UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Università degli Studi di Padova
Department of Mathematics

Ph.D. COURSE IN MATHEMATICAL SCIENCES
CURRICULUM COMPUTATIONAL MATHEMATICS
CYCLE XXX.

# Variational Methods and Numerical Algorithms for Geometry Processing

Thesis written with the financial contribution of Fondazione Cariparo

**Coordinator:** Prof. Martino Bardi

**Supervisor:** Prof. Serena Morigi

**Ph.D. student:** Martin Huska

# Abstract

In this work we address the problem of shape partitioning which enables the decomposition of an arbitrary topology object into smaller and more manageable pieces called partitions. Several applications in Computer Aided Design (CAD), Computer Aided Manufactury (CAM) and Finite Element Analysis (FEA) rely on object partitioning that provides a high level insight of the data useful for further processing. In particular, we are interested in 2-manifold partitioning, since the boundaries of tangible physical objects can be mathematically defined by two-dimensional manifolds embedded into three-dimensional Euclidean space. To that aim, a preliminary shape analysis is performed based on shape characterizing scalar/vector functions defined on a closed Riemannian 2-manifold. The detected shape features are used to drive the partitioning process into two directions – a human-based partitioning and a thickness-based partitioning. In particular, we focus on the Shape Diameter Function that recovers volumetric information from the surface thus providing a natural link between the object's volume and its boundary, we consider the spectral decomposition of suitably-defined affinity matrices which provides multi-dimensional spectral coordinates of the object's vertices, and we introduce a novel basis of sparse and localized quasi-eigenfunctions of the Laplace-Beltrami operator called $\mathrm{L}_p$ Compressed Manifold Modes.

The partitioning problem, which can be considered as a particular inverse problem, is formulated as a variational regularization problem whose solution provides the so-called piecewise constant/smooth partitioning function. The functional to be minimized consists of a fidelity term to a given data set and a regularization term which promotes sparsity, such as for example, $\mathrm{L}_p$ norm with $p \in (0, 1)$ and other parameterized, non-convex penalty functions with positive parameter, which controls the degree of non-convexity.

The proposed partitioning variational models, inspired on the well-known Mumford Shah models for recovering piecewise smooth/constant functions, incorporate a non-convex regularizer for minimizing the boundary lengths. The derived non-convex non-smooth optimization problems are solved by efficient numerical algorithms based on Proximal Forward-Backward Splitting and Alternating Directions Method of Multipliers strategies, also employing Convex Non-Convex approaches.

Finally, we investigate the application of surface partitioning to patch-based surface quadrangulation. To that aim the 2-manifold is first partitioned into zero-genus patches that capture the object's arbitrary topology, then for each patch a quad-based minimal surface is created and evolved by a Lagrangian-based PDE evolution model to the original shape to obtain the final semi-regular quad mesh. The evolution is supervised by asymptotically area-uniform tangential redistribution for the quads.

**Keywords: Shape Analysis, Manifold Partitioning, Variational Methods, Optimization Algorithms.**

# Abstract

In questo lavoro affrontiamo il problema della partizione delle forme il cui scopo è la decomposizione di un oggetto di topologia arbitraria in parti più piccole e meglio gestibili chiamate partizioni. Svariate applicazioni in Computer Aided Design (CAD), Computer Aided Manufactury (CAM) e Finite Element Analysis (FEA) sfruttano tali decomposizioni in quanto forniscono un'informazione globale sulla forma. In particolare, siamo interessati al partizionamento di varietà topologiche di dimensioni 2, in quanto il bordo di oggetti fisici tangibili può essere definito matematicamente da varietà bidimensionali immerse nello spazio euclideo tridimensionale. A tale scopo, viene eseguita un'analisi preliminare sulla forma che fa uso di diverse funzioni scalari/vettoriali definite sulla varietà. Il processo di partizionamento si può affrontare da due punti di vista: uno basato sulla percezione visiva umana e un altro basato sullo spessore delle componenti della forma in esame. In particolare, ci concentriamo sulla funzione 'Diametro di forma' che recupera informazioni volumetriche dalla superficie, fornendo così un naturale legame tra il volume dell'oggetto e il suo bordo; inoltre studiamo la decomposizione spettrale di opportune matrici di affinità che fornisce coordinate spettrali multidimensionali caratterizzanti la forma dell'oggetto; infine introduciamo una nuova base, denominata $L_p$ Compressed Manifold Modes, di quasi-autofunzioni sparse e localizzate dell'operatore Laplace-Beltrami.

Il problema di partizionamento può essere considerato un particolare problema inverso, pertanto è fomulato come un problema di regolarizzazione variazionale che ha come soluzione la cosiddetta funzione di partizionamento. Il funzionale da minimizzare è somma di un termine di fedeltà a un determinato set di dati e di un termine di regolarizzazione che promuove la sparsità, come ad esempio la norma $L_p$ con $p \in (0, 1)$ o altre funzioni di penalizzazione non convesse e parametrizzate, con parametro positivo, che controlla il grado di non convessità.

I metodi proposti per ottenere la funzione di partizione, ispirati ai modelli variazionali di Mumford-Shah di funzionali costanti o smooth a tratti, incorporano un regolarizzatore non convesso per ridurre al minimo le lunghezze del contorno delle partizioni. Per la soluzione dei problemi di ottimizzazione non convessi e non smooth si propongono metodi

numerici basati su Proximal Forward-Backward Splitting, Alternating Directions Method of Multipliers e strategie Convex Non-Convex.

Inoltre, studiamo un'applicazione del partizionamento di forma nell'ambito della patch-based surface quadrangulation. A questo scopo la varietà viene prima suddivisa in patch di genere zero che catturano la topologia arbitraria dell'oggetto, quindi per ogni patch viene creata una superficie minima ad elementi quadrilateri che si evolve secondo un modello differenziale alle derivate parziali, seguendo un approccio Lagrangiano per ottenere una rappresentazione a griglie quadrilatere semi-regolari. L'evoluzione è supervisionata da una ridistribuzione tangenziale uniforme dell'area-asintotica dei quadrilateri.

**Parole Chiave: Analisi della Forma, Partizionamento di Varietà, Metodi Variazionali, Algoritmi di Ottimizzazione**

# Table of contents


**1 Introduction** **1**

**2 Variational Formulation of the Shape Partitioning Problem** **7**

2.1 Mumford-Shah Variational Models . . . . . 10

**3 First Order Optimization Methods** **13**

3.1 Convex Optimization Methods . . . . . 14

3.2 Non-Convex Optimization Methods . . . . . 21

3.3 Convex-NonConvex Strategy . . . . . 23

**4 Spatial Discretization** **25**

4.1 Discrete Differential Operators . . . . . 27

4.2 Quad-based Finite Volumes Spatial Discretization . . . . . 32

**5 Shape Analysis** **35**

5.1 Meshless approach to SDF computation by particles flow . . . . . 35

5.1.1 SDF: Examples . . . . . 41

5.2 Affinity Matrix . . . . . 49

5.3 Compact Support $L_p$CMs as Localized Shape Descriptors . . . . . 52

5.3.1 The sparsity-inducing variational model for $L_p$CMs . . . . . 55

5.3.2 Discretization of the variational model . . . . . 59

5.3.3 Applying ADMM to the proposed model . . . . . 61

**6 Shape Partitioning** **67**

6.1 Sparsity-Inducing Non-Convex Variational Shape Partitioning . . . . . 69

6.1.1 Discretization of the SMCMR model . . . . . 72

6.1.2 Algorithm SMCMR . . . . . 75

6.2 Localized Shape Descriptors in Non-Convex Shape Partitioning . . . . . 77

6.3 Convex Non-Convex approach in Segmentation over Surfaces . . . . . 81

6.3.1 Non-convex penalty functions . . . 82
6.3.2 Convexity Analysis . . . 83
6.3.3 Applying ADMM to the proposed CNC model . . . 85
6.4 Experiments on the Partitioning Algorithms . . . 89
6.4.1 Data Set and Hardware Specifications . . . 89
6.4.2 Experimental results of SMCMR Framework . . . 91
6.4.3 Experimental Results of Partitioning Driven by $L_p$ Compressed Modes . . . 96
6.4.4 Experimental Results of CNC Segmentation on Surfaces . . . 102

**7 Application in Surface-Patch Quadrangulation 109**
7.1 Related Work . . . 110
7.2 Mesh Partitioning . . . 112
7.3 Build of the Topology Skeleton . . . 113
7.3.1 Step 1: Construction of the boundaries of $S_i$ . . . 113
7.3.2 Step 2: Discretization of the boundaries into rectangular topology 114
7.3.3 Step 3: Construction of the topology-skeleton S . . . 117
7.4 Skeleton Evolution . . . 118
7.4.1 Lagrangian Evolution . . . 118
7.4.2 Numerical scheme . . . 123
7.5 Numerical experiments . . . 128
7.5.1 Example 1: Mesh Quadrangulation . . . 129
7.5.2 Example 2: Different resolutions . . . 131

**8 Conclusions 135**

**References 139**

**Appendix A Proofs in Section 5.3 on $L_p$ Compressed Manifold Modes 149**

**Appendix B Proofs in Section 6.3 on the CNC Approach in Segmentation Over Surfaces 155**

**Appendix C Proofs in Section 7.4 on the Tangential Evolution 161**

# Chapter 1

# Introduction

The recent development of 3D scanning technology for reverse engineering and sophisticated scan devices for medical imaging, have incredibly increased the generation and availability of digital models of 3D physical objects simply represented by a set of 3D points on the surface of the object. These raw 3D data connected into spatial triangulations, called 3D meshes, provide only local information on the structure of the overall surface. A high level insight of the raw 3D data set is required to make the digital model useful for further processing required in a variety of applications including computer graphics, computer-aided design (CAD), computer-aided modelling (CAM), computer-aided engineering (CAE) and computer vision. At this aim, shape analysis is preliminarily applied to detect high level model features, that can be useful for further tasks such as shape comparison, classification and retrieval or even the simple reconstruction of the data as a consistent CAD model.

Driven by the shape analysis, the 3D object partitioning is one of the fundamental geometry processing which provides a global insight on the model structure that can be exploited to deduce some depth level insight on the model. The 3D data points are partitioned into non-overlapping parts covering the entire object, which represent distinct attribute values of the virtual object. There are numerous object partitioning techniques based on various shape attributes. The most used features are principal curvatures, mean and Gaussian curvature, dihedral angles, normal directions, and so on. In [51] the authors proposed a survey of existing local shape descriptors. Specific criteria dictate which elements belong to the same partition and these criteria are built upon the segmentation objective which in turn depends on the application.

*Convexity/Concavity* and *thickness* are popular shape criteria used in object decomposition. The convexity-driven segmentation of a shape finds a very intuitive match with the decomposition of an object made by the human vision system [31, 130]. This is

to the fact that an approximate convex decomposition can more accurately represent the important structural features of the model by ignoring insignificant features, such as wrinkles and other surface texture. Unlike, the thickness of parts of a shape is a less intuitive detection strategy for a human eye. Nevertheless, this geometry feature represents a strategic quantity in shape analysis in the context of industrial design and production. In the field of industrial production of 3D models, in fact, the measuring of the thickness of different parts of an object is of fundamental importance to be able to better estimate the cost of production of molds that allow to realize the object with the desired shape, and then to estimate the cost of production of the objects themselves.

Much work has been done on approximate decomposition of a shape into convex components. Concavity-aware partitioning is proposed in [5] and [76]. In Asafi et al. [3] weakly convex components are obtained by a point-visibility test. The same idea is followed in [61] to approximate convex components of shape represented by, eventually incomplete, point clouds.

Partitioning methods based on spectral analysis emphasize mostly the concavity attribute, being able to partition even shallow concavities. The spectral analysis method uses the eigenvalues of properly defined matrices based on the connectivity of the graph in order to partition a mesh. Liu and Zhang [78] use spectral analysis on the dual graph of the mesh. They define an affinity matrix using both geodesic distances and angular distances as proposed by the fuzzy clustering method in [63]. This type of matrix has been used successfully for clustering since it groups elements having high affinity, see for example [130].

The basis generated by the eigenfunctions of Laplace-Beltrami Operator LBO, used in [130], called Manifold Harmonic Basis (MHB), has been proposed in [118] in analogy to Fourier analysis.

However, in the shape partitioning context, rather than a multiresolution representation of the shape, which is the peculiarity of the MHB on manifolds, the focus is on identifying the observable features of the manifold which represent for example protrusions, ridges, details in general localized in small regions.

Hence, in the partitioning context, a more suitable alternative to the MHB is represented by the Compressed Manifold Basis (CMB), introduced in [88], which is characterized by compact support quasi-eigenfunctions of the LBO obtained by imposing sparsity constraints.

Focusing on thickness as a segmentation property, the Shape Diameter Function (SDF), proposed in [108], is a measure of thickness that recovers volumetric information from the surface boundaries, thus providing a natural link between the object's volume

and its boundary. The SDF is a scalar function which maps for every point on the surface its distance to the opposite inner part of the object. As successfully proved in [108] this definition of the SDF is invariant to rigid body transformations of the whole object, and very robust to any deformation that does not alter the volumetric shape locally. This consistency over pose changes made SDF a very attractive shape-attribute for partitioning of a 3D object. Moreover, there is also a noticeable connection between SDF and Medial Axis, since distance from a point to Medial Axis is approximately half of the SDF value in the point. We will refer to this shape descriptor also as *SDF map* of a 3D object, whose target is not to mimic the partitioning of an object made by the human vision system into perceptually meaningful components, but rather to define a thickness-oriented partitioning.

In addition to the criteria deduced by the shape analysis that dictate the rules of the division into parts, the partitioning methods can be grouped into a few categories according to their computational methodology: (i) Region growing; (ii) Watershed-based; (iii) Reeb graphs; (iv) Model-based; (v) Skeleton-based; (vi) Clustering; (vii) Spectral analysis; (viii) Explicit Boundary Extraction; (ix) Critical points-based; (x) Multiscale Shape Descriptors; (xi) Markov Random Fields and (xii) Variational partitioning/segmentation. A detailed analysis of the aforementioned categories is given in [1] and exhaustive surveys are provided in [4],[107].

The concept of iteratively seeking a partition that minimizes a given error metric, named variational partitioning, has been introduced in [35] where the authors presented an optimization cost function based on clustering face normal of the mesh. Since then, several variational mesh partitioning have been proposed mostly for surface-based segmentation. In [128] a variational mesh segmentation framework based on fitting general quadrics (including planes as a special case) is proposed. Wu and Kobbelt [125] extend [35]'s work by introducing sphere, circular cylinder and rolling ball patch as basic primitives. An important result on part-based segmentation has been presented in [130], where a convexified version of the variational Mumford-Shah model is presented and extended to 3D meshes. The cost function contains a data term measuring the variation within a segment and a regularization term based on the total variation of the gradient, measuring the length of the boundary between segments.

In this work, we present a study in the wide field of object partitioning via variational methods. In particular, we deal with the 2-manifold partitioning which mimics the human

vision system with possible adjustment to obtain a patch-based manifold partitioning, and in addition, we address the thickness-oriented partitioning which is commonly applied, in contrast to the one driven by human-vision system, in industrial applications.

Summarizing, the contributions in this thesis can be grouped into following topics:

1. Shape Analysis
   We realize a numerical method to evaluate the so-called Shape Diameter Function that recovers the volumetric information of an object from a closed surface, thus, forming a natural connection of the object's volume and its boundary; Section 5.1, presented in [58].
   Next, we propose the spectral decomposition of an appropriately defined Affinity matrix that allows for obtaining a multidimensional spectral coordinates of the surface; Section 5.2, presented in [59].
   At last, we introduce a new orthonormal basis of sparse quasi-eigenfunctions of the Laplace-Beltrami operator by interpreting a sparsity-inducing term in the variational formulation used for obtaining the so-called Compressed Manifold Modes; Section 5.3, presented in [56].

2. Decomposition Variational Models
   We focus on a new variational formulation of the partitioning problem where the functional to be minimized, similarly to the Mumford-Shah models, consists of a fidelity term to the observed data and a regularization term where we propose a sparsity-inducing penalty either to the solution or to its gradient using $L_p$ norm with $p \in (0, 1)$ to minimize the boundary lengths between segmented parts; Chapter 6, presented in [59, 56, 55].
   The resulting functionals turn out to be non-convex or even non-differentiable thus requiring particular care in the study of resolvability of the problem itself and in its numerical solution.

3. Numerical Optimization
   We propose efficient and effective numerical algorithms for solving the non-convex decomposition problem, using techniques of proximal operators, Forward-Backward scheme, and methods based on Alternating Directions Method of Multipliers. We promote the Convex Non-Convex strategy which admits a non-convex regularization, while maintaining the overall functional convex via a suitable tuning of the model parameters; Sections 6.1, 5.3, 6.3, presented in [59, 56, 55].

4. Applications to Patch-Surface Quadrangulation
   We focus on an application of the surface decomposition into zero genus patches to

the reconstruction of a quadrilateral, semi-regular mesh.
To that aim, we utilize a Lagrangian PDE surface evolution model with supervised tangential redistribution of the points towards asymptotically uniform areas of the surface; Chapter 7, presented in [57].

Overall, the work is organized as follows. In Chapter 2 we introduce the variational formulation for of the shape partitioning problem, in Chapter 3 we present some optimization methods that provide solutions to convex and non-convex optimization problems. Then in Chapter 4 we present the discretization of an embedded 2-manifold in $\mathbb{R}^3$ as well as the discretization of differential operators defined on surfaces that will be used throughout the next chapters. In Chapter 5 we analyse the so-called shape descriptors which drive the Partitioning procedures described in Chapter 6 and we conclude with an application of the Partitioning in Patch-Surface Quadrangulation summarized in Chapter 7.

# Chapter 2

# Variational Formulation of the Shape Partitioning Problem

This work is devoted to variational models and their related numerical algorithms for solving geometry processing problems. Variational approaches allow us to solve many inverse problems by minimization and regularization using different tools from optimization, numerical analysis, scientific computing and utilizing knowledge about the model, geometry constraints and a priori information. The variational method with regularization has been proved to be one of the most powerful techniques for solving many image processing tasks [119]. More recently, many variational regularization approaches have been proposed in geometry processing to solve important problems such as surface reconstruction [11], fairing [85], simplification [84] and partitioning [130].

Our work is focused on the shape partitioning problem by variational regularization methods. The aim is to partition a shape into its main constituent parts for further analysis and understanding. Shape partitioning is used in CAD, finite element method (FEM) and many others computer graphics and vision applications.

The variational approach proposes to compute an approximated solution $u \in \Omega$ by minimizing an error metric defined by a functional $\mathcal{J}(u)$

$$\min_{u \in \Theta} \{\mathcal{J}(u) := \mathcal{F}(u)\} \ . \tag{2.1}$$

The functional $\mathcal{F}$ named *fidelity* it captures the error metric in terms of closeness to the observed data $f$. Common choices for the fidelity term are the Euclidean $L_2$ norm

and the $L_1$ norm

$$\begin{aligned} \mathcal{F}(u) &= \|F(u) - f\|_2^2 \,, \\ \mathcal{F}(u) &= \|F(u) - f\|_1 \,, \end{aligned} \tag{2.2}$$

for $F(\cdot)$ a possible continuous linear operator which definition depends on the problem assumed.

When dealing with inverse problems one usually add constraints on the solution for expressing certain prior knowledge into the optimization problem. A *regularization* term (or *regularizer*) $\mathcal{R}(u)$ is added to the functional (2.1), which now reads as

$$\min_{u \in \Theta} \{\mathcal{J}(u) := \mathcal{F}(u) + \mu \mathcal{R}(u)\} \tag{2.3}$$

where $\mu > 0$ is a weighting parameter balancing the trade-off between $\mathcal{F}$ and $\mathcal{R}$, often referred to as the regularization parameter.

The usual requirements are on the smoothness and sparsity of the solution or of its gradient. Popular convex choices for $\mathcal{R}(\cdot)$ are the $L_2$ norm and $L_1$ norm terms which read as

$$\mathcal{R}(u) = \|u\|_2^2 \;, \qquad \mathcal{R}(u) = \|\nabla u\|_2^2 \,, \tag{2.4}$$

$$\mathcal{R}(u) = \|u\|_1 \;, \qquad \mathcal{R}(u) = \|\nabla u\|_1 \,. \tag{2.5}$$

The $L_1$ norm penalty (2.5) is the commonly used convex relaxation of the $L_0$ pseudo-norm, while the non-convex $L_p$ norm for $0 < p < 1$ leads to a pure non-convex regularization term

$$\mathcal{R}(u, p) = \|u\|_p \;, \qquad \mathcal{R}(u, p) = \|\nabla u\|_p \,. \tag{2.6}$$

The penalty functions are applied either on the function values or its gradient, producing different results in the function smoothness and sparsity. In addition to the $L_p$ norm, other similar-in-shape parametrized functions $\phi(t, a)$ have been introduced in recent literature [89, 30] such as

$$\phi_{\log}(t; a) = \frac{\log(1 + at)}{a}, \;\; \phi_{\text{rat}}(t; a) = \frac{t}{1 + at/2}, \;\; \phi_{\text{atan}}(t; a) = \frac{\operatorname{atan}\left(\frac{1+2at}{\sqrt{3}}\right) - \frac{\pi}{6}}{a\sqrt{3}/2} \tag{2.7}$$

where the concavity parameter $a$ controls their shape from nearly convex for $a \to 0^+$ to strongly non-convex for $a \gg 0$. These parametrized functions satisfy appealing properties that will be discussed in Section 6.3.1.

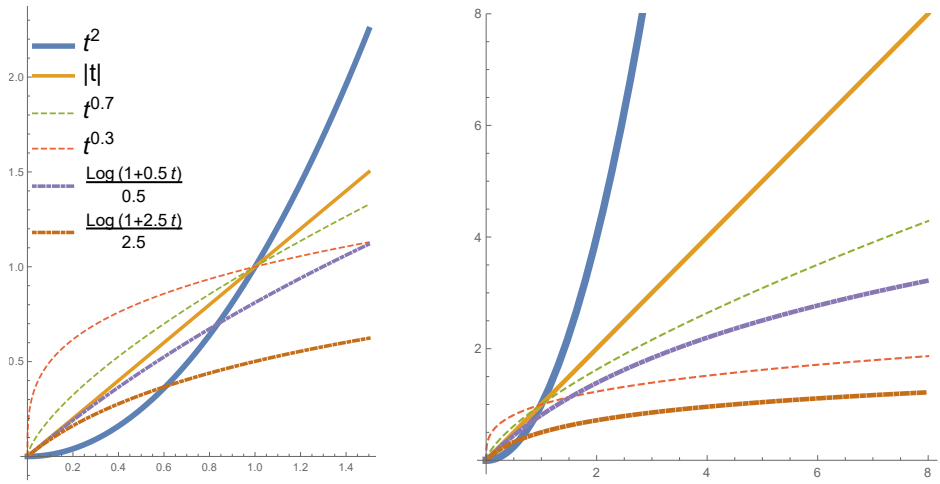



Fig. 2.1 Plots of different penalty functions, for small $t$ values (left), for large $t$ values (right).

The effect of the regularization can be illustrated in Fig. 2.1 where we visualize the plots of $L_2$ norm, $L_1$ norm, $L_p$ norm for $p = 0.7$ and $p = 0.3$, and the logarithmic penalty function $\phi_{\log}(t; a)$ for concavity parameter $a = 0.5$ and $a = 2.5$. Focusing on these penalties, the $L_2$ norm strongly penalizes large values and mildly the small ones. The $L_1$ norm slightly improves the $L_2$ norm penalty by penalizing more small values around zero, Fig. 2.1 left, but still the large values are quite strongly penalized, thus, for example the original large values (intensities) are also decreased. The $L_1$ norm enlightens the concept of sparsity, as those function values that are small, are strongly penalized unless they diminish to zero. The $L_p$ norm, as $p$ goes to zero, enforces this concept of sparsity. Small function values are penalized even more, while the large ones, Fig. 2.1 right, are not penalized so strongly, as in $L_1$ or $L_2$ case, thus, the original expected intensities are better preserved. At last, the $\phi_{\log}(t; a)$ parametrized penalty function acts in-between $L_1$ and $L_p$ norm, as for the concavity parameter $a \to 0^+$, it asymptotically converges to $L_1$ norm, and with increasing $a$, it gets more concave, having similar properties to $L_p$ norm with $p < 1$.

The penalties in (2.6) and (2.7) are non-convex functions. Therefore, one can no longer deploy the classical tools for convex optimization, but has to turn to different theoretical and numerical strategies.

In case the regularization is applied to the magnitude of the function gradients, the $L_2$ norm penalty acts as diffusion, the $L_1$ norm penalty creates a piecewise-constant functions with smoothed jumps in function gradients, while the non-convex penalty leads to a piecewise-constant functions, with better preserved jumps in function gradient.

In this work we strongly take advantage of non-convex penalties in variational shape partitioning. In particular, we adopt a two steps approach that consists of the following steps:

1. Regularization and reconstruction of the observed data producing the partitioning function $u$. More precisely, the approximated solution $u^*$ will be obtained by minimizing a variational model of the form

$$u^* = \arg\min_u \{\mathcal{J}(u) = \mathcal{F}(u) + \mu\mathcal{R}(u)\} \ . \tag{2.8}$$

2. Apply a thresholding strategy or a similar simple clusterization method, e.g. K-means, on the partitioning function $u^*$, to produce the final decomposition into $K$ parts.

   Eventually, the tracking/smoothing of the partitions' boundary can be required, according to the application context.

A state-of-the-art variational model that can be written in form (2.8) is the well-known Mumford-Shah functional introduced in [86] for image segmentation. In the following we briefly provide its overview since this model has strongly motivated the partitioning variational models proposed in this work.

## 2.1 Mumford-Shah Variational Models

Let $\Omega \subset \mathbb{R}^d$ be a given bounded open set and $f : \Omega \to \mathbb{R}^d$ a measurable function on it. The Mumford-Shah functional provides a partition $\Omega = \Omega_1 \cup \Omega_2 \cup \cdots \cup \Omega_K$ of the domain $\Omega$ with respect to $f$ such that

($\iota$) the function $f$ varies smoothly and/or slowly within each $\Omega_i$

($\iota\iota$) the function $f$ varies discontinuously and/or rapidly across most of the boundary $\Gamma \subset \Omega$ between different $\Omega_i$.

The problem is formulated as the minimization of the following functional

$$\mathcal{J}(u, \Gamma) = \int_\Omega |f - u|^2 d\Omega + \mu \, Length(\Gamma) + \eta \int_{\Omega\setminus\Gamma} |\nabla u|^2 d\Omega, \tag{2.9}$$

which is known as the *piecewise-smooth Mumford-Shah model*.

This model approximates $f$ by a piecewise-smooth function $u : \Omega \to \mathbb{R}^d$ whose restrictions $u_i$ to the pieces $\Omega_i$ of a partition $i$ are continuous or differentiable everywhere and maybe discontinuous across the $(d-1)$-dimensional jump set $\Gamma$. The weight $\mu > 0$ controls the length of the jump set $\Gamma$ and $\eta > 0$ enforces the smoothness of $u$ away from $\Gamma$. The second term in (2.9) imposes that the boundaries $\Gamma$ be as short as possible. The first term in (2.9) is the so-called fidelity term, which represents the closeness of $u$ to the observed data $f$. The restriction of (2.9) for the limiting case $\eta \to \infty$, imposes zero gradient outside $\Gamma$, that is $u$ is required to assume the constant value $\bar{f}_i$ on each connected component $\Omega_i$. The resulting minimization problem, known as the *piecewise-constant Mumford-Shah model*, often referred to as the special case of the Chan-Vese model [29], considers the following functional

$$\mathcal{J}(\Gamma) = \sum_{i=1}^{K} \int_{\Omega_i} |f - \bar{f}_i|^2 d\Omega + \sum_{i=1}^{K} \mu \, Length(\Gamma_i) \tag{2.10}$$

where $Length(\Gamma_i) = |\partial\Omega_i|$ and $\bar{f}_i := \text{mean}_{\Omega_i} f$. The minimization of the functionals (2.9) or (2.10) represents a non-convex optimization problem due to the second term, so the obtained solutions are in general local minimizers.

In [27] the authors presented a convex relaxation of the Mumford-Shah model in the form

$$\mathcal{J}(u) = \int_{\Omega} |u(x) - f|^2 d\Omega + \mu \int_{\Omega} |\nabla u| d\Omega \,, \tag{2.11}$$

which is proved to be equivalent to the solution of (2.10). This result on convex relaxation led to new studies that improve the model (2.11) [99, 72]. Nevertheless, recently, non-smooth, non-convex functionals have shown remarkable advantages over convex ones, for example in the image restoration context; theoretical explanation and numerical examples can be found in numerous articles [69],[91],[68].

We introduce a general form of the *piecewise-smooth* Mumford-Shah functional, applied to the so-called *partitioning function* $u$, that will be considered in this work as

$$\mathcal{J}(u) = \int_{\Omega} |f - u|^2 d\Omega + \mu \int_{\Omega} \phi(|\nabla u|) d\Omega + \eta \int_{\Omega} |\nabla u|^2 d\Omega \tag{2.12}$$

where the first term represents the fidelity to the observed data $f$ produced by preliminary shape analysis techniques, the second term approximates the measurement of the boundary lengths via the regularization of $u$ through a non-convex penalty function $\phi(\cdot)$, weighted by the parameter $\mu > 0$. The last term represents the quadratic regularization of the gradient of $u$, thus smoothing the gradient away from the boundary.

The usual choice for the penalty function $\phi(\cdot)$ in (2.12) is the $L_1$ norm that leads to the well known Total Variation (TV) regularizer [22]. Nevertheless, in this work, we focus on different, non-convex choices for $\phi(\cdot)$ as the sparsity-inducing $L_p$ pseudo-norm for $0 < p < 1$ or a set of non-convex parametrized function penalties that are described in Section 6.3.1.

Moreover, let us notice that the *piecewise-constant* relaxed Mumford-Shah functional is obtained from (2.12) by setting $\eta = 0$.

The variational model (2.12) and its variants will be applied both on the partitioning of 3D shape and to the segmentation of scalar fields defined over a 2-manifold. 2D images mapped over surfaces are special cases of the latter context. For example QR-codes mapped on the object's surface.

# Chapter 3

# First Order Optimization Methods

In this chapter we present some recent first-order numerical optimization methods for the solution of convex/non-convex and possibly non-smooth unconstrained minimization problems with cost-function $\mathcal{J}(u)$ defined as

$$u^* = \arg\min_{u\in\Theta} \{\mathcal{J}(u) := \mathcal{F}(u) + \mu\mathcal{R}(u),\, u \in \Theta\} \tag{3.1}$$

where the parameter $\mu > 0$ controls the trade-off between the fidelity term $\mathcal{F}(u)$ and the regularization term $\mathcal{R}(u)$ which stabilizes the inverse problem.

The set $\Theta$ is a finite dimensional Euclidean space with inner product $\langle\cdot,\cdot\rangle$ and norm $\|\cdot\| = \langle\cdot,\cdot\rangle^{1/2}$. For a *convex* $\mathcal{J}(\cdot)$, the natural optimization strategy is to apply a method from the well-known *convex* optimization tools and thus to use results from convex analysis that provide a unique, global minimizer for (3.1). In case $\mathcal{J}(\cdot)$ is non-convex, we can still utilize *convex* optimization methods, or the ones devised for the *non-convex* optimization. However, the solution provided by these numerical methods applied to non-convex functional results in a minimizer that depends on the initialization and the convergence is guaranteed to the local minima only. Since, as we will see in the next chapters, the shape partitioning is neater if the regularization term is non-convex rather then if it is convex, in order to maintain convexity of the objective function $\mathcal{J}(u)$ while using non-convex regularizers, we can follow the *Convex Non-Convex* (CNC) strategy to derive conditions on the parametrized penalty functions $\phi(t, a)$ that guarantee that the overall objective functional $\mathcal{J}(u)$ is convex.

We restrict this brief survey to the First-Order Methods (FOM), which are iterative methods that only exploit information of the objective function and its gradient (sub-gradient) and converge rather slowly. However, for very large-scale optimization problems, as the considered shape partitioning problem, usually a fast lower-precision solution is

favoured. Appropriate references to the methods described in this chapter can be found in various literature, see for example [9, 18, 44, 19, 94, 26, 109].

## 3.1 Convex Optimization Methods

We first recall a basic result on the existence and uniqueness of a solution to the problem (3.1), see [36]. Under the following assumptions

($\iota$) $\mathcal{F}(u)$ and $\mathcal{R}(u)$ are proper, lower semi-continuous and convex

($\iota\iota$) $\mathcal{F}(u)$ is differentiable with $\nabla\mathcal{F}(u)$ is Lipschitz continuous with constant $L$

($\iota\iota\iota$) $\mathcal{J}(u)$ is coercive

problem (3.1) possesses at least one solution. Moreover, if $\mathcal{J}(u)$ is strictly convex, the solution is unique.

In the following we briefly describe the most representative first order methods and strategies used in case of optimization problems with splitted objective function in the form (3.1).

### Gradient Descent Method

In case $\mathcal{J}$ is convex and smooth, the simplest and the most straightforward method to solve (3.1) is to follow the gradient-descent or steepest descent method, probably one of the oldest optimization methods, which generates a sequence $\{u^k\}$ for initialized $u^0$ as follows

$$u^{k+1} = u^k - \alpha\nabla\mathcal{J}(u^k)\,, \quad k = 0, 1, \ldots \tag{3.2}$$

where the scalar parameter $\alpha > 0$ stands for a suitable step-size in the steepest-descent direction, which can be either constant or varying throughout the iterations in a more sophisticated way.

The iterations (3.2) create a sequence $u^0, u^1, \ldots$, such that $\mathcal{J}(u^0) \geq \mathcal{J}(u^1) \geq \ldots$. After $k$ iterations $u^k$ will be close enough to a minimizer. In case of non-differentiable $\mathcal{J}$, the gradient is replaced by the subgradient at the points which are not differentiable. On the other hand, the subgradient method [109] results in the same steepest directions at differentiable points.

### Proximal Gradient Method

This method is useful in particular when dealing with a functional defined in the form (3.1) with $\mathcal{F}(u)$ and $\mathcal{R}(\cdot)$ proper, convex, and $\mathcal{J}(u)$ that can be splitted into a differentiable part $\mathcal{F}(\cdot)$ and a possibly non-differentiable term $\mathcal{R}(\cdot)$. Such splitting is not unique as it can be performed in various way, thus, leading to different implementations when applied.

The $k$-th iteration of the proximal gradient method reads as follows

$$u^{k+1} = \mathbf{prox}_{\alpha^k \mathcal{R}}\left(u^k - \alpha^k \nabla \mathcal{F}(u^k)\right) \tag{3.3}$$

where $\alpha^k > 0$ is the step-size scalar parameter and the proximal operator $\mathbf{prox}_{\alpha f}(x)$ of a convex function $f$ at $x$ with positive parameter $\alpha$ is defined as

$$\mathbf{prox}_{\alpha f}(x) = \arg\min_u \left( f(u) + \frac{1}{2\alpha}\|u - x\|_2^2 \right) . \tag{3.4}$$

Many FOM methods are based on finding the minimizer of a quadratic approximation $\mathcal{Q}_\alpha(u, x)$ to $\mathcal{J}(u)$ which consists of sum of the regularization term $\mathcal{R}(u)$ and the quadratic approximation to the fidelity $\mathcal{F}(u)$ at a given point $x$ defined as

$$\mathcal{Q}_\alpha(u, x) := \mathcal{F}(x) + \langle u - x, \nabla \mathcal{F}(x)\rangle + \frac{1}{2\alpha}\|u - x\|_2^2 + \mu \mathcal{R}(u) \tag{3.5}$$

where $\alpha > 0$ represents the step-size. Applying the proximal operator defined in (3.4) to (3.3), we obtain

$$u^{k+1} = \arg\min_u \left\{ \mu \mathcal{R}(u) + \frac{1}{2\alpha}\left\| u - \left(u^k - \alpha \nabla \mathcal{F}(u^k)\right)\right\|_2^2 \right\} \tag{3.6}$$

that can be rewritten into the following form

$$u^{k+1} = \arg\min_u \left\{ \mathcal{F}(u^k) + \left\langle u - u^k, \nabla \mathcal{F}(u^k)\right\rangle + \frac{1}{2\alpha}\|u - u^k\|_2^2 + \mu \mathcal{R}(u) \right\} , \tag{3.7}$$

which exactly coincides with the minimization of the quadratic approximant $\mathcal{Q}_\alpha(u, u^k)$ in (3.5) evaluated at the previous time-step $u^k$.

Let $\mathcal{F}(u)$ be a differentiable function and $\nabla \mathcal{F}(u)$ be Lipschitz continuous with constant $L > 0$, i.e.

$$\|\nabla \mathcal{F}(u) - \nabla \mathcal{F}(x)\|_2 \leq L\|u - x\|_2 , \quad \forall u, x \in \Theta .$$

Assuming that the step-size is in the interval $\alpha^k \in (0, 1/L]$ or estimated through the line search, the sequence $\{u^k\}$ generated by the proximal gradient method (3.3) converges to an optimal solution $u^*$ of the problem (3.1), not necessarily unique, with convergence rate of order $O(1/k)$, see Th. 1.2 in [9]. However, the typical line search parameter value used is $1/2$.

The method itself is a generalization of other well-known approaches, as it reduces for $\mathcal{F} = 0$ to proximal minimization [94] and for $\mathcal{R} = 0$ to the classical gradient-descent algorithm.

