appropriate kernel rescaling.

In the graph domain, the non-local formulation translates into an accurate spatial discretization and leads to the definition of implicit (unconditionally stable) and explicit (conditionally stable) evolutive schemes. We observed that the model, applied for editing color functions defined on meshes, without changing the underlying geometry, behaves as in the image context, giving the possibility to extend all the editing tasks to any kind of data localized on 3-dimensional domains.

In Chapter 6, we presented a non-linear extension of the linear osmosis model, involving a diffusivity term that balances the diffusion intensity on different points of the domain.

On image domains, defining the diffusivity as a function of the gradient of the data allows to preserve the underlying features on the boundary (i.e., edges), thus preventing smoothing artifacts. Suitable finite difference schemes return a spatial discretization that can be viewed as a particular case of the one computed for graphs. Moreover, an unconditionally stable semi-implicit scheme is used to deal with the non-linear term in a computationally efficient way, obtaining fast convergence to the steady-state solution.

The efficiency and the numerical accuracy of the presented schemes are validated through three imaging applications: shadow/spot-light removal and compact data representation. The results are accurate, can be computed efficiently and outperform the ones obtained by alternative osmosis models and state-of-the-art approaches both visually and in terms of Structural Similarity Index. The approach only requires a preliminary segmentation of the region of interest (shadow, light-spot, edge mask) without any other tuning of the model hyperparameter, which makes it, essentially, fully automatic.

An interesting direction for future research is the study of a combined non-linear and anisotropic osmosis model as well as the extension to higher-order differential models so as to reduce the reconstruction drawbacks observed especially when large masks are considered.

Finally, in Chapter 7, we use the osmosis model to modify the actual geometry of surfaces. This is done by applying the osmotic evolution process to shape descriptors as Euclidean coordinates or Normal Controlled Coordinates (NCC), and consequently reconstruct the corresponding surface. In particular, we propose two frameworks for

cloning and inpainting.

- For the cloning task, taking inspiration from image cloning, we define the osmosis model through a drift term that depends on the Euclidean coordinates of the patch and of the Region of Interest to be combined. A pre-processing step is needed in advance to define these data functions on a single graph, constituted by a mixed triangulation of the two surfaces, computed on a common parametrization domain.

Preliminary tests show the potentiality of the proposed method, whose main strength is the reasonable computational cost, due to the use of a stable implicit scheme for osmosis evolution. Furthermore, using a mixed triangulation avoids the possible detail loss that may happen when resampling is applied. On the other hand, the robustness of the method can be improved by a further analysis of the alignment and parametrization steps.

- For the inpainting task, the osmosis model acts on the NCC extracted from the original damaged mesh, imposing a null drift term on the corrupted region, where the correct NCC field from the surroundings must be smoothly diffused.

The framework does not require any pre-processing step and is computationally cheap, thanks to the implicit evolution scheme.

# Bibliography

- [1] S. AMGHIBECH, Eigenvalues of the discrete p-laplacian for graphs, Ars Combinatoria, 67 (2003), pp. 283–302.
- [2] G. AN ZOU, X. WANG, AND T. W. SHEU, Finite element analysis of a new phase field model with p-laplacian operator, Mathematics and Computers in Simulation, 185 (2021), pp. 134–152.
- [3] F. ANDREU-VAILLO, J. M. MAZÓN, J. D. ROSSI, AND J. J. TOLEDO-MELERO, Nonlocal diffusion problems, no. 165, American Mathematical Soc., 2010.
- [4] E. ARBEL AND H. HEL-OR, Texture-preserving shadow removal in color images containing curved surfaces, in 2007 IEEE Conference on Computer Vision and Pattern Recognition, 07 2007, pp. 1–8.
- [5] O. ARGUDO, P. BRUNET, A. CHICA, AND ÀLVAR VINACUA, Biharmonic fields and mesh completion, Graphical Models, 82 (2015), pp. 137–148.
- [6] H. AVRON, A. SHARF, C. GREIF, AND D. COHEN-OR,  $\ell_1$ -sparse reconstruction of sharp point set surfaces, ACM Trans. Graph., 29 (2010), pp. 135:1–135:12.
- [7] M. BABA AND N. ASADA, Shadow removal from a real picture, in ACM SIGGRAPH 2003 Sketches & Applications, SIGGRAPH '03, New York, NY, USA, 2003, Association for Computing Machinery, p. 1.
- [8] M. BALASHOV AND R. KAMALOV, The gradient projection method with armijo's step size on manifolds, Computational Mathematics and Mathematical Physics, 61 (2021), pp. 1776–1786.
- [9] G. BARAVDISH, Y. CHENG, O. SVENSSON, AND F. ÅSTRÖM, Extension of p-laplace operator for image denoising, in System Modeling and Optimization, L. Bociu, J.-A. Désidéri, and A. Habbal, eds., Cham, 2016, Springer International Publishing, pp. 107–116.
- [10] S. BENALIA AND M. HACHAMA, A nonlocal method for image shadow removal, Computers & Mathematics with Applications, 107 (2022), pp. 95–103.