### Forward-Backward Splitting Method

The forward-backward splitting method can be derived from the relative difference between two successive iterations of the gradient-descent method

$$\frac{u^{k+1} - u^k}{\alpha} = -\nabla\mathcal{F}(u^k) - \nabla\mathcal{R}(u^{k+1}) \tag{3.8}$$

where the gradient $\nabla\mathcal{F}$ is evaluated explicitly from iteration $k$ (forward step) while the gradient $\nabla\mathcal{R}$ is implicit at iteration $k+1$ (backward step). Rearranging (3.8), one obtains the known formula of Forward-Backward splitting that can be written into a two-step iteration as

$$\begin{aligned}
\text{Fw. step} : x^{k+1} &= (I - \alpha\nabla\mathcal{F})u^k \\
\text{Bw. step} : u^{k+1} &= (I + \alpha\nabla\mathcal{R})^{-1}x^{k+1} && (3.9) \\
&= \arg\min_u \left\{\mathcal{R}(u) + \frac{1}{2\alpha}\|u - x^{k+1}\|_2^2\right\} && (3.10) \\
&= \mathbf{prox}_{\alpha\mathcal{R}}(x^{k+1}) \,,
\end{aligned}$$

which exactly coincides with the Proximal Gradient Iteration.

Assuming that the problem (3.1) has a solution and that the time-step $\alpha$ is in the interval $\alpha \in (0, 2/L)$, the sequence $\{u^k\}$ generated by the F-B Splitting Method converges to a solution of problem (3.1). This is proved in [36] for the case of the proximity operator in (3.10) available in exact form, and in [120] when it can only be approximated up to a certain precision. The convergence rate is of order $O(1/k)$.

### Iterative Shrinkage-Thresholding Algorithms

The class of iterative shrinkage-thresholding algorithms (ISTA) is related to forward-backward splitting method. The shrinkage-thresholding algorithms were designed under

the assumption that $\mathcal{J}(u)$ can be splitted into convex, smooth $\mathcal{F}(u)$ and convex, possibly non-smooth $\mathcal{R}(u)$ parts, moreover, $\mathcal{R}(u)$ is considered simple, separable function in form

$$\mathcal{R}(u) = \sum_{i=1}^{n} \mathcal{R}_i(u_i) \ . \tag{3.11}$$

This assumption allows to split the minimization problem into $n$ one-dimensional problems that are solved via a shrinkage operator or a thresholding defined as

$$\mathcal{T}_\alpha(x)_i = (|x_i| - \alpha)_+ \ \mathrm{sgn}(x_i) \ .$$

The most popular algorithm accelerating ISTA is Fast ISTA (FISTA) [8], however, the idea of accelerating ISTA was first introduced by Nesterov in [87] ISTA represents an extension to the gradient descent method with acceleration that aims to solve the problem (3.1) via minimization of the quadratic approximation $\mathcal{Q}_\alpha(u, x)$ in (3.5) to $\mathcal{J}(u)$ defined as

$$p_\alpha(x) = \arg\min_u \{\mathcal{Q}_\alpha(u, x)\} \ . \tag{3.12}$$

By ignoring the constant terms, we can rewrite (3.5) as

$$p_\alpha(x) = \arg\min_u \left\{ \mu\mathcal{R}(u) + \frac{1}{2\alpha} \left\| u - (x - \alpha\nabla\mathcal{F}(x)) \right\| \right\} \ , \tag{3.13}$$

from which the iteration $k$ of ISTA reads as

$$u^{k+1} = p_\alpha(u^k) \ . \tag{3.14}$$

Since the $\mathcal{R}(u)$ term is separable, the computation of $u^{k+1}$ reduces to solving a one-dimensional minimization problem for each of its components, which by simple calculus produces

$$u^{k+1} = \mathcal{T}_\alpha \left( u^k - \alpha\nabla\mathcal{F}(u^k) \right)$$

where $\mathcal{T}_\alpha$ is the shrinkage operator.

Assuming the problem (3.1) has at least one solution, and that the step-size $\alpha \in (0, 1/L)$ then the sequence $\{u^k\}$ converges to a minimizer of $\mathcal{J}(u)$ with convergence rate order of $O(1/k)$.

ISTA speeds up the convergence of sub-gradient and proximal gradient methods under the assumption (3.11).

FISTA [8] on the other hand, instead of the actual iteration $u^{k+1}$ uses in its iteration a specific combination of $u^{k+1}$ and $u^k$ and sets the relaxation parameter $\alpha$ automatically

and iteratively as $\alpha^k = \frac{t^k - 1}{t^{k+1}}$. The iteration $k$ for initialized $u^0$, $x^0 = u^0$ and $t^0 = 1$ reads

$$\begin{aligned} u^{k+1} &= p_\alpha(x^k) \quad \text{in (3.13)} \\ t^{k+1} &= \frac{1 + \sqrt{1 + 4(t^k)^2}}{2} \\ x^{k+1} &= u^{k+1} + \alpha^k(u^{k+1} - u^k) \ . \end{aligned} \tag{3.15}$$

FISTA speeds up the convergence rate of ISTA to $O(1/k^2)$, thus the convergence of sub-gradient and proximal gradient methods as well. The assumption (3.11) of splitting $\mathcal{R}$ to easier, lower dimensional problems still holds in this case.

### Alternating Minimization Methods

In this category we recognize various methods based on the Lagrangian, its variations and splitting strategies that enclose methods as Augmented Lagrangian (ALM) [12] , Alternating Directions Method of Multipliers (ADMM) [18] or the Split-Bregman Method [45].

The Augmented Lagrangian method, often referred to as the Multipliers Method, was derived from the classical penalty method. The aim is to solve the optimization problem

$$\min_{u \in \Theta} \{\mathcal{J}(u) := \mathcal{F}(u) + \mu \mathcal{R}(Au)\} \ , \tag{3.16}$$

with $A$ a given linear map defined in $\Theta$. The unconstrained optimization problem (3.16) is equivalent via the standard variable splitting/substitution trick to the equality constrained problem in form

$$\begin{aligned} &\min_{u \in \Theta,\, x \in \Theta_1} \{\mathcal{J}(u) := \mathcal{F}(u) + \mu \mathcal{R}(x)\} \\ &s.t. \quad Au = x \ . \end{aligned} \tag{3.17}$$

The main difference from the penalty method is that the quadratic penalty function is not added to the cost function itself, but rather to the Lagrangian of the problem that is defined as follows

$$\mathcal{L}(u, x; \rho) = \mathcal{F}(u) + \mu \mathcal{R}(x) + \langle \rho, Au - x \rangle + \frac{\beta}{2} \|Au - x\|_2^2 \tag{3.18}$$

where $\rho$ represents the dual variable of Lagrangian multipliers associated to the so called consensus constraint $Au = x$, and $\beta > 0$ is additional trade-off penalty parameter associated with the quadratic penalty term. The Augmented Lagrangian function defined

in (3.18) represents a sort of classical Lagrangian penalized with a quadratic term and is equivalent to the Lagrangian of the problem

$$\begin{aligned} &\min_{u\in\Theta,\, x\in\Theta_1} \left\{ \mathcal{J}(u) := \mathcal{F}(u) + \mu\mathcal{R}(u) + \frac{\beta}{2}\|Au - x\|_2^2 \right\} \\ &s.t. \quad Au - x = 0 \; . \end{aligned} \tag{3.19}$$

The main idea behind ALM is that minimization of the classical Lagrangian for problem (3.17) is difficult to handle, however, its penalized version (3.18) by a quadratic term is an easier task. The iterations of the Augmented Lagrangian algorithm read as follows

$$\begin{aligned} (u^{k+1}, x^{k+1}) &= \arg\min_{u,x} \mathcal{L}(u, x; \rho^k) \\ \rho^{k+1} &= \rho^k + \beta(Au^{k+1} - x^{k+1}) \; . \end{aligned} \tag{3.20}$$

In (3.20), the issue arises when the optimization of $\mathcal{F} + \mathcal{R}$ in the original functional in (3.17) is not easy to perform, while it is easy for $\mathcal{F}$ and $\mathcal{R}$ separately.

The original ALM leads to joint minimization of $\mathcal{L}$ in two variables $u$ and $x$ what can be a challenging task. Another popular optimization algorithm is the so-called Alternating Directions Method of Multipliers (ADMM).

In ADMM the solution of the saddle-point problem defined in (3.18) is determined by minimizing alternatively w.r.t. $u$ and $x$ as follows

$$\begin{aligned} u^{k+1} &= \arg\min_{u} \mathcal{L}(u, x^k; \rho^k) \\ x^{k+1} &= \arg\min_{x} \mathcal{L}(u^{k+1}, x; \rho^k) \\ \rho^{k+1} &= \rho^k + \beta(Au^{k+1} - x^{k+1}) \; , \end{aligned} \tag{3.21}$$

with initialized $\rho^0$ and $x^0$.

The advantage of the above iterations is that at iteration $k$, the update is done by minimizing over just one variable, while keeping the others fixed in the most recent values, thus, solving a lower dimensional problem that is also presumably easier to solve.

The sub-problems in (3.21) can be reformulated in terms of the proximal operator as

$$\begin{aligned} u^{k+1} &= \mathbf{prox}_{\frac{1}{\beta}\mathcal{F}}(x^k - \rho^k) \\ x^{k+1} &= \mathbf{prox}_{\frac{1}{\beta}\mathcal{R}}(u^{k+1} + \rho^k) \\ \rho^{k+1} &= \rho^k + \beta(Au^{k+1} - x^{k+1}) \; . \end{aligned} \tag{3.22}$$

Under the assumption that $\mathcal{F}(u)$ and $\mathcal{R}(u)$ are closed, proper and convex; and assuming the Augmented Lagrangian $\mathcal{L}$ in (3.18) has a saddle point, then the ADMM iterates (3.21) converge to $(u^*, x^*)$ that is the solution of (3.17) while strong duality for $\rho^*$ holds, without any additional assumptions on $A$ [18, 44]. The obtained solution $(u^*, x^*)$ does not necessarily has to be optimal.

The ADMM framework is popular also for non-convex optimization problems, however, there is no general theory for convergence in presence of non-monotone operators or non-convex functionals to be optimized [44]. The recent development [52, 75, 123] suggests that a general convergence theory in the non-convex case may be developed, although, for some specific problems it may be still not possible to successfully prove convergence of ADMM-based algorithms.

### Primal-Dual Algorithm

The Primal-Dual strategy was introduced for additional structures present in $\mathcal{J}$. In particular, the presence of a linear continuous operator $A$ defined in $\Theta$ and applied on $u$ in one of the functionals $\mathcal{F}(Au)$ such that its presence makes the evaluation of the proximal operator for $\mathcal{F}(Au)$ non-trivial.

Thus, the primal problem is defined as

$$\min_u \{\mathcal{F}(Au) + \mathcal{R}(u)\} \ . \tag{3.23}$$

Let us recall the convex Legendre conjugate for $\mathcal{F}$

$$\mathcal{F}^*(y) = \langle Au, y\rangle - \mathcal{F}(Au) \ ,$$

from which we can write the primal-dual problem as the saddle-point problem

$$\min_u \max_y \langle Au, y\rangle + \mathcal{R}(u) - \mathcal{F}^*(y) \ . \tag{3.24}$$

From (3.24), the dual problem is defined as

$$\max_y - (\mathcal{F}^*(y) + \mathcal{R}^*(-A^*y)) \ . \tag{3.25}$$

The Primal-Dual Algorithm [26] for initialized $(u^0, y^0)$ updates the variables as follows

$$\begin{aligned} y^{k+1} &= \mathbf{prox}_{\mathcal{F}^*}(y^k + \alpha A\bar{u}^k) \\ u^{k+1} &= \mathbf{prox}_{\mathcal{R}}(u^k - \beta A^* y^{k+1}) \\ \bar{u}^{k+1} &= u^{k+1} + \theta(u^{k+1} - u^k) \end{aligned} \tag{3.26}$$

where $\alpha > 0$, $\beta > 0$ are chosen step sizes in order to satisfy $\alpha\beta L^2 < 1$ for $L$ the Lipschitz constant $L = \|A\|$ and $\theta \in [0, 1]$ is the extrapolation step size.

The primal-dual gap is defined as the difference of the primal (3.23) and the dual problem (3.25) that vanished for $(\hat{u}, \hat{y})$ being the optimal saddle-point for convex functions.

In relation to the previous methods, for $A = I$ the primal-dual algorithm is equivalent to ADMM algorithm.

## 3.2 Non-Convex Optimization Methods

In this section, we focus on the algorithms used when $\mathcal{J}$ is non-convex. Many of the algorithms from the previous Section 3.1 can be in general used also for the non-convex optimization, if treated properly. Non-convex regularizers $\mathcal{R}(u)$ are advantageous since they usually result in sparser solutions for a given residual energy. On the other hand, non-convex formulations are generally more difficult to solve due to suboptimal local minima, initialization issues, etc.. However, in many applications, the local minima solutions provide satisfactory results w.r.t. sparsity/closeness-to-optimal-solution.

In what follows, we summarize the state-of-the-art methods and strategies.

### Iteratively Re-weighted Methods

In this category fall the methods that approximate the non-convex penalty norm, usually enclosed in $\mathcal{R}$, through a weighted convex one for which the weights are re-computed at each iteration. In particular, the main methods are Iteratively Re-weighted Least Squares (IRLS) [124, 46], $L_1$ Norm (IRL1) [23, 80], and in general Iteratively Re-weighted Norm (IRN) [101].

All the methods are based on the same principle of relaxing $L_1$ or $L_p$, $p \in (0, 1)$ norm by weighted $L_1$ or $L_2$ norms what allows for a convex optimization at each iteration.

### Majorization-Minimization Methods

This strategy seeks minimum of the problem (3.1) in two steps, MAJORIZATION and MINIMIZATION. At each iteration $k$ in the majorization step one constructs the so-called surrogate function $g(u, u^k)$ to the concave functional $\mathcal{J}$ at point $u^k$ such that the following properties are satisfied

$$(\iota)\ \ \mathcal{J}(u^k) = g(u^k, u^k)\,, \qquad \forall\, u \in \Theta$$

$$(\iota\iota)\ \ \mathcal{J}(u) \leq g(u, u^k)\,, \qquad \forall\, u, u^k \in \Theta\,.$$

The above majorization relation between functions is closed under the formation of sums, non-negative products and limits. This means that $g(u, w)$ lies above $\mathcal{J}(u)$ and is tangent at $u \equiv w$. In general, this strategy [54] is suited also for maximization problems, where the surrogate function $g(u, u^k)$ can be defined as minorizating one. In general, a surrogate function is not necessarily convex and can be constructed as

- Jensen's inequality – leading to Expectation-Maximization (EM) algorithms
- Chord above the function graph – used in image reconstruction
- Supporting hyperplanes
- Quadratic, piecewise linear functions
- Arithmetic-geometric mean inequality
- The Cauchy-Schwartz inequality – multidimensional scaling

The minimization step consists in solving

$$u^{k+1} = \arg\min_{u \in \Theta} g(u, u^k)\,. \tag{3.27}$$

This scheme immediately implies that

$$g(u^k, u^{k-1}) \leq g(u, u^{k-1})$$

for every $u \in \Theta$ and hence

$$\mathcal{J}(u^k) \leq \mathcal{J}(u^{k-1})$$

for every $k \geq 1$. Thus, naturally producing a descent sequence for minimizing problem (3.1).

In [32] an MM strategy is applied to solve a non-convex optimization problem in the form (3.1) and the convergence of the algorithm is investigated by using recent results in non-convex optimization.

## 3.3 Convex-NonConvex Strategy

The CNC strategy is a rather novel strategy for obtaining a global optimal solution while using non-convex penalty functions. Originally, the idea was formulated, apart from greedy strategies, in Graduated Non-Convexity Algorithm (GNC) by Blake and Zisserman [13] and Nikolova [89]; and has been successfully applied to image restoration and reconstruction.

More precisely, the key idea is to control the degree of non convexity of the regularizer $\mathcal{R}(u)$ in order to guarantee that the total objective function $\mathcal{J}(u)$ in (3.1) is convex. This is obtained by balancing the positive second derivatives in the fidelity term $\mathcal{F}$ against the negative second derivatives in the $\mathcal{R}$ term [13].

CNC strategies have been successfully applied by Selesnick in [30, 106] where he proposes a set of parametrized non-convex penalty functions $\phi(\cdot; a)$ to control the convexity of $\mathcal{J}(u)$. The parameter $a$ in $\phi(\cdot; a)$ stands for the concavity parameter which controls the shape of the penalty function $\phi$. In CNC regime, suitable conditions on $a$ should be derived to guarantee the convexity of $\mathcal{J}(u)$. Other CNC approaches have been proposed by Morigi et al. [68, 70, 67] in the context of image restoration.

# Chapter 4

# Spatial Discretization

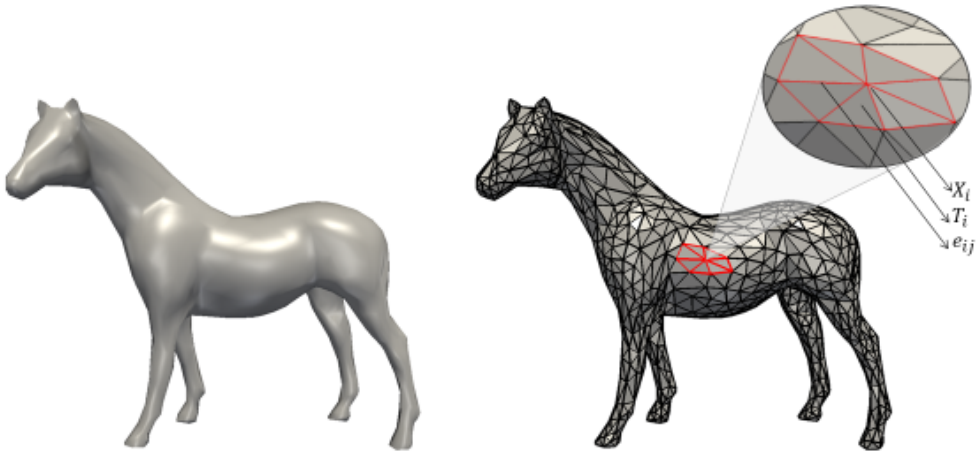



Fig. 4.1 Approximation of a 2-manifold (left) by a triangular mesh (right).

In this chapter, we will focus on the spatial discretization and approximation of a continuous domain $\Omega$ which can represent a closed/open 2-manifold $\mathcal{M}$ and the discretization of local, continuous differential operators defined on it. A triangular mesh $M := (V, T)$ represents a piecewise-linear approximation to $\mathcal{M}$ embedded in $\mathbb{R}^3$. $V$ is the set of $n$ vertices $V \in \mathbb{R}^{n\times 3} = \{X_1, \ldots, X_n\}$ sampled over the surface $\mathcal{M}$. The connectivity set $T \in \mathbb{N}^{n_T\times 3}$ defines how vertices in $V$ are interconnected into $n_T$ triangles. This connectivity graph $T$ provides automatically the set of $\frac{3n_T}{2}$ unique edges $E \subseteq V \times V$, see an example illustrated in Fig. 4.1.

In this work, we will deal with different polygonal mesh representations having different properties. The constituents of a two-dimensional polygonal mesh are *vertices*, *edges* and *facets* (*elements*). In general, facets may share vertices and edges. We consider only *conforming* meshes, in which every two faces share either a single vertex, or an entire common edge. By this assumption, we do not consider meshes having the so-called *T-joints*.

The *valence* of a vertex is the number of its incident edges, while its *star* is the set of its incident facets and edges. A *1-ring neighbourhood* is the set of adjacent vertices, if not stated otherwise.

A *triangle mesh* is a mesh in which all facets are triangles, while a *quad mesh* is a mesh in which all facets are quadrilaterals that not necessarily have to be planar. In quad mesh, a vertex is called *regular* if it has valence equals to 4 while for a triangular meshes the *regular* vertex has valence 6. The vertices having valence different from 4 (or 6 for triangular meshes) are called *extraordinary* or *irregular* vertices (EV).

A *regular mesh* is a mesh that can be globally mapped to a rectangular domain. We will refer to this type of mesh as patch. They have limited applicability as they are suitable for surfaces of disk or toroidal topology only.

A mesh is called *valence semi-regular* if most of its vertices are regular ones. Often a valence semi-regular mesh is referred to simply as *semi-regular*. However, some literatures distinguish these two types of mesh and they define a *semi-regular mesh* as a mesh created by gluing together a set of different patches where each patch is a regular mesh with boundary.

On the opposite, a mesh having a large number of irregular vertices is called *unstructured* mesh [15].

The local 1-ring vertex neighbourhood of a vertex $X_i$, denoted by $N(X_i)$, can be defined as

$$N(X_i) = \{X_j : \forall e_{ij} \in E\} \ .$$

Therefore, $v_i = |N(X_i)|$ is the valence of vertex $X_i$.

A 1-ring triangle neighbourhood of a vertex $X_i$, denoted by $N_\triangle(X_i)$, is defined as the set of triangles $\tau_j$ which share the vertex $X_i$.

For non-uniform unordered point cloud $V$ no direct topologic relation between points is provided, thus the set $N(X_i)$ will be defined by the indexes of the points $X_j \in V$ which are inside the ball whose diameter is related to the density measure (4.1) defined in the following.

Appropriate density features of a scattered data set $V$ are the separation distance $q_V$ and the fill distance $h_V$ of the set $V$, which, for a compact, connected region $\Omega$, are

respectively defined as:

$$\begin{aligned} q_V &:= \tfrac{1}{2} \min_{1\le i<j\le n} \|X_i - X_j\|_2, \\ h_V &:= \sup_{X\in\Omega} \min_{1\le i\le n} \|X - X_i\|_2. \end{aligned} \tag{4.1}$$

The separation distance $q_V$ is half the minimum distance between two points in $V$, and the local sample density or fill distance $h_V$ is defined as the radius of the largest inner empty disk. We can measure the uniformity of the set $V$ by using the positive scalar value $\rho = \rho(V) = \frac{q_V}{h_V}$. Optimal values for $\rho$ depend on the type of data distribution; e.g. for planar domains, if the data are points of an equilateral triangular grid, then $\rho_{opt} = \sqrt{3}/2$, while for a regular square grid we have $\rho_{opt} = \sqrt{2}/2$, [24].

The usefulness of the local neighbourhood shows advantage in the estimation of the normals, or the approximation to local differential operators of a function defined over the manifold $\mathcal{M}$ and sampled over the vertices $V$ of the associated mesh $M$.

## 4.1 Discrete Differential Operators

Assuming that a scalar function $u\colon \mathcal{M} \to \mathbb{R}$, is sampled over the vertices $V$ of the mesh $M$. Denoting by $u_i := u(X_i)$, $i = 1, \ldots, n$, the sampled functions are represented by the vectors $u \in \mathbb{R}^n$. Since polygonal meshes are piecewise linear surfaces, the concepts of continuous differential operators of a function $u$ cannot be applied directly. The assumption required is that meshes can be interpreted as piecewise linear approximations of smooth surfaces in order to compute the discrete differential operators directly from the mesh data.

The general idea is to compute discrete differential properties as spatial averages over a local neighbourhood $N(X_i)$ of a point $X_i$ on the mesh. Towards that aim more ways of constructing the averaging region from local neighbourhoods have been used ranging from Barycentric cells, Voronoi and Mixed Voronoi cells used in up to the whole $N(X_i)$ used in Finite volume approximations, or the elements (triangles) themselves are considered in Finite Elements Method.

Alternatively, we can consider the mesh also as a graph structure, adapting the results from Finite Volumes/Elements methods to this simpler structure.

### 1) Weighted Gradient

At each vertex $X_i$, the 1-ring neighbourhood $N(X_i)$ defines $v_i$ discrete directional derivatives, each calculated along a different direction identified by the edges $e_{ij}$, $j = 1, \ldots, v_i$. The discrete analogue of the *directional derivative* on a 2-manifold $\mathcal{M}$ as the edge derivative of $u$ at a vertex $X_i \in V$ along an edge $e_{i,j} \in E$ is defined by the following difference operator $du(X_i, X_j)$

$$\frac{\partial u}{\partial e_{i,j}} \approx du(X_i, X_j) := w(X_i, X_j)(u(X_j) - u(X_i)) \tag{4.2}$$

where $w : V \times V \to \mathbb{R}_+$ is a symmetric measure defined between points $X_i$ and $X_j$ and $w(X_i, X_j) = 0$ if $(X_i, X_j) \notin E$.

Hence, the intrinsic (weighted) gradient operator $\nabla_w u(X_i)$ of a function $u$ evaluated at vertex $X_i$ is a vector in the tangent space $T_{X_i}\mathcal{M}$, whose magnitude can be approximated as follows

$$\|\nabla_w u(X_i)\|_2^2 = \sum_{j \in N(X_i)} w(X_i, X_j)^2 (u(X_j) - u(X_i))^2. \tag{4.3}$$

In the partitioning model that we will introduce in Chapter 6, we need a suitable proposal for the weights in (4.3). To that aim, taking into account the geometry of $M$, the corresponding weights can be defined as

$$w_{ij} := w(X_i, X_j) = \frac{1}{|e_{ij}|}\,. \tag{4.4}$$

Alternatively, we can focus on a natural property that the boundaries should correspond to strong affinity changes on the function values between adjacent regions. Towards that aim the weights can be chosen as boundary detecting functions

$$w(X_i, X_j) \;=\; e^{-\|f(X_i) - f(X_j)\|_2^2 / 2\sigma} \tag{4.5}$$

where $f$ represents the observed data, i.e. a shape describing function. The parameter $\sigma \in (0, 1]$ in (4.5) controls how much the similarities of two local neighbours are penalized. Smaller values of $\sigma$ preserve smaller differences in the function $f$.

### 2) Laplace-Beltrami Operator

Probably the most popular and wildly used discretization of the Laplace-Beltrami operator is the cotangent formula [82, 98] that has been derived using the mixed Finite Elements/Volumes method. In the following we briefly sketch how the formula is obtained

[17]. Considering the Barycentric local averaging region $V_i$ to vertex $X_i$, the cells joints midpoints of adjacent edges $e_{ij}$, $e_{ik}$ and the barycentre of adjacent triangles $\tau$ in $N_\triangle(X_i)$.

Integrating the Laplace-Beltrami Operator of a function $u$ over this region $V_i$ and applying the divergence theorem we can write

$$\int_{V_i} \Delta_{LB} u(X)\, \mathrm{d}V_i = \int_{V_i} \mathrm{div}\nabla u(X)\, \mathrm{d}V_i = \int_{\partial V_i} \nabla u(X) \cdot n(X)\, \mathrm{d}s \tag{4.6}$$

where $n(u)$ denotes the normal vector to the boundary of $V_i$. For the barycentric cell-region $V_i$, (4.6) can be splitted into sum of integrals over all triangles $\tau_j$ in $N_\triangle(X_i)$. Thus, considering the evaluation of (4.6) on a triangle $\tau = (X_i, X_j, X_k)$, we can approximate it as

$$\int_{\partial V_i \cap \tau} \nabla u(X) \cdot n(X)\, \mathrm{d}s = \frac{1}{2} \nabla u(X) \cdot e_{jk}^{\perp} \tag{4.7}$$

where $\frac{1}{2} e_{jk}^{\perp}$ represents the counter-clockwise rotation of $e_{jk}$ that approximate the normal $n(X)$ orthogonal to midpoints-edge $\left(\frac{X_i+X_j}{2}, \frac{X_i+X_k}{2}\right)$. The gradient $\nabla u(X)$ is constant over $\tau$ an can be expressed as

$$\nabla u(X) = (u_j - u_i) \frac{e_{ik}^{\perp}}{2A(\tau)} + (u_k - u_i) \frac{e_{ij}^{\perp}}{2A(\tau)} \tag{4.8}$$

where $\frac{e_{ik}^{\perp}}{2A(\tau)}$ and $\frac{e_{ij}^{\perp}}{2A(\tau)}$ come from the derivatives of linear basis functions for barycentric interpolation over $\tau$; and $A(\tau)$ stands for the area of $\tau$. Plugging the gradient formula in (4.7) we obtain

$$\int_{\partial V_i \cap \tau} \nabla u(X) \cdot n(X)\, \mathrm{d}s = (u_j - u_i) \frac{e_{ik}^{\perp} \cdot e_{jk}^{\perp}}{4A(\tau)} + (u_k - u_i) \frac{e_{ij}^{\perp} \cdot e_{kj}^{\perp}}{4A(\tau)} \,. \tag{4.9}$$

Denoting the angles at $X_j$ and $X_k$ as $\gamma_j$, $\gamma_k$, the triangle area can be expressed as

$$A(\tau) = \frac{1}{2} \sin\gamma_j |e_{ij}|\, |e_{kj}| = \frac{1}{2} \sin\gamma_k |e_{ik}|\, |e_{jk}| \,.$$

Following the same notation, $\cos\gamma_j$ and $\cos\gamma_k$ can be computed as

$$\cos\gamma_j = \frac{e_{kj} \cdot e_{ij}}{|e_{kj}|\, |e_{ij}|} \,, \qquad \cos\gamma_k = \frac{e_{ik} \cdot e_{jk}}{|e_{ik}|\, |e_{jk}|} \,.$$

Plugging these expressions into (4.9) we obtain

$$\int_{\partial V_i \cap \tau} \nabla u(X) \cdot n(X)\, \mathrm{d}s = \frac{1}{2} \left(\cot\gamma_k (u_j - u_i) + \cot\gamma_j (u_k - u_i)\right) \,, \tag{4.10}$$

thus, each triangle contributes to difference of $u$ at $X_i$ w.r.t. two different vertices in its neighbourhood. Thus, labelling $\gamma_j$ and $\delta_j$ the opposite angles to edge $e_{ij}$ in the triangle tuple connected by it, we can write the approximation to LBO as sum of differences along $e_{ij}$

$$\Delta_{LB}u(X_i) := \frac{1}{2|N_\triangle(X_i)|}\sum_{j\in N(X_i)}\frac{1}{2}(\cot\gamma_j + \cot\delta_j)(u_j - u_i)\ , \tag{4.11}$$

and applied to the mesh coordinates

$$\Delta_{LB}X_i := L(X_i) = \frac{1}{2|N_\triangle(X_i)|}\sum_{j\in N(X_i)}\frac{1}{2}(\cot\gamma_j + \cot\delta_j)(X_j - X_i)\ . \tag{4.12}$$

The expression (4.12) can be rewritten also in terms of normalized weighted formula

$$L(X_i) = \frac{1}{2o}\sum_{j\in N(X_i)}\omega_{ij}\,(X_j - X_i)\ , \tag{4.13}$$

$$\omega_{ij} = \frac{1}{2}(\cot\gamma_j + \cot\delta_j)\ , \qquad o = |N_\triangle(X_i)|\ , \tag{4.14}$$

with the cotangent weights $\omega_{ij}$ and the normalization term $o$.

In matrix notation, the operator $\Delta_{LB}$ for a triangle mesh $M$, may be realized by $D^{-1}L$, where $L \in \mathbb{R}^{n\times n}$ is a symmetric, positive semi-definite, sparse matrix (weight matrix) defined as

$$L(i,j) := \begin{cases} \omega_{ij} = \frac{1}{2}(\cot\gamma_j + \cot\delta_j) & j \in N(X_i) \\ -\sum_{k\in N(X_i)}\omega_{ik} & i = j \\ 0 & otherwise \end{cases} \tag{4.15}$$

and $D$ is a lumped mass matrix defined as $D = diag\{|N_\triangle(X_1)|, \cdots, |N_\triangle(X_n)|\}$, which relates to the area/volume around the vertices of the discretized manifold.

Popular and simpler choice for the weights and the normalization term in (4.14) in formula (4.13) is the following

$$\omega_{ij} = \frac{1}{\|X_i - X_j\|}\ , \qquad o = \sum_{j\in N(X_i)}\|X_i - X_j\|\ , \tag{4.16}$$

which gives rise to the umbrella discretization of the Laplace-Beltrami operator.

### 3) Weighted p-Laplacian

The p-Laplacian of a function $u$ on a connected, oriented Riemannian manifold without boundary $\mathcal{M}$ is defined by

$$\Delta_p u := \operatorname{div}(|\nabla u|^{p-2}\nabla u) \; . \tag{4.17}$$

It is the Euler-Lagrange operator associated with the functional

$$\int_M |\nabla u|^p, \quad p \in (1, \infty) \; .$$

In the following proposition we report the relation between the weighted continuous p-Laplacian operator and its discretization through the weighted directional derivatives defined in (4.2).

**Proposition 1** *Given a set of points $V = \{X_i\}_{i=1}^n$ on a 2-manifold $\mathcal{M}$, the nonlinear operator $L_p^w$ of a twice differentiable function $u$ defined as:*

$$L_p^w u(X_i) = \frac{1}{2} \sum_{j \in N(X_i)} \gamma_p^w(X_i, X_j)(u(X_j) - u(X_i)), \tag{4.18}$$

$$\gamma_p^w(X_i, X_j) = w(X_i, X_j)^2(|\nabla u(X_j)|^{p-2} + |\nabla u(X_i)|^{p-2}), \tag{4.19}$$

*represents the discrete approximation of the weighted p-Laplacian operator :*

$$\Delta_p^w u := \nabla_w \cdot (|\nabla_w u|^{p-2}\nabla_w u) \tag{4.20}$$

*where $\nabla_w$ is the weighted gradient of $u$ on $\mathcal{M}$.*

*Proof.* Let $b(u) := |\nabla u|^{p-2}$, and $b_i$ be the evaluation of $b(u)$ at $X_i \in V$, by applying (4.2), the discretization of the weighted p-Laplacian operator (4.20) is given by

$$\begin{aligned} L_p^w u(X_i) &= \sum_j w_{ij}(b_j du(X_j, X_i) - b_i du(X_i, X_j)) \\ &= \sum_j w_{ij}^2(b_j\,(u(X_i) - u(X_j)) - b_i(u(X_j) - u(X_i))) \\ &= \sum_j w_{ij}^2(b_i + b_j)(u(X_i) - u(X_j)). \end{aligned} \tag{4.21}$$

Replacing (4.19) in (4.21) we get (4.18). □

The p-Laplacian is a non-linear operator, with the exception of the special case when $p = 2$, where it reduces to the regular Laplacian operator $\Delta_2 f = div(\nabla f)$, while for $p = 1$, we get $\Delta_1 f = \nabla \cdot (\frac{\nabla f}{|\nabla f|})$, which is the mean curvature operator.

#### 4) Mean Curvature and Normals

The mean curvature $H$ of a surface $\mathcal{M}$ is an extrinsic measure of curvature that comes from differential geometry and that locally describes the curvature of an embedded surface in $\mathbb{R}^3$.

The most straightforward estimation of the mean curvature measure $H$ and normal $\mathbf{N}$ at $X$ is obtained by exploiting the well-known relation

$$\Delta_{LB} X = -2H\mathbf{N}, \tag{4.22}$$

which relates the Laplace-Beltrami differential operator to the mean curvature vector $H\mathbf{N}$ when $\Delta_{LB}$ is applied to the coordinate function $X$ of a surface.

The mean curvature field $H(X_i)$ at a vertex $X_i$, that is used for the solution of the shape partitioning problem, for example in (5.11), is then obtained by constructing the local neighbourhood $N(X_i)$, and taking norm of the value evaluated by (4.13) for suitable choice of the weights (4.12),(4.16) based on the input type. The corresponding outward normal vector $\mathbf{N}_i$ is obtained by normalizing the value evaluated by (4.13).

Alternatively, the vertex normals can be also computed as spatial averages of normal vectors in a local one-ring triangle neighbourhood as follows

$$\mathbf{N}_i = \frac{1}{o} \sum_{j \in N_\triangle(X_i)} \frac{\mathbf{N}_{\tau_j}}{A(\tau_j)}, \qquad o = \sum_{j \in N_\triangle(X_i)} A(\tau_j), \tag{4.23}$$

where $\mathbf{N}_{\tau_j}$ is the normal vector to the triangle $\tau_j$.

## 4.2 Quad-based Finite Volumes Spatial Discretization

In the previous section we describe the discretization of a triangular mesh approximation to a 2-manifold $\mathcal{M}$ embedded in $\mathbb{R}^3$. The formulas to evaluate the normals and some differential operators defined on the surface assumed the mesh to be a graph structure what leads to very fast and effective discrete approximations of the reported operators.

In what follows, we introduce the Finite Volumes Method to the spatial discretization of a surface that is used to discretize the quad-based surface PDE evolution model in Chapter 7.

Since we are dealing with a quad mesh $M := (V, Q)$ where $Q$ is the set of quads, we can define the quad 1-ring neighbourhood as $N_{\square}(X_i) = \{Q_j \in Q;\ j = 1, \ldots, m_i\,,\ X_i \in Q_j\}$ where $m_i$ defines the number of quads surrounding (sharing) the vertex $X_i$.

The Finite Volumes Method assumes a general, continuous surface $\mathcal{M}$ to be approximated by the union of so-called *control volumes* $V_i$, $i = 1, \ldots, n$ built around each vertex $X_i$ and the function values at $X_i$ are assumed to be constant over $V_i$. In case of an evolving surface, the control volume $V_i$ naturally changes with the evolving vertex.

Let us introduce the local vertex and quad indexing in barycentric control volume $V_i$ around vertex $X_i$ as illustrated in Figure 4.2 for $m_i = 5$.

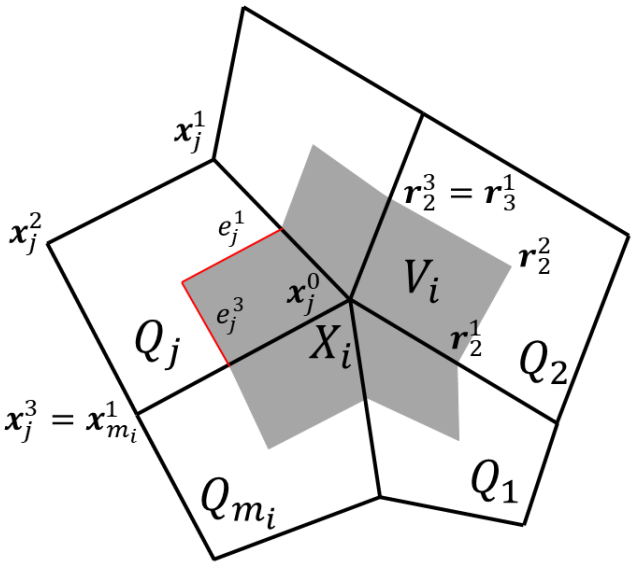



Fig. 4.2 Finite volume $V_i$ (shaded grey area) with local indices of its representative points for the case $m_i = 5$.

The local vertex indices in a quad $Q_j$ are denoted as $\mathbf{x}_j^k$, $k \in \{0, \ldots, 3\}$ satisfying $\mathbf{x}_j^0 = X_i$ and $j$ index refers to the neighbouring quad.

Therefore, the barycentric finite volume $V_i$ around vertex $X_i$ consists of $m_i$ quarter-quads of the neighbouring quads $Q_j$ as illustrated in Fig. 4.2. The boundary of $V_i$ in one

quarter-quad is described by vertices $\mathbf{r}_j^k$ as follows

$$\begin{aligned}
\mathbf{r}_j^0 &= \mathbf{x}_j^0 \\
\mathbf{r}_j^1 &= \left(\mathbf{x}_j^0 + \mathbf{x}_j^1\right)/2 \\
\mathbf{r}_j^2 &= \left(\mathbf{x}_j^0 + \mathbf{x}_j^1 + \mathbf{x}_j^2 + \mathbf{x}_j^3\right)/4 \\
\mathbf{r}_j^3 &= \left(\mathbf{x}_j^0 + \mathbf{x}_j^3\right)/2 \,.
\end{aligned}$$

This method aims to approximate the solution of an evolution PDE model in terms of integrals over the control volumes $V_i$, unlike the Finite Differences Method. In particular, we do not approximate directly the before-mentioned operators, but rather their integrals over the finite volume, which is e.g. in case of the Laplace-Beltrami operator further transformed, to sum of integrals over the boundary of $V_i$. Therefore, we excuse ourselves to postpone the approximation formulas of the appropriate operators integrated over $V_i$ in Section 7.4.2 together with the rest of the numerical scheme for the model.

# Chapter 5

# Shape Analysis

In this chapter, we introduce the shape-describing single- and multi-channel functions that will drive the variational partitioning models described in details in Chapter 6.

In Section 5.1 we propose an alternative approach to the computation of the Shape Diameter Function [108] in order to measure the thickness of a solid closed object.

Section 5.2 describes a new proposal for an affinity matrix whose eigen-decomposition is used on the human-based partitioning described in Section 6.1. Finally in Section 5.3 we propose the $L_p$ Compressed Modes basis as the alternative to the eigen-decomposition of the affinity matrix.

## 5.1 Meshless approach to SDF computation by particles flow

In the original proposal [108] the basic operation in the SDF computation is a ray-mesh intersection. Given a point on a mesh, several rays inside a cone centred around the inward-normal direction are sent towards the opposite side of the mesh. The intersected points are filtered to remove false intersections and finally their ray-lengths are summed as weighted contributions to the final SDF value of that point. An octree built around the mesh is used as a spatial structure for the intersection search. In [64] the authors introduced an optimization to the original algorithm proposed in [108]. The SDF values, computed only for a small number of randomly distributed faces, were propagated over the mesh by using Poisson-based interpolation. The robustness of the original proposal in [108] is further improved in [102] by introducing an adaptive ray casting strategy.

In this section we propose a new approach to approximate SDF values for a point cloud representing the boundary of a closed 2-manifold. Unlike the original proposal,

our strategy is mesh-less and relies on a particle flow of the point cloud representing the boundary of a closed 2-manifold and a simple collision test. We can assume that the cloud of points is noisy-free; in case of noisy data a fairing procedure can be preliminarily applied [85]. The normals to these particles set are then approximated, and used to drive the particle evolution towards the inward-normal direction. At each time step a simple collision test is evaluated to eventually stop the particle flow, the distance covered by the stopped particle represents its SDF value.

Having an algorithm that computes the shape diameter analysis on a set of points without requiring global connectivity information is of fundamental importance since it makes it independent from the type of representation adopted by the geometric model, i.e. parametric form, B-rep, mesh or even solid model. The only requirement that the model must satisfy is that its boundary is evaluable at a given point set to provide the input point cloud.

Figure 5.1 shows the steps of our proposal: from left to right: the initial `ant` data set, the associated voxel grid data structure, the resulting SDF which can be used to define a thickness-oriented partitioning.

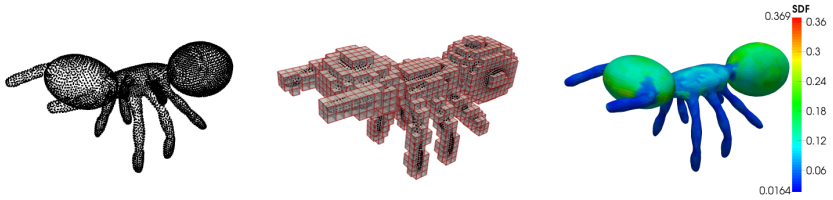



Fig. 5.1 From left to right: data set, initial voxel grid, resulting SDF.

Let us assume that the motion of a particle $X_i$ in a particle cloud $V$ is driven by the following flow

$$\frac{\partial x_i}{\partial t} = -\mathbf{N}_i\,, \quad x(0) = X_i\,, \tag{5.1}$$

where the unknown is the position in space of the particle, $x_i(t)$, at the time $t$, $\mathbf{N}_i$ is the outward unit normal at $X_i$ that remains constant during the whole evolution, and $t \in [0, T_{max})$ is the particle lifetime. From a mathematical point of view the unknown is a single variable vector-valued function in space: $x_i : \mathbb{R} \to \mathbb{R}^3$.

Evolution prescribed in (5.1) causes the motion of the particle in the inward-normal direction, towards the inner part of the object.

The flow of a particle $x_i$ is stopped as soon as it first collides with another particle $x_j$ which was moving towards the opposite direction. Therefore the lifetime of $x_i$ and of its antipodal surface point $x_j$ is the same $T_{max}$. For example, points on a sphere will evolve towards the center of it and will stop their evolution at the same time $T_{max}$.

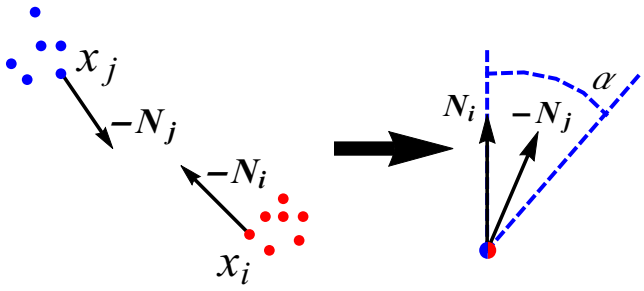



Fig. 5.2 Collision test (5.2) for two particles $x_i$ and $x_j$.

The lifetime $T_{max}$ is established by the following collision test, illustrated in Fig. 5.2.

*Two evolving particles $x_i$ and $x_j$ collide if their trajectories intersect and they have the same, but opposite, velocity direction under a certain tolerance $\alpha$, that is:*

$$\measuredangle\left(\mathbf{N}_i, -\mathbf{N}_j\right) \leq \alpha\,. \tag{5.2}$$

The angle value $\alpha$ allows for a cone of admissible directions that, relaxing the ideal case where $\mathbf{N}_i = -\mathbf{N}_j$, better fits the spatial discretization of the surface.

When the particle $x_i$ (and $x_j$) stops its flow, at $T_{max}$, a scalar function value $f_{SDF}(\cdot)$, $f_{SDF} : V \to \mathbb{R}^+$, is computed as the distance between $X_i$ and $X_j$, with $X_i, X_j \in V$, and associated both to the particle $X_i$ and $X_j$.

The particle flow (5.1) is simply discretized in time using forward Euler explicit scheme

$$x_i^{(t)} = x_i^{(t-1)} - dt\mathbf{N}_i\,. \tag{5.3}$$

The algorithm for shape diameter analysis of a set of points $V$ is described in Algorithm 1: *SDF_Flow*.

The normals $\{\mathbf{N}_i\}_{i=1}^n$ to the particle set $V$ are approximated according to the type of data set, using (4.13) with (4.16) for point clouds, and (4.13) with (4.14), or (4.23) for meshes.

In order to make the SDF computation object independent, a preliminary minimal uniform scale is applied to $V$ to fit the vertices inside the Euclidean space $[0,1]^3$, and a uniform voxel grid of $[0,1]^3$ is built with voxel size $dh$ computed according to the point cloud density. Assuming a unitary velocity, the time step $dt$ in (5.3) is then chosen to be approximately equal to the voxel size $dh$.

From the voxel grid we extract the *envelope* $\mathcal{E}$ of the set of voxels containing the object. $\mathcal{E}$ consists of non-empty voxels that include points in $V$, together with empty

**Algorithm 1 SDF Flow**

**Input**: data $V$
**Output**: $f_{SDF}(V)$, SDF values of $V$
**Parameter**: cone angle $\alpha$

Preliminary Process:
· compute normals $\{\mathbf{N}_i\}_{i=1}^n$
· compute uniform voxel grid of $[0,1]^3$
· compute the envelope $\mathcal{E}$

· set $V^0 = V$
· set $t = 0$
**while** $|V^t| \neq \emptyset$
· $V^* \leftarrow$ move $V^t$ according to (5.3)
· assign $V^*$ to the voxel grid
· $t = t + 1$
· $V^t =$ `Collision_Detect`$(\mathcal{E}, V^*, \alpha)$
**end**

voxels which are inside the object. Since we suppose to deal with a set of points $V$ on a water-tight surface, a water-tight envelope is obtained by choosing $dh \geq h_V$, where $h_V$ is the fill distance defined in (4.1); for the reported experiments we fixed $dh = 2h_V$.

The core of the algorithm *SDF_Flow* is simply based on a cycle which evolves, for each time step $t$, the set of the alive particles in $V$, which is denoted by $V^t$. The dimension of this set decreases in time and it is updated by the function `Collision_Detect()` detailed below.

**Algorithm 2 Collision Detection**

**function**: `Collision_Detect`$(\mathcal{E}, V^*, \alpha)$
· update envelope $\mathcal{E}$ for $V^*$
**for** each non-empty voxel $v_i \in \mathcal{E}$
**for** every $x_j \in v_i$
· $\mathcal{S} =$ `Valid_Neigh`$(x_j, v_i)$
**if** (`SDF_assignment`$(x_j, \mathcal{S}, \alpha)$)
· update $V^*$
**end**
**end**

In the function `Collision_Detect()`, described in Algorithm 2, first the envelope $\mathcal{E}$ is updated by labelling as non-empty the voxels that contain $V^*$. Then, for each non-empty voxel $v_i \in \mathcal{E}$, we check for potential collisions between particles located in $\mathcal{E}$. In particular, the `SDF_assignment()` function checks if there are colliding particles for $x_j$ first inside the voxel $v_i$ itself and then in a small voxel neighbourhood $\mathcal{S}$.

The latter check is required to enforce the robustness of the `SDF_Flow` in Algorithm 1 since we are dealing with time and space discretization.

In the time-continuous case, two particles moving in the perfect opposite directions, but in the same trajectory, are guaranteed to collide in a certain voxel. However, due to the time discretization assumed, our ideal evolving particles could skip across the voxel where they were supposed to collide.

Moreover, due to the spatial discretization of the surface boundary of an object, given a particle $x_i$ evolving in its normal direction is not guaranteed the existence of its antipodal surface particle $x_j$ on the perfect opposite side.

Therefore, in order to avoid both the swapping problem and the non-uniformity of the data set $V$ the investigation for potential collisions is enlarged farther than the 1-ring voxel neighbourhood, towards a limited set of neighbourhood voxels $\mathcal{S}$ detected by the function `Valid_Neigh()`.

The `Valid_Neigh()` function thus returns the set $\mathcal{S} \subset \mathcal{E}$ of neighbourhood voxels, which contain potentially colliding particles, that belong to the cone centred at $\mathbf{N}_i$ with angle $\beta$ inside the 2-ring voxel neighbourhood.

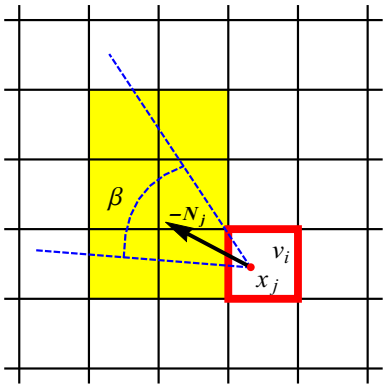



Fig. 5.3 Voxel-grid scheme for a particle $x_j$ which moves towards direction $-\mathbf{N}_j$ and whose collision-detection check is enlarged to the the set $\mathcal{S}$ (yellow shaded area and voxel $v_i$) which consists of the 2-ring voxels belonging to the cone width $\beta$.

In the synthetic two-dimensional example, depicted in Fig. 5.3, the set of admissible voxels $\mathcal{S}$ is yellow coloured and it consists of the voxel $v_i$ and the voxels inside the 2-voxel

rings of $v_i$ which lie in the $\beta$ cone centred around the inward-normal direction of $x_j$. From our numerical experience, the number of parameters required can be limited by setting $\beta = \alpha$ without any significant impact on the performance. The choice of the parameter $\alpha$ will be discussed in Section 5.1.1.

Finally, the function `SDF_assignment()` looks for a possible matching between the input particle $x_j$ and a particle $x_k$ in the set of voxels $\mathcal{S} \subset \mathcal{E}$.

From those particles which satisfies the collision test (5.2), we select the particle $x_k$ that has minimal normals angle deviation with respect to $N_j$, that is

$$\alpha_k = \arg\min_{x_* \in \mathcal{S}} \measuredangle (\mathbf{N}_j, -\mathbf{N}_*) \ . \tag{5.4}$$

Then the $f_{SDF}$ value associated to $X_j$ is computed as the distance between the two particles $X_j$ and $X_k$ in $V$

$$f_{SDF}(X_j) = \|X_j - X_k\|_2 \, . \tag{5.5}$$

In case $x_k$ is alive, then also its associated initial particle $X_k$ has to be set with the same SDF value, that is $f_{SDF}(X_k) = f_{SDF}(X_j)$, and both the particles $x_j, x_k$ are removed from $V^t$ since they have reached their lifespan.

When the user-choice of $\alpha$ turns out to be not adequate to the data set $V$, an evolving particle $x_j$ can outflow the envelope $\mathcal{E}$ at any time step $t$, i.e. it moves through the object without collision. In this case the particle is removed from $V^t$ and labelled as outliers. Finally, an SDF value $f_{SDF}(X_j)$ is assigned to each outliers particle $X_j$, computed as the SDF average of its non-outliers neighbourhood particles $X_k$, $k \in N(X_j)$.

In the proposition below we give sufficient conditions for the theoretical convergence of Algorithm 1.

**Proposition 2** *Let $V$ be a data set of points in $\mathbb{R}^3$ representing the boundary of a 2D manifold $\mathcal{M}$, and $h_V$ represents its fill distance. Then, setting the voxel size $dh > h_V$, Algorithm 1 assigns an estimated SDF value $f_{SDF}(X_j)$ to each particle $X_j \in V$ with analytical shape diameter value $f^*$ with respect to $\mathcal{M}$, satisfying one of two assumptions:*

- *If* $\measuredangle (\mathbf{N}_i, -\mathbf{N}_j) = 0$, *then*

$$f_{SDF}(X_j) = f_{SDF}(X_i) = f^*, \quad X_i \in V.$$

- *If* $\measuredangle(\mathbf{N}_i, -\mathbf{N}_j) < \alpha$, *then*

$$|f_{SDF}(X_j) - f^*| \leq f_{SDF} - \left[ f_{SDF}\cos(\alpha) \pm \sqrt{h_V^2 - f_{SDF}\sin(\alpha)} \right] . \tag{5.6}$$

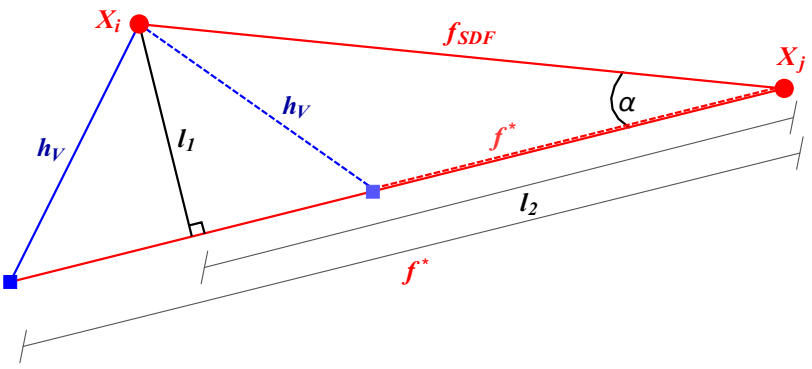



Fig. 5.4 Approximation error when the antipodal point for $X_j$ is not a vertex of $V$: rounded red points represent particles in $V$, and squared blue points represent possible antipodal points on the 2-manifold, where $l_1 = f_{SDF}\sin(\alpha)$ and $l_2 = f_{SDF}\cos(\alpha)$.

The proof follows straightforward from Fig. 5.4. According to Proposition 2, Algorithm 1 converges to the exact SDF value if there exist two particles $X_j, X_i$ in the perfect opposite direction, otherwise it will assign at $X_j, X_i$ an estimated SDF value, where the approximation error is limited by (5.6).

In our experience, summarized in the numerical section 5.1.1, we can assume that a suitable configuration of the $dh$ voxel size and $\alpha$ parameter, always leads to convergence. If $V$ was entirely contained in a single voxel, a collision between particle would be always captured by enlarging $\alpha$ since we assumed that the 2-manifold is closed. However, the larger the voxel dimension, the worse is the computational accuracy of Algorithm 1. A good compromise is obtained by automatically setting $dh$ related to the fill distance $h_V$ of $V$, and let the parameter $\alpha$ be in a suitable interval introduced in Section 5.1.1.

### 5.1.1 SDF: Examples

In this section we report the evaluations of the proposed method. We first analyse in Example 1 the performance of Algorithm 1 for computing SDF in terms of accuracy and efficiency; in Example 2 we compare the proposed method with the ray-trace based SDF algorithm by Shapira et al. [108].

We tested the proposed algorithms on a collection of point clouds and meshes downloaded from the data repository website `http://segeval.cs.princeton.edu`, [31].

Table 5.1 Data sets and timing results for SDF computation by Algorithm 1.

| Data set | Vertices $\|V\|$ | Outliers (%) | Time (s) | Data set | Vertices $\|V\|$ | Outliers (%) | Time (s) |
|---|---|---|---|---|---|---|---|
| ant | 7038 | 0.34 | 1.054 | mech_2 | 10400 | 0.49 | 0.965 |
| armadillo | 25193 | 0.42 | 1.924 | mech_3 | 1512 | 3.31 | 0.096 |
| bird_1 | 6475 | 6.52 | 0.258 | mech_4 | 14872 | 0.85 | 1.223 |
| bird_2 | 8946 | 6.23 | 0.461 | mech_5 | 14956 | 0.87 | 1.536 |
| blocks | 6146 | 7.31 | 0.587 | octopus_1 | 7251 | 0.87 | 1.640 |
| camel | 9757 | 1.99 | 0.578 | octopus_2 | 1010 | 0.40 | 0.184 |
| chair | 10121 | 0.10 | 1.042 | octopus_3 | 243 | 30.45 | 0.041 |
| cup | 15037 | 0.84 | 1.491 | octopus_4 | 1343 | 16.08 | 0.166 |
| dolphin | 7573 | 0.96 | 0.213 | plane | 7470 | 2.60 | 0.591 |
| fish | 6656 | 0.53 | 0.990 | pliers | 3906 | 1.02 | 0.482 |
| giraffe | 9239 | 1.34 | 0.357 | sphere | 16386 | 0.00 | 0.864 |
| glasses | 7407 | 0.65 | 0.349 | table | 14587 | 0.61 | 0.694 |
| hand | 6607 | 0.73 | 0.451 | teddy | 9548 | 2.79 | 1.267 |
| horse | 7268 | 1.37 | 0.419 | torus | 7200 | 1.47 | 1.626 |
| human_1 | 9508 | 0.01 | 0.346 | vase_1 | 14859 | 1.10 | 0.764 |
| human_2 | 15385 | 0.00 | 0.331 | vase_2 | 14476 | 0.66 | 0.761 |
| mech_1 | 8759 | 1.96 | 0.567 | wolf | 4712 | 1.25 | 0.256 |

The chosen dataset represents geometric models with different characteristics in terms of details, "thickness", and level of refinement, and presents a medium dense vertex distribution. Data sets information is given in the second and sixth column of Table 5.1.

In the reported figures, the SDF values are visualized using false colours superimposed onto the object. In particular, since the $f_{SDF}$ is a positive scalar function defined on the object domain, we made use of the HSV colour model such that the hue value $f_{SDF}(X_i)$ at the vertex $X_i$ is assigned linearly to the interval $(min(f_{SDF}(V)), max(f_{SDF}(V)))$. In Fig. 5.9 where we compare our SDF results with the ones obtained using the original algorithm by Shapira et al. [108], we depict screenshots of the application downloaded from the author's website. This application uses rendering and lighting procedures slightly different from ours specified in Section 6.4.1. Nevertheless the results are easily comparable.

To evaluate the *accuracy* of our algorithm, we introduce a local measure of accuracy with respect to the so-called "ground-truth", given by the relative error vector

$$err(x_i) = \frac{f^{SDF}(x_i) - f_*^{SDF}(x_i)}{f_*^{SDF}(x_i)}, \quad i = 1, \cdots, N \tag{5.7}$$

where $f^{SDF}$ and $f_*^{SDF}$ denote the approximated and ground-truth values, respectively, evaluated at $x_i$.

Then the global accuracy is measured by the following norms defined for the vector $err = [err^{(1)}, \ldots, err^{(N)}]^T \in \mathbb{R}^N$:

- the weighted Euclidean $L_2$-norm

$$L_2(err) = \left( \frac{1}{N} \sum_{i=1}^{N} \left( err^{(i)} \right)^2 \right)^{\frac{1}{2}}, \tag{5.8}$$

- the $L_\infty$-norm

$$L_\infty(err) = \max_i \left| err^{(i)} \right|. \tag{5.9}$$

**Example 1 – Efficiency and accuracy of the SDF flow**

Algorithm 1 for the SDF computation operates with a user-chosen input parameter $\alpha$ representing the width of the cone at the particle direction, which affects the output results in terms of accuracy and amount of outliers. When the $\alpha$ value is large, the number of outliers is approaching to 0, but the collision test (5.2) can result positive also

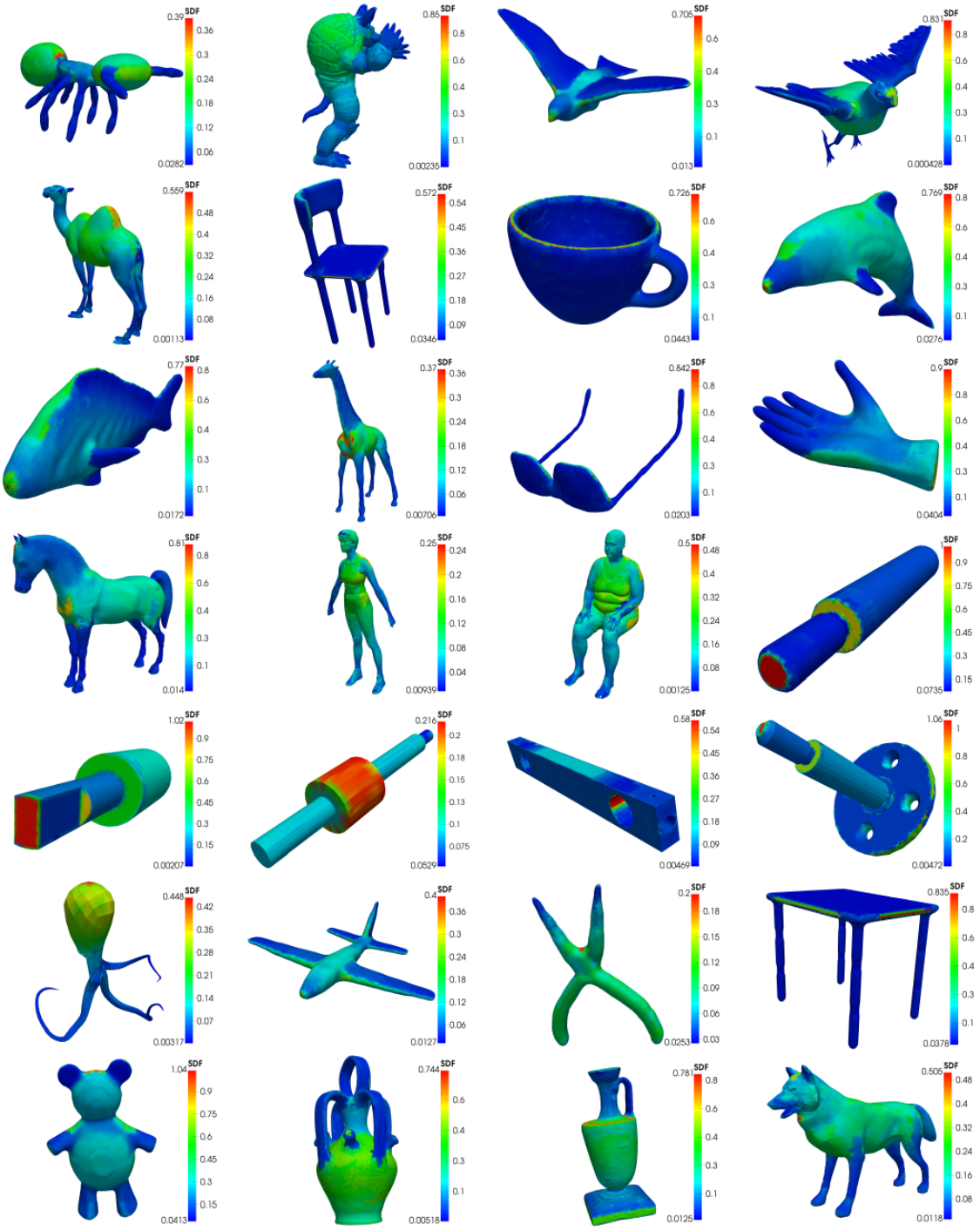


Fig. 5.5 Example 1: SDF results obtained by applying Algorithm 1 to several data sets. The results are sorted by appearance in Table 5.1.

for couple of particles that naturally should not collide each other. Decreasing $\alpha$, both the precision and the number of outliers increase. The substantial experimentation done allowed us to determine an interval for optimal angle values $\alpha \in [\frac{\pi}{6}, \frac{2\pi}{5}]$.

For all the examples reported the number of neighbourhood voxel rings is 2.

The *efficiency* of our algorithm for SDF approximation has been evaluated by applying Algorithm 1 to several data sets. In Table 5.1 we report the performance results for the data sets shown in Fig.5.6, Fig. 5.7, Fig. 5.9 and Fig. 5.5. The third and seventh column in Table 5.1 reports the percentage of outlier particles for each data set, that is particles which lifetimes have been ended prematurely since their trajectories went out from the object. In the fourth and eighth column the timing results are shown in seconds for an optimal configuration of the parameter $\alpha$. We observe that the computational time is affected by the shape, density and dimension of the input particle cloud. For dense, but relatively thin point clouds, for example `cup`, the computational times are comparable with rather sparse and relatively thick point clouds, for example `teddy`. Algorithm 1 turns out to be particularly fast when the input point cloud has a lot of thin features (in animals category – the legs, tentacles, wings); in this case, a large part of the moving particles from $V^t$ reaches its lifespan in a few time steps, thus saving computational time in the `Collision_Detect` function.

The *robustness* of Algorithm 1 with respect to the density of the data set has been evaluated by applying Algorithm 1 to decreasing point clouds of the same object. We report the results for the input point clouds `octopus_1`, `octopus_2`, and `octopus_3` illustrated in Fig. 5.6 (top row) which presents a decreasing density of particles from left to right. This example allows us to demonstrate the capacity of our algorithm to obtain accurate results even in case of poorly dense input data set $V$, preserving the shape diameter features of the represented object. SDF qualitative results are illustrated in Fig.5.6 (bottom row). This robustness is attributed to the right tuning of the parameters in Algorithm 1. In particular, the decreasing density causes enlargement of the separation and the fill distances (4.1) that naturally leads to enlargement of the voxel size $h$, the threshold $\alpha$ in the collision test (5.2) and the $\beta$, what allows more admissible neighbourhood voxels. We remark that since our algorithm is point-wise based and the resulting SDF value estimation is computed as a distance between two points, the accuracy error naturally increases for decreasing mesh density. Setting as ground truth the highest density point cloud represented on the left of Fig.5.6, we can evaluate the loss of accuracy in terms of the metrics defined in (5.8) and (5.9) for the two down-sampled point clouds decimated up to 14% and 3% and illustrated in Fig.5.6 centre and right, respectively. The global accuracy results for $f^{SDF}$ are reported in Table 5.2.

Table 5.2 Example 1: global accuracy measures with respect to the `octopus_1` as ground-truth.

| Data set | $L_2$ | $L_\infty$ |
|---|---|---|
| `octopus_2` | 0.32964 | 0.22472 |
| `octopus_3` | 0.76988 | 0.19654 |

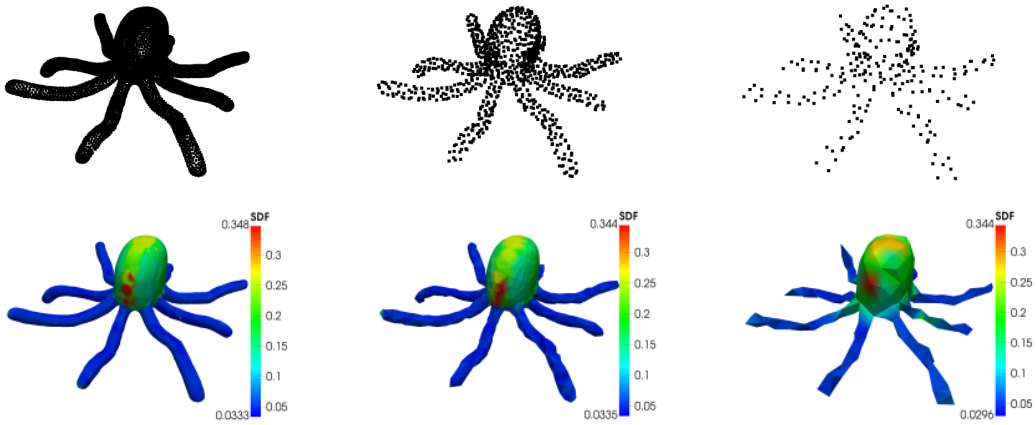



Fig. 5.6 Example 1: three different initial point clouds (top) with decreasing density, resulting in comparable results (bottom). From left to right: `octopus_1`, `octopus_2`, `octopus_3`.

**Example 2 – Comparisons with the Ray Tracing approach.**

To provide a comparison to state-of-the-art SDF algorithms, we evaluate Algorithm 1 with the SDF algorithm based on ray tracing, introduced in [108], and implemented in the free software downloadable from Lior Shapira's website `http://www.liors.net/shape-diameter-function`. The input parameters required by the original proposal are mainly the number of cones, the number of rays per cone and the cone separation. We denote by Ray Tracing Basic (RT-Basic) the settings: 1 cone with 15 rays and separation 1, and by Ray Tracing Best (RT-Best) the tuned parameter setting which provided the best result. In order to maintain the correspondence among the methods the colour function option has been chosen as the mapping to the interval $[\min f_{SDF}, \max f_{SDF}]$.

In Table 5.3 we report the comparison in terms of computational time in seconds for a few meshes. The performance of Algorithm 1 is always better with respect to the basic setting (RT-Basic) and outperforms the best setting (RT-Best) of the ray tracing algorithm.

To allow for a visual inspection of the performance, we report the SDF computed maps in Figure 5.9. The visual results produced by the faster RT-Basic, Figure 5.9(second row), present several failures. For the mesh `cup`, false red spots appear on the outer side of the object (large $f_{SDF}$) although there are no correspondence in the inner part of it. In the `mech_2` SDF result, the RT approaches were not able to detect the difference in volume between the top and the middle part which is yellow-coloured in our result, see Figure 5.9 (first row). The `pliers` result also presents false red spots on the handlers even if the thickness is uniform. For the `plane` result, where the wings are joined to the rest of the body, the diameter analysis should produce values definitely smaller than those on the tip of the nose. However, all but two inaccuracies were repressed by tuning the parameters in the ray-tracing approach, Figure 5.9(third row), thus obtaining visual results similar to those produced by Algorithm 1 and shown in Figure 5.9(first row). For the `mech_2` mesh, the appropriate result has not been obtained, even when the RT-Best was applied, while the shape analysis failed to produce the correct SDF map for the wings and the tip in the `plane` mesh.

Compared with the timing results shown in Table 1 in [108] and the performance obtained by the RT approaches in Table 5.3, we noticed a significant improvement of performance, according to the data set dimensions. Certainly we can optimize the performance of both the approaches by exploiting GPU computational power or applying greedy strategies like for example the ASDF approach proposed in [64] or the MeshLab implementation [33] of the ray tracing approach based on [108]. We would like to stress again that Algorithm 1 works on cloud of points while the RT approaches require input meshes.

We finally investigate the global accuracy of the two methods when applied to three test objects, namely `blocks`, `sphere` and `torus`, depicted in Figure 5.7, which are characterized by a known shape diameter. The object `blocks` is composed of three blocks with the same rectangular base $0.2 \times 0.4$, and heights 0.2, 0.6 and 1, respectively. The second object is a `sphere` with diameter 1, while the `torus` object has larger radius $R = 0.4$ and smaller radius $r = 0.1$.

The histograms in Figure 5.8 illustrate the exact distributions of thickness for the three objects in the left column, and the errors between the exact and the computed SDF in the remaining two columns for our algorithm and the RT-based approach, respectively in the middle and right columns. The error histograms have been computed as absolute difference of the histogram bins between the ground truth and the estimated values. A quantitative evaluation of the accuracy errors produced by the two algorithms are summarized in Table 5.4. We can observe that Algorithm 1 always outperformed the RT-

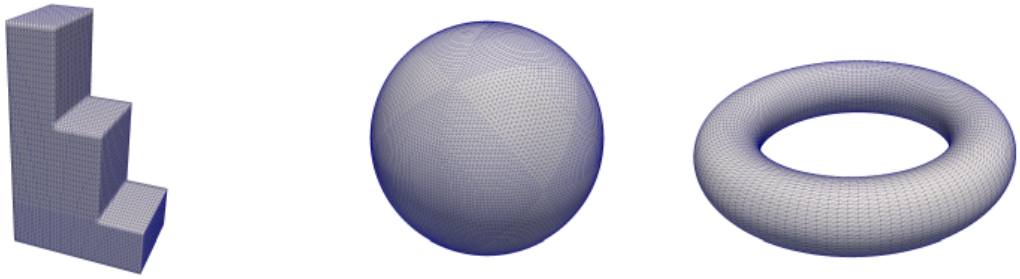


Fig. 5.7 Example 2: test objects used in accuracy comparison.

Table 5.3 Example 2: time comparison in seconds between Algorithm 1 and the ray tracing algorithm in [108].

| Data set | Alg. 1 (s) | RT-Basic (s) | RT-Best (s) |
|---|---|---|---|
| `cup` | 1.491 | 3.373 | 12.579 |
| `mech_2` | 0.965 | 10.807 | 10.807 |
| `plane` | 0.591 | 2.005 | 14.977 |
| `pliers` | 0.482 | 2.958 | 8.564 |
| `octopus_4` | 0.166 | 0.482 | 1.673 |
| `blocks` | 0.587 | 2.938 | 2.938 |
| `sphere` | 0.864 | 12.328 | 12.328 |
| `torus` | 1.626 | 3.130 | 3.130 |

based algorithm. This can be due to the fact that the RT-based algorithm systematically averages multiple lengths inside a cone, and this overestimates the diameter. The timing results for the three test objects are reported in Table 5.3.