- [11] G. H. BENDELS AND R. KLEIN, Mesh forging: editing of 3d-meshes using implicitly defined occluders, in Proceedings of the 2003 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing, SGP '03, Goslar, DEU, 2003, Eurographics Association, p. 207–217.
- [12] M. BERGOU, M. WARDETZKY, D. HARMON, D. ZORIN, AND E. GRINSUN, Discrete quadratic curvature energies, in ACM SIGGRAPH 2006 Courses, SIGGRAPH '06, New York, NY, USA, 2006, Association for Computing Machinery, p. 20–29.
- [13] A. BERMAN AND R. J. PLEMMONS, Nonnegative Matrices in the Mathematical Sciences, Society for Industrial and Applied Mathematics, 1994.
- [14] R. J. BIEZUNER, G. ERCOLE, AND E. M. MARTINS, Computing the first eigenvalue of the p-laplacian via the inverse power method, Journal of Functional Analysis, 257 (2009), pp. 243–270.
- [15] J. BLINN, Simulation of wrinkled surfaces, in Siggraph 1978, Association for Computing Machinery, Inc., January 1978, pp. 286–292.
- [16] A. I. BOBENKO, A conformal energy for simplicial surfaces, Combinatorial and Computational Geometry, 52 (2005), pp. 133–143.
- [17] A. I. BOBENKO AND P. SCHRÖDER, Discrete willmore flow, SGP '05, Goslar, DEU, 2005, Eurographics Association.
- [18] F. BORG, What is osmosis? explanation and understanding of a physical phenomenon, (2003).
- [19] N. BOUMAL, An introduction to optimization on smooth manifolds, Cambridge University Press, 2023.
- [20] H. BREZIS, Operateurs maximaux monotones et semi-groupes de contractions dans les espaces de Hilbert, Elsevier, 1973.
- [21] —, Functional analysis, Sobolev spaces and partial differential equations, vol. 2, Springer, 2011.
- [22] T. BÜHLER AND M. HEIN, Spectral clustering based on the graph p-laplacian, in Proceedings of the 26th Annual International Conference on Machine Learning, ICML '09, New York, NY, USA, 2009, Association for Computing Machinery, p. 81–88.