Table 5.4 Example 2: global accuracy comparison.

| | Alg. 1 | | RT-Best | |
|---|---|---|---|---|
| Data set | $L_2$ | $L_\infty$ | $L_2$ | $L_\infty$ |
| `blocks` | $2.414\times10^{-2}$ | 0.399 | $8.309\times10^{-2}$ | 0.526 |
| `sphere` | $4.99\times10^{-7}$ | $2\times10^{-6}$ | $1.917\times10^{-4}$ | $10^{-3}$ |
| `torus` | $1.057\times10^{-3}$ | $1.093\times10^{-3}$ | $3.943\times10^{-2}$ | 0.1 |

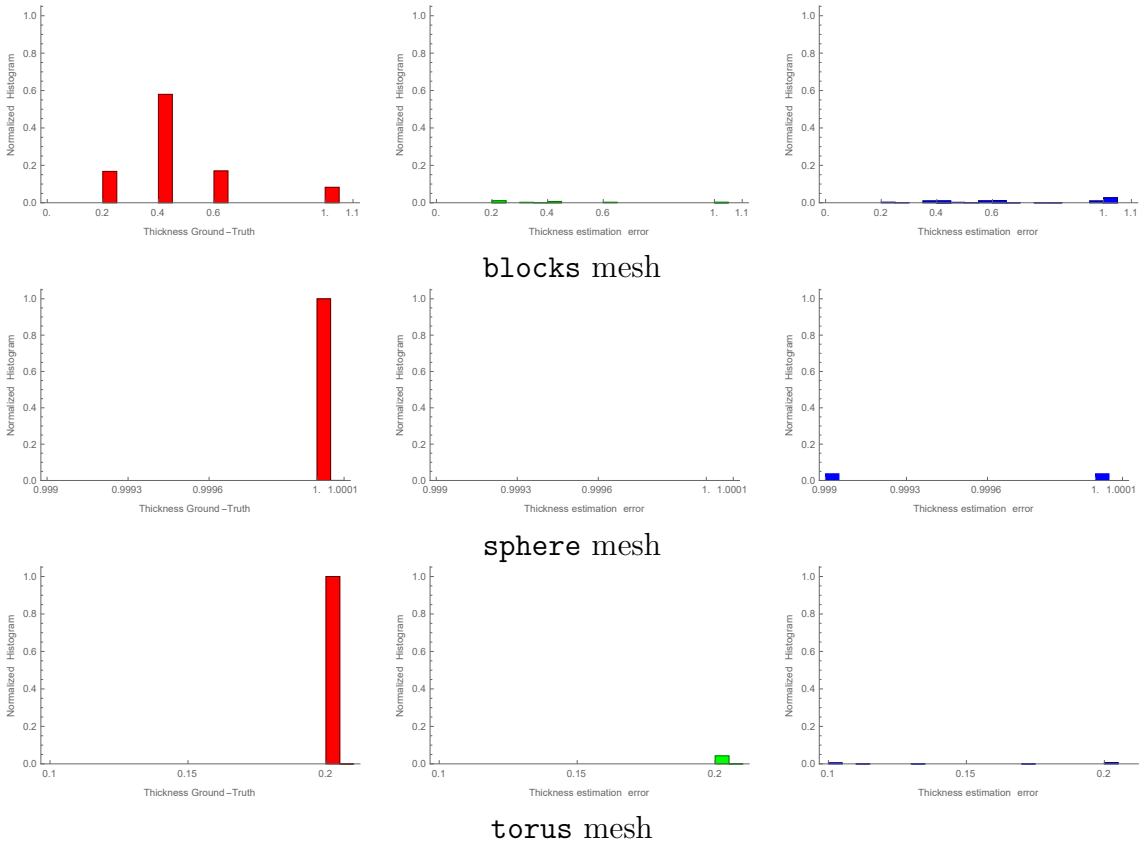



Fig. 5.8 Example 2: quantitative evaluation: exact shape diameter (left), differences between the exact and the computed $f_{SDF}(X)$ values by Algorithm 1 (centre), and between the exact and the computed $f_{SDF}(X)$ values by RT-algorithm in [108] (right).

## 5.2 Affinity Matrix

In this section, we describe the graph matrix which plays the role of the affinity matrix that will be used for human perception segmentation [79].

In general, decomposition methods based on spectral analysis emphasize mostly the concavity attribute, being able to partition even shallow concavities. The spectral analysis method uses the eigenvalues of properly defined matrices based on the connectivity of the graph in order to partition a mesh. Liu and Zhang [78] use spectral analysis on the dual graph of the mesh. They define an affinity matrix using both geodesic distances and angular distances as proposed by the fuzzy clustering method in [63]. This type of matrix has been used successfully for clustering since it groups elements having high affinity, see for example [130].

The affinity matrix associated to mesh $M$ should encode mesh structural information which reflects how vertices are grouped in accordance with human perception.

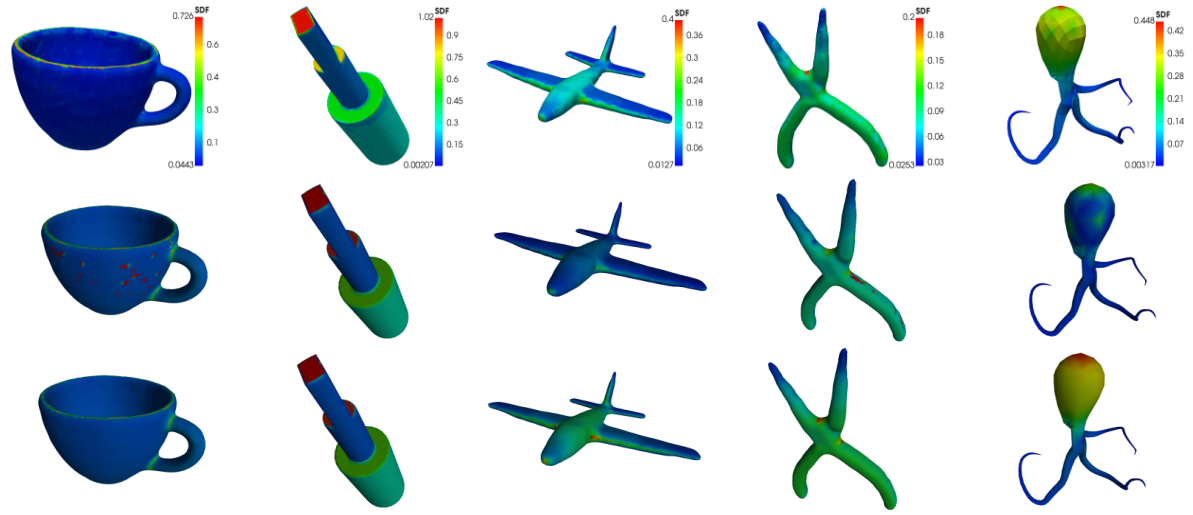


Fig. 5.9 Example 2: qualitative comparison between Algorithm 1 and the original ray tracing approach [108]: (Top) SDF results of Algorithm 1, (Middle) SDF results obtained with RT-Basic settings, (Bottom) SDF results with RT-Best settings. The associated computational timings are reported in Table 5.3.

Taking into account the curvature as shape information, we want to determine a perceptual partition of $M$ such that the edges between different parts have very low weights (vertices in different clusters are dissimilar from each other), and the edges belonging to the same part have high weights (vertices within the same cluster are similar to each other). At this aim we define the affinity matrix $\mathcal{L} \in \mathbb{R}^{n \times n}$:

$$\mathcal{L}_{i,j} = \begin{cases} -w_{ij}, & i \neq j \ \text{ and } \ e_{ij} \in E \\ \sum_{j \in N(X_i)} w_{ij}, & i \ = \ j \\ 0 & \text{otherwise} \end{cases}, \tag{5.10}$$

with the following similarity non-negative weights:

$$w_{ij} \ := \ \frac{|N_{\triangle}(X_j)|}{\#N(X_i)} \, e^{-\|H(X_i) - H(X_j)\|_2^2/(2\sigma^2)} \tag{5.11}$$

where the parameter $\sigma \in (0, 1]$ in (5.11) controls the width of the local neighbourhoods, $H(X_i)$ represents the mean curvature at vertex $X_i$ and the normalization terms represent the area of triangle neighbourhood divided by its cardinality.

The spectral decomposition of $\mathcal{L}$, defined in the following, provides a set of $(n-1)$ non trivial, smooth, shape intrinsic isometric-invariant maps. We refer the reader to [118] for details.

**Proposition 3** *The matrix $\mathcal{L} \in \mathbb{R}^{n\times n}$ defined in (5.10)-(5.11) associated to a connected mesh $M$ of $n$ vertices, satisfies the following properties:*

*1) $\mathcal{L}$ is symmetric and positive semi-definite;*

*2) $\mathcal{L} = U\,\Lambda\,U^T$, $\Lambda = diag(\lambda_i)$, $0 = \lambda_0 < \lambda_1 < \ldots < \lambda_n$ ;*

*3) $\lambda_i\,, \forall i$ are real eigenvalues, $U^T\,U = I_n$ with $I_n$ the identity matrix of order $n$, $U = \{v_0, v_1, \ldots, v_n\}$ form an orthogonal basis of $\mathbb{R}^n$ ;*

*4) If $f = \sum_{i=1}^{n} \langle f, v_i\rangle\, v_i$, the $k$-term approximation of $f$ is given by*

$$f_k = \sum_{i=1}^{k} \langle f, v_i\rangle\, v_i\ .$$

The first $k$ eigenvectors associating to the smallest nonzero eigenvalues, correspond to smooth and slowly varying functions, while the last one show more rapid oscillations. Property 4) defines the truncated spectral approximation of the $\mathcal{L}$ matrix, that considers the contribution of the first $k$ eigenpairs related to the smallest eigenvalues which hold for identifying the main shape features at different scale forming a signature for shape characterization.

In case of eigen-decomposition-based mesh partitioning the shape describing function used to detect salient shape parts is a vector function $f$ defined as the truncated spectral coordinates of a vertex $X_i$ and denoted by

$$f(X_i) = (v_1(X_i),\, v_2(X_i),\, \ldots,\, v_d(X_i))\,, \quad d \leq k\,, \tag{5.12}$$

where each $v_j$ obtained by the spectral decomposition of $\mathcal{L}$ in (5.10)-(5.11) is normalized in the range $[-1, 1]$.

The number $k$, which represents the number of computed eigenpairs, should be chosen according to the shape resolution. In Fig. 5.10 the first $k = 6$ eigenfunctions of the affinity matrix (5.10) corresponding to the first six non-zero eigenvalues are illustrated for the `horse_2`, the `glasses` and the `chair` meshes, visualized in false colours in the range [blue,red]. In the case of `glasses` mesh, one can observe that the third and fourth eigenfunctions yet somehow localize by the extremal function values the separate lenses and the temples' curved endings, however, values of the last two eigenfunctions already oscillate too much, thus, do not provide a meaningful description of the salient shape parts anymore.

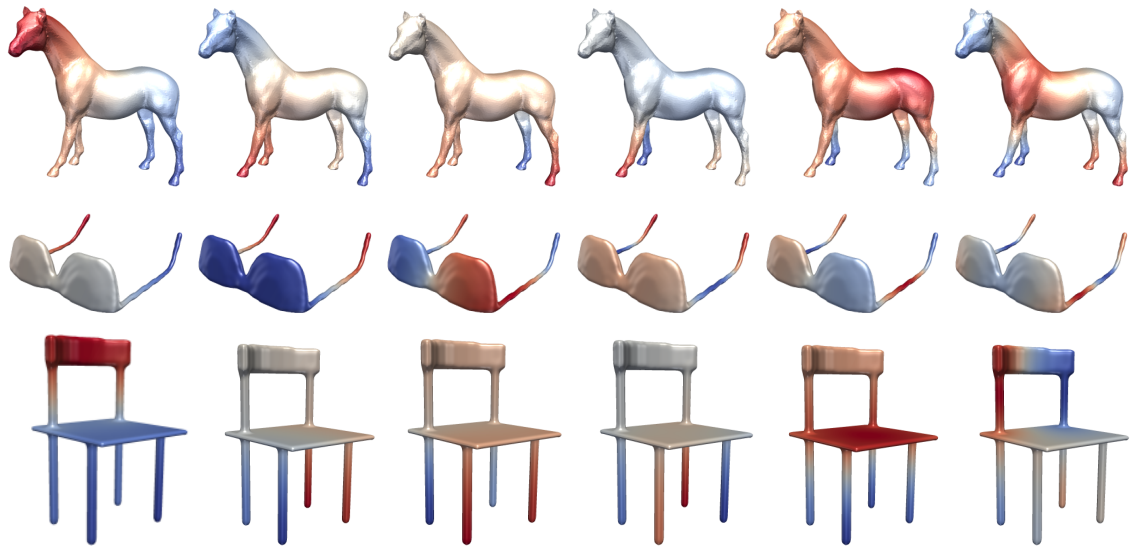


Fig. 5.10 The smallest $k = 6$ eigenfunctions of the affinity matrix (5.10)-(5.11) for `horse_2`, `glasses` and `chair` meshes.

The multi-channel function $f$ in the form (5.12) which is a vector function defined for each vertex $X_i$ of the mesh, can be assumed as a multi-channel input function for the segmentation algorithm. However, we are not limited to spectral information, and many other shape properties can be analogously exploited as multi-channel input function $f$.

Properties of the Laplacian spectrum have been widely investigated in shape analysis and applied for several applications in surface processing, such as shape segmentation, matching and retrieval, see [100],[74]. The choice of the Laplacian matrix influences the spectral segmentation results as documented, for example, in [130], where instead of the more common cotangent based Laplacian proposed in [82, 98], the Laplacian matrix of the dual graph (triangle-based) is considered, weighted by the dihedral angles.

## 5.3 Compact Support $L_p$CMs as Localized Shape Descriptors

In Section 5.2, we described the basis generated by the eigenfunctions of the Affinity matrix. These provide a spectral coordinates of the vertices, thus, jointly create a dense, multi-dimensional characteristic of the salient parts of the object.

In recent years, the basis generated by the eigenfunctions of Laplace-Beltrami Operator (LBO), called Manifold Harmonic Basis (MHB) has been proposed in [118] and commonly used for multi-resolution shape approximation in analogy to Fourier analysis.

It is well known that the eigenfunctions of the Laplace Beltrami operator, called Manifold Harmonics (MH), define a function basis. In particular, for a smooth manifold $\mathcal{M}$ embedded in $\mathbb{R}^3$ the Laplace-Beltrami operator induces a set of eigenfunctions $\{\phi_k\}$ and associated eigenvalues $\{\lambda_k\}$ determined by

$$-\Delta\phi_k = \lambda_k\phi_k \qquad k \in \mathbb{N},\ \lambda_k \in \mathbb{R}\ . \tag{5.13}$$

The self-adjointness of $\Delta$ implies that the eigenvalues are real and that the eigenfunctions are orthogonal with respect to the $L_2$-inner product: $< f, g >= \int_{\mathcal{M}} f\ g$.

One major drawback of this basis is that, similarly to the Fourier spectrum, the MHs are dense and have global spatial support. This means that the functions do not give intuitive insight on the features of the manifold, thus reducing their use in practical shape processing applications [74]. It is also well known that using a reduced number of eigenfunctions corresponding to the smallest eigenvalues $\lambda$, the MHs allow to approximate the shape of the manifold in an improved manner as the number of eigenfunctions increases.

However, in the shape partitioning context, rather than a multi-resolution representation of the shape, which is the peculiarity of the MHB on manifolds, in this section, the focus is on identifying the observable features of the manifold which represent for example protrusions, ridges, details in general localized in small regions.

Hence, in the partitioning context, a more suitable alternative to the MHB is represented by the Compressed Manifold Basis (CMB), introduced in [88], which is characterized by compact support quasi-eigenfunctions of the LBO obtained by imposing sparsity constraints.

Motivated by the advantages in terms of control on the compact support obtained by using the $L_1$ norm to force the sparsity of the solution discussed in [92] and [88], we devised to replace the $L_1$ norm by a more effective sparsity-inducing $L_p$ norm term, with $0 < p \leq 1$, which stronger enforces the locality of the resulting basis functions. The set of functions $\Psi = \{\psi_k\}_{k=1}^N$, that we will call $L_p$ Compressed Modes ($L_p$CMs), is computed by solving the following variational model

$$\min_{\Psi} \sum_{k=1}^{N} \left( \frac{1}{\mu}\|\psi_k\|_p^p - \frac{1}{2} < \psi_k, \Delta\psi_k > \right) \quad s.t. \quad < \psi_j, \psi_k >= \delta_{jk}\ , \tag{5.14}$$

where $\delta_{jk}$ is the Kronecker delta, and $\mu > 0$ is a penalty parameter. They form an orthonormal basis for the $L^2(\Omega)$ space, where $\Omega$ is the domain in consideration, and they represent a set of quasi-eigenfunctions of the Laplace-Beltrami operator.

The second term in the objective function of (5.14) is the fidelity term which represents the accuracy of the shape approximation provided by the set of functions $\Psi$, while the first term, so-called penalty term, forces the sparsity in the functions $\Psi$ thus imposing spatially sparse solutions. We remark that at the aim to construct a basis which is sparse but also localized in space it is necessary to further demonstrate that the functions $\Psi$ determined by solving (5.14) have compact support. This aspect will be proved in this section.

The penalty parameter $\mu$ controls the compromise between the two aspects. It is well known that the sparsity is better induced by the $L_p$ norm for $0 < p < 1$, rather than the $L_1$ norm. For $p = 1$ model (5.14) reduces to the proposal in [92], where the sparsity is forced only by acting on the $\mu$ value to increase the contribution of the penalty term, thus decreasing the shape approximation guaranteed by the fidelity term.

The parameter $p$ plays a crucial role since it allows to force the sparsity while maintaining the approximation accuracy without excessively stressing the penalty via the $\mu$ value. The accuracy is fundamental to localize the support of the functions in specific local features of the shape such as protrusions and ridges.

(a) Supports of $L_p$CM Basis, $p$=1, $\mu = 100$;

(b) Supports of $L_p$CM Basis, $p = 0.8$, $\mu = 1000$;

Fig. 5.11 Partitioning of the 2-manifold `hand` using $L_1$CM basis (a) and $L_p$CM basis (b).

Some evidence of the benefit obtained by the sparsity-inducing proposal, is shown in Fig.5.11 where we try to answer the following question: Can we identify the most salient parts of the manifold `hand` using only six compressed modes? Fig.5.11 compares the supports of the compressed modes determined as solution of the variational problem (5.14) with $p = 1$ (Fig.5.11(a)) and $p = 0.8$ (Fig.5.11 (b)) where the $\mu$ parameter has been chosen, for each $p$ value, to provide the most natural salient part identification. An automatic strategy for the choice of the optimal $\mu$ parameter will be also discussed in this work. The supports of the six quasi-eigenfunctions are colored in red and we can observe that using $p < 1$ strengthens the sparsity, while if $p = 1$ no $\mu$ value has allowed to correctly identify all the fingers.

**Background on the Compressed Modes**

In the preliminary work [92] the authors show how to produce a basis of localized functions $\{\psi_k\}_{k=1}^N$ in $\mathbb{R}^d$, called Compressed Modes (CMs), by solving the following variational problem

$$\min_{\{\psi_1,\psi_2,..,\psi_N\}} \sum_{k=1}^{N} \left( \frac{1}{\mu} \|\psi_k\|_1 + \langle \psi_k, H\psi_k \rangle \right) \quad s.t. \quad < \psi_j, \psi_k >= \delta_{jk} \ , \tag{5.15}$$

where $H = -\frac{1}{2}\Delta + V(x)$ is the Hamiltonian operator corresponding to potential $V(x)$, the $L_1$ norm is defined as $\|f\|_1 = \int_\Omega |f| dx$ and $\langle f, g \rangle = \int_\Omega f^* g dx$ $\Omega \subset \mathbb{R}^d$. Here the $L_1$ norm is a penalty term used to achieve spatial sparsity. The orthonormality constraints in (5.15), which enforce the orthonormality of the basis functions, lead to a non-convex variational problem, with many local minimizers.

A theoretical analysis of the CMs, provided in [7], allows for finding the minimizer of the variational formulation of the Schrodinger equation, showing the spatial localization property of CMs, and establishing an upper bound on the volume of their support. Consistency results for the CMs were proved in [6].

In [88] the variational problem (5.15) in $\mathbb{R}^d$ domains has been extended to deal with Laplace-Beltrami eigenfunctions on 2-manifolds discretized by three-dimensional meshes. These new basis functions, named Compressed Manifold Modes (CMM), form the Compressed Manifold Basis (CMB) and define an alternative to the classical MHB, proposed in [118].

### 5.3.1 The sparsity-inducing variational model for $L_p$CMs

Let $\Omega = B(0, R) \subset \mathbb{R}^d$ denote the $d$-dimensional ball of radius $R$ centered at the origin. Relation (5.14), can be rewritten as:

$$\min_{\Psi} \sum_{k=1}^{N} \left( \frac{1}{\mu} \int_\Omega |\psi_k|^p \, dx - \frac{1}{2} \int_\Omega \psi_k \Delta \psi_k dx \right) \quad s.t. \quad \int_\Omega \psi_j \, \psi_k \, dx \ = \ \delta_{jk} \ , \tag{5.16}$$

where we denoted the $L_p$ norm of a function by $\|f\|_p = (\int_\Omega |f|^p dx)^{1/p}$.

Before proceeding with the solution of the variational model (5.16) we demonstrate in the following two important properties of the $L_p$ compressed functions such as local support and completeness, which also hold for the CMs determined by solving (5.15).

### On the support of the $L_p$CMs

In the following we establish an asymptotic upper bound on the volume of the support of the $L_p$CMs in terms of the penalty parameter $\mu$ and the sparsity parameter $p$. At this aim, we first reformulate (5.16) by using integration by parts and imposing zero boundary conditions on $\Omega$. It follows that the first $N$ compressed modes $\{\psi_i\}_{i=1}^N$ solve the following constrained optimization problem:

$$\min_{\Psi} \sum_{i=1}^{N} \left( \frac{1}{\mu} \int_\Omega |\psi_i|^p \, dx + \frac{1}{2} \int_\Omega |\nabla \psi_i|^2 \, dx \right) \quad s.t. \quad \int_\Omega \psi_j \psi_k \, dx = \delta_{jk} \, . \tag{5.17}$$

We first introduce the following result on the volume support of the first compressed mode.

**Proposition 4** *$\forall x \in \Omega$, any $0 < p < 1$, and $\mu$ sufficiently small, we have*

$$\int_\Omega \left( \frac{1}{\mu} |\psi_1|^p + \frac{1}{2} |\nabla \psi_1|^2 \right) dx \leq m(\Omega)^{\frac{1}{p}-1} \mu^{-\frac{4}{4+d}} \tag{5.18}$$

*where $m(\Omega)$ is the finite measure of the domain $\Omega \subset \mathbb{R}^d$.*

*Proof.* From the relation between $L_p$ norm and $L_q$ norm, with $0 < p < q \leq \infty$

$$\|f\|_p \leq m(\Omega)^{\frac{1}{p}-\frac{1}{q}} \|f\|_q \, , \tag{5.19}$$

for $q = 1$ and $0 < p < 1$, it follows that

$$\int_\Omega \left( \frac{1}{\mu} |\psi_1|^p + \frac{1}{2} |\nabla \psi_1|^2 \, dx \right) \leq m(\Omega)^{\frac{1}{p}-1} \cdot \int_\Omega \left( \frac{1}{\mu} |\psi_1| + \frac{1}{2} |\nabla \psi_1|^2 \right) dx. \tag{5.20}$$

By using Proposition 3.4 of [7], namely

$$\int_\Omega \left( \frac{1}{\mu} |\psi_1| + \frac{1}{2} |\nabla \psi_1|^2 \right) dx = C_1 \mu^{-\frac{4}{4+d}},$$

where $C_1$ is some fixed constant depending on $d$, we easily obtain the bound in (5.18). □

**Theorem 1** *There exist $\mu_0$, such that for $\mu < \mu_0$ the corresponding $L_p$ compressed modes $\{\psi_i\}_{i=1}^N$ satisfy*

$$|supp(\psi_i)| \leq C \mu^{-\frac{8}{4+d}+1} m(\Omega)^{\frac{1}{p(1-p)}-2} \tag{5.21}$$

*where $C$ depends on $N$ and $p$.*

The proof is postponed to the Appendix.

The result in Theorem 1 is fundamental for the construction of a compact support $L_p$CM basis and it will represent the key aspect for the shape partitioning method based on the $L_p$CMs that will be described in Section 6.2.

An example demonstrating the essence of Theorem 1 is shown in Figure 5.12. In each row three $L_p$CM functions are illustrated obtained for a particular $\mu$ value, and fixed $N$ and $p$ values. Since the upper bound given in Theorem 1 depends on $\mu$, $p$, $N$, for increasing values of $\mu$, as we expected, we notice an enlargement of the compact support of each function.

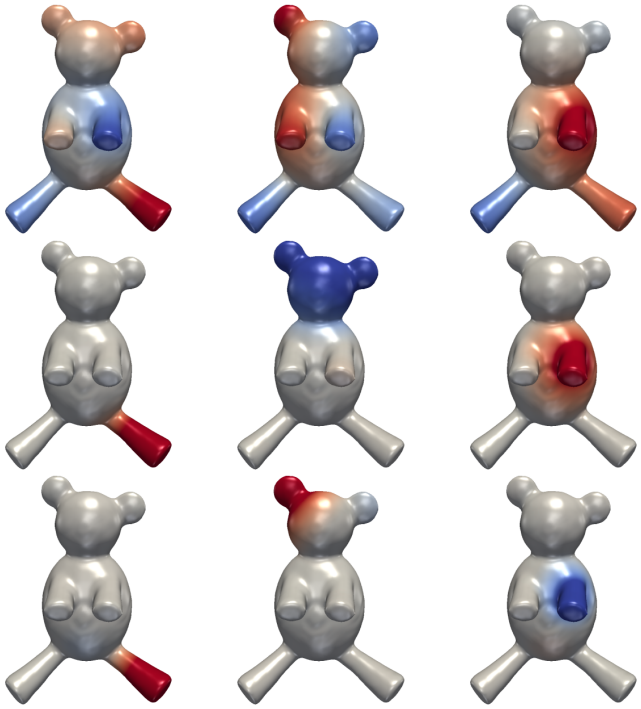


Fig. 5.12 Three $L_p$CMs generated for the `teddy_2` mesh for three different $\mu$ values, row-wise $\mu = \{250, 160, 50\}$ .

### Completeness of the $L_p$CMs

We now investigate a completeness result on the $L_p$CMs and its effect on shape approximation. In particular, we prove that, for a fixed $\mu$ value in (5.16), under some unitary transformations, the $L_p$CM functions $\{\psi_i\}_{i=1}^{N}$ approximate the eigenfunctions of the Laplacian operator in an improved manner as $N$ increases.

Let $\Phi = \{\phi_i\}_{i=1}^{M}$ be the set of orthonormal eigenfunctions of $-\frac{1}{2}\Delta$ corresponding to the eigenvalues $\{\lambda_i\}_{i=1}^{M}$, defined by (5.13) where the eigenvalues are arranged in non-decreasing order.

The following result of completeness for the $L_p$CMs holds.

**Theorem 2** *Given fixed parameters $\mu$ and $p$ in (5.16), for a fixed integer $M < N$, the first $N$ functions $L_p$CMs $\{\psi_i\}_{i=1}^{N}$ up to an unitary transformation, satisfy*

$$\lim_{N\to\infty} \|\phi_i - \psi_i\|_2^2 = 0, \quad i = 1, \ldots, M. \tag{5.22}$$

*Proof.* The proof follows from [129], where the authors demonstrate the result in the case of $L_1$-norm, but it still holds if the $L_1$ norm term is replaced by any functional bounded by $L_2$ norm. In fact, for the relation (5.19) between $L_p$ and $L_q$ norms, $0 < p < q \leq \infty$, if we set $q = 2$, it follows that $L_p$-norm, $0 < p < 2$, is bounded by $L_2$-norm. □

The completeness result confirms that using the $L_p$CM orthogonal basis, analogously to the $\Phi$ basis, we can reconstruct any function defined on the shape, up to an arbitrary degree of precision. However, for a small number $N$ of functions, the approximated reconstructions show significant differences.

By the way of illustration, let us consider the geometric reconstruction of the 2-manifold `horse`. The shape approximation process, described for MHs in [118], also holds for L$_p$CMs. The reconstruction obtained by using all the $N$ eigenfunctions of the LBO, where $N = 868$, is shown in Fig. 5.13 (top). In the second and third row of Fig. 5.13 we show, respectively, the shape reconstructions obtained using the basis MHB computed by solving (5.13), and the proposed L$_p$CM basis obtained by (5.16), formulated in the 2-manifold context which will be discussed in the following sections. For a fixed value of the dimension $N$ in the range $N = 8, 15, 30$, the reconstruction obtained by MHB is smoother but less representative of the underlying shape, while the L$_p$CM approximation looks like a more stylized shape, roughly a skeleton of the shape. Moreover, while the MHB approximations for increasing dimensions tend to refine the basic shape, the L$_p$CM basis enriches the skeleton shape with smaller features while maintaining the structure of the shape. This can be observed in the horse reconstructions in Fig. 5.13 (bottom) where the horse's legs and ears are well represented only using the L$_p$CM bases.

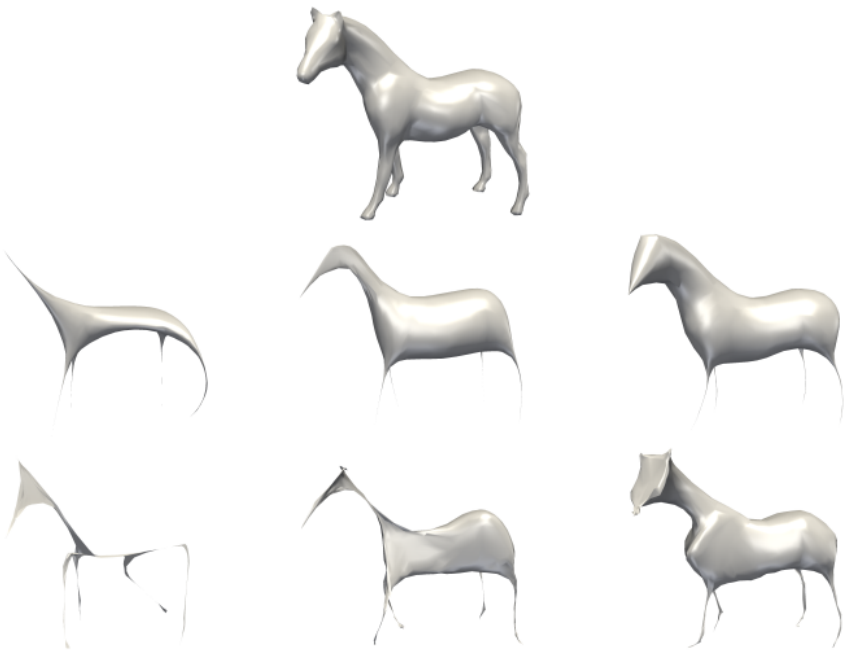


Fig. 5.13 Reconstruction of the `horse` shape (top) using MH (middle row) and $L_p$CM (bottom row) bases for increasing space dimension $N$; from left to right $N = 8$, $N = 15$, and $N = 30$.

### 5.3.2 Discretization of the variational model

We are interested in the application of model (5.14) to compute the $L_p$ Compressed Modes induced by the LBO on a manifold $\mathcal{M}$ approximated by mesh $M$.

By applying the discretization $D^{-1}L$ (4.15) for the LBO on $M$, and arranging the discretized $L_p$CMs in columns of a matrix $\Psi = [\psi_1, \ldots, \psi_N]$, with $\Psi \in \mathbb{R}^{n \times N}$, the constrained minimization problem (5.16) on $M$ reads as follows

$$\Psi^* = \arg\min_{\Psi} \frac{1}{\mu} \|\Psi\|_p^p + \mathrm{Tr}\left(\Psi^T L \Psi\right) \quad s.t. \quad \Psi^T D \Psi = I, \tag{5.23}$$

where $Tr(\cdot)$ denotes the trace operator, and $\|\Psi\|_p^p = \sum_{i,j} d_i |\Psi_{i,j}|^p$, with $d_i$ diagonal elements of the matrix $D$. In [21] the authors propose to approximate discrete norms, in particular the $L_1$ norm, as a weighted sum in the vertex neighbourhood

$$\|\Psi\|_p^p = \sum_{i,j} |\Psi_{i,j}|^p w(X_i) \ ,$$

where the given weights $w(\Psi_{i,\cdot})$ can be set apart from the uniform choice $w(\Psi_{i,\cdot}) = 1$ as the area around associated vertex $X_i$ or polyhedra volumes determined by linear basis

functions $b_i(X)$ defined over every triangle $\tau \in T$ as

$$w(\Psi_{i,\cdot}) = |N_\triangle(X_i)| \,, \tag{5.24}$$

$$w(\Psi_{i,\cdot}) = \sum_{\tau \in N_\triangle(X_i)} \int_\tau b_i(X) \cdot \mathrm{sign}(\hat{\Psi}(X)) \tag{5.25}$$

where the piecewise-linear hat functions $b_i(X)$ are defined as

$$b_i(X) = \begin{cases} 1 & X = X_i \\ 0 & X \in M \backslash N(X_i) \\ \text{linear} & \text{on } N(X_i) \end{cases} \,, \tag{5.26}$$

and $\hat{\Psi}(X) \approx \sum_{i=1}^{n} \Psi_{i,\cdot} b_i(X)$. We follow the area-weights option.

A discussion on the existence of a minimizer for a constrained variational problem relies on conditions on the associated Lagrangian and on the constraints. In particular, the orthogonality constraints in problem (5.23) are bounded above by quadratic functions. The Lagrangian function of (5.23) is defined as

$$\mathcal{J}(\Psi, \Lambda) = \frac{1}{\mu} \|\Psi\|_p^p + \mathrm{Tr}\left(\Psi^T L \Psi\right) - \mathrm{Tr}\left(\Lambda(\Psi^T D \Psi - I)\right) \tag{5.27}$$

where $\Lambda$ is the matrix of Lagrangian multipliers. The function (5.27) is proper, lower semi-continuous, bounded from below and coercive. If $\Psi$ is a local minimizer of (5.23) then $\Psi$ satisfies the first-order optimality conditions

$$\mathcal{D}_\Psi \mathcal{J}(\Psi, \Lambda) = \frac{1}{\mu} \nu^* + (2L - 2\Lambda D)\Psi = 0 \tag{5.28}$$

where $\nu^* \in \partial_\Psi \left[ \|\Psi\|_p^p \right] (\Psi^*)$ represents the subdifferential (with respect to $\Psi$, calculated at $\Psi^*$), defined in (A.1), and we used results from [104] for trace derivative.

The above-described optimization problem allows us to determine the $\mathrm{L}_p$ Compressed Modes for given dimension $N$ and given regularization parameter $\mu$. Two sets of $\mathrm{L}_p$CM basis generated from the meshes `wolf` and `fawn` are shown in Fig. 5.14. The change of the compact support of $\psi_1 \in \Psi$, in case of `bird` mesh, for different values of $\mu$ is demonstrated in Figure 5.15.

However, the numerical method to obtain a basis that covers the whole mesh $M$, either just for given $N$ or for given $\mu$, will be described together with the partitioning algorithm in Section 6.2.

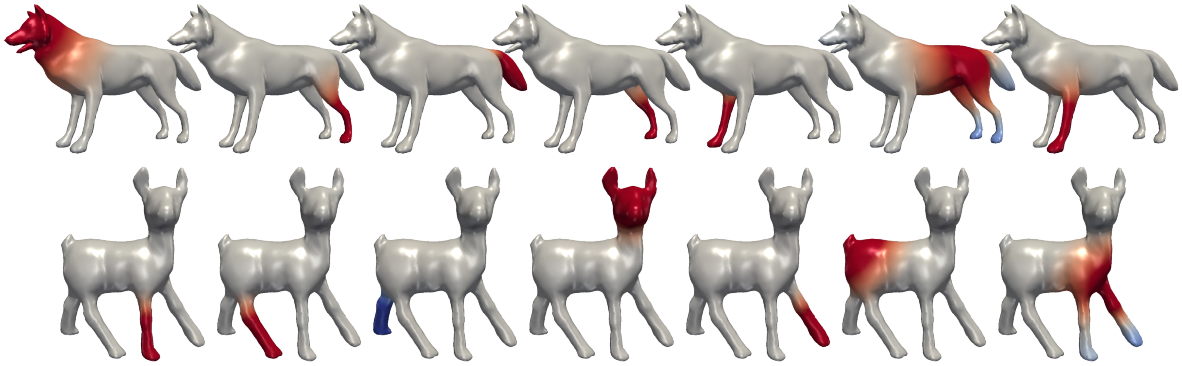


Fig. 5.14 Example of two sets of $L_p$CM basis $\Psi$ generated for the `wolf` and the `fawn` meshes.

Fig. 5.15 $L_p$CM $\psi_1$ of $\Psi$ generated for 4 different $\mu$ values of `bird` mesh. From left to right: enlarging of the support obtained by increasing the parameter $\mu$ for a fixed number of basis functions.

The rest of this chapter will be devoted to the numerical solution of the optimization problem (5.23) by an ADMM-based strategy.

### 5.3.3 Applying ADMM to the proposed model

In this section, we illustrate in detail the ADMM-based iterative algorithm used to numerically solve the proposed model (5.23). Two different splitting methods for solving problem (5.23) have been proposed in [92] and [88]. In [92] the authors solve the minimization problem by the splitting orthogonality constraint (SOC) method introduced in [65], while in [88] an ADMM approach is introduced that improves the empirical convergence performance of the former. Our approach follows the ADMM strategy, and mainly differs from [88] in the proximal map sub-problem.

First, we replace the orthogonality constraint in (5.23) using an indicator function

$$\iota(\Psi) = \begin{cases} 0 & \text{if} \quad \Psi^T D \Psi = I \\ \infty & \text{otherwise} . \end{cases}$$

Then problem (5.23) can be rewritten as:

$$\Psi^* \;=\; \arg\min_{\Psi} \frac{1}{\mu}\|\Psi\|_p^p + \mathrm{Tr}(\Psi^T L \Psi) + \iota(\Psi)\;. \tag{5.29}$$

We can resort to the variable splitting technique for the orthogonality constraint and introduce two new auxiliary matrices, $E, S \in R^{n\times N}$, the problem (5.29) is then rewritten as

$$\min_{\Psi,S,E} \frac{1}{\mu}\|S\|_p^p + \mathrm{Tr}(E^T L E) + \iota(\Psi) \quad s.t. \quad \Psi = S\,,\;\; \Psi = E\;. \tag{5.30}$$

To solve problem (5.30), we define the augmented Lagrangian functional

$$\begin{aligned}\mathcal{L}(\Psi, S, E; U_E, U_S; \mu) \quad = \quad & \frac{1}{\mu}\|S\|_p^p + \mathrm{Tr}(E^T L E) + \iota(\Psi) \\ & - \langle\, U_S, \Psi - S \,\rangle \;\; + \frac{\rho}{2}\,\|\Psi - S\|_F^2 \\ & - \langle\, U_E, \Psi - E \,\rangle + \frac{\rho}{2}\,\|\Psi - E\|_F^2 \end{aligned} \tag{5.31}$$

where $\rho > 0$ is scalar penalty parameter and $U_S \in \mathbb{R}^{n\times N}$, $U_E \in \mathbb{R}^{n\times N}$ are the matrices of Lagrange multipliers associated with the linear constraints $\Psi = S$ and $\Psi = E$ in (5.30), respectively.