- [23] L. BUNGER, M. BURGER, AND D. TENBRINCK, Computing nonlinear eigenfunctions via gradient flow extinction, in *Scale Space and Variational Methods in Computer Vision: 7th International Conference, SSVM 2019, Hofgeismar, Germany, June 30–July 4, 2019, Proceedings 7*, Springer, 2019, pp. 291–302.
- [24] L. CALATRONI, C. ESTATICO, N. GARIBALDI, AND S. PARISOTTO, Alternating direction implicit (adi) schemes for a pde-based image osmosis model, *Journal of Physics: Conference Series*, 904 (2017), p. 012014.
- [25] L. CALATRONI, M. HUSKA, S. MORIGI, AND G. A. RECUPERO, A unified surface geometric framework for feature-aware denoising, hole filling and context-aware completion, *Journal of Mathematical Imaging and Vision*, 65 (2023), pp. 82–98.
- [26] L. CALATRONI, S. MORIGI, S. PARISOTTO, AND G. A. RECUPERO, Fast and stable schemes for non-linear osmosis filtering, *Computers & Mathematics with Applications*, 133 (2023), pp. 30–47.
- [27] J. CALDER, The game theoretic p-laplacian and semi-supervised learning with few labels, *Nonlinearity*, 32 (2018), p. 301.
- [28] S. CARLSSON, Sketch based coding of grey level images, *Signal Process.*, 15 (1988), pp. 57–83.
- [29] G. CASCIOLA, D. LAZZARO, L. MONTEFUSCO, AND S. MORIGI, Fast surface reconstruction and hole filling using positive definite radial basis functions, *Numer Algor*, 39 (2005), pp. 289–305.
- [30] M. CHAHHOUE, L. MOUMOUN, M. E. FAR, AND T. GADI, Segmentation of 3d meshes using p-spectral clustering, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36 (2014), pp. 1687–1693.
- [31] A. CHAMBOLLE, An algorithm for total variation minimization and applications, *Journal of Mathematical Imaging and Vision*, (2004).
- [32] T. CHAN AND J. SHEN, Image Processing and Analysis: Variational, PDE, Wavelet, and Stochastic Methods, *Society for Industrial and Applied Mathematics*, Philadelphia, 2005.
- [33] X. CHUNXIA, S. RUIYUN, X. DONGLIN, AND M. KWANÂ-LIU, Fast shadow removal using adaptive multi-scale illumination transfer, *Computer Graphics Forum*, 32 (2013), pp. pp. 207–218.
- [34] U. CLARENZ, U. DIEWALD, G. DZIUK, M. RUMPF, AND R. RUSU, A finite element method for surface restoration with smooth boundary conditions, *Computer Aided Geometric Design*, 21 (2004), pp. 427–445.

- [35] I. COHEN AND G. GILBOA, Energy dissipating flows for solving nonlinear eigenpair problems, *Journal of Computational Physics*, 375 (2018), pp. 1138–1158.
- [36] I. COHEN AND G. GILBOA, Introducing the p-laplacian spectra, *Signal Processing*, 167 (2020), p. 107281.
- [37] R. L. COOK, L. C. CARPENTER, AND E. E. CATMULL, The reyes image rendering architecture, *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, (1987).
- [38] J. DAVIS, S. MARSCHNER, M. GARR, AND M. LEVOY, Filling holes in complex surfaces using volumetric diffusion, in *Proceedings. First International Symposium on 3D Data Processing Visualization and Transmission*, 2002, pp. 428–441.
- [39] P. DEIDDA, The graph p-laplacian eigenvalue problem, (2023).
- [40] P. DEIDDA, M. PUTTI, AND F. TUDISCO, Nodal domain count for the generalized graph p-laplacian, *Applied and Computational Harmonic Analysis*, 64 (2023), pp. 1–32.
- [41] EL BOUCHAIRI, IMAD, FADILI, JALAL M., AND ELMOATAZ, ABDERRAHIM, Continuum limit of p-laplacian evolution problems on graphs: Lq graphons and sparse graphs, *ESAIM: M2AN*, 57 (2023), pp. 1795–1838.
- [42] G. ELBER, Geometric deformation-displacement maps, in *10th Pacific Conference on Computer Graphics and Applications*, 2002. *Proceedings.*, 2002, pp. 156–165.
- [43] G. ELBER, Geometric texture modeling, *IEEE Computer Graphics and Applications*, 25 (2005), pp. 66–76.
- [44] L. C. EVANS, Partial differential equations, *American Mathematical Society*, Providence, R.I., 2010.
- [45] M. EYIYUREKLI AND D. E. BREEN, Detail-preserving level set surface editing and geometric texture transfer, *Graphical Models*, 93 (2017), pp. 39–52.
- [46] G. FINLAYSON, M. DREW, AND C. LU, Entropy minimization for shadow removal, *International Journal of Computer Vision*, 85 (2009), pp. pp. 35–57.
- [47] G. FINLAYSON, S. HORDLEY, C. LU, AND M. DREW, On the removal of shadows from images, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28 (2006), pp. pp. 59–68.
- [48] M. S. FLOATER, Mean value coordinates, *Computer Aided Geometric Design*, 20 (2003), pp. 19–27.