We then consider the following saddle-point problem:

$$\begin{aligned} \text{Find} \quad & (\Psi^*, S^*, E^*; U_S^*, U_E^*) \;\in\; \mathbb{R}^{n\times N}\times\; \mathbb{R}^{n\times N}\times\; \mathbb{R}^{n\times N}\times\; \mathbb{R}^{n\times N}\times\; \mathbb{R}^{n\times N} \\ \text{s.t.} \quad & \mathcal{L}\,(\Psi^*, S^*, E^*; U_E, U_S; \mu) \;\le\; \mathcal{L}\,(\Psi^*, S^*, E^*; U_E^*, U_S^*; \mu) \;\le\; \mathcal{L}\,(\Psi, S, E; U_E^*, U_S^*; \mu) \\ & \forall\,(\Psi, S, E; U_E, U_S) \;\in\; \mathbb{R}^{n\times N}\times\; \mathbb{R}^{n\times N}\times\; \mathbb{R}^{n\times N}\times\; \mathbb{R}^{n\times N}\times\; \mathbb{R}^{n\times N}\,, \end{aligned} \tag{5.32}$$

with the augmented Lagrangian functional $\mathcal{L}$ defined in (5.31).

In the following we present the ADMM-based iterative algorithm used to compute a saddle-point solution of (5.31)–(5.32) which provides a minimizer of problem (5.23).

Given the previously computed (or initialized for $k = 0$) matrices $S^{(k)}$, $E^{(k)}$, $U_S^{(k)}$ and $U_E^{(k)}$, the $k$-th iteration of the proposed ADMM-based iterative scheme applied to the solution of the saddle-point problem (5.31)–(5.32) reads as follows:

$$\Psi^{(k+1)} \leftarrow \arg\min_{\Psi\in\mathbb{R}^{n\times N}} \mathcal{L}(\Psi, S^{(k)}, E^{(k)}; U_S^{(k)}, U_E^{(k)}) \tag{5.33}$$

$$S^{(k+1)} \leftarrow \arg\min_{S\in\mathbb{R}^{n\times N}} \mathcal{L}(\Psi^{(k+1)}, S, E^{(k)}; U_S^{(k)}, U_E^{(k)}) \tag{5.34}$$

$$E^{(k+1)} \leftarrow \arg\min_{E\in\mathbb{R}^{n\times N}} \mathcal{L}(\Psi^{(k+1)}, S^{(k+1)}, E; U_S^{(k)}, U_E^{(k)}) \tag{5.35}$$

$$U_S^{(k+1)} \leftarrow U_S^{(k)} - \rho\left(\Psi^{(k+1)} - S^{(k+1)}\right) \tag{5.36}$$

$$U_E^{(k+1)} \leftarrow U_E^{(k)} - \rho\left(\Psi^{(k+1)} - E^{(k+1)}\right). \tag{5.37}$$

In the following we show in detail how to solve the three minimization sub-problems (5.33)–(5.35) for the primal variables $\Psi$, $S$ and $E$, respectively, while the ADMM dual variable updates (5.36)–(5.37) admit closed-form solutions.

### Solution of subproblem (5.33) for $\Psi$

We observe that the subproblem (5.33) can be rewritten as:

$$\Psi^{(k+1)} \leftarrow \arg\min_{\Psi} \frac{\rho}{2}\|\Psi - (S + \frac{1}{\rho}U_S)\|_F^2 + \frac{\rho}{2}\|\Psi - (E + \frac{1}{\rho}U_E)\|_F^2 + \iota(\Psi)\,. \tag{5.38}$$

If we omit the constant terms, problem (5.38) is equivalent to the following

$$\Psi^{(k+1)} \leftarrow \arg\min_{\Psi} \rho\|\Psi - Y\|_F^2 \quad s.t. \quad \Psi^T D\Psi = I \tag{5.39}$$

where $Y = \frac{1}{2}(S + \frac{1}{\rho}U_s + E + \frac{1}{\rho}U_E)$.

**Theorem 3** *The costrained quadratic problem (5.39), assuming $Y$ has full rank, has the closed-form solution*

$$\Psi^{(k+1)} = YV\Sigma^{-1/2}V^T \tag{5.40}$$

*where $V \in \mathbb{R}^{N\times N}$ is a orthogonal matrix and $\Sigma$ is a diagonal matrix satisfying the SVD factorization $Y^T DY = V\Sigma V^T$.*

*Proof.* Setting

$$\Psi = D^{-\frac{1}{2}}\Phi\,, \tag{5.41}$$

then the constraint in (5.39) is equivalent to $\Phi^T\Phi = I$, and a solution of (5.39) can be obtained by solving:

$$\min_{\Phi} \rho\|D^{-\frac{1}{2}}\Phi - Y\|_F^2 \;\; s.t. \;\; \Phi^T\Phi = I\,. \tag{5.42}$$

A closed-form solution of the minimization problem (5.42) can be derived by considering the Lagrangian of the constrained problem (5.42)

$$\mathcal{L}(\Phi, \Lambda) = \rho \| D^{-\frac{1}{2}} \Phi - Y \|_F^2 + Tr(\Lambda(\Phi^T \Phi - I)) \tag{5.43}$$

where $\Lambda$ is the matrix of Lagrangian multipliers, and its first-order optimality conditions which read as

$$\begin{cases} \dfrac{\partial \mathcal{L}}{\partial \Phi} = & 2\rho D^{-\frac{1}{2}}(D^{-\frac{1}{2}}\Phi - Y) + \Phi(\Lambda + \Lambda^T) = 0 \\ \\ \Phi^T \Phi = & I \end{cases} . \tag{5.44}$$

Multiplying by $D$ the first eq. in (5.44) we obtain:

$$\begin{cases} 2\rho(\Phi - D^{\frac{1}{2}}Y) + D\Phi(\Lambda + \Lambda^T) = & 0 \\ \Phi^T \Phi = & I \end{cases} , \tag{5.45}$$

from which it follows that

$$D^{\frac{1}{2}} Y = \Phi(I + \hat{D}(\Lambda + \Lambda^T)) \tag{5.46}$$

where $\hat{D} = \frac{1}{2\rho} D$, and then,

$$\Phi = D^{\frac{1}{2}} Y (I + \hat{D}(\Lambda + \Lambda^T))^{-1} . \tag{5.47}$$

We set $Z = D^{\frac{1}{2}} Y$, by recalling the second relation of (5.44) and using (5.46), it follows

$$Z^T Z = (I + \hat{D}(\Lambda + \Lambda^T))^T (I + \hat{D}(\Lambda + \Lambda^T)) . \tag{5.48}$$

Since $Z^T Z \in \mathbb{R}^{N \times N}$, with $N << n$, is symmetric and semi-definite positive, following [65], we apply the Singular Value Decomposition (SVD), namely $Z^T Z = V \Sigma V^T$.
Then $I + \hat{D}(\Lambda + \Lambda^T) = \pm V \Sigma^{\frac{1}{2}} V^T$ are two square roots of $Z^T Z$. The principal square root

$$(I + \hat{D}(\Lambda + \Lambda^T)) = V \Sigma^{\frac{1}{2}} V^T \tag{5.49}$$

is the one we desired. If $Z^T Z$ is full rank, then $V \Sigma^{\frac{1}{2}} V^T$ is invertible. Thus, relation (5.47) can be rewritten as:

$$\Phi = D^{\frac{1}{2}} Y V \Sigma^{-\frac{1}{2}} V^T$$

and by (5.41) it follows that

$$\Psi^{(k+1)} = D^{-\frac{1}{2}} D^{\frac{1}{2}} Y V \Sigma^{-\frac{1}{2}} V^T \,,$$

thus (5.40) holds. □

**Remark.** The problem (5.42) is known as orthogonal Procrustes problem. Following [47] a solution $\Phi$ of (5.42) reads as

$$\Phi = \tilde{U}\tilde{V}^T \,, \tag{5.50}$$

computed by applying the SVD to the matrix $B = (D^{-\frac{1}{2}})^T Y$, thus obtaining $B = \tilde{U}\tilde{\Sigma}\tilde{V}^T$. Since the SVD computation of an $m \times n$ matrix takes time that is proportional to $O(km^2n + k'n^3)$ with $k$ and $k'$ constants, the computational cost for computing the SVD of the $n \times N$ matrix $B$ is $O(n^2N + N^3)$, while in the proposed solution, as shown, we computed the SVD of a matrix $Z^TZ$ of dimensions $N \times N$, with a cost of $O(2N^3)$. Due to the fact that $n >> N$, we conclude that the proposed minimization proved in Theorem 3 is much more computational efficient than the use of the decomposition given in (5.50).

### Solution of subproblem (5.34) for $S$

Given $\Psi^{(k+1)}$, $E^{(k)}$, $U_S^{(k)}$,and $U_E^{(k)}$, and recalling the definition of the augmented Lagrangian functional in (5.31), the minimization sub-problem for $S$ in (5.34) can be rewritten as follows:

$$S^{(k+1)} \leftarrow \arg\min_S \frac{1}{\mu}\|S\|_p^p + \frac{\rho}{2}\|\Psi - (S + \frac{1}{\rho}U_S)\|_F^2 \,. \tag{5.51}$$

We can use the Generalized Iterated Shrinkage (GISA) strategy for Non-convex Sparse Coding proposed in [133], where the authors extended the popular soft-thresholding operator to $l_p$-norm, or its generalization given in [69]. Rewriting component-wise Eq. (5.51), the minimization problem is equivalent to the following $n \times N$ independent scalar problems:

$$s_{i,j}^{(k+1)} \leftarrow \arg\min_{s_{i,j}\in\mathbb{R}} \left\{ f(s_{i,j}) = \frac{d_i}{\rho\mu}|s_{i,j}|^p + \frac{1}{2}(s_{i,j} - q_{i,j})^2 \right\} \,, \quad \begin{matrix} i = 1,\ldots,n\,, \\ j = 1,\ldots,N \end{matrix} \tag{5.52}$$

where $q_{i,j} = \psi_{i,j} - \frac{1}{\rho}(U_S)_{i,j}$. Following Theorem 1 in [133] each of the optimization problems (5.52) has a unique minimum given by

$$\text{prox}_f^{\rho\mu}(q_{i,j}) = \begin{cases} 0 & \text{if } |q_{i,j}| \leq \hat{s} \\ \text{sign}(q_{i,j})\, s^* & \text{if } |q_{i,j}| > \hat{s} \end{cases} \tag{5.53}$$

where the thresholding value is

$$\hat{s} = \left(\frac{2d_i}{\rho\mu}(1-p)\right)^{1/(2-p)} + \frac{d_i}{\rho\mu}p\left(\frac{2d_i}{\rho\mu}(1-p)\right)^{(p-1)/(2-p)}$$

and $s^*$ is the unique solution of the following nonlinear equation:

$$s_{i,j} - q_{i,j} + p\frac{d_i}{\rho\mu}(s_{i,j})^{p-1} = 0\,, \tag{5.54}$$

that can be easily solved by a few iterations of an iterative zero-finding algorithm.

**Solution of subproblem (5.35) for $E$**

Given $\Psi^{(k+1)}, S^{(k+1)}, U_S^{(k)}$, and $U_E^{(k)}$, the minimization problem of the augmented Lagrangian functional in (5.31) with respect to $E$ in (5.35) can be rewritten as follows:

$$E^{(k+1)} \leftarrow \arg\min_E Tr(E^T L E) + \frac{\rho}{2}\|\Psi - (E + \frac{1}{\rho}U_E)\|_F^2\,. \tag{5.55}$$

To solve minimization problem (5.55), we consider the optimality conditions, namely

$$2LE + \rho(\Psi - (E + \frac{1}{\rho}U_E)) = 0\,,$$

which reduce to the solution of $N$ linear systems for $E$ in the following form

$$(\rho I - 2L)E = \rho(\Psi - \frac{1}{\rho}U_E)\,. \tag{5.56}$$

The solution of the optimization problem (5.23) via the above-described ADMM-based strategy will provide the set of vectors $\{\psi_i\}_{i=1}^N$ representing discretization of the $N$ quasi eigen-functions $\text{L}_p$CM obtained from a mesh $M$.

# Chapter 6

# Shape Partitioning

Shape partitioning enables the decomposition of arbitrary topology objects into smaller and more manageable pieces called partitions. In particular we are interested in Manifold Partitioning, since the boundaries of tangible physical objects can be mathematically defined by two-dimensional manifolds embedded into three-dimensional Euclidean space.

Let us introduce the following formulation of the shape partitioning problem.

**Definition 1 (Manifold Partitioning)** *Given a compact 2-manifold $\mathcal{M}$, find the partition into $N$ sub-manifolds defined by the pairs of topological spaces $\{(U_k, \partial U_k)\}_{k=1}^{N}$, with boundary $\partial U_k$, such that all of the following conditions hold:*

*P1)* $U_k, k = 1, \ldots, N$, *is a non-empty connected sub-manifold;*

*P2)* $\bigcup_{k=1}^{N} U_k = \mathcal{M}$

*P3) The intersection of any two distinct sub-manifolds $U_i, U_j$ in $\mathcal{M}$ is equal to a simple curve:*

$$U_i \cap U_j = \partial U_i \cap \partial U_j = \textit{1-manifold.}$$

The sub-manifolds $\{U_k\}_{k=1}^{N}$ are said to cover $\mathcal{M}$ and provide the so-called *segmentation*, or partitioning, of the object represented by $\mathcal{M}$.

Many shape processing applications rely on a more stringent characterization of partitioning which requires a global parametrization of the manifold. However, smooth global parameterization does not always exist or is easy to find. Only the simplest 2-manifolds indeed can be adequately parameterized. In general, a topology decomposition

of the manifold is required to describe it as a a collection of parameterized surfaces (charts).

We briefly review some useful definitions.

A chart for a 2-manifold $\mathcal{M}$ is a homeomorphism $\varphi$ from a subset $U$ of $\mathcal{M}$ to a subset of the two-dimensional Euclidean space. The chart is traditionally recorded as the ordered pair $(U, \varphi)$. A collection $\{(U_k, \varphi_k)\}$ of charts on $\mathcal{M}$ such that $\bigcup U_k = \mathcal{M}$ forms an atlas for $\mathcal{M}$.

When a manifold is constructed from multiple overlapping charts, the regions where they overlap carry information essential for understanding the global structure. In this context, as specified by *P3)* in Definition 1, the overlap is reduced to boundary curves shared by two adjacent patches.

A *patch-based* partitioning can be then defined as follows.

**Definition 2 (Patch-Based Manifold Partitioning)** *Given a compact 2-manifold $\mathcal{M}$, find the partition into $N$ sub-manifolds $\{(U_k, \partial U_k)\}_{k=1}^{N}$ such that conditions P1) - P3) hold, together with the following*

*P4) $U_k$ is a 0-genus sub-manifold that defines a chart.*

*P5) $U_k$ has at most two boundaries.*

Given a chart decomposition of a triangle mesh, each chart can be parameterized on a planar domain (e.g., a circle or a rectangle) using different methods, whose selection depends on its genus and number of boundary components. More precisely, a disk-like charts (i.e., 0-genus patches with one boundary component) are parameterized using the barycentric coordinates method [43]; while a 0-genus chart with more than one boundary component, or more generally charts with an arbitrary genus, are converted to disk-like regions by cutting them along cut-graphs and then embedded on the plane using the barycentric coordinates method [96],[110].

For approximation purposes and in order to reduce the parameterization distortion, it is preferable to work with disk-like patches.

In [95] a topology-based decomposition of the shape is computed and used to segment the shape into primitives, which define a chart decomposition of the mesh. The charts considered in [95] are all 0-genus but can present more than one boundary components. In contrast, in this work we restrict the chart $U_k$ to be a disk-like patch bounded by one or two closed curves. The latter requires a simple cut between the two boundaries to avoid internal holes in the planar parameterization.

Once the proposed patch-based manifold partitioning is built, we can associate a parameterization $\varphi_k$ to each sub-manifold $U_k$. However, we omit the construction of a parameterization, as discussing these details goes beyond the scope of this work.

## 6.1 Sparsity-Inducing Non-Convex Variational Shape Partitioning

In this section we introduce the partitioning framework which exploits global or local shape information represented by a generic vector function $f : \Omega \to \mathbb{R}^d, d \geq 1$ at the points $V$ to infer a decomposition of the surface $\mathcal{M}$ in salient parts. In Section 6.4.2 we present segmentation results for a well-known single-channel (scalar) function $f$, the Shape Diameter Function, which measures the thickness property of an object, as well as results for a vector function defined in (5.12) (multi-channel) derived from spectral decomposition which better reflects the human perception of shape decomposition. In the latter case $d$ is thus the number of considered eigenvectors of the affinity matrix $\mathcal{L}$ in (5.10).

Our proposal, as well as the variational mesh decomposition introduced in [130], are based on the relaxation of Mumford-Shah variational model (2.10) proposed by Nikolova et al. (2.11) in [27], introduced in Chapter 2.

However, model (2.11) used in [130] works well only if the intensity function $f$ is homogeneous in each region. When this is not the case, that is in the presence of inhomogeneities inside the regions to be segmented, then variational model (2.12) for piecewise-smooth partitioning behaves better. For image segmentation, authors in [22] introduced a convex relaxation where the boundary information is extracted from the total variation term.

Our goal is to develop an object partitioning framework that has the following properties:

- work on multi-channel (vector-valued) functions characterizing arbitrary object features;
- exploit an ad hoc sparsity-inducing regularizer for minimizing the total length of the boundaries while preserving their geometric features (corner, flat, etc.);
- utilize a two-step procedure such that Step 1 is independent on the number $K$ of segments required; no need to solve the whole problem again for different $K$ values;
- work both for homogeneous and piecewise smooth function $f$ over each channel;

- detect portion of objects whose boundaries are characterized by significant changes both in $f$, and in the local curvature.

To that aim, in the following, we present a strategy for the partitioning of meshes based on a new variant of the Mumford-Shah models (2.9) and (2.10) where we adopt an $L_p$-norm approximation of the total length of the boundaries.

Let $f = (f_1, \ldots, f_d)$ be a given vector-valued function with channels $f_i : \Omega \to \mathbb{R}$, $i = 1, .., d$, and $u = (u_1, \ldots, u_d)$ be a vector function on $\Omega$, eventually non-smooth, named the *partition function*. Unlike for the colour image segmentation process where all image channels participate jointly in driving the segmentation process [111], here we apply the variants of the Mumford-Shah models (2.9) and (2.10) to each channel $u_i$ of $u$, for $i = 1, \ldots, d$. In particular, in the first step, each channel $u_i$ is separately computed by minimizing the *piecewise smooth partitioning functional*:

$$\min_{u_i} \{\mathcal{J}_s(u_i)\} \tag{6.1}$$

$$\mathcal{J}_s(u_i) \;:=\; \frac{1}{2}\int_\Omega |f_i - u_i|^2 \, d\Omega + \frac{\alpha}{p}\int_\Omega \phi\left(\|\nabla u_i\|\right) \, d\Omega + \frac{\beta}{2}\int_\Omega |\nabla u_i|^2 \, d\Omega, \tag{6.2}$$

or the *piecewise constant partitioning functional*:

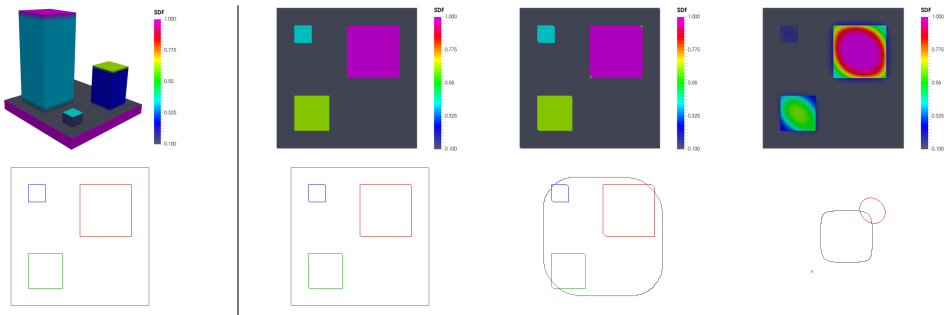
$$\min_{u_i} \{\mathcal{J}_c(u_i)\} \tag{6.3}$$

$$\mathcal{J}_c(u_i) \;:=\; \frac{1}{2}\int_\Omega |f_i - u_i|^2 \, d\Omega + \frac{\alpha}{p}\int_\Omega \phi\left(\|\nabla u_i\|\right) \, d\Omega, \tag{6.4}$$

where $\phi(t) := |t|^p$, is the penalty function with $p \in (0, 2]$, sparsity-inducing for $p < 1$, and $\beta := \beta(x)$, $\beta : \Omega \to [0, 1]$ is an adaptive function which approaches to zero at the high curvature points of $\Omega$. The models (6.1) and (6.3) represent an adaptation of (2.12).

In the second step, we apply a multi-channel clusterization procedure to the vector function $u$ to finalize the object partitioning. The number of parts $K$ (phases) is only required in this second step, so users can choose or change it without the need of solving the previous stage again.

The $L_p$ penalty term is introduced in (6.2) and (6.4) to better control the length of the boundaries and substantially improves upon the $L_1$ norm results. In particular, for $p = 1$ the penalty term in (6.2) and (6.4) corresponds to the Total Variation (TV) term which have been used in [130] to measure the length of the boundaries.



(a) SDF partitioning ground truth (b) $p = 0.2$ (c) $p = 0.8$ (d) $p = 1.0$

Fig. 6.1 Effect of the $L_p$ regularizer wrt the $L_1$ regularizer for the SDF partitioning of the `blocks_2` mesh.

The benefit of using $p < 1$ is illustrated in Fig. 6.1 for the segmentation of a mesh composed of variable size boxes (Fig.6.1 (a)). Since the thickness property is used as criteria for partitioning, from the top view, the expected results are four boxes which are shown in Fig.6.1 (a). The true thicknesses (heights) were used as thresholds. The top row of Fig.6.1(b), (c), and (d), shows the results obtained from the proposed variational model for different $p$, which are used in Step 2 to produce the simple partitions, according to the given thresholds, which represent the true heights. In the bottom row, we plot the partition boundaries, obtained as iso-contours of $u^*$ in the top row, according to the thresholds. For the choice $p < 1$ in (6.4) our model preserves the sharp boundary shape, as illustrated in Fig.6.1 (b) and (c), while for $p = 1$ the boundaries shrink and the small features disappear as illustrated in Fig.6.1 (d). In particular for $p$ approaching to zero the boundary shape improves and the original intensities are preserved.

This behaviour is justified from the fact that the well-known TV regularizer is defined as the continuous $L_1$ norm, $p = 1$, which inevitably curtails originally salient boundaries to penalize their magnitudes. In particular, as discussed in [112], the TV of a feature is directly proportional to its boundary size, so that one way of minimizing the TV of that feature would be to reduce its boundary size, in particular by smoothing corners. Moreover, the change in intensity due to TV regularization is inversely proportional to the scale of the feature, so that very small-scaled features are removed.

In order to make the model independent from the scale of the feature to segment, we could use the $L_0$ measure of the discrete gradient explicitly defined as $\| \nabla u \|_0 := \#\{x \,|\, \|\nabla u\|_2 \neq 0\}$, where $\#$ is the counting operator, which outputs how many times $u$ changes its value. We propose to approximate the $L_0$ measure of the gradient with the non-smooth non-convex and non-Lipschitz regularization term, $L_p$ quasi-norm, $\phi(t) = |t|^p$,

with $0 < p < 1$, which has recently been proposed in image processing and compressed sensing since it promotes gradient-sparser solutions or sparser solutions, substantially improving upon the $L_1$ norm results [69].
This choice may lead to a challenging computation problem since it requires non-convex (when $p < 1$), non-smooth minimization which, involving many minima, can get stuck in shallow local minima. However, in Section 6.1.1, we show how to efficiently solve these optimization problems.

### 6.1.1 Discretization of the SMCMR model

In this section we focus on the numerical solution of the Sparsity-inducing Multi-Channel Multiple Region (SMCMR) models. Finite dimensional approximate solutions to the shape partitioning problems (6.2) and (6.4) read respectively as the minimizations of the following functions

$$\mathcal{J}_s(u_i) := \frac{1}{2}\|u_i - f_i\|_2^2 + \frac{\alpha}{p}\sum_{j=1}^{n}\phi\left(\|(\nabla_w u_i)_j\|_2\right) + \frac{\beta}{2}\sum_{j=1}^{n}\|(\nabla_w u_i)_j\|_2^2 , \tag{6.5}$$

$$\mathcal{J}_c(u_i) := \frac{1}{2}\|u_i - f_i\|_2^2 + \frac{\alpha}{p}\sum_{j=1}^{n}\phi\left(\|(\nabla_w u_i)_j\|_2\right) , \tag{6.6}$$

where $f_i \in \mathbb{R}^n$ is a vector of values associated to the set of vertices $V$, and $u_i \in \mathbb{R}^n$ represents the discretization of the $ith$ component of the partition function $u$ to be estimated. The discrete operator $(\nabla_w u(v))$ denotes the discretization of the weighted local variation of the function $u$ at vertex $v$ given in (4.2) and (4.3).
The regularization terms in (6.5) and (6.6) encode a prior knowledge on the local variation of the partition function, expressed as (4.3).

The classical gradient descent method for the numerical integration of the optimization problems (6.1) and (6.3) would involved the p-Laplacian flow that follows from Lemma 1 and is for example used in [84] for polygonal mesh simplification. However, while its numerical implementation could be straightforward, because of stability constraints, the gradient descent has rather undesirable asymptotic convergence properties which can make it very inefficient.

In the rest of this section we propose a fast iterative method to approximate faithfully the minimizer of (6.5) and (6.6) which represent the discretized versions of (6.1) and (6.3), for $p \in (0, 2]$. The method presents a global minimum for $1 \leq p \leq 2$, while when $p < 1$, (6.5) and (6.6) are non-convex, and a global optimal solution is not insured. The

proposed iterative method has been implemented and evaluated as described in Section 6.1.2.

In what follows we focus on the minimization of $\mathcal{J}_s$ in (6.5), since the functional $\mathcal{J}_c$ in (6.6) can be seen as a special case of $\mathcal{J}_s$ when $\beta = 0$. However, since $\mathcal{J}_c$ is non-smooth, in our unified treatment of the two optimal problems, we will adapt a proximal forward backward (PFB) strategy for non-smooth optimization.

We first split the objective function into two terms, $h : \Omega \to \mathbb{R}$ and $g : \Omega \to \mathbb{R}$ where $h(u)$ is differentiable but $g(u)$ may not be differentiable (in case $\mathcal{J}_c$ in (6.6) is applied), then (6.5) reads as

$$\mathcal{J}_s(u_i) \;=\; \underbrace{\frac{1}{2}\|u_i - f_i\|_2^2}_{h(u)} + \underbrace{\frac{\alpha}{p}\sum_{j=1}^{n}\phi\left(\|(\nabla_w u_i)_j\|_2\right) + \frac{\beta}{2}\sum_{j=1}^{n}\|(\nabla_w u_i)_j\|_2^2}_{g(u)} \;.$$

In the following for simplicity of notations we drop the subscripts $i$.

The optimization problem *minimize* $h(u) + g(u)$, is then solved by applying an iterative PFB-based scheme [20], where each iteration step $k$ is given by

$$\begin{aligned} v^{(k)} &:= u^{(k-1)} - \lambda_k \nabla h(u^{(k-1)}) && (6.7)\\ u^{(k)} &:= \arg\min_u \left\{ g(u) + \frac{1}{2\lambda_k}\|u - v^{(k)}\|_2^2 \right\} && (6.8)\\ &:= (I + \lambda_k \partial\,[\,g\,]\,(u^{(k)}))^{-1} v^{(k)} && (6.9)\\ &:= prox_{\lambda_k g}(v^{(k)}) && (6.10) \end{aligned}$$

where $\partial_x[\,\phi\,](x^*)$ denotes the sub-differential with respect to $x$ of the function $\phi$ calculated at $x^*$, and when $\phi$ is differentiable, we have $\partial_x[\,\phi\,](x^*) = \{\nabla\phi(x^*)\}$ for all $x^*$. The explicit updating (6.7) represents the forward step, whereas the evaluation of the proximity operator (6.10) represents the implicit backward step which leads to the following system of equations:

$$(I + \lambda_k(\beta L_2^w + \alpha L_p^w))u^{(k)} = (1 - \lambda_k)u^{(k-1)} + \lambda_k\, f \qquad (6.11)$$

where $I$ denotes the identity matrix of order $n$, and $L_p^w$ denotes the discretization of the weighted p-Laplacian operator given in (4.3). The presence in $L_p^w$ of the (diffusivity) coefficient $\gamma_p^w(X_i, X_j)$ defined in (4.19) makes it highly non-linear and for arbitrary $p$ even non-differentiable. Solvers like the Newton's method which converges rapidly near a minimizer provided the objective functional depends smoothly on the solution, does not work satisfactorily on it or eventually fail. Therefore, we introduce a gradient linearization

technique for the non-linear equations (6.11), resulting in the *lagged diffusivity fixed point algorithm* [121] based on the following idea.

In order to solve the equation $\nabla J(\hat{x}) = 0$, we write

$$\nabla J(x) = L(x)x - z$$

where $z$ is independent of $x$. Then at each iteration $k$, one finds $x^{(k+1)}$ by solving the linear problem:

$$L(x^{(k)})x^{(k+1)} = z \ . \tag{6.12}$$

A connection between the gradient linearization approach, the lagged diffusivity fixed point iterations and the half quadratic minimization has been investigated in [90] where it is shown that the methods construct exactly the same sequence of iterates $x^{(k+1)}$.

Setting $u^0 = f$ and following (6.12), the backward iteration (6.11), is then replaced by the following linear system:

$$(I + \lambda_k(\beta L_2^w + \alpha L_p^w(u^{(k-1)}))) \, u^{(k)} \ = \ (1 - \lambda_k)u^{(k-1)} + \lambda_k \, f \tag{6.13}$$

where the non-linear (diffusion) operator $L_p^w$ has been linearized by $L_p^w(u^{(k-1)})$ applied on a function $u^{(k)}$.

The coefficient matrix of the linear system is positive definite, symmetric, and the linear system (6.13) is solvable, and the unique solution is the approximate solution of (6.10). The linear convergence of the of the Lagged Diffusivity Fixed Point Method for $p = 1$ is discussed in [28].

The backward iteration (6.13) can be further simplified using (4.18) thus obtaining $n$ independent linear equations for each vertex $X_\ell \in V$:

$$u^{(k)}(X_\ell) = \frac{(1 - \lambda_k)u^{(k-1)} + \lambda_k \, (f + \sum_j (\alpha \gamma_{\ell j}^{(k-1)} + \beta w_{\ell j}) u^{(k-1)}(X_j))}{1 + \lambda_k \sum_j (\alpha \gamma_{\ell j}^{(k-1)} + \beta w_{\ell j})} \tag{6.14}$$

where we omitted the $\gamma$ dependence on $w$ and $p$ notation to improve readability. Since for each vertex $X_\ell$, at each iteration $k$, the solution of the linear system (6.13) is reduced to an explicit solution of a linear diffusion equation, whose diffusivity depends on the previous iterate $u^{(k-1)}$, the overall computational cost for the solution of this problem is linear in the number of vertices.

For all $p \in [1, \infty)$ if the algorithm converges, then it converges to the solution of the minimized function $u$ in (6.5). However, when $p < 1$ (non-convex case) if the algorithm

converges to some function $u$ the latter is not guaranteed to be the global minimum of the minimized function (6.5).

To finalize the partitioning algorithm, we need a suitable proposal for the weights in (4.3). Towards this aim we remark that a natural property that a genuine partitioning algorithm should satisfy is that the boundaries should correspond to strong affinity changes on the function values between adjacent regions. Therefore the weights are chosen to be boundary detecting functions (4.5) defined as

$$w(X_\ell, X_j) \;=\; e^{-\|f(X_\ell)-f(X_j)\|_2^2/\sigma}\;. \tag{6.15}$$

The parameter $\sigma \in (0, 1]$ in (6.15) controls how much the similarities of two local neighbours are penalized. Smaller values of $\sigma$ preserve smaller differences in the function $f$.

By using (6.15) we get a good measurement of similarity, which penalizes in (4.18)-(4.19) the spatial clusterization flow of the vertices with different features.

### 6.1.2 Algorithm SMCMR

To summarize previous results, in Algorithm 3 we report the main steps of the proposed algorithm for mesh decomposition based on the variational formulations (6.5) and (6.6).

The partitioning algorithm consists of two steps; the first, for each i-*th* channel $i = 1, \ldots, d$, computes the minimizer $u_i$ by the PFB-based iterative scheme described in Section 6.1.1. In particular, the partition function $u_i$ is obtained by iterating (6.14) with weights given in (6.15) for each vertex $X_j \in V$, until the relative change of $u_i$ is below a fixed small tolerance $\epsilon$. The step sizes $\lambda_k$ can be found by a line search, that is their values are chosen in each iteration, however we followed a strategy to set $\lambda_0 = 10$ at the beginning, and at each iteration $\lambda_k$ is updated by factor 0.9.

STEP 2 is an automatic thresholding/clusterization procedure, and we could follow the classical K-means algorithm, with the K-means++ algorithm for cluster centre initialization [81, 2]. However, the K-means method is strongly sensitive to the initialization of cluster centres due to its non-convexity. In particular, it favours the centroids as far away as possible from each other, while for the proposed segmentation model, two salient parts of the object can have centroids not too far away. Therefore, instead of using K-means++ initialization, we can set the cluster centroids $c_i$, $i = 1, \ldots, K$ simply by assigning to each $c_i$ the value of $u^*$ at a point in each salient part. Then the clusterization is achieved

**Algorithm 3 SMCMR segmentation**

**Input**: mesh data $\Omega$, $f \in \mathbb{R}^d$, $K$ number of parts
**Output**: classification vector $Label \in \mathbb{R}^n$

**Parameters**: · penalty $p$ , tolerance $\epsilon$
· length regularizer $\alpha > 0$
· smooth regularizer $\beta > 0$
· similarity coefficient $\sigma > 0$ for $w$ in (6.15)

STEP 1: Compute $u^* \in \mathbb{R}^d$ through PFB Iterations

· PFB Initialization: $u^{(0)} = f$,
**for** $i = 1, \ldots d$ **do**:
· $k := 1, \quad \lambda_0 = 10$,
**repeat**
· FS: $v_i^{(k)} := (1 - \lambda_{k-1})u_i^{(k-1)} + \lambda_{k-1} f_i$
· BS: compute $u_i^{(k)}$ by (6.14)
· Update: $\lambda_k := 0.9\,\lambda_{k-1}$ , $k = k + 1$
**until** $\|u_i^{(k)} - u_i^{(k-1)}\|_2 < \epsilon$
**end for**
· $u^* = u^{(k)}$

STEP 2: Segmentation of the mesh into $K$ parts, using $u^*$

· $Label(X_i) = J$ , $J \in \{1, \ldots, K\}$ , $\forall X_i \in V$ by (6.16).

in only one iteration by labelling each vertex as

$$Label(X_i) = \arg \min_{j=1,\dots,K} ||u^*(X_i) - c_j||_2 \ , \tag{6.16}$$

without updating the cluster centroids, as it is instead required in the K-means algorithm.

The simple procedure mentioned above, in case of a single-channel function used, coincides with thresholding.

The output of Step 2 in Algorithm 3 is a manifold partitioning of $\Omega$, according to Definition 1.

## 6.2 Localized Shape Descriptors in Non-Convex Shape Partitioning

In Section 5.3.3 we described an optimization method to compute a basis of $N$ functions $L_p$CMs induced by the LBO of a manifold $\mathcal{M}$ represented by a mesh $M$ with $n$ vertices. Each $L_p$CM has compact support: it is non-zero only in a confined region of the domain, and the size of the compact support can be controlled by $\mu$ and $p$.

Relying on this result, in this section we propose a unified framework to perform both mesh segmentation and patch-based partitioning. The proposed method carries out two approaches, one completely unsupervised, namely the number of segments is determined automatically, and the other supervised, by performing a segmentation into a given number of parts. We refer the reader to [115] for unsupervised state-of-the-art methods, and to [10, 62] for supervised competitors. The unsupervised numerical algorithm to partition a mesh iteratively increases the support of $N$ functions $L_p$CMs, with $N << n$, until their supports cover the entire mesh without overlapping. The set of vertices in the support of $\psi_i$ defines a sub-mesh. A partitioning of $M$ is defined as the union of the $N$ sub-meshes $\psi_i, i = 1, \dots, N$.

The algorithm consists of three main steps illustrated in Algorithm 4, which takes as input the initial mesh $M$, the number of partitions $N$ or the initial $\mu$ value, and returns a set of sub-meshes $S$. As concerning mesh segmentations, given in Def. 1, the set $S$ is directly the output of Step 2, while for patch-based partitioning a further step (Step 3) is required to suitably refine the partition $S$ according to Def. 2.

In Step 1 an iterative process is applied to generate a basis $\{\psi_i\}_{i=1}^N$ by solving (5.23) with the ADMM procedure described in Sec. 5.3.3. This task can be realized following

**Algorithm 4 $L_p$CM Mesh Partitioning**

**Input**: mesh $M$ , $\mu$ or $N$
**Output**: patch set $S = \{S_k\}_{k=1}^{N}$
**Parameters**: tolerance $\epsilon = 0.01$

STEP 1: Compute $\Psi \in \mathbb{R}^{n \times N}$

STEP 1a (given $\mu$):
· set *uncovered* = true, $N = 1$
**while** (*uncovered*) **do**:
· $N \leftarrow N + 1$
· Compute $\{\psi_i\}_{i=1}^{N}$ by solving (5.23)
· set *uncovered* = $(\exists X_j : \psi_i(X_j) = 0 \; \forall i)$
**end while**

STEP 1b (given $N$):
· set *uncovered* = true, $\mu = 2$
**while** (*uncovered*) **do**:
· update $\mu \leftarrow 4\mu$
· Compute $\{\psi_i\}_{i=1}^{N}$ by solving (5.23)
· set *uncovered* = $(\exists X_j : \psi_i(X_j) = 0 \; \forall i)$
**end while**

STEP 2: Region Growing

**for** $k = 1, \ldots N$ **do**:
· set seeds $s_k$ according to (6.17),
· set initial buffer $b_k \leftarrow N_{\triangle}(s_k)$
**while** $b_k \neq \{\oslash\}$ **do**:
**if** $(||\psi_k(\tau)| - \max_{i=1,..,N} |\psi_i(\tau)|| \leq \epsilon)$
· add $\tau$ in $S_k$
· update $b_k$ by inserting $N_{\triangle}(\tau)$
**end if**
· update $b_k$ by removing $\tau$
**end while**
**end for**

STEP 3: Refinement for Patch-Based Manifold Partitioning

two different approaches, named Step 1a and Step 1b, that terminate when all the vertices $V$ are in the support of at least one $\mathrm{L}_p$CM.