- [49] C. FREDEMBACH AND G. FINLAYSON, Simple shadow removal, in 18th International Conference on Pattern Recognition (ICPR'06), vol. 1, 2006, pp. 832–835.
- [50] G. FU, P. ZHAO, AND Y. BIAN,  $p$ -laplacian based graph neural networks, in International Conference on Machine Learning, PMLR, 2022, pp. 6878–6917.
- [51] G. GILBOA, Iterative methods for computing eigenvectors of nonlinear operators, Handbook of Mathematical Models and Algorithms in Computer Vision and Imaging: Mathematical Imaging and Vision, (2021), pp. 1–28.
- [52] R. GUO, Q. DAI, AND D. HOIEM, Paired regions for shadow detection and removal, IEEE Transactions on Pattern Analysis and Machine Intelligence, 35 (2013), pp. pp. 2956–2967.
- [53] K. HAGENBURG, M. BREUSS, J. WEICKERT, AND O. VOGEL, Novel schemes for hyperbolic pdes using osmosis filters from visual computing, in Scale Space and Variational Methods in Computer Vision, A. M. Bruckstein, B. M. ter Haar Romeny, A. M. Bronstein, and M. M. Bronstein, eds., Berlin, Heidelberg, 2012, Springer Berlin Heidelberg, pp. 532–543.
- [54] L. P. HARI, D. GIVOLI, AND J. RUBISTEIN, Computation of open willmore-type surfaces, Applied Numerical Mathematics, 37 (2001), pp. 257–269.
- [55] L. HE AND S. SCHAEFER, Mesh denoising via l0 minimization, ACM Trans. Graph., 32 (2013).
- [56] A. HERTZ, R. HANOCKA, R. GIRYES, AND D. COHEN-OR, Deep geometric texture synthesis, ACM Trans. Graph., 39 (2020).
- [57] M. HUSKA, S. H. KANG, A. LANZA, AND S. MORIGI, A variational approach to additive image decomposition into structure, harmonic, and oscillatory components, SIAM Journal on Imaging Sciences, 14 (2021), pp. pp. 1749–1789.
- [58] M. HUSKA, A. LANZA, S. MORIGI, AND I. SELESNICK, A convex-nonconvex variational method for the additive decomposition of functions on surfaces, Inverse Problems, 35 (2019), p. 124008.
- [59] M. HUSKA, A. LANZA, S. MORIGI, AND F. SGALLARI, Convex non-convex segmentation of scalar fields over arbitrary triangulated surfaces, J. Computational and Applied Mathematics, 349 (2019), pp. 438–451.
- [60] M. HUSKA, D. LAZZARO, AND S. MORIGI, Shape partitioning via  $L_p$  compressed modes, J. Math. Imaging Vis., 60 (2018), p. 1111–1131.

- [61] M. HUSKA, S. MORIGI, AND G. A. RECUPERO, Sparsity-aided variational mesh restoration, in *Scale Space and Variational Methods in Computer Vision*, A. Elmoataz, J. Fadili, Y. Quéau, J. Rabin, and L. Simon, eds., Cham, 2021, Springer International Publishing, pp. 437–449.
- [62] M. HUSKA, S. MORIGI, AND G. A. RECUPERO, Geometric texture transfer via local geometric descriptors, *Applied Mathematics and Computation*, 451 (2023), p. 128031.
- [63] Z. KARNI AND C. GOTSCHAN, Spectral compression of mesh geometry, in *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '00, USA, 2000*, ACM Press/Addison-Wesley Publishing Co., p. 279–286.
- [64] B. KAWOHL AND J. HORÁK, On the geometry of the  $\mu_i$ -laplacian operator, 2017.
- [65] M. KAZHDAN, M. BOLITHO, AND H. HOPPE, Poisson surface reconstruction, in *Proceedings of the fourth Eurographics symposium on Geometry processing*, vol. 7, 2006.
- [66] A. KOVNATSKY, K. GLASHOFF, AND M. M. BRONSTEIN, Madmm: A generic algorithm for non-smooth optimization on manifolds, in *Computer Vision – ECCV 2016*, B. Leibe, J. Matas, N. Sebe, and M. Welling, eds., Cham, 2016, Springer International Publishing, pp. 680–696.
- [67] V. KRAEVOY AND A. SHEFFER, Mean-value geometry encoding, *Int. J. Shape Model.*, 12 (2006), pp. 29–46.
- [68] Y.-K. LAI, S.-M. HU, D. X. GU, AND R. R. MARTIN, Geometric texture synthesis and transfer via geometry images, in *Proceedings of the 2005 ACM Symposium on Solid and Physical Modeling, SPM '05, New York, NY, USA, 2005*, Association for Computing Machinery, p. 15–26.
- [69] A. LANZA, S. MORIGI, AND G. RECUPERO, Variational graph p-laplacian eigendecomposition under p-orthogonality constraints, *Computational Optimization and Applications*, (2024).
- [70] A. LANZA, S. MORIGI, AND F. SGALLARI, Convex image denoising via non-convex regularization, in *Scale Space and Variational Methods in Computer Vision*, J.-F. Aujol, M. Nikolova, and N. Papadakis, eds., Cham, 2015, Springer International Publishing, pp. 666–677.

- [71] H. LE AND D. SAMARAS, Shadow removal via shadow image decomposition, in Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 8578–8587.
- [72] R. B. LEHOUCQ, D. C. SORENSEN, AND C. YANG, ARPACK Users' Guide, Society for Industrial and Applied Mathematics, 1998.
- [73] G. LI, J. LI, J. MERTEN, Y. XU, AND S. ZHU, Adaptive finite element approximations of the first eigenpair associated with p-laplacian, 2024.
- [74] P. LIEPA, Filling holes in meshes, in Proceedings of the 2003 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing, SGP '03, Goslar, DEU, 2003, Eurographics Association, p. 200–205.
- [75] Y. LIPMAN, O. SORKINE, D. COHEN-OR, D. LEVIN, C. ROSSI, AND H. SEIDEL, Differential coordinates for interactive mesh editing, in Proceedings Shape Modeling Applications, 2004., 2004, pp. 181–190.
- [76] F. LIU AND M. GLEICHER, Texture-consistent shadow removal, in Computer Vision – ECCV 2008, D. Forsyth, P. Torr, and A. Zisserman, eds., Berlin, Heidelberg, 2008, Springer Berlin Heidelberg, pp. 437–450.
- [77] W. LIU, X. MA, Y. ZHOU, D. TAO, AND J. CHENG,  $p$ -laplacian regularization for scene recognition, IEEE Transactions on Cybernetics, 49 (2019), pp. 2927–2940.
- [78] W. LIU, Z.-J. ZHA, Y. WANG, K. LU, AND D. TAO,  $p$ -laplacian regularized sparse coding for human activity recognition, IEEE Transactions on Industrial Electronics, 63 (2016), pp. 5120–5129.
- [79] Z. LIU, R. LAI, H. ZHANG, AND C. WU, Triangulated surface denoising using high order regularization with dynamic weights, SIAM J. Sci. Comp., 41 (2019), pp. B1–B26.
- [80] W. E. LORENSEN AND H. E. CLINE, Marching cubes: A high resolution 3d surface construction algorithm, in Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '87, New York, NY, USA, 1987, Association for Computing Machinery, p. 163–169.
- [81] D. LUO, H. HUANG, C. DING, AND F. NIE, On the eigenvectors of p-laplacian, Machine Learning, 81 (2010), pp. 37–51.
- [82] M. LYSAKER, S. OSHER, AND XUE-CHENG TAI, Noise removal using smoothed normals and surface fitting, IEEE Transactions on Image Processing, 13 (2004), pp. 1345–1357.