In Step 1a the parameter $\mu$ in (5.23) is assigned. Starting from the construction of a small set of functions $L_p$CMs, the space dimension is enlarged at each iteration by adding a new function $\psi_i$ until any vertex of $M$ is covered by at least one function in $\Psi$. Starting from a small dimension space, Step 1a ends up with a space of dimension $N$ spanned by the $L_p$CMs. In this approach the final number of partitions $N$ is unpredictable in advance.

Alternatively, Step 1b overcomes the problem to identify an a priori value for $\mu$ and requires instead a fixed number for $N$. At each iteration, $N$ basis functions $L_p$CM are built by solving (5.23) with a given $\mu$. If there exists a vertex of $M$ not covered by any function in $\{\psi_i\}_{i=1}^N$, then $\mu$ is increased, thus causing an enlargement of the function supports. The solution of (5.23) is then re-iterated with the new value for $\mu$.

Once the $N$ functions discretized in $\Psi \in \mathbb{R}^{n\times N}$ are determined by either Step 1a or Step 1b, the whole set of vertices $V$ is covered but many regions can be over-covered by more $\mathrm{L}_p$CMs. Mesh partitioning satisfying Def. 1 is then carried out in Step 2. At this aim, $N$ initial seeds $(s_1, \ldots, s_N)$ are selected as the $L_p$CM extrema, as follows

$$s_k = \arg\max_{i=1,..,n} |\psi_k(X_i)| \quad k = 1, \ldots, N \; . \tag{6.17}$$

Then a region growing strategy is applied which consists of a buffer of adjacent neighbours of a given element set, and a loop in which the buffer and the element set are updated according to some decision rule. Starting from the initial buffer $b_k = N_\triangle(s_k), k = 1, \ldots, N$, we examine each triangle $\tau \in b_k$ to decide if it will be added to $S_k$ which is the set of triangles associated with the function $\psi_k$. We denote by $\psi_k(\tau)$ the value obtained interpolating $\psi_k$ on its vertices.

There are two cases that may occur when an unassigned triangle $\tau$ is considered:

- In the first case, $\tau$ is covered by one support, i.e. $\psi_k(\tau) \neq 0$ and $\psi_i(\tau) = 0 \;\; \forall i \neq k$. We remove $\tau$ from $b_k$ and assign it to $S_k$. Then the buffer $b_k$ is updated by adding the $\tau$'s neighbours $N_\triangle(\tau)$.

- The second case occurs when the supports of at least two basis functions overlap, i.e. $\psi_k(\tau) \neq 0$ and $\exists\, i \neq k \;:\; \psi_i(\tau) \neq 0$. This case locates over the bands of overlapped supporting functions. If the difference from the extrema is under a certain threshold $\epsilon$, which reads as

$$\Big| |\psi_k(\tau)| - \max_{i=1,\ldots,N} |\psi_i(\tau)| \Big| \leq \epsilon \; , \tag{6.18}$$

then $\tau$ is added to $S_k$ and $b_k$ is updated accordingly, as in the previous case. Otherwise, $\tau$ will be assigned to a different set and the only action taken in this case will be to remove $\tau$ from $b_k$.

We notice that condition (6.18) is trivially satisfied in the first case.

A better understanding of condition (6.18) is provided in Fig. 6.2. The region growing step has been applied to partition the `horse` mesh into $N = 6$ parts; the partitioning results of Step 2 are illustrated in Fig. 6.16.


Fig. 6.2 L$_p$CMs plotted over the magenta curve on the `horse_2` mesh. Over the curve only $\psi_3$ and $\psi_6$ are non-zero, and the red box shows the band where the supports overlap.

Along the magenta coloured curve depicted on the mesh (Fig. 6.2, left), from the horse's head to its bottom, we plot the values of the L$_p$CMs (Fig. 6.2, right). Only the two functions $\psi_3$ and $\psi_6$ of $\Psi$ are non-zero. For the sake of clarity we plot also $\psi_4$, which localizes rear-left leg of the `horse` mesh and over the line evaluates zero. The red box locates the band of overlapping. When the functions values, e.g. $\psi_3$ and $\psi_6$, are too close (below $\epsilon$), even in case of some minor numerical perturbation, a corresponding set of successive triangles may tend to over-leap in the cluster assignment. However, the condition (6.18) satisfyingly overcomes this practice issue.

Step 2 ends when the buffers are empty, i.e. all the triangles of M have been assigned to $S$.

An a posteriori procedure approximates the boundaries of the sub-meshes $\{S_k\}_{k=1}^{N}$ by smooth spline curves.

Step 3 of Algorithm 4 is applied to finalize the Patch-Based Manifold Partitioning following Definition 2. The refinement is required only for a few patches $\{S_k\}_{k=1}^{\bar{N}}$, $\bar{N} < N$, with genus greater than zero, and for 0-genus patches with more than two closed-loop boundaries. The refinement is an adaptive process that consists in the re-iteration of Step 1 and Step 2 for every patch $S_k$ that needs to be further subdivided, by imposing the initial number of partitions $N = 2$.

## 6.3 Convex Non-Convex approach in Segmentation over Surfaces

In this section we introduce a new framework for segmentation of manifolds based on scalar-valued features which exploits global or local shape information represented by a generic real-valued function $f : \mathcal{M} \to \mathbb{R}$ defined on a 2-manifold $\mathcal{M}$, to infer a decomposition of $\mathcal{M}$ in salient parts.

The basic step in the proposed framework is represented by the solution of the following variation of the Mumford-Shah variational model:

$$\begin{gathered} \min_u \mathcal{J}(u;\lambda,\eta,a)\,, \\ \mathcal{J}(u;\lambda,\eta,a) := \frac{\lambda}{2}\int_{\mathcal{M}}(u-f)^2\,d\mathcal{M} + \frac{\eta}{2}\int_{\mathcal{M}}|\nabla_w u|^2\,d\mathcal{M} + \int_{\mathcal{M}}\phi\left(|\nabla_w u|;a\right)\,d\mathcal{M}\,, \end{gathered} \tag{6.19}$$

where $\lambda > 0, \eta \geq 0$ are regularization parameters, $\nabla_w u$ is the intrinsic (Riemannian) gradient defined on $\mathcal{M}$ and $d\mathcal{M}$ is the manifolds element measure, while $|\cdot|$ is the Riemannian norm, $u : \mathcal{M} \to \mathbb{R}$ represents the manifold-valued partition function, and $\phi(\cdot\,;a) : [0,+\infty) \to \mathbb{R}$ is a parametrized, non-convex penalty function with parameter $a \geq 0$, which controls the degree of non-convexity and will be referred to as the *concavity parameter*. The functional is composed by the sum of smooth convex (quadratic) terms and a non-smooth non-convex regularization term designed for penalizing simultaneously the non-smoothness of the inner regions and the length of the segmented boundaries. Thus the functional $\mathcal{J}$ in (6.19) is non-smooth and can be convex or non-convex depending on the parameters $\lambda, \eta$ and $a$. We are interested in solving the segmentation problem via construction and then optimization of the CNC functional $\mathcal{J}$ in (6.19). From the seminal works in [13], and [89] for CNC image denoising, very interesting developments have been presented by Selesnik and others for different purposes, see [30, 68, 70, 67] for more details. The attractiveness of such CNC approach resides in its ability to promote sparsity more strongly than it is possible by using only convex terms while at the same time maintaining convexity of the total optimization problem, so that well-known reliable convex minimization approaches can be used to compute the (unique) solution.

A first contribution of this section is the derivation of conditions that ensure the functional $\mathcal{J}$ in problem (6.19) is convex – despite the regularization term being non-convex. The inclusion of the manifold's geometry is done by characterizing the convexity parameter $a$ locally. Therefore the model (6.19) has a unique global minimizer. In this section, we propose a three-stage variational segmentation method inspired by the piecewise smoothing proposal in [22] which is a convex variant of the classical Mumford-

Shah model. In the first stage an approximate solution $u^*$ to the optimization problem (6.19) is computed. Once $u^*$ is obtained, then in the second stage the segmentation is done by thresholding $u^*$ into different parts. The thresholds can be given by the users or can be obtained automatically using any clustering methods, such as for example the K-means algorithm. As discussed in [22], this allows for a K-phase segmentation ($K \geq 2$) by choosing $(K-1)$ thresholds after $u^*$ is computed in the first stage. In contrast, many multiphase methods require $K$ to be given in advance which implies that if $K$ changes, the minimization problem has to be solved again. Finally, a contour track phase is computed to extract boundary curves delimiting the segmented regions on the manifold.

### 6.3.1 Non-convex penalty functions

In this section we characterize the non-convex penalty functions $\phi(\,\cdot\,;a)$ used in (6.19). We denote the sets of non-negative and positive real numbers as $\mathbb{R}_+ := \{\, t \in \mathbb{R} : t \geq 0\}$ and $\mathbb{R}_+^* := \{\, t \in \mathbb{R} : t > 0\}$, respectively. Analogously to [30, 68, 70], we consider parametrized penalty functions $\phi(t;a) : \mathbb{R}_+ \to \mathbb{R}_+$ such that for any value of the parameter $a \in \mathbb{R}_+$ the following assumptions are satisfied:

A1) $\phi(t;a) \in \mathcal{C}^2(\mathbb{R}_+)$ ($\phi$ twice continuously differentiable in $t$ on $\mathbb{R}_+$)

A2) $\phi'(t;a) > 0 \quad \forall\, t \in \mathbb{R}_+$ ($\phi$ strictly increasing in $t$ on $\mathbb{R}_+$)

A3) $\phi''(t;a) \leq 0 \quad \forall\, t \in \mathbb{R}_+$ ($\phi$ concave in $t$ on $\mathbb{R}_+$)

A4) $\phi(0;a) = 0,\ \phi'(0^+;a) = 1,$ ($\phi, \phi', \phi''$ normalization)
$\inf_{t\in\mathbb{R}_+} \phi''(t;a) = \phi''(0^+;a) = -a\ .$

We notice that the popular $L_p$ quasi-norm with $0 < p < 1$, namely $\phi(t;p) = |t|^p$, satisfies assumptions A1)–A3) but not assumption A4), since $\phi'(0^+;p) = +\infty,\ \phi''(0^+;p) = -\infty$. Thus, such penalty function does not allow for applying the CNC strategy that asks for assumption A4) as mandatory.

We remark that $a$ represents a scalar indicator of the "degree of concavity" of the penalty function $\phi$, thus justifying the name *concavity parameter*. Moreover, we set $\phi(t;0) := t$, such that the $L_1$-norm penalty is recovered as a special case of $\phi(\cdot;a)$ when $a \to 0^+$. We refer to [70, 30] for a detailed discussion on commonly used penalty functions satisfying assumptions A1)–A4) above, such as

$$\phi_{\log}(t;a) = \frac{\log(1+at)}{a}\ ,\ \phi_{\mathrm{rat}}(t;a) = \frac{t}{1+at/2}\ ,\ \phi_{\mathrm{atan}}(t;a) = \frac{\operatorname{atan}\left(\frac{1+2at}{\sqrt{3}}\right) - \frac{\pi}{6}}{a\sqrt{3}/2}\ . \quad (6.20)$$

In Figure 6.3 we show the plots of these penalty functions, for three different values $a \in \{0.2, 2, 4\}$ of the concavity parameter together with the plot of the absolute value $|t|$.

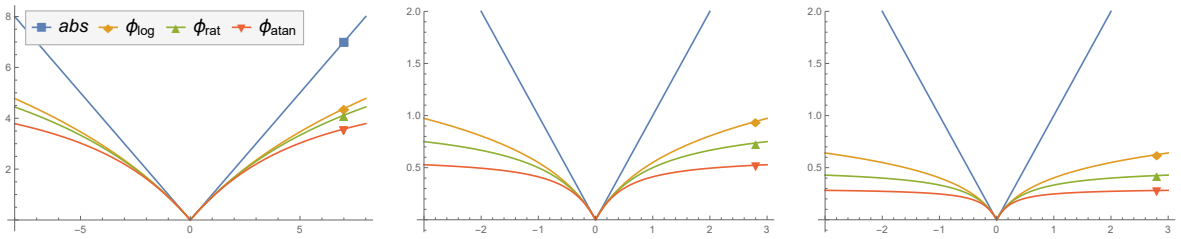



Fig. 6.3 Plots of the penalty functions $\phi_{\log}(t;a)$, $\phi_{\mathrm{rat}}(t;a)$, $\phi_{\mathrm{atan}}(t;a)$ defined in (6.20), for different values of the concavity parameter $a$: $a = 0.2$ (left), $a = 2$ (center), $a = 4$ (right); in comparison to the absolute value function.

### 6.3.2 Convexity Analysis

In this section we seek for a sufficient condition on the parameters $\lambda > 0$, $\eta, a \geq 0$, such that the objective functional $\mathcal{J}(\,\cdot\,;\lambda,\eta,a)$ in (6.19) is strictly convex.

Using the discretization of an embedded manifold $\mathcal{M}$ in $\mathbb{R}^3$ specified in Chapter 4, we can rewrite $\mathcal{J}$ in (6.19) in discrete vector-indexed form as

$$\begin{aligned} \mathcal{J}(u;\lambda,\eta,a) \quad &= \quad \sum_{i=1}^{n} \frac{\lambda}{2}\,(u_i - f_i)^2 \;+\; \sum_{i=1}^{n} \frac{\eta}{2}\Big[\sum_{j\in N_i} w_{ij}^2\,(u_j - u_i)^2\Big] \\ &+ \quad \sum_{i=1}^{n} \phi\left(\sqrt{\sum_{j\in N_i} w_{ij}^2\,(u_j - u_i)^2}\,;\, a_i\right), \end{aligned} \tag{6.21}$$

where the discretization scheme (4.2)-(4.4) has been used in (6.21). In view of the convexity conditions that will be derived, the convexity parameter $a$ in (6.19) is locally defined as $a_i$ in (6.21) for each vertex $X_i$, to take into account the manifold's geometry.

In the following, we give two lemmas which allow us to reduce convexity analysis from the original functional $\mathcal{J}(\,\cdot\,;\lambda,\eta,a)$ of $n$ variables to easier functions $f_v(\,\cdot\,;\lambda,\eta,a)$ and $g_v(\,\cdot\,;\lambda,\eta,a)$ of $v+1$ and $v$ variables, respectively. Then in Theorem 4 we finally give sufficient conditions for strict convexity of $\mathcal{J}$ in (6.21).

**Lemma 1** *Let* $f_v(\,\cdot\,;\lambda,\eta,a)\colon \mathbb{R}^{v+1}\to\mathbb{R}$ *be the function defined by*

$$\begin{aligned} f_v(x_0,x_1,\ldots,x_v;\lambda,\eta,a) \quad &:= \quad \frac{\lambda}{2}\sum_{j=0}^{v}\frac{1}{v_j+1}\,x_j^2 \;+\; \frac{\eta}{2}\sum_{j=1}^{v} w_{0j}^2\,(x_j-x_0)^2 \\ &+ \quad \phi\left(\sqrt{\sum_{j=1}^{v} w_{0j}^2\,(x_j-x_0)^2}\,;\,a\right)\,. \end{aligned} \tag{6.22}$$

*Then, the functional* $\mathcal{J}(\,\cdot\,;\lambda,\eta,a):\mathbb{R}^n\to\mathbb{R}$ *defined in (6.21) is strictly convex if all the functions* $f_{v_i}(\,\cdot\,;\lambda,\eta,a_i):\mathbb{R}^{v_i+1}\to\mathbb{R}$, $i=1,\ldots,n$, *are strictly convex.*

*Proof.* The functional $\mathcal{J}$ in (6.21) can be rewritten in the following equivalent form:

$$\mathcal{J}(u;\lambda,\eta,a) \quad = \quad \mathcal{A}(u)+\sum_{i=1}^{n} f_{v_i}(u_i,u_{N_i(1)},\ldots,u_{N_i(v_i)};\lambda,\eta,a_i) \tag{6.23}$$

where $\mathcal{A}(u)$ is an affine function of $u$ and the function $f_v$ is defined in (6.22). We remark that the generic $i$-th term of the last two sums in (6.21) involves $v_i+1$ different vertices and that, globally, the last two sums involve the generic $i$-th vertex $v_i+1$ times. Since the affine function $\mathcal{A}(u)$ does not affect convexity, we can conclude that the functional $\mathcal{J}$ in (6.23) - or, equivalently, in (6.21) - is strictly convex if all the functions $f_{v_i}$ in (6.23) are strictly convex. □

**Lemma 2** *The function* $f_v(\,\cdot\,;\lambda,\eta,a):\mathbb{R}^{v+1}\to\mathbb{R}$ *defined in (6.22) is strictly convex if the function* $g_v(\,\cdot\,;\lambda,\eta,a):\mathbb{R}^{v}\to\mathbb{R}$ *defined by*

$$g_v(y_1,\ldots,y_v;\lambda,\eta,a) \;=\; \frac{1}{2}\left(\eta+\frac{\lambda}{\kappa}\right)\sum_{j=1}^{v} y_j^2 \;+\; \phi\left(\sqrt{\sum_{j=1}^{v} y_j^2}\,;\,a\right) \tag{6.24}$$

*is strictly convex, where*

$$\kappa=(1+\tilde{v})\lambda_1\,, \tag{6.25}$$

*with* $\tilde{v}:=\max_j v_j$, *and* $\lambda_1$ *is the largest eigenvalue of matrix* $Q\in\mathbb{R}^{(v+1)\times(v+1)}$ *defined as*

$$Q=\begin{bmatrix} \sum w_i^2 & -w_1^2 & -w_2^2 & -w_3^2 & \cdots & -w_v^2 \\ -w_1^2 & w_1^2 & 0 & 0 & \cdots & 0 \\ -w_2^2 & 0 & w_2^2 & 0 & \cdots & 0 \\ -w_3^2 & 0 & 0 & w_3^2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & 0 \\ -w_v^2 & 0 & 0 & 0 & 0 & w_v^2 \end{bmatrix} \tag{6.26}$$

*where* $(w_1, w_2, \ldots, w_v)^T \in \mathbb{R}^v$ *are the weights defined in (4.4) associated with a generic vertex in* $V$ *of valence* $v$*.*

The proof is provided in the Appendix.

**Theorem 4** *Let the function* $\phi(\,\cdot\,; a) : \mathbb{R}_+ \to \mathbb{R}$ *satisfy assumptions A1)–A4) in Section 6.3.1. Then, a sufficient condition for functional* $\mathcal{J}(\,\cdot\,; \lambda, \eta, a)$ *in (6.21) to be strictly convex is that the parameters* $(\lambda, \eta, a)$ *satisfy:*

$$a_i \;<\; \eta + \frac{\lambda}{\kappa_i} \quad \Longleftrightarrow \quad a_i \;=\; \tau_c \left( \eta + \frac{\lambda}{\kappa_i} \right), \quad \tau_c \in [0, 1) \;, \tag{6.27}$$

*for every* $i \in \{1, \ldots, n\}$*, with* $\kappa_i$ *defined in (6.25).*

*Proof.* Based on Lemmas 1–2, the functional $\mathcal{J}(\,\cdot\,; \lambda, \eta, a)$ is strictly convex if all the functions $g_{v_i}, i \in \{1, \ldots, n\}$, defined in (6.24) are strictly convex. Then, based on Proposition 2 in [30], the statement (6.27) follows. □

We conclude by highlighting some properties of functional $\mathcal{J}$ in (6.21).

**Definition 3** *Let* $Z : \mathbb{R}^n \to \mathbb{R}$ *be a (not necessarily smooth) function. Then,* $Z$ *is said to be* $\mu$*-strongly convex iff there exists a constant* $\mu > 0$*, called the modulus of strong convexity of* $Z$*, such that function* $Z(x) - \frac{\mu}{2} \|x\|_2^2$ *is convex.*

**Proposition 5** *Let* $\phi(\,\cdot\,; a) : \mathbb{R}_+ \to \mathbb{R}$ *be a penalty function as defined in (6.20) and let the parameters* $(\lambda, \eta, a)$ *satisfy condition (6.27). Then, the functional* $\mathcal{J}(\,\cdot\,; \lambda, \eta, a)$ *in (6.21) is proper, continuous, bounded from below by zero, coercive and* $\mu$*-strongly convex with* $\mu$ *equal to*

$$\mu \;=\; \lambda \;+\; \min_{i=1,\ldots,n} \{\kappa_i(\eta - a_i)\} \;. \tag{6.28}$$

***Remark:*** *Note that* $\mu$ *is not necessarily maximal.*

The proof is provided in the Appendix.

### 6.3.3 Applying ADMM to the proposed CNC model

In this section, we illustrate the ADMM-based iterative algorithm used to compute minimizers of our discrete objective functional $\mathcal{J}$ in (6.21) in case that parameters $\lambda, \eta, a$ satisfy condition (6.27), such that $\mathcal{J}$ is strictly convex.

Let us first introduce sparse matrix representations of the discrete gradient and divergence operators to speed up the computations. In particular, the discrete gradient

operator is represented by the matrix $M \in \mathbb{R}^{(\hat{v}n)\times n}$, $M = \left(M_1^T, M_2^T, \ldots, M_{\hat{v}}^T\right)^T$, where $\hat{v}$ denotes the maximum valence in $\Omega$ and each sub-matrix $M_j \in \mathbb{R}^{n\times n}$, $j = 1, \ldots, \hat{v}$, represents the linear operator which simultaneously computes the $j$-th directional derivative at all vertices, that is

$$(M_j)_{ik} = \begin{cases} w_{iz} & \text{for } k = z \\ -w_{iz} & \text{for } k = i \\ 0 & \text{otherwise} \end{cases}, \quad z = N_i(j)\,. \tag{6.29}$$

At the $i$-th row of $M_j$, corresponding to the $j$-th directional derivative at the $i$-th vertex, the only non-zero elements are the one on the main diagonal and the one corresponding to the $j$-th neighbour of the $i$-th vertex. Thus, each sub-matrix $M_j$ has exactly $2n$ non-zero elements. In addition, we have $M^T M = \left(M_1^T M_1 + M_2^T M_2 + \ldots + M_{\hat{v}}^T M_{\hat{v}}\right)$. To proceed with ADMM on triangular mesh surfaces, we introduce the auxiliary variable $t \in \mathbb{R}^{\hat{v}n}$, and reformulate problem (6.21) in the equivalent form:

$$\{\, u^*, t^* \} \;\leftarrow\; \arg\min_{u,t} \quad \left\{ \frac{\lambda}{2} \left\| u - f \right\|_2^2 + \sum_{i=1}^{n} \left[ \frac{\eta}{2} \left\| t_i \right\|_2^2 + \phi\Big( \left\| t_i \right\|_2 ; a_i \Big) \right] \right\} \tag{6.30}$$

$$\text{subject to :} \quad t = M u\,, \tag{6.31}$$

where $t_i := \Big( (M_1 u)_i, (M_2 u)_i, \ldots, (M_{v_i} u)_i \Big)^T \in \mathbb{R}^{v_i}$ represents the discrete gradient of $u$ at vertex $X_i$. To solve problem (6.30)–(6.31), we define the augmented Lagrangian functional

$$\begin{aligned} \mathcal{L}(u, t; \rho; \lambda, \eta, a) \quad = \quad & \frac{\lambda}{2} \left\| u - f \right\|_2^2 + \sum_{i=1}^{n} \left[ \frac{\eta}{2} \left\| t_i \right\|_2^2 + \phi\Big( \left\| t_i \right\|_2 ; a_i \Big) \right] \\ & - \left\langle\, \rho, t - M u \,\right\rangle + \frac{\beta}{2} \left\| t - M u \right\|_2^2 \end{aligned} \tag{6.32}$$

where $\beta > 0$ is a scalar penalty parameter and $\rho \in \mathbb{R}^{\hat{v}n}$ is the vector of Lagrange multipliers associated with the linear constraint $t = M u$ in (6.31). We then consider the following saddle-point problem:

$$\begin{aligned} \text{Find} \quad & (u^*, t^*; \rho^*) \;\in\; \mathbb{R}^n \times\; \mathbb{R}^{\hat{v}n} \times\; \mathbb{R}^{\hat{v}n} \\ \text{s.t.} \quad & \mathcal{L}\,(u^*, t^*; \rho; \lambda, \eta, a) \;\leq\; \mathcal{L}\,(u^*, t^*; \rho^*; \lambda, \eta, a) \;\leq\; \mathcal{L}\,(u, t; \rho^*; \lambda, \eta, a) \\ & \forall\, (u, t; \rho) \;\in\; \mathbb{R}^n \times\; \mathbb{R}^{\hat{v}n} \times\; \mathbb{R}^{\hat{v}n}\,. \end{aligned} \tag{6.33}$$

Given the previously computed (or initialized for $k = 0$) vectors $u^{(k)}$ and $\rho^{(k)}$, the $k$-th iteration of the proposed ADMM-based iterative scheme applied to the solution of the saddle-point problem (6.32)–(6.33) reads as follows:

$$t^{(k+1)} \leftarrow \arg\min_{t\in\mathbb{R}^{\hat{v}n}} \mathcal{L}(u^{(k)}, t; \rho^{(k)}; \lambda, \eta, a) \tag{6.34}$$

$$u^{(k+1)} \leftarrow \arg\min_{u\in\mathbb{R}^{n}} \mathcal{L}(u, t^{(k+1)}; \rho^{(k)}; \lambda, \eta, a) \tag{6.35}$$

$$\rho^{(k+1)} \leftarrow \rho^{(k)} - \beta\left(t^{(k+1)} - Mu^{(k+1)}\right). \tag{6.36}$$

In the following we show in detail how to solve the two minimization sub-problems (6.34) and (6.35) for the primal variables $t$ and $u$, respectively.

Although the minimization sub-problems are all strictly convex and admit a unique solution, convergence of the overall ADMM algorithm is clearly not guaranteed.

### Solving the sub-problem for t

The minimization sub-problem for $t$ in (6.34) can be rewritten as follows:

$$t^{(k+1)} \leftarrow \arg\min_{t\in\mathbb{R}^{\hat{v}n}} \left\{\sum_{i=1}^{n}\left[\frac{\eta}{2}\|t_i\|_2^2 + \phi\Big(\|t_i\|_2; a_i\Big)\right] + \frac{\beta}{2}\left\|t - r^{(k+1)}\right\|_2^2\right\} \tag{6.37}$$

where constant terms have been omitted and where the vector $r^{(k+1)} \in \mathbb{R}^{\hat{v}n}$ is constant with respect to the optimization variable $t$ and is given by:

$$r^{(k+1)} = Mu^{(k)} + \frac{1}{\beta}\rho^{(k)}. \tag{6.38}$$

The minimization problem in (6.37) rewritten in component-wise (vertex-by-vertex) form, is equivalent to the following $n$ independent lower dimensional problems:

$$t_i^{(k+1)} \leftarrow \arg\min_{t_i\in\mathbb{R}^{v_i}} \left\{\phi\Big(\|t_i\|_2; a_i\Big) + \frac{\eta}{2}\|t_i\|_2^2 + \frac{\beta}{2}\left\|t_i - r_i^{(k+1)}\right\|_2^2\right\}, \tag{6.39}$$

with $i = 1, \ldots, n$, $r_i^{(k+1)} := \left(Mu^{(k)}\right)_i + \left(\rho^{(k)}\right)_i / \beta$ and $\left(Mu^{(k)}\right)_i$, $\left(\rho^{(k)}\right)_i \in \mathbb{R}^{v_i}$ denote the discrete gradient and the associated vector of Lagrange multipliers at vertex $X_i$, respectively, with valence $v_i$.

Since we are imposing that condition (6.27) is satisfied, such that the original functional $\mathcal{J}(u; \lambda, \eta, a)$ in (6.19) is strictly convex, we aim at avoiding non-convexity of the ADMM sub-problems (6.39). In the first part of Proposition 6 below, we give necessary and sufficient conditions for strict convexity of the cost functions in (6.39). In particular,

based on (6.41)–(6.42), we can state that the problems in (6.39) are strictly convex if and only if the following conditions hold:

$$\beta > \max_{i=1,\dots,n} a_i - \eta \,. \tag{6.40}$$

In case conditions in (6.40) are satisfied, the unique solutions of the strictly convex problems in (6.39) can be obtained by the soft-thresholding operator defined in (6.44)–(6.45). We remark that the nonlinear equation in (6.45) can be solved up to a sufficient accuracy by very few steps of the iterative Newton method.

**Proposition 6** *Let $\eta, a \geq 0$, $\beta > 0$ and $r \in \mathbb{R}^v$ be given constants, and let $\phi(\,\cdot\,; a) : \mathbb{R}_+ \to \mathbb{R}$ be a function satisfying assumptions A1)–A4) in Section 2. Then, the function*

$$\theta(x) := \phi\left(\|x\|_2; a\right) + \frac{\eta}{2}\|x\|_2^2 + \frac{\beta}{2}\|x - r\|_2^2 \,, \quad x \in \mathbb{R}^v \,, \tag{6.41}$$

*is strictly convex if and only if the following condition holds:*

$$\beta > a - \eta \,. \tag{6.42}$$

*Moreover, in case that (6.42) holds, the strictly convex minimization problem*

$$\arg\min_{x \in \mathbb{R}^v} \theta(x) \tag{6.43}$$

*admits the unique solution $x^* \in \mathbb{R}^v$ given by the following shrinkage operator:*

$$x^* = \xi^* r \,, \quad \text{with } \xi^* \in [0, 1[, \text{ and} \tag{6.44}$$

$$\begin{aligned} &\text{a)} \quad \xi^* = 0 && \text{if } \|r\|_2 \leq \frac{1}{\beta} \\ &\text{b)} \quad \xi^* \in \,]0, 1[ \text{ unique solution of :} \\ &\qquad \phi'\left(\|r\|_2\,\xi; a\right) + \|r\|_2\Big((\eta + \beta)\xi - \beta\Big) = 0 && \text{otherwise.} \end{aligned} \tag{6.45}$$

The proof is provided in the Appendix.

**Solving the sub-problem for u**

The minimization sub-problem for $u$ in (6.35) can be rewritten as follows:

$$u^{(k+1)} \leftarrow \arg\min_{u\in\mathbb{R}^n} \left\{ \frac{\lambda}{2} \|u - f\|_2^2 + \left\langle \rho^{(k)}, Mu \right\rangle + \frac{\beta}{2} \|t^{(k+1)} - Mu\|_2^2 \right\} \tag{6.46}$$

where constants have been omitted. The quadratic minimization problem (6.46) has first-order optimality conditions which lead to the following linear system:

$$\left(I + \frac{\beta}{\lambda} M^T M\right) u \;=\; f \;+\; \frac{\beta}{\lambda} M^T \left(t^{(k+1)} - \frac{1}{\beta}\rho^{(k)}\right) . \tag{6.47}$$

Since the product $M^T M$ is symmetric and the ratio $\frac{\beta}{\lambda}$ is positive, the $n \times n$ coefficient matrix of the linear system (6.47) is symmetric semi pos. def. and highly sparse. Hence, (6.47) admits a unique solution obtained very efficiently by the iterative (preconditioned) conjugate gradient method.

## 6.4 Experiments on the Partitioning Algorithms

### 6.4.1 Data Set and Hardware Specifications

We tested the proposed algorithms on a collection of point clouds and meshes downloaded from the data repository website `http://segeval.cs.princeton.edu`, [31]. The chosen dataset represents geometric models with different characteristics in terms of details, "thickness", and level of refinement, and presents a medium dense vertex distribution. In general, the benchmark [31] contains irregular triangulated meshes which can be divided into few categories representing living things (human (+ hand), mammals, birds, insect (ant), fishes (+ octopus)), daily objects (bust, cup, glasses, chair, table, teddy bear, vase) and engineering-oriented objects (air planes, bearings, mechanical object and pliers). In Table 6.1 we report the data sets we have used in this work, listing the number of vertices and triangles respectively in the second and third column of 6.1. We have used also additional meshes or synthetic data for which we knew the ground truths. These are also reported in the third column of Table 6.1.

In the reported figures the SDF values are visualized using false colours superimposed onto the object. In particular, we made use of the HSV colour model and of the colour transfer function offered by software ParaView (`http://www.paraview.org/`). The hue value $u(X_i)$ at the vertex $X_i$ is assigned linearly to the interval $(min(f(V)),$

Table 6.1 Data sets.

| Data set | $\|V\|$ | $\|T\|$ | Data set | $\|V\|$ | $\|T\|$ | Data set | $\|V\|$ | $\|T\|$ |
|---|---|---|---|---|---|---|---|---|
| ant | 7038 | 14072 | horse_2 | 8078 | 16152 | teddy | 9548 | 19092 |
| armadillo | 25193 | 50382 | human_1 | 9508 | 19012 | teddy_2 | 12831 | 25658 |
| bird_1 | 6475 | 12946 | human_2 | 15385 | 30766 | vase_1 | 14859 | 29734 |
| bird_2 | 8946 | 17888 | mech_1 | 8759 | 17514 | vase_2 | 14476 | 28952 |
| bust | 25467 | 50930 | mech_2 | 10400 | 20796 | vase_3 | 10637 | 21274 |
| camel | 9757 | 19510 | mech_3 | 1512 | 3020 | wolf | 4712 | 9420 |
| chair | 10121 | 20242 | mech_4 | 14872 | 29764 | | | |
| cup | 15037 | 30074 | mech_5 | 14956 | 29924 | blocks | 6146 | 12288 |
| cup_2 | 15127 | 30254 | octopus_1 | 7251 | 14498 | blocks_2 | 105990 | 208896 |
| dolphin | 7573 | 15142 | octopus_2 | 1010 | 2016 | ellipsoid | 16386 | 32768 |
| fawn | 3911 | 7818 | octopus_3 | 243 | 482 | fertility | 19994 | 40000 |
| fish | 6656 | 13308 | octopus_4 | 1343 | 2682 | hand_im | 26422 | 52840 |
| fish_2 | 5121 | 10238 | octopus_5 | 5944 | 11888 | horse_im | 129218 | 258432 |
| giraffe | 9239 | 18474 | plane | 7470 | 14936 | plate | 85709 | 171356 |
| glasses | 7407 | 14810 | pliers | 3906 | 7808 | sphere | 16386 | 32768 |
| hand | 6607 | 13210 | pliers_2 | 5110 | 10216 | torus | 7200 | 14400 |
| horse | 7268 | 14532 | table | 14587 | 29170 | vase_im | 52028 | 104056 |

$max(f(V))$). The software ParaView, and its VTK reader, was also used to visualize and to produce all figures reported in this work and has been taken from our published work [58, 59, 56, 55]. The only exception is illustrated in Fig. 5.9 where we compare our SDF results with the ones obtained using original algorithm by Shapira et al. [108]. Figure 5.9 depicts screenshots of the application downloaded from the author's website `http://www.liors.net/shape-diameter-function`; and Figures 6.14, 6.15 depict figures taken from [130]. The applications use slightly different rendering and lighting procedures, nevertheless, the results are easily comparable.

The performance has been evaluated by experimental tests run on Intel®Core™i7-4720HQ Quad-Core 2.6 GHz machine, with 12 GB/RAM and Nvidia GeForce GTX 860M graphics card in a Windows OS. The codes were executed without any additional machine support, e.g. parallelization, GPU support, register usage.

The code for results presented in Sections 5.1.1 and 6.4.2 was written in C++ language using EIGEN mathematical library. In the latter case, to compute the solutions of the large sparse eigenproblems required for multi-channel segmentation, we used the wrapper EIGEN/Spectra (`http://yixuan.cos.name/spectra/`) which provides an efficient implementation of the Arnoldi method. For a more pleasing visualization, in a

final stage of the object partitioning we could expect the smoothing of the boundaries between parts. However, in the preliminary computations this process is sometimes omitted so as not to distort the raw results obtained from the application of the variational partitioning model.

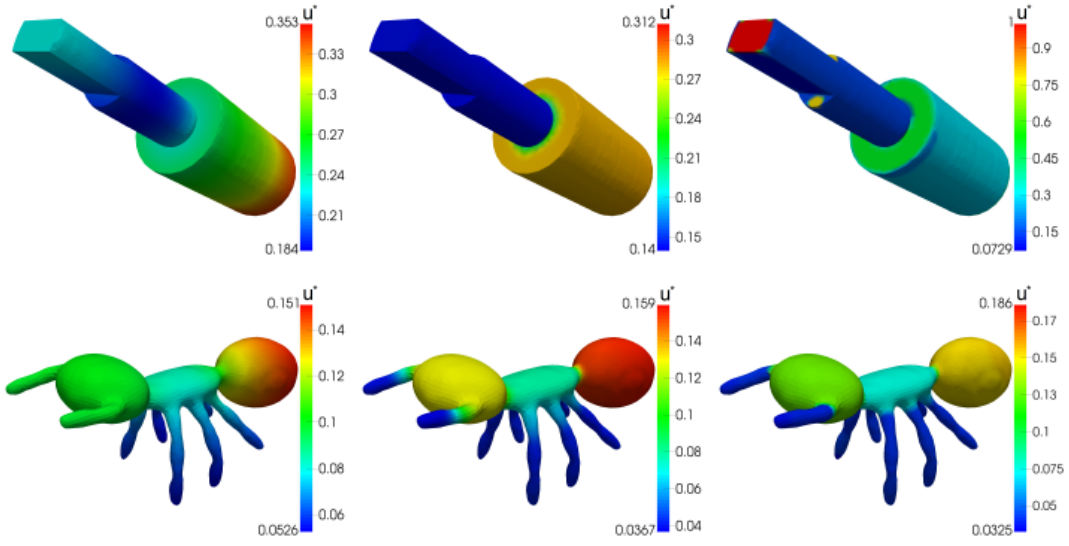
The other algorithms related to the results presented in Section 6.4.3 and 6.4.4 are written in MATLAB, also executed without any additional machine support.

### 6.4.2 Experimental results of SMCMR Framework

In this section we describe the experimental results which demonstrate the performance of our segmentation approach, described in Section 6.1 and summarized in Algorithm 3 (SMCMR), both in case of single channel input function, $f \in \mathbb{R}$, in regime of piecewise constant segmentation, and in case of multi-channel piecewise smooth segmentation.

In the examples illustrated in this section, we did not apply any post-process (smoothing, etc.) to the boundaries between the segmented parts in order to not alter the results computed by the application of the variational partitioning model.

#### Single-channel partitioning based on SDF




(a) (b) (c)

Fig. 6.4 Effect of the parameter $p$ on the STEP 1 in **Algorithm 3**. (a) $p = 2$, (b) $p = 1$, (c) $p = 0.8$.

In this example we aim to decompose the surface boundary of an object into meaningful parts using the shape diameter values as shape attribute to distinguish the salient parts.

Table 6.2 Timing results for single-channel (SDF) partitioning in seconds: the computational time of one iteration (Iter), and total computational time using the tolerances $\epsilon = 10^{-2}$ and $\epsilon = 10^{-4}$.

| Data set | $\lvert V\rvert$ | Iter | $\epsilon = 10^{-2}$ | $\epsilon = 10^{-4}$ |
|---|---|---|---|---|
| `ant` | 7038 | 0.008 | 0.025 | 1.832 |
| `armadillo` | 25193 | 0.048 | 0.209 | 12.718 |
| `blocks` | 6146 | 0.008 | 0.015 | 0.328 |
| `bird_2` | 8946 | 0.014 | 0.090 | 4.258 |
| `camel` | 9757 | 0.013 | 0.049 | 2.793 |
| `dolphin` | 7573 | 0.011 | 0.040 | 4.174 |
| `mech_2` | 10400 | 0.012 | 0.022 | 0.389 |
| `mech_3` | 1512 | 0.002 | 0.004 | 0.086 |
| `octopus_1` | 7251 | 0.009 | 0.029 | 1.740 |
| `octopus_3` | 243 | 0.001 | 0.002 | 0.029 |
| `pliers` | 3906 | 0.004 | 0.014 | 1.025 |
| `wolf` | 4712 | 0.006 | 0.017 | 1.645 |

Therefore we expect the solution to be composed of homogeneous regions surrounded by closed contours which separates parts with significative different thicknesses. We applied Algorithm 3, with $\beta = 0$ and the input data $f \in \mathbb{R}$ which was the SDF map computed by the dynamic algorithm described in Section 5.1. The result is a piecewise constant approximation of the given SDF initial data enforcing sparsity in the gradient magnitude of the solution. The model data of the mesh samples from the data repository reported for this single-channel partitioning example are illustrated in Table 6.2.

The decomposition results strongly depend on the parameter $p$ which forces the sparsity in the gradient of the solution $u^*$. The effects of the parameter $p$ can be observed, for the `ant` and `mech_1` data sets, in Fig. 6.4 where the colours (from red (large) to blue (small)) indicate the value of the solution $u^*$ from STEP 1 of **Algorithm 3**. In this experiment, we fixed the value of $\alpha = 1$ to highlight the effect of parameter $p$, however, similar results can be obtained for different $\alpha$ values. When $p > 1$ the solution of the optimization process behaves like a smoothing flow, as illustrated in Fig. 6.4(a), thus destroying the boundaries between parts. This effect is easily justified in terms of the p-Laplacian operator which, for $p = 2$, turns to the classical Laplace-Beltrami operator $\Delta_2$. For the choice of $p > 1$, the tuning of $\alpha$ parameter does not help to improve the result. When $p \leq 1$, as shown in Fig. 6.4(b) and (c), the regularization term induces the sparsity of the $u^*$ function leading to cleaner, straightforward partitioning clues for the underlying object.

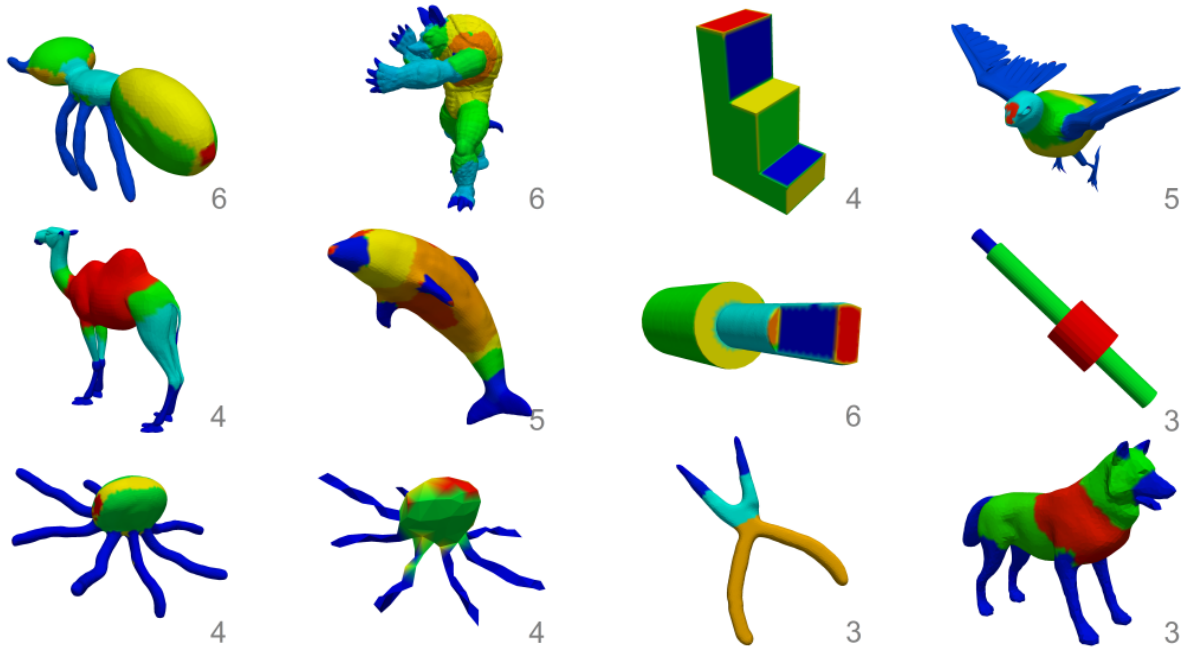
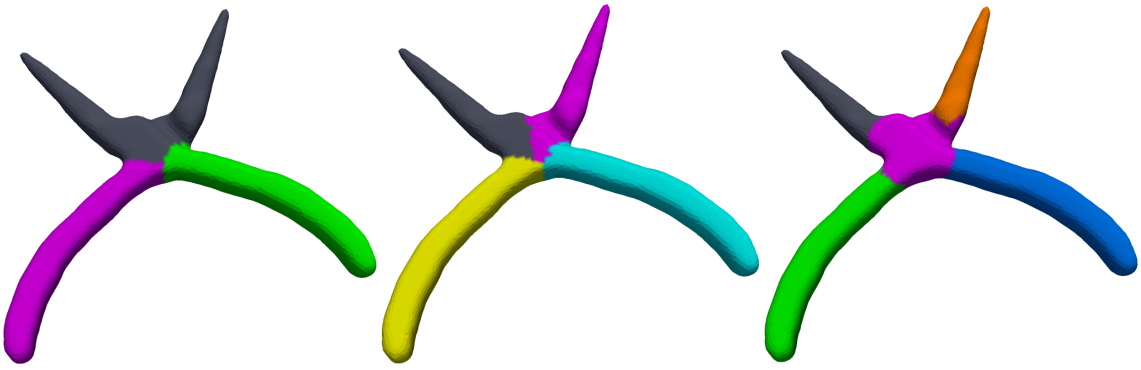



Fig. 6.5 Examples of single-channel partitioning based on SDF into patches with similar thickness.

In Fig. 6.5 a sample set of objects partitioned into patches of different thickness is shown. The results were obtained by applying Algorithm 3 with $d = 1$, $p = 0.8$, tolerance $\epsilon = 10^{-4}$ and $\alpha = 1$. At the bottom right of each object we report the value of $K$, which in this type of partitioning associates to the number of clusters having similar thickness. The associated computational times are reported in Table 6.2: for one iteration $k$ (third column), the total time required by STEP 1 with $\epsilon = 10^{-4}$ and $\epsilon = 10^{-2}$ are reported in the fourth and fifth columns, respectively. Although we used the more stringent $\epsilon$ tolerance in the shown examples in Fig. 6.5, we noticed that for many input shapes the results for $\epsilon \leq 10^{-2}$ are very favourable, too. It is also worth noting that it is not necessary to require large scale models to generate good results. In fact our algorithm generates acceptable segmentation results independently from the resolution of the meshes, as illustrated for the two `octopus` meshes in Fig. 6.5 (last row – left) which present different resolutions.

### Multi-channel partitioning based on spectral analysis

For the spectral partitioning, which aims to simulate the human being decomposition, the eigen-decomposition of the affinity matrix described in Section 5.2 is preliminary applied, obtaining 15 non-constant eigenvectors for each object in the repository data set. The affinity matrix weights (5.11) were computed using $\sigma = 0.5$ for every object. The



(a) (b) (c)

Fig. 6.6 Multi-channel partitioning of the `pliers_2` mesh into $K$ parts: $K = 3$ (a), $K = 4$ (b), $K = 5$ (c)

.

dimension $d$ of the multi-channel function $f$ used as input of the Algorithm SMCMR can be less or equal to the number of eigenvectors, that is $d \leq 15$. The usual choice is to take the first $d$ significant(well shape-describing) ones. Our choice is showed in the third column of Table 6.3. Finally, the peculiarity of our algorithm allows us to consider a number of partitions $K$ independent on $d$. An example of this benefit is illustrated in Fig. 6.6. First STEP 1 of Algorithm 3 is applied using $d = 3$ channels, by considering the first three eigenfunctions among the 15 computed ones. Then STEP 2 is recomputed for $K = 3$ (Fig. 6.6 (a)), $K = 4$ (Fig. 6.6 (b)), and $K = 5$ (Fig. 6.6 (c)).

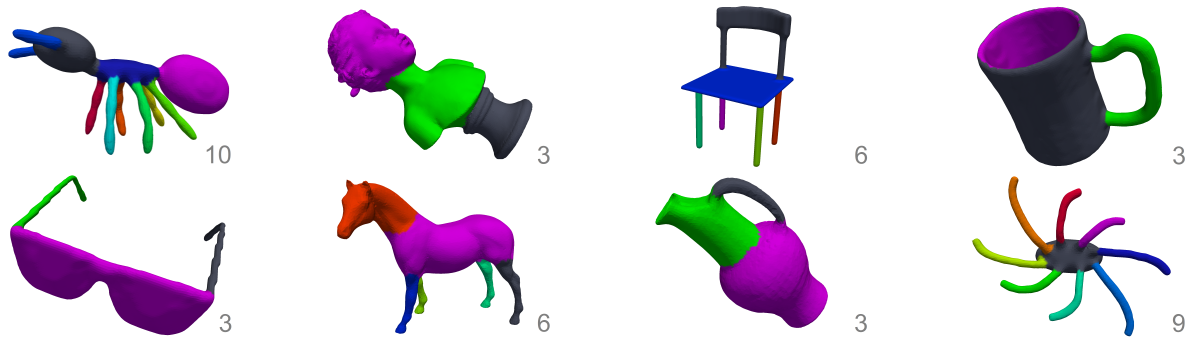



Fig. 6.7 Examples of multi-channel partitioning into patches simulating human being segmentation.

The effectiveness of the proposed Algorithm 3 to partition the surface patches is shown in Figure 6.7 for a selected set of objects from the repository. At the bottom right of each object, we report the $K$ value – number of partitions produced. Details on the model sizes ($|V|$), the number of channels considered ($d$), and the computational times for these objects are reported in Table 6.3. In particular with *Spectra* we denote the

Table 6.3 Timing results for multi-channel partitioning in seconds: the computational time for the eigen-decomposition (*Spectra*) and the overall computational time (*Time*) for the ($d$)-channel Algorithm 3.

| Data set | $\lvert V\rvert$ | $d$ | Spectra ($s$) | Time ($s$) |
|---|---|---|---|---|
| `ant` | 7038 | 8 | 0.406 | 5.015 |
| `bust` | 25467 | 2 | 3.639 | 5.156 |
| `chair` | 14372 | 5 | 1.247 | 6.787 |
| `cup_2` | 15127 | 2 | 9.999 | 2.256 |
| `glasses` | 7407 | 2 | 0.455 | 0.593 |
| `horse_2` | 8078 | 6 | 0.451 | 5.084 |
| `octopus_5` | 5944 | 8 | 0.310 | 4.856 |
| `pliers_2` | 5110 | 3 | 0.293 | 1.654 |
| `vase` | 10637 | 3 | 0.591 | 3.235 |

timing for eigen-decomposition to compute the first 15 non-constant eigenvectors, while with *Time* we report the overall computational time for running the $d$-channel Algorithm SMCMR in seconds.

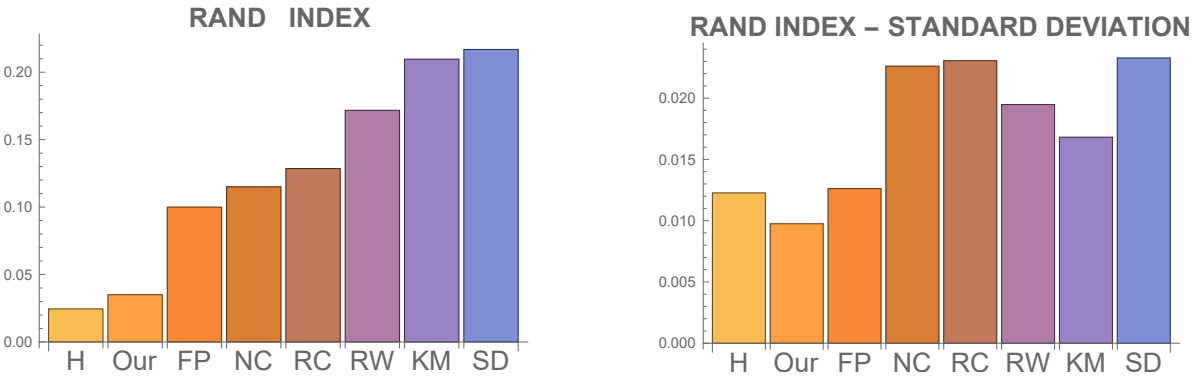



Fig. 6.8 Averaged dissimilarity ($1 - RI$) for the comparison of Algorithm 3 ("Our") with other methods. Left: comparison w.r.t the human-generated segmentation. Right: standard deviation from the average. The lower the better.

We compared the results of Algorithm 3 with other popular segmentation methods and with the human-generated segmentation, both provided by the benchmark in [31]. Namely, the methods considered are: fitting primitives ("FP"), normalized cuts ("NC"), randomized cuts ("RC"), random walks ("RW"), K-means ("KM"), shape diameter function ("SD").

For the choice of a unifying comparison measure, we considered the *Rand Index* metric, denoted by RI, which measures the likelihood that a pair of faces are either in

the same segment in two segmentations, or in different segments in both segmentations. If we denote $S_1$ and $S_2$ as two segmentations, $s_i^1$ and $s_i^2$ as the segment IDs of face $i$ in $S_1$ and $S_2$, and M as the number of faces in the polygonal mesh, $C_{ij} = 1$ iff $s_i^1 = s_j^1$ and $P_{ij} = 1$ iff $s_i^2 = s_j^2$, then we can define Rand Index as:

$$RI(S_1, S_2) = \binom{M}{2}^{-1} \sum_{i,j,i<j} \left[C_{ij}P_{ij} + (1 - C_{ij})\,(1 - P_{ij})\right].$$

This measure reflects similarity between two segmentations, i.e. $C_{ij}P_{ij} = 1$ indicated that faces $i$ and $j$ have the same ID in both segmentations, and $(1 - C_{ij})\,(1 - P_{ij})$ indicates that faces $i$ and $j$ have different IDs in the segmentations being compared.

As well as in [31], we report in Fig. 6.8 the estimates $1 - RI$ to show rather dissimilarities from the human-based segmentation averaging the results for each object in the repository data set. Therefore, the lower bars represent better results. The chart presented was computed as the average over $RI$ for each segmentation method mentioned w.r.t. the human segmentation. In Fig. 6.8 the bar labelled "H" was computed as the average of the human-generated segmentations, in order to track the dissimilarities over human-produced results. We also report in Fig. 6.8 (right) the standard deviation from the average. We can conclude that Algorithm 3 is quite consistent comparing to the other methods and the human-generated segmentations.

### 6.4.3 Experimental Results of Partitioning Driven by $L_p$ Compressed Modes

In this section we describe the experimental results which demonstrate the performance of **Algorithm 4** ($L_p$CM Mesh Partitioning). In particular, we first evaluate the performance of Step 1 for the computation of the $L_p$ Compressed Modes $\Psi \in \mathbb{R}^{n\times N}$, then we illustrate the results of Step 2 and Step 3 for part-/patch-based partitioning, respectively.

In the examples illustrated we applied a post-process smoothing to the boundaries between the segmented parts $\{S_k\}_{k=1}^N$ by projecting the boundary vertices onto the cubic spline obtained by least-squares approximation.

#### STEP 1: Computing the $L_p$CMs

The two strategies Step 1a and Step 1b in **Algorithm 4**, described in section 6.2, generate the basis functions $\Psi$.

In all the experiments we used a randomized matrix as initial iterate $\Psi^{(0)}$ for the ADMM computation of (5.23), and we terminated the ADMM iterations as soon as the

relative change between two successive iterates satisfies

$$err_{\Psi} = \frac{\|\Psi^{(k)} - \Psi^{(k-1)}\|_F}{\|\Psi^{(k-1)}\|_F} < 10^{-3}. \tag{6.48}$$

As already observed in [88], where the $L_1$ penalty term is used, different runs converge to the same set of basis functions, although their ordering might be different. In our experiments the $p$ values were tested in the range $[0.5, 0.8]$. However, since small $p$ values affect mainly the efficiency, we decided to set the sparsity parameter $p = 0.8$ for all the examples reported in this section.

Figure 6.9 illustrates how Step 1a works when the parameter value $\mu$ is fixed, $\mu = 300$. At the first iteration, only two initial quasi-eigenfunctions are computed with the given $\mu$. The control of the local support volume resulted in localizing two legs of the `horse` mesh, leaving the rest uncovered (highlighted in magenta at the end of the first row). In the second iteration (second row), the space dimension is enlarged ($N = 3$), resulting in optimization of $\Psi^{n\times 3}$. The support of the third function $\psi_3$ shrinks the uncovered area under the head and neck, leaving just two legs and part of the `horse`'s body uncovered. The algorithm terminates after five iterations, enlarging the space up to six functions $\psi_1, \ldots, \psi_6$ and leaving no more vertices of $M$ uncovered. The result of the last iteration is depicted in the bottom row of Figure 6.9. Notice that over iterations, the corresponding functions describing the same parts of the mesh retain their order in the set $\Psi$. This, in general, does hold, but the order may change when re-running the whole algorithm due to the randomized initialization of $\Psi$.

Step 1*b* iteratively recomputes a given number $N$ of basis functions increasing the value of the $\mu$ parameter, thus enlarging the local support at each iteration, until all the vertices of $M$ are covered by at least one function $\psi$.

In Figure 6.10 we show the enlargement of the support of $\psi_5 \in \Psi$ for increasing values of $\mu$ and a fixed basis dimension $N = 5$. The initial $\mu = 8$ is increased by a factor 4 at each iteration. From left to right, the result is shown for $\mu = 8$, $\mu = 32$, $\mu = 128$ and $\mu = 512$.

In order to further demonstrate how the $L_p$CMs localize the details much better than the Laplacian eigenvectors, we consider a synthetic example of an ellipsoid with a growing bump. The ellipsoid's principal semi-axes are $\{2.5, 1.5, 1.5\}$ long and it was approximated by triangulated mesh of $|V| = 16386$ vertices and $|T| = 32768$ triangles. In the top row of Fig. 6.11 we report the first five non-constant eigenvectors of LBO corresponding to the first five non-zero eigenvalues obtained by solving the generalized eigenvalue problem (5.13). The eigenvectors present global support and neither the first

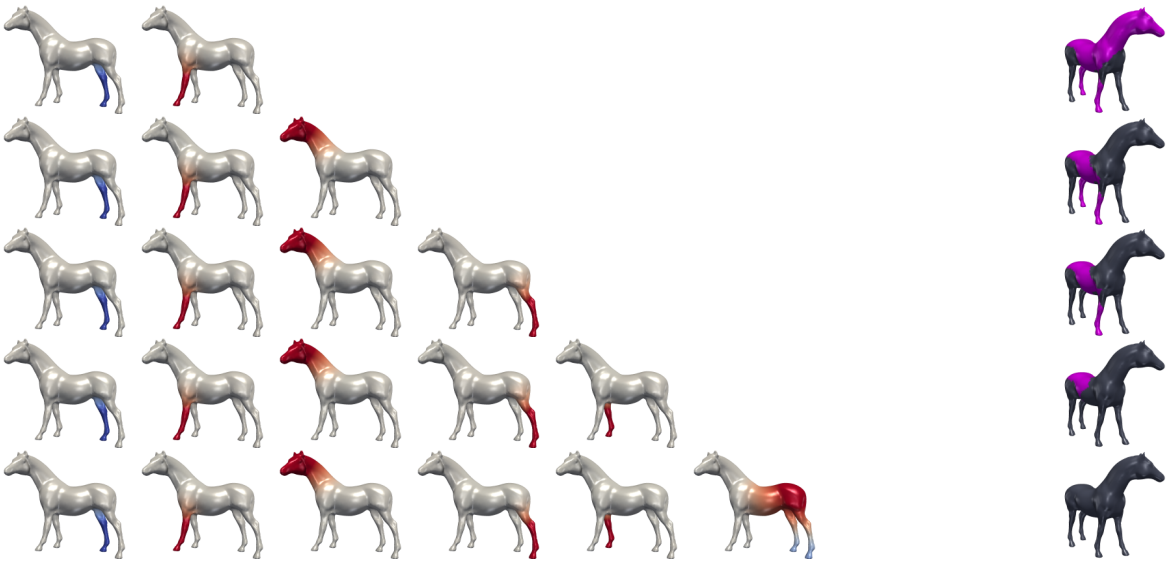


Fig. 6.9 $L_p$CMs generated at each iteration of Step 1a in Algorithm 4, on the `horse_2` mesh.

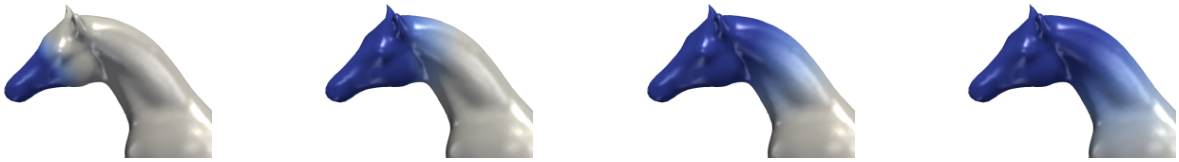


Fig. 6.10 $L_p$CM $\psi_5$ of $\Psi$ generated for 4 iterations of Step 1b in Algorithm 4, on the `horse_2` mesh. From left to right: enlarging of the support obtained by increasing the parameter $\mu$ for a fixed number of basis functions.

five nor the rest of the eigenvectors, which are not illustrated here for space constraints, are able to localize the bump. In the bottom rows of Fig. 6.11 we show the first five $L_p$CMs for $p = 0.8$ and $\mu = 125$, for different bump dimensions. In the first and last and row of Fig. 6.11 the bump dimensions correspond. The compact support $L_p$CMs which localize the bumps are highlighted in red boxes.

We conclude this example presenting an empirical investigation on the numerical convergence of the proposed ADMM-based minimization scheme.

In our formulation (5.23), we deal with a non-convex orthogonality constraint and non-convex penalty term, i.e. the sparsity-inducing $L_p$ norm. Therefore, the convergence to an optimal solution in the global sense is not guaranteed and we assume that the algorithm converges to at least a local minima, which is still a sufficient result for our application.

At that aim, we investigated the empirical convergence via the relative change of the primal variables, and, following [18], the squared primal residual norm $\|\mathbf{r}\|_2^2$ which,

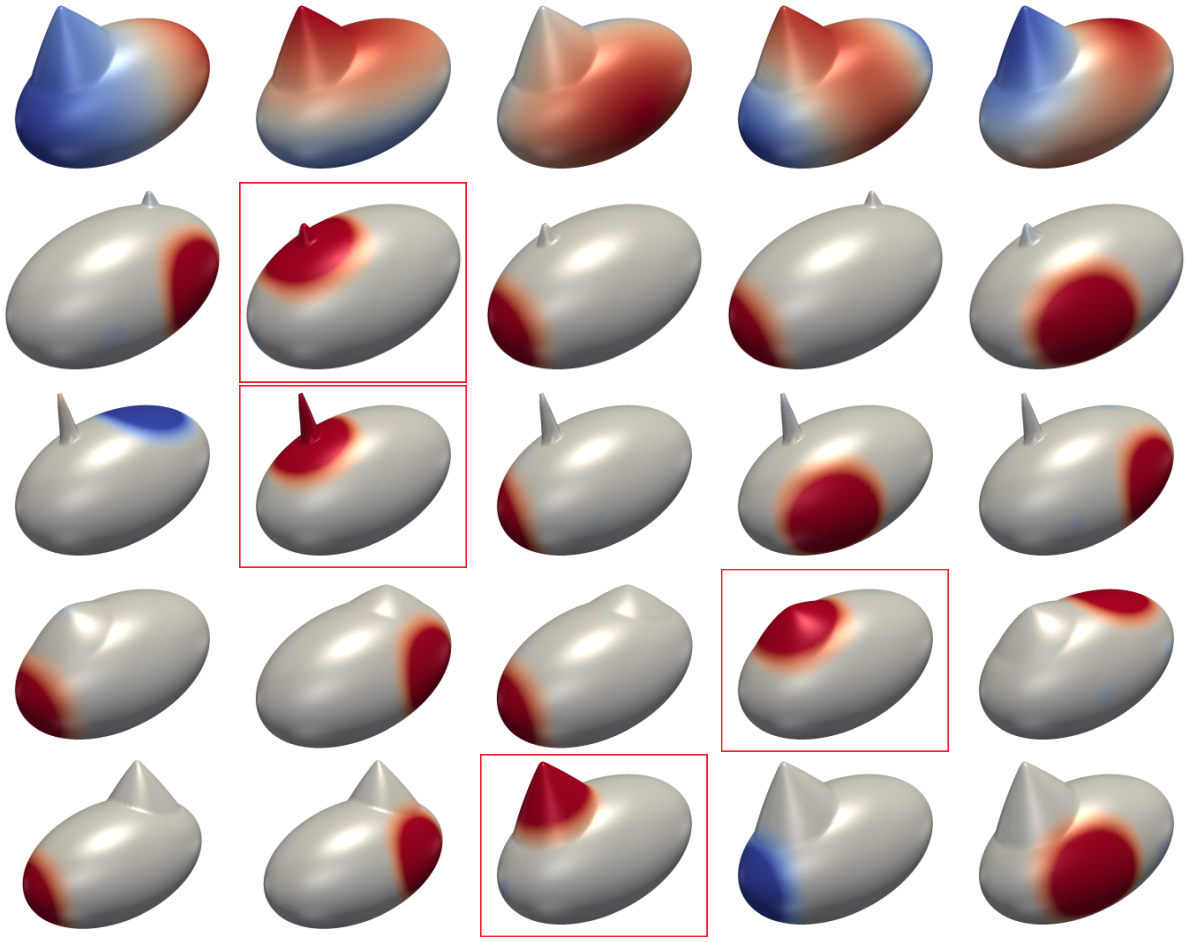


Fig. 6.11 Comparison of the first five MHB functions and $L_p$CMs in case of an ellipsoid with a bump. Top row: first 5 non-constant eigenvectors of LBO. Bottom rows: first 5 $L_p$CMs for different bump dimensions.

according to our implementation, is defined as

$$\|\mathbf{r}^{(k)}\|_2^2 = \|\Psi^{(k)} - S^{(k)}\|_F^2 + \|\Psi^{(k)} - E^{(k)}\|_F^2 \,.$$

By the way of illustration, in Figure 6.12 we report the convergence plots concerning some models used for these examples. The plots in Fig. 6.12(left) show that the relative errors $err_\Psi$ defined in (6.48) on the ADMM iterates $\Psi^{(k)}$ computed by Step 1 in Algorithm 4, converge to some limit, which indeed indicated convergence of the proposed method (at least to local minimizers), whereas the plots in Fig. 6.12(right) demonstrate that the primal residual norms $\|\mathbf{r}^{(k)}\|_2^2$ reduce.

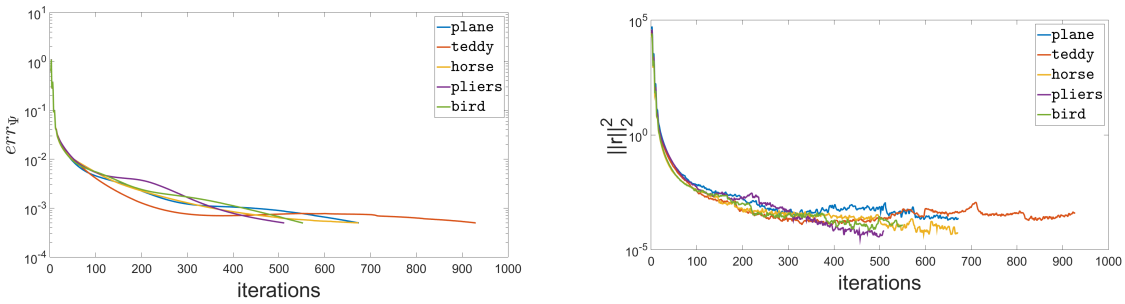



Fig. 6.12 Relative change of primal variable $\Psi$, $err_\Psi$, (left) and primal residual norm $\|\mathbf{r}_k\|_2^2$ (right) in terms of ADMM iterations.

### STEP 2: Mesh Segmentation

In Step 2 we apply the region growing algorithm detailed in Section 6.2 to obtain the partition $S = \{S_k\}_{k=1}^N$ which represents a decomposition of the mesh into its salient parts. Several examples of mesh segmentation are shown in Figure 6.13. The number of partitions produced ($N$) is reported on the bottom right of each segmented object.

Details of the datasets are given in Table 6.4. In particular, for each mesh, we report the number of partitions $K$ ($N$), the value of $\mu$ automatically computed by Step 1b, the time in seconds to obtain the $L_p$CM basis of dimension $N$ in Step 1b, and the time for the mesh segmentation procedure in Step 2 of **Algorithm 4**.

It is worth mentioning that our segmentation procedure is naive compared with many other spectral segmentation approaches proposed in literature, which are enriched by many heuristic strategies based on curvature criteria or edge detection, which, however, can be easily applied also to our basic algorithm. Nevertheless, the obtained results enhance the good properties of our proposal.

The model `fish_2`, illustrated in Fig. 6.14, is considered a particularly difficult challenge since its featured parts (fins, head, tail) are smoothly joined with the rest of the body thus presenting weak boundary strength but good degree of protrusion. In Fig. 6.14 we show a comparison between our $L_p$CM basis (top left) and the eigenfunctions computed by the truncated spectral decomposition used in [130] (bottom left). The latter is considered the state-of-the-art among the variational methods using spectral analysis.

The salient parts are nicely identified by the $L_p$CMs using only $N = 8$ functions, mimicking the human driven segmentation shown in Fig. 6.14 (right). In [130] the authors claim that even for higher space dimensions their method was not able to localize the salient parts. We notice that in Fig. 6.14 (top row, left) the fish meshes for $\psi_2$ and $\psi_8$ are visualized upside-down to better show which fins are localized by these supporting functions.

Table 6.4 Performance of the mesh segmentation algorithm 4.

| Data set | $\|V\|$ | $\|T\|$ | $K$ | $\mu$ | STEP 1 (s) | STEP 2 (s) |
|---|---|---|---|---|---|---|
| `ant` | 7038 | 14072 | 9 | 150 | 9.69 | 4.79 |
| `armadillo` | 25319 | 50542 | 12 | 140 | 47.26 | 13.23 |
| `bird` | 6475 | 12946 | 4 | 300 | 7.32 | 4.54 |
| `dolphin` | 7573 | 15142 | 7 | 150 | 9.40 | 3.48 |
| `fawn` | 3911 | 7818 | 6 | 150 | 4.10 | 2.54 |
| `fertility` | 19994 | 40000 | 7 | 300 | 26.56 | 11.57 |
| `fish_2` | 5121 | 10238 | 8 | 130 | 7.45 | 2.11 |
| `giraffe` | 9239 | 18474 | 13 | 130 | 14.55 | 3.52 |
| `glasses` | 7407 | 14810 | 6 | 150 | 7.75 | 3.36 |
| `hand` | 6607 | 13210 | 8 | 150 | 8.27 | 3.32 |
| `horse_2` | 8078 | 16152 | 6 | 300 | 9.59 | 3.81 |
| `octopus_1` | 7251 | 14498 | 9 | 150 | 11.14 | 4.08 |
| `plane` | 7470 | 14936 | 7 | 150 | 7.54 | 3.34 |
| `pliers_2` | 5110 | 10216 | 6 | 130 | 5.21 | 2.82 |
| `teddy` | 9548 | 19092 | 7 | 130 | 12.25 | 4.53 |
| `teddy_2` | 12831 | 25658 | 16 | 30 | 22.83 | 4.47 |
| `wolf` | 4712 | 9420 | 7 | 150 | 5.94 | 2.45 |

The spectral segmentation results are shown in Fig. 6.15. The starting seeds (left) computed by Step 1 of **Algorithm 4** are placed correctly, then the region growing algorithm in Step 2 ends up with the partitioning in Fig. 6.15(middle).

On the right of Fig. 6.15 we report the mesh decomposition shown in [130] which has been produced with the help of an edge detection strategy introduced in the variational formulation. A visual insight allows us to observe some defects for both the top fins, on the cluster boundaries which indeed go through the middle of the fins.

### STEP 3: Patch-based Partitioning

The third step of **Algorithm 4** refines the partitioning obtained by Step 2 finalizing a patch-based manifold partitioning, see Def. 2. To this end, first we select from $S$ those parts $S_k$ which have genus higher than zero and/or more boundaries, and we re-run Step 1 and Step 2 for each of them until every $S_k$ has 0-genus and at most two boundaries.

In Figure 6.16 we illustrate a few examples of patch-based partitioning resulting from Step 3 (bottom row) in comparison with the mesh partitioning obtained in Step 2 (top row). For all the meshes reported in this figure, just one part (from left to right yellow/red/magenta/red) was further subdivided. The `fertility` mesh (left), characterized by four holes, represents a closed mesh of higher genus. Also in this case,

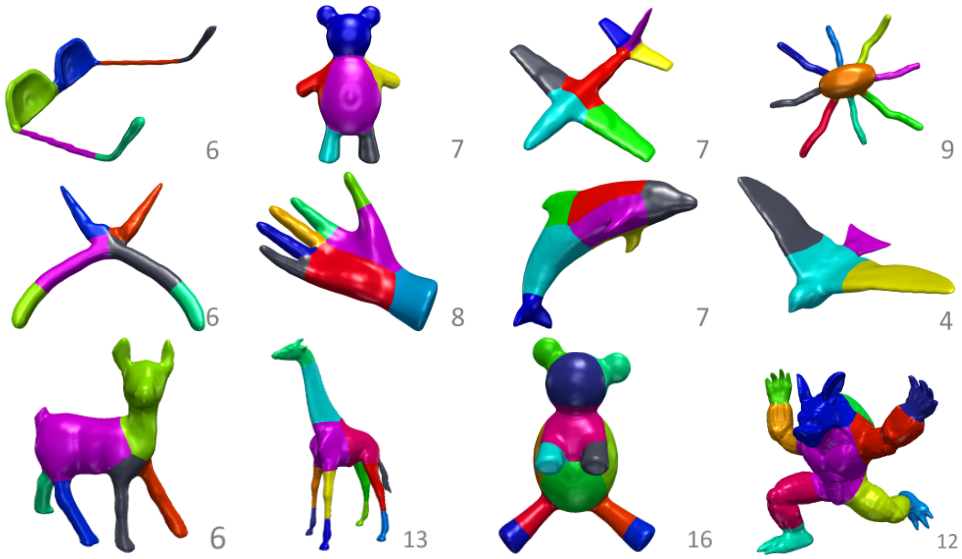



Fig. 6.13 Mesh partitioning into salient parts obtained in Step 2 of the **Algorithm 4**.

the algorithm was able to both localize salient parts of the mesh and create a satisfying 0-genus patching.