- [83] A. LÊ, Eigenvalue problems for the p-laplacian, *Nonlinear Analysis: Theory, Methods & Applications*, 64 (2006), pp. 1057–1099.
- [84] F. MAGGIOLI, S. MELZI, M. OVSJANIKOV, M. M. BRONSTEIN, AND E. RODOLÀ, Orthogonalized fourier polynomials for signal approximation and transfer, *Computer Graphics Forum*, 40 (2021), pp. 435–447.
- [85] M. MAINBERGER, A. BRUHN, J. WEICKERT, AND S. FORCHHAMMER, Edge-based compression of cartoon-like images with homogeneous diffusion, *Pattern Recognition*, 44 (2011), pp. 1859–1873.
- [86] M. MEYER, M. DESBRUN, P. SCHRÖDER, AND A. H. BARR, Discrete differential-geometry operators for triangulated 2-manifolds, in *Visualization and Mathematics III*, H.-C. Hege and K. Polthier, eds., Berlin, Heidelberg, 2003, Springer Berlin Heidelberg, pp. 35–57.
- [87] S. MORIGI AND M. RUCCI, Multilevel mesh simplification, *The Visual Computer*, 30 (2014), pp. 479–492.
- [88] S. MORIGI, M. RUCCI, AND F. SGALLARI, Nonlocal surface fairing, in *SSVM*, A. M. Bruckstein, B. M. ter Haar Romeny, A. M. Bronstein, and M. M. Bronstein, eds., Berlin, Heidelberg, 2012, Springer Berlin Heidelberg, pp. 38–49.
- [89] A. NG, M. JORDAN, AND Y. WEISS, On spectral clustering: Analysis and an algorithm, in *Advances in Neural Information Processing Systems*, T. Dietterich, S. Becker, and Z. Ghahramani, eds., vol. 14, MIT Press, 2001.
- [90] M. NIKOLOVA, Energy minimization methods, in *Handbook of Mathematical Methods in Imaging*, O. Scherzer, ed., Springer Reference, Springer, Jan. 2011, pp. 138–186.
- [91] P. OCHS, Y. CHEN, T. BROX, AND T. POCK, ipiano: Inertial proximal algorithm for non-convex optimization, *SIAM Journal on Imaging Sciences*, 7 (2014), pp. 1388–1419.
- [92] S. PARISOTTO, L. CALATRONI, A. BUGEAU, N. PAPADAKIS, AND C.-B. SCHÖNLIEB, Variational osmosis for non-linear image fusion, *IEEE Transactions on Image Processing*, 29 (2020), pp. 5507–5516.
- [93] S. PARISOTTO, L. CALATRONI, M. CALIARI, C.-B. SCHÖNLIEB, AND J. WEICKERT, Anisotropic osmosis filtering for shadow removal in images, *Inverse Problems*, 35 (2019), p. 054001.
- [94] J.-H. PARK, J.-H. MOON, S. PARK, AND S.-H. YOON, Geostamp: Detail transfer based on mean curvature field, *Mathematics*, 10 (2022).

- [95] S. PARK, X. GUO, H. SHIN, AND H. QIN, Surface completion for shape and appearance, *Vis. Comput.*, 22 (2006), p. 168–180.
- [96] D. PASADAKIS, C. L. ALAPPAT, O. SCHENK, AND G. WELLEIN, Multiway p-spectral graph cuts on grassmann manifolds, *Machine Learning*, (2022), pp. 1–39.
- [97] M. PAULY, R. KEISER, L. P. KOBELT, AND M. GROSS, Shape modeling with point-sampled geometry, *ACM Trans. Graph.*, 22 (2003), p. 641–650.
- [98] Y. PERES AND S. SHEFFIELD, Tug-of-war with noise: A game-theoretic view of the  $p$ -laplacian, *Duke Mathematical Journal*, 145 (2008), pp. 91 – 120.
- [99] P. PÉREZ, M. GANGNET, AND A. BLAKE, Poisson image editing, *ACM Trans. Graph.*, 22 (2003), p. pp. 313–318.
- [100] P. PERONA, T. SHIOTA, AND J. MALIK, Anisotropic Diffusion, Springer Netherlands, Dordrecht, 1994, pp. 73–92.
- [101] L. PIEGL AND W. TILLER, The NURBS book, Springer Science & Business Media, 2012.
- [102] M. REUTER, S. BIASOTTI, D. GIORGI, G. PATANÈ, AND M. SPAGNUOLO, Discrete laplace-beltrami operators for shape analysis and segmentation, *Computers & Graphics*, 33 (2009), pp. 381–390. IEEE International Conference on Shape Modelling and Applications 2009.
- [103] S. A. B. S. CHEN AND W. LUO, Orthogonal least squares methods and their application to non-linear system identification, *International Journal of Control*, 50 (1989), pp. 1873–1896.
- [104] Y. SAAD, Iterative Methods for Sparse Linear Systems, Other Titles in Applied Mathematics, Society for Industrial and Applied Mathematics, 2003.
- [105] B. SCHMIDT AND F. FRATERNALI, Universal formulae for the limiting elastic energy of membrane networks, *Journal of the Mechanics and Physics of Solids*, 60 (2012), pp. 172–180.
- [106] T. SCHOENEMANN, S. MASNOU, AND D. CREMERS, On a linear programming approach to the discrete willmore boundary value problem and generalizations, in *Curves and Surfaces*, J.-D. Boissonnat, P. Chenin, A. Cohen, C. Gout, T. Lyche, M.-L. Mazure, and L. Schumaker, eds., Berlin, Heidelberg, 2012, Springer Berlin Heidelberg, pp. 629–646.



- [107] T. SCHUSTER, B. KALTENBACHER, B. HOFMANN, AND K. S. KAZIMIERSKI, Regularization Methods in Banach Spaces, De Gruyter, Berlin, Boston, 2012.
- [108] C.-B. SCHÖNLIEB, Partial Differential Equation Methods for Image Inpainting, Cambridge Monographs on Applied and Computational Mathematics, Cambridge University Press, 2015.
- [109] H. S. SEUNG AND D. R. NELSON, Defects in flexible membranes with crystalline order, Phys. Rev. A, 38 (1988), pp. 1005–1018.
- [110] A. SHARF, M. ALEXA, AND D. COHEN-OR, Context-based surface completion, ACM Trans. Graph., 23 (2004), p. 878–887.
- [111] A. SHEFFER AND V. KRAEVOY, Pyramid coordinates for morphing and deformation, in Proceedings. 2nd International Symposium on 3D Data Processing, Visualization and Transmission, 2004. 3DPVT 2004., 2004, pp. 68–75.
- [112] Y. SHOR AND D. LISCHINSKI, The shadow meets the mask: Pyramid-based shadow removal, Comput. Graph. Forum, 27 (2008), pp. pp. 577–586.
- [113] D. SLEPCEV AND M. THORPE, Analysis of p-laplacian regularization in semisupervised learning, SIAM Journal on Mathematical Analysis, 51 (2019), pp. 2085–2120.
- [114] O. SORKINE, D. COHEN-OR, Y. LIPMAN, M. ALEXA, C. RÖSSL, AND H.-P. SEIDEL, Laplacian surface editing, in Proceedings of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing, SGP '04, New York, NY, USA, 2004, Association for Computing Machinery, p. 175–184.
- [115] O. SORKINE-HORNUNG AND D. COHEN-OR, Least-squares meshes, Proceedings - Shape Modeling International SMI 2004, (2004), pp. 191–199.
- [116] E. SOUBIES, L. BLANC-FÉRAUD, AND G. AUBERT, A unified view of exact continuous penalties for  $\ell_2$ - $\ell_0$  minimization, SIAM Journal on Optimization, 27 (2017), pp. 2034–2060.
- [117] X. SUN, P. L. ROSIN, R. MARTIN, AND F. LANGBEIN, Fast and effective feature-preserving mesh denoising, IEEE Trans. on Vis. and Comp. Graph., 13 (2007), pp. 925–938.
- [118] Y. SUN, S. SCHAEFER, AND W. WANG, Denoising point sets via l0 minimization, Computer Aided Geometric Design, 35-36 (2015), pp. 2 – 15.
- [119] T. TASDIZEN, R. WHITAKER, P. BURCHARD, AND S. OSHER, Geometric surface processing via normal maps, ACM Trans. Graph., 22 (2003), p. 1012–1033.

- [120] G. TAUBIN, Curve and surface smoothing without shrinkage, 07 1995, pp. 852–857.
- [121] W. T. TUTTE, How to draw a graph, Proceedings of the London Mathematical Society, s3-13 (1963), pp. 743–767.
- [122] K. UHLENBECK, Regularity for a class of non-linear elliptic systems, Acta Mathematica, 138 (1977), pp. 219 – 240.
- [123] P. UPADHYAYA, E. JARLEBRING, AND F. TUDISCO, The self-consistent field iteration for p-spectral clustering, arXiv preprint arXiv:2111.09750, (2021).
- [124] O. VOGEL, K. HAGENBURG, J. WEICKERT, AND S. SETZER, A fully discrete theory for linear osmosis filtering, in Scale Space and Variational Methods in Computer Vision, A. Kuijper, K. Bredies, T. Pock, and H. Bischof, eds., Berlin, Heidelberg, 2013, Springer Berlin Heidelberg, pp. 368–379.
- [125] J. WANG AND M. M. OLIVEIRA, Filling holes on locally smooth surfaces reconstructed from point clouds, Image and Vision Computing, 25 (2007), pp. 103 – 113.
- [126] P.-S. WANG, Y. LIU, AND X. TONG, Mesh denoising via cascaded normal regression, ACM Trans. Graph., 35 (2016).
- [127] S. WANG, Y. CAI, Z. YU, J. CAO, AND Z. SU, Normal-controlled coordinates based feature-preserving mesh editing, Multimedia Tools Appl., 71 (2014), p. 607–622.
- [128] Y. WANG, W. YIN, AND J. ZENG, Global convergence of admm in nonconvex nonsmooth optimization, J Sci Comput, 78 (2019), pp. 29–63.
- [129] J. WEICKERT, K. HAGENBURG, M. BREUSS, AND O. VOGEL, Linear osmosis models for visual computing, in Energy Minimization Methods in Computer Vision and Pattern Recognition, A. Heyden, F. Kahl, C. Olsson, M. Oskarsson, and X.-C. Tai, eds., Berlin, Heidelberg, 2013, Springer Berlin Heidelberg, pp. 26–39.
- [130] J. WEICKERT, W. WERDEGANG, S. ZUCKER, A. DOBBINS, L. IVERSON, B. KIMIA, AND A. TANNENBAUM, Anisotropic diffusion in image processing, (1996).
- [131] T. J. WILLMORE, Surfaces in conformal geometry, Annals of Global Analysis and Geometry, 18 (2000), pp. 255–264.
- [132] J.-H. WON, K. LANGE, AND J. XU, A unified analysis of convex and non-convex lp-ball projection problems, Optimization Letters, 17 (2023), pp. 1133–1159.

- [133] T.-P. WU AND C.-K. TANG, A bayesian approach for shadow extraction from a single image, in Proceedings of the Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1, USA, 2005, IEEE Computer Society, pp. 480–487.
- [134] T.-P. WU, C.-K. TANG, M. S. BROWN, AND H.-Y. SHUM, Natural shadow matting, ACM Transactions on Graphics (TOG), 26 (2007), pp. pp. 8–28.
- [135] X. YAO AND J. ZHOU, Numerical methods for computing nonlinear eigenpairs: Part i. iso-homogeneous cases, SIAM Journal on Scientific Computing, 29 (2007), pp. 1355–1374.
- [136] W. YIFAN, L. RAHMANN, AND O. SORKINE-HORNUNG, Geometry-consistent neural shape representation with implicit displacement fields, 2021.
- [137] Y. YU, K. ZHOU, D. XU, X. SHI, H. BAO, B. GUO, AND H.-Y. SHUM, Mesh editing with poisson-based gradient field manipulation, ACM Trans. Graph., 23 (2004), p. 644–651.
- [138] S. K. ZAVRIEV AND F. V. KOSTYUK, Heavy-ball method in nonconvex optimization problems, Computational Mathematics and Modeling, 4 (1993), pp. 336–341.
- [139] C. ZHANG, Nearly unbiased variable selection under minimax concave penalty, Ann. Statist., 38 (2010), pp. 894–942.
- [140] D. ZHANG, Homological eigenvalues of graph p-laplacians, Journal of Topology and Analysis, (2021).
- [141] D. ZHANG, X. WANG, J. HU, AND H. QIN, Interactive modeling of complex geometric details based on empirical mode decomposition for multi-scale 3d shapes, Computer-Aided Design, 87 (2017), pp. 1–10.
- [142] W. ZHANG, B. DENG, J. ZHANG, S. BOUAZIZ, AND L. LIU, Guided mesh normal filtering, Comput. Graph. Forum, 34 (2015), p. 23–34.
- [143] Y. ZHENG, H. FU, O. K. AU, AND C. TAI, Bilateral normal filtering for mesh denoising, IEEE Trans. on Visualization and Computer Graphics, 17 (2011), pp. 1521–1530.