### 6.4.4 Experimental Results of CNC Segmentation on Surfaces

In this section we describe the experimental results which assess the performance of the segmentation algorithm proposed in Section 6.3, where in (6.19) we used the $\phi_{\log}$ penalty function defined in (6.20). For what concerns the ADMM, the parameter $\beta$ is chosen according to condition (6.40), and the ADMM iterations are terminated as soon as the relative change between two successive iterates satisfies

$$err := \|u^{(k)} - u^{(k-1)}\|_2 \,/\, \|u^{(k-1)}\|_2 < 10^{-4}. \tag{6.49}$$

#### Image Segmentation over Surfaces

In our first example, we consider an image function defined on a smooth manifold, a plane, discretized by a mesh of resolution $|V| = 85709$ and $|T| = 171356$. We investigate the usefulness of the considered model both in the convex, CNC and non-convex regimes. In particular, the convex case corresponds to the classical TV-$L_2$ model introduced for image denoising in [103], and is obtained by setting $\tau_c = 0$ in (6.27), the CNC model is given for $\tau_c = 0.99$ thus maximizing the non-convexity of the penalty term while preserving the overall convexity of the functional $\mathcal{J}$. Finally, for $\tau_c = 100$ we get a strictly

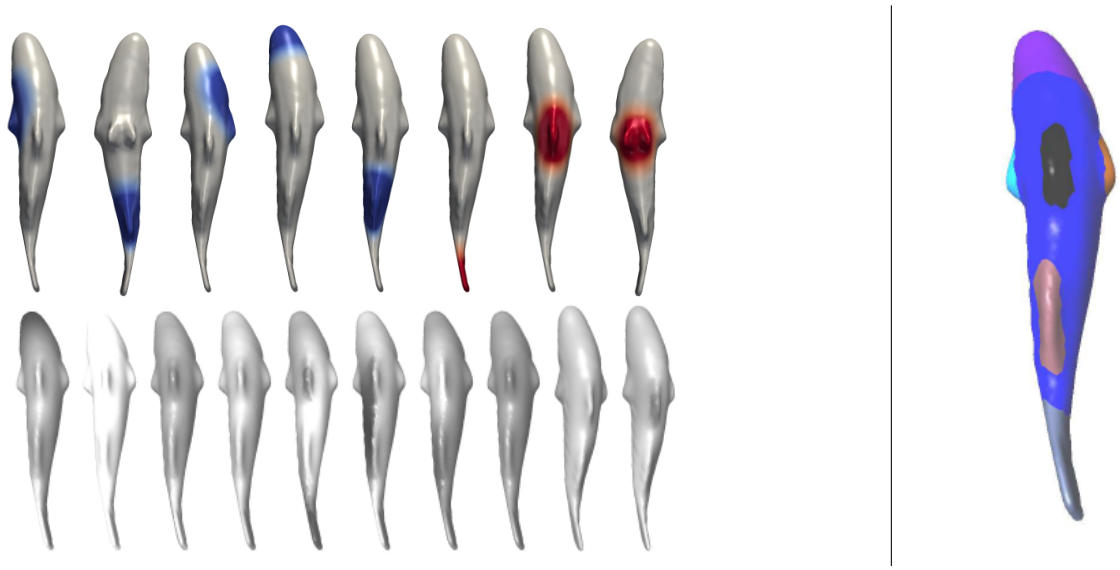


Fig. 6.14 Supports of the eigenfunctions for the `fish_2` mesh: $L_p$CMs results (top row) and human segmentation (top row, right), eigenfunctions of the affinity matrix proposed in [130] (bottom row).

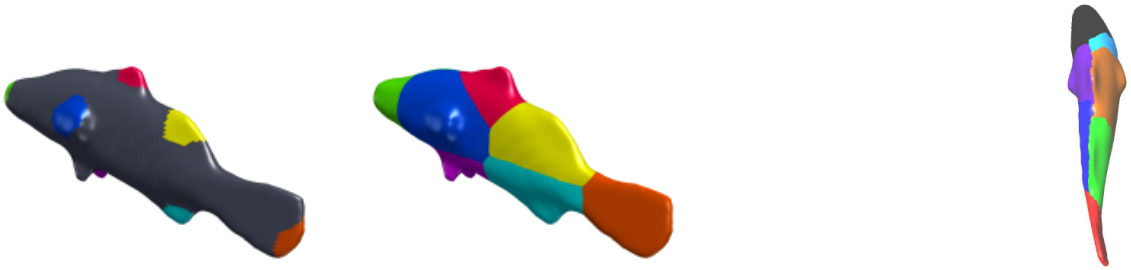


Fig. 6.15 Segmentation of the `fish_2` mesh: seed areas of Step 2(left), mesh segmented by Step 2 (center), and mesh segmentation using [130].

non-convex model solved by the same ADMM procedure. We set the regularization parameter $\eta = 0$ and the fidelity parameter $\lambda$ in the range $[500, 700]$. The input image textured on the plane was corrupted by an additive white Gaussian noise with standard deviation $\sigma = 5 \times 10^{-2}$. The results $u^*$ of the first phase of our segmentation algorithm are shown in Fig. 6.17 top row, and the associated values along the two lines indicated by red arrows are illustrated in Fig. 6.17, bottom row. The original intensity values are plotted in red dashed line. For the convex regime in Fig. 6.17(a) we can observe the typical behavior of TV-$L_2$ model: corners are smoothed and the contrast is decreased. The algorithm is stopped after 59 iterations, with a Root Mean Square Error (RMSE) of 18.88, with respect to the uncorrupted textured image plane. The CNC regime in Fig. 6.17(b) presents sharper edges and the loss of contrast is decreased. The number

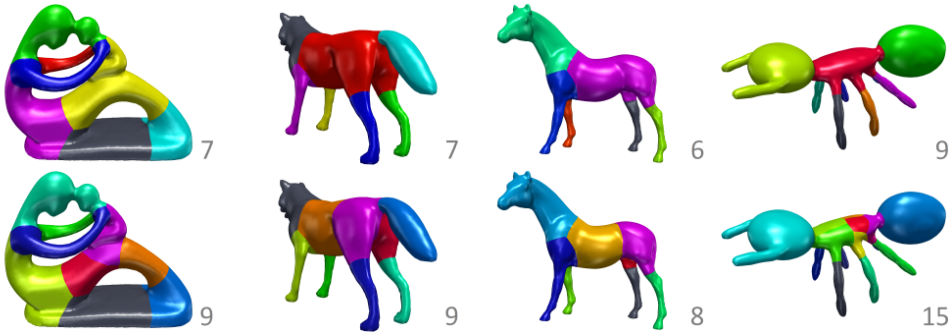
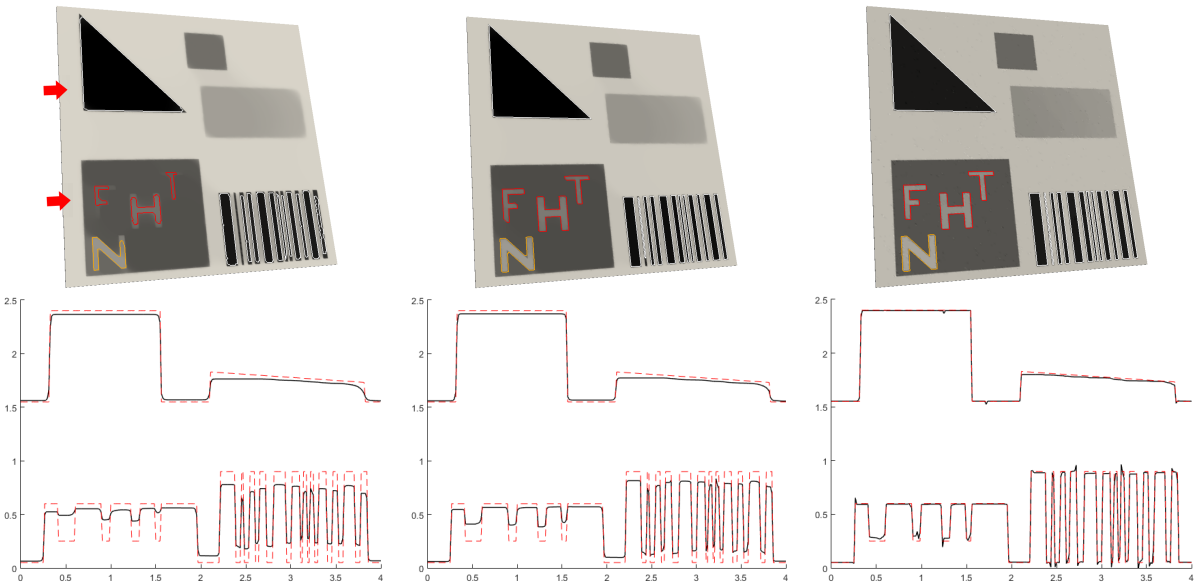



Fig. 6.16 Mesh segmentation into salient parts obtained by applying Step 2 of **Algorithm 4** (top row); patch-based partitioning into 0-genus patches by applying the refinement in Step 3 (bottom row).

of iterations needed for convergence was 55 and RMSE=13.20. The non-convex regime produces the result shown in Fig. 6.17(c), with RMSE=4.19. In this case the algorithm did not converge to the given threshold within the maximum number of iterations 500. Results in Fig. 6.17 indicate that higher quality restorations can be achieved by pushing model (6.19) beyond its convexity limits, however the numerical convergence is affected by the non-convexity of the functional $\mathcal{J}$.

### Thickness-based Segmentation

In the second example, we aim to segment a 2-manifold into $K$ disjoint regions that are "homogeneous" according to a certain feature, in this context represented by the thickness map of the object, computed on the mesh by the so-called Shape Diameter Function (SDF) with the method described in 5.1. The real-valued SDFs are shown in Fig. 6.18 top. We applied our algorithm in the CNC regime, namely with $\tau_c = 0.99$, $\lambda = \{50, 100, 30, 100\}$ and $\eta = 10$, which produced the partition function. Afterwards, the simple thresholding is used in order to clusterize the object into $K$ homogeneously thick parts. The segmented parts, together with the number of partitions $K$ on bottom right, are shown in Fig. 6.18 bottom, using false colors. We conclude this example presenting an empirical investigation on the numerical convergence of the proposed ADMM-based minimization scheme. By the way of illustration, in Fig. 6.19 we report the convergence plots concerning the `ant` model. The plots in Fig. 6.19(left column) show the relative change *err* defined in (6.49) and the functional values over the iterations in the CNC regime, whereas the right column of Fig. 6.19 shows the same quantities for the non-convex regime. The plots reported, and the similar results obtained for the other 2-manifolds, confirm that the numerical convergence in the non-convex regime is slower than in the CNC regime.



(a) (b) (c)

Fig. 6.17 Example 1: (a) Convex regime for $\tau_c = 0$, (b) CNC regime for $\tau_c = 0.99$, (c) Non-convex regime for $\tau_c = 100$. Boundary contours detected by the third stage of the segmentation algorithm are over-imposed for three different partitions.

### Boundary Tracking

In this example we show how the salient parts are nicely identified by the boundary tracking in the third phase of the segmentation algorithm. In Fig. 6.20 we show the segmentation of a real-valued function on the mesh `hand_im`, $|V| = 26422$, $|T| = 52840$; which has been corrupted by an additive Gaussian noise with $\sigma = 10^{-2}$. From left to right: the input noisy data, the result $u^*$ of the ADMM algorithm using $\lambda = 200$ and $\eta = 4$, the boundary curves overimposed to the clusterization of $u^*$, and a zoomed detail. The boundaries between the segmented parts (black solid line) is detected and then smoothed by projecting the boundary vertices onto the cubic spline obtained by least-squares approximation (white solid line).

### Application Examples

We conclude with two additional examples presented in Fig. 6.21. The noisy textured meshes of `horse_im` and `vase_im` (Fig. 6.21 top) represent synthetic examples of a real-life case – 3D data scanned together with texture. The Gaussian noise was added with $\sigma = 10^{-2}$ and the resulting partitioning function $u^*$ was obtained from the ADMM algorithm using $\lambda = \{120, 180\}$ respectively and $\eta = 0$. In the bottom of Figure 6.21 we

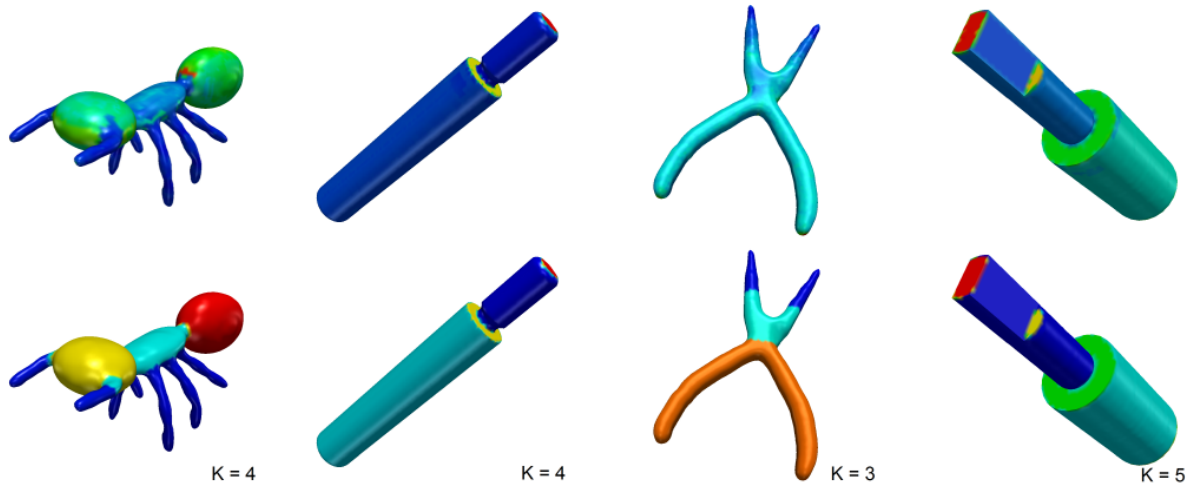



Fig. 6.18 Example 2: segmentation results (bottom row) obtained from the input SDF maps (top row). From left to right: `ant` $|V| = 7038$, $|T| = 14072$; `mech_1` $|V| = 8759$, $|T| = 17514$; `pliers` $|V| = 3906$, $|T| = 7808$; `mech_2` $|V| = 10400$, $|T| = 20796$.

plot the segmentation of $u^*$ in false colours together with the traced boundary curves coloured in white.

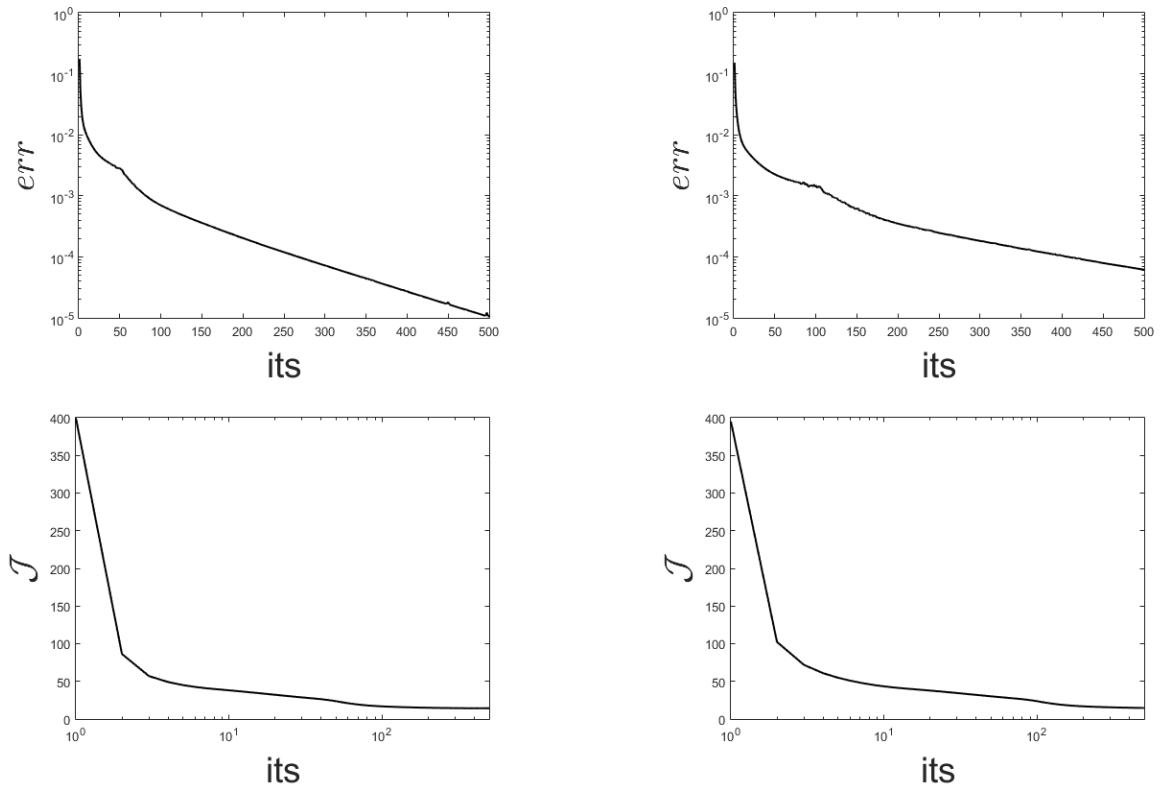


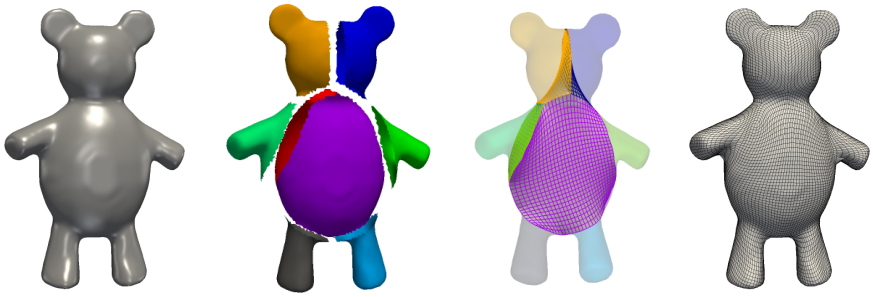

Fig. 6.19 Example 2: top: relative errors over iterations for `ant` mesh in CNC (left) and non-convex (right) regimes in logarithmic scale in y-axis. bottom: corresponding values of $\mathcal{J}$ in (6.21) with logarithmic scale of x-axis.

Fig. 6.20 Example 3: noisy textured mesh, result of the ADMM algorithm, segmented parts in false colours with overimposed boundary curves, detailed zoom.


Fig. 6.21 Example 4: noisy textured mesh (top) and its associated segmentations (bottom).

# Chapter 7

# Application in Surface-Patch Quadrangulation



*(a)* $M_{\triangle}$ *(b)* $\bigcup_i M_i$ *(c)* $\bigcup_i S_i$ *(d)* $M_{\square}$

Fig. 7.1 Overview of **Algorithm 5**, from left to right: Input Mesh $M_{\triangle}$; Mesh Partitioning; Topology-Skeleton $S$; Resulting pure Quadrilateral Mesh $M_{\square}$

In this chapter, we present a novel approach to Surface-Patch Quadrangulation. Starting from an unstructured triangle mesh $M_{\triangle}$ approximating a 2-manifold $\mathcal{M}$, we provide as a result a semi-regular, or valence semi-regular, quad mesh $M_{\square}$ representing the same 2-manifold $\mathcal{M}$. Note that the valence semi-regular mesh generated by the proposed algorithm can be easily converted in semi-regular one.

The resulting mesh has the following properties:

- Semi-regularity
- Uniform tessellation density
- Well-shaped quads

- Minimal number of patches – structural fidelity, thus few extraordinary vertices
- Patches well-representing the salient features of the shape

The overall process consists of three phases summarized in Algorithm 5 Surface-Patch Quadrangulation [57].

**Algorithm 5 Surface-Patch Quadrangulation**

**Input**: Triangular Mesh $M_\triangle$ , Quad edge length $h$
**Output**: Pure Quadrilateral Mesh $M_\square$

$\{M_i\}_{i=1}^K \longleftarrow$ `Mesh_Partitioning`$(M_\triangle)$
$\{S_i\}_{i=1}^K \longleftarrow$ `Topology_Skeleton`$(\{M_i\}_{i=1}^K, h)$
$M_\square \longleftarrow$ `Skeleton_Evolution`$(\{S_i\}_{i=1}^K, \{M_i\}_{i=1}^K)$

The input to Algorithm 5 is expected to be a pure triangular mesh $M_\triangle$, Fig. 7.1*(a)*, which is partitioned in Phase 1 into $K$ patches of 0-genus having at most two boundaries, Fig. 7.1*(b)*. This Phase is further discussed in Section 7.2.

From a given partitioning(patching), in Phase 2, we create a topology-skeleton structure $S$ of mesh $M_\triangle$, see Fig. 7.1*(c)*. This structure consists of a union of $K$ minimal surfaces, each discretized by a quad grid according to a given edge length, that match the boundaries of patches. An exhausting discussion of this procedure is formalized in Section 7.3.

At last, Phase 3 evolves the topology-skeleton $S$ towards vertices in $M_\triangle$ to create a pure quadrilateral approximation to the given shape obtaining $M_\square$ as shown in Fig. 7.1*(d)*. A detailed description of the evolution process is presented in Section 7.4.

## 7.1 Related Work

There are numerous surface quadrangulation techniques that can be categorized according to [15] into following classes.

**Direct Tri to Quad conversion** targets directly the problem of converting triangular (or polygonal) input mesh into a quadrangular one. Usually, the works utilize local connectivity operations, e.g. gluing two adjacent triangles into one quad, what makes it highly input-dependent. In case of even number of triangles in the mesh, the produced result consists of pure unstructured quad mesh whose quad-shape quality depends on the input regularity. The state of the art representatives are [114] and the work of Catmull and Clark [25] which are also implemented in the open source software Meshlab [34]. In

[114] the authors propose a tri-to-quad re-meshing as preliminary step to their mesh simplification algorithm. First they create a quad-dominant mesh by pairwise merging of adjacent triangles, prioritizing quads with nearly right angles. Next, by series of edge-flip iterations the remaining triangles are brought together and merged in order to create a pure quadrangulized mesh.

**Voronoi based** methods are based on generating a set of vertices on the surface and then utilizing Centroidal Voronoi Tessellation (CVT) [40] to optimize the sampling quality measure. Originally, the methods were developed to re-mesh a triangular surface, which led to enrich anisotropic feature on the vertices [105]. In [73] the authors propose $L_p$-CVT as a generalization of $L_2$-CVT [127] for a fully automatic feature-sensitive remeshing. As the authors declare, after performing interleaved optimization iterations of their objective function, as done in [117], they obtain a quad-dominant mesh by merging the triangles in priority order. In **Patch Based** class of methods the common technique used consists of constructing a one-to-one mapping of the original surface onto a set of square patches. Afterwards, the final quadrangulation is obtained by sampling each patch by a regular grid. Thus, the obtained result is a pure-quad mesh, semi-regular in valence. A few methods based on multiresolution [42], normal meshes [50] and abstract domains [97] are designed for triangle meshes, however, the methods can be adapted to quad-based setting or alternatively, transformed by an approach from the first class of methods.

On the other hand, methods developed in [14, 37, 113] are direct techniques for quad mesh generation. Authors in [37] take inspiration from the well-known triangle remeshing technique MAPS [71]. First, they utilize the Catmull-Clark subdivision [25] to get a pure quad mesh; then they simplify it via a variation of [38] to base domain corresponding to the input mesh. During the process, the original vertices are mapped to the simplified domain as well as the base-domain parametrization of the original surface is generated. Then, the resulting mesh is obtained by sampling the base domain and mapping the vertices back on the surface. Oppositely, in [14] the authors proceed by normal- and then spatial-based clustering to create a coarse mesh which they subdivide by Catmull-Clark iterations. The work [113] introduce application of polycube-maps to meshes in order to ease texture mapping. This work opened another sub-class of studies and methods based on polycube mappings both in automatic [77] and user controlled [126] construction, since the resulting re-sampling has several good properties as few irregular vertices and well-shaped quads.

The largest class gathers **Parametrization based** methods. The main idea is to construct a mapping from the surface embedded in 3D to a domain in 2D in which the quadrangulation is trivial [16]. There are many approaches concentrating on different

targets varying from conformal homology-based and harmonic-based parametrization [49, 116, 39] to field-guided methods, e.g. using RoSy field [66] and symmetric cross field [16]. In [93], the authors consider also symmetry of the processed meshes via RoSy field. Work [39] use the scalar harmonic field generated by eigendecomposition of Laplace-Beltrami operator computed on the triangular input mesh. They select an eigenvector corresponding to a higher frequency eigenvalue, which critical points are used as the input for Morse-Smale complex of that scalar field. The resulting extracted cells are quadrilateral and trivial to sample. In [53] the authors introduce to the original proposal a directional control.

At last the **Field-guided** methods are characterized by explicit control over local properties of quad elements in the mesh by means of the guiding fields. Traditionally, the problem of quadrangulation is divided into simpler sub-problems which uses results from the preceding steps as input for the actual one and can be used independently. Usually the problem is divided into three steps: Orientation Field Generation, Sizing Field Generation, and Quad Mesh Synthesis. The state of the art method introduced in [60] uses the three-step approach to produce both triangulation and quadrangulation of an input mesh or a pointcloud. In the first step, the Orientation Field Generation, the authors compute locally approximated smooth RoSy fields. Then they compute the position field for the new vertices of the mesh and at last they extract a quad dominant mesh from the position field. In order to obtain a pure quadrilateral mesh, they perform one iteration of the Catmul-Clark subdivision algorithm. In comparison to other works in this class of methods, since [60] is using local solutions for the field driving the quadrangulation, the optimization steps take very short time. On the other hand, w.r.t. globally optimized methods, the number of singularities (extraordinary vertices) is higher.

We categorize our work in the **Patch Based** class of methods.

## 7.2 Mesh Partitioning

The partitioning of $M_\triangle$ is the decomposition of $M_\triangle$ into $K$ open sub-meshes(patches) $M_i$ of 0-genus with one or two boundaries such that $M_\triangle = \bigcup_{i=1}^{K} M_i$. The algorithm is presented in Section 6.2. Nevertheless, for the purpose of this chapter we restrict ourselves to the case when each patch has exactly one boundary.

Each sub-mesh $M_i$ is characterized by a triplet $M_i = (V_i, T_i, B_i)$ where $V_i$ is a set of vertices, $T_i$ a set of triangles and $B_i$ is a set of ordered lists of boundary vertices in $V_i$.

Let us notice that in this settings, $M_{\triangle}$ can be considered also as a cloud of points with boundary, however, the triangle information is used in the last Phase 3, the Skeleton Evolution.

## 7.3 Build of the Topology Skeleton

The aim is to create from a given mesh partitioning an associated topology-skeleton $S = \bigcup_{i=1}^{K} S_i$ where each sub-part $S_i$ is a surface with boundary curves approximating $B_i$, thus matching $C^0$ the boundary of $M_i$, see Fig. 7.1 $(b) - (c)$. The benefit of $S$ will reveal later in the surface approximation by quadrilateral mesh.

The overall procedure to this aim consists of the following three main steps:

1. Construction of the boundaries of $S_i$.
2. Discretization of the boundaries of $S_i$ into rectangular topology.
3. Construction of $S$ by solving minimal surface PDE model for each $S_i$, $i = 1, \ldots, K$

The result of this constructional step is the skeleton $S$ represented by $K$ patches $S_i$ approximated by $K$ geometrically-regular grids.

### 7.3.1 Step 1: Construction of the boundaries of $S_i$

The goal of this step is to split the common boundaries $B_i$, $i = 1, \ldots, K$ between sub-meshes into a set $C$ of pieces $C_{ij}$ and approximate them by spline curves.

The global set $C$ of boundary pieces $C_{ij}$ is defined as

$$C := \left\{ C_{ij} : C_{ij} = M_i \bigcap M_j \, \forall (i,j) \right\} .$$

The piece $C_{ij}$, illustrated by dashed red lines in Fig. 7.2, is the common boundary of exactly two sub-meshes $M_i$ and $M_j$. We notice that $C_{ij} = C_{ji}\,, \forall (i,j)$.

We first locate a global set of vertices $P \subset V$ (marked in blue in Fig. 7.2) that lie on the common boundary of at least three patches of $M$. Formally, each element of $P$ is defined as $B_i \bigcap B_j \bigcap B_k$, $i \neq j \neq k$ where the triplet $\{i, j, k\}$ is taken as all the unique combinations out of $\{1, \ldots, K\}$. These vertices define the locations for the split of each $B_i$, $i = 1, \ldots, K$.

Once the set $P$ is selected, we proceed by looping through each patch boundary $B_i$ and by splitting it into pieces $C_{ij}$, which start and end with elements of $P$, forming the

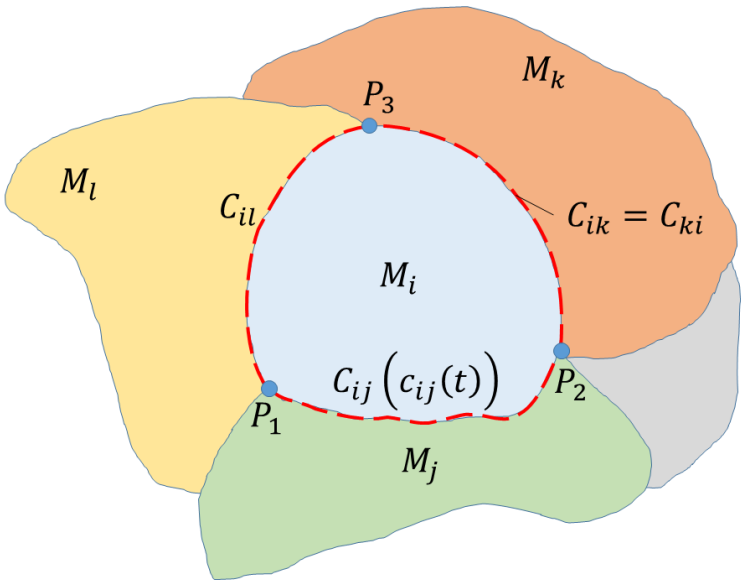



Fig. 7.2 A synthetic scheme of Construction of the $S_i$ boundary.

global set $C$. In case a boundary $B_i$ does not contain any vertex of $P$, the corresponding boundary piece $C_{ij} \equiv B_i$, i.e. the single boundary piece represents the whole boundary between two patches $M_i$ and $M_j$.

Finally, we create an index list $I_i$ for each $B_i$ of the references to elements of $C$ such that $B_i = \bigcup_{j \in I_i} C_{ij}$. In Fig. 7.2 we have $I_i = (j, k, l)$.

At last, we represent each boundary piece $C_{ij}$, by a least square approximating cubic spline $c_{ij}(t)$, $t \in [0, 1]$ such that $c_{ij}(0) = C_{ij}^1$ and $c_{ij}(1) = C_{ij}^{\#C_{ij}}$ where $\#C_{ij}$ stands for the cardinality of the list $C_{ij}$, i.e. the spline is clamped at the first and the last point of $C_{ij}$.

### 7.3.2 Step 2: Discretization of the boundaries into rectangular topology

The goals of this step are the sampling of the boundary of each $S_i$, computation of the grid resolution that describe the surface $S_i$, and the definition of the quad-side topology of its boundary.

The flow of the processes in this step can be summarized in three steps:

1. Re-sampling of $B_i$ in terms of $b_i$ and resolution settings by applying Principal Component Analysis (PCA) on $b_i$

2. Classify the patch

3. Corner Placement.

**Re-sampling of $B_i$ and resolution setting for $S_i$**

Assuming we know the desired minimal edge length $h$ of each quad in the final $M_\square$ mesh, we discretize uniformly the parametric domain $[0,1]$ of each $c_{ij}(t)$ into a number of points $\frac{|C_{ij}|}{h}$, so that to obtain for each $S_i$ a list $b_i$ of boundary points defined as

$$b_i = \bigcup_{j \in I_i} \left\{ c_{ij}(t_k),\, t_k \in [0,1],\; k = 1, \ldots, \frac{|C_{ij}|}{h} \right\}$$

where by $|C_{ij}|$ we denote the chord length of a piecewise linear curve computed by

$$|C_{ij}| = \sum_{k=1}^{\#C_{ij}-1} \|C_{ij}^{k+1} - C_{ij}^{k}\| \,,$$

with $C_{ij}^k$ the $k-$th element of the point list $C_{ij}$.

Once the boundary list $B_i$ is re-sampled into $b_i$ by taking into account the edge length $h$, we need to split $b_i$ into four parts which will correspond to the rectangular-topology boundary of a quad grid of resolution $(m_i, n_i)$, representing $S_i$. To that aim, the PCA is applied to the points $b_i$ for each $S_i$, $i = 1, \ldots, K$. This provides us with the axes $W = \{w_1, w_2, w_3\}$ of the ellipsoid enclosing $b_i$. The first two vectors $w_1$ and $w_2$ represent the two main directions of the data set $b_i$, while the ratio between the two vectors characterizes how much the ellipsoid is stretched. This allows us to detect the orientation of the rectangular-topology boundary of $S_i$, and the ratio between the number of uniformly sampled points $m_i$ and $n_i$ into the two directions.

The resolution $(m_i, n_i)$ of the quad grid representing $S_i$ is then obtained by imposing the following constraints

$$\begin{cases} 2(m_i + n_i) & = \#b_i \\ \dfrac{n_i}{m_i} & = c \end{cases} \tag{7.1}$$

where the first constraint imposes the perimeter of a rectangular domain and the second constraint imposes the ratio between the two sides to be $c = \frac{\|w_1\|_2}{\|w_2\|_2}$.

**Classification of the patch $S_i$**

Each patch $S_i$, $i = 1, \ldots, K$ is classified according to its corresponding patch $M_i$ w.r.t. its shape as

- $M_i^{(1)}$ – flat surface

- $M_i^{(2)}$ – protrusion.

In these cases $M_i$ has just one boundary. Examples of the patch types $M_i^{(1)}$ and $M_i^{(2)}$

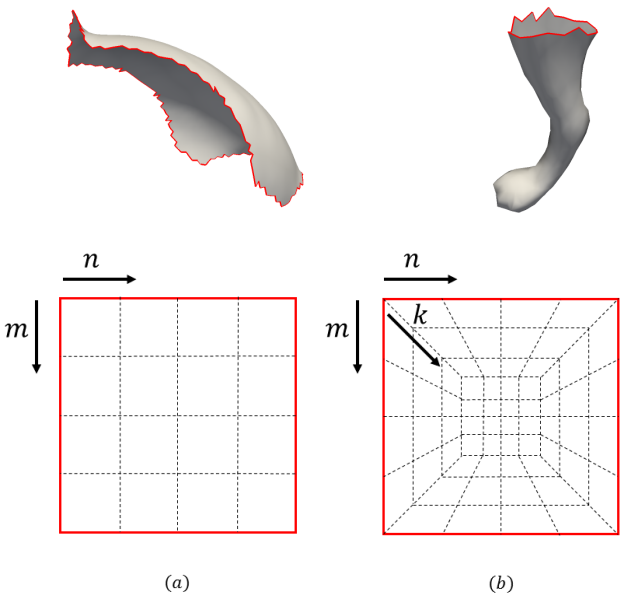



Fig. 7.3 Top: Examples of $M_i$ patches associated to types $M_i^{(1)}$ (a) and $M_i^{(2)}$ (b). Bottom: A sketch of the two approaches to discretize the domain $D$ according to the patch type. The boundary is red-coloured.

are illustrated in Fig. 7.3 (top), together with their domains $D = [0,1]^2$, in Fig. 7.3 (bottom).

Each $S_i$ is discretized according to its $M_i$ type. A regular grid discretization of the domain $D$ forms $q_i = m_i \times n_i$ quads for $M_i^{(1)}$ type, see Fig. 7.3 $(a)$ bottom; and $q_i = m_i \times n_i + 2k_i(m_i + n_i)$ quads for type $M_i^{(2)}$, Fig. 7.3 $(b)$ bottom. Let us notice that for the protrusion patch in Fig. 7.3 $(b)$, the discretization follows its natural shape much better.

The classification is done as follows. For each patch $M_i$ we compute the triangulation area $A_i = \sum_j |T_i^j|$ that we want to cover by quads of size $h \times h$. Thus, the expected number of quads in $S_i$ for corresponding $M_i$ is given by $q_i^e = \frac{A_i}{h^2}$.

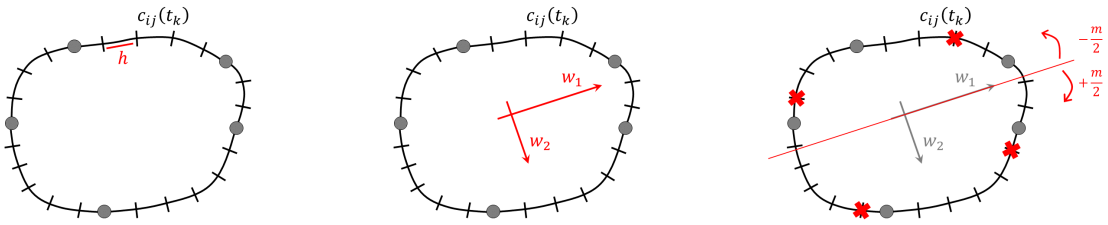
We compare each $q_i^e$ to the number of quads $q_i$. The classification between $M_i^{(1)}$ and $M_i^{(2)}$ is driven by thresholding with $\delta = min_i \frac{q_i^e}{q_i}$. Then all the patches $M_i$ for which $q_i > \delta$ are classified as $M_i^{(2)}$ with the number of rings in $k$ direction, see Fig.7.3 bottom $(b)$, such that $k$ is the minimum $k$ for which the relation $q_i \leq \delta$ holds; $M_i^{(1)}$ otherwise.

### Corner Placement

Given the resolution settings $(m_i, n_i)$ and the sampled boundary $b_i$, we can just place two corners of the rectangular domain while the remaining two will be placed automatically. From a visual inspection it is more appealing to see the mesh lines following the natural symmetry of an object, in our case the boundary $b_i$.

Thus, we proceed by reflecting $b_i$ along $w_1$ and $w_2$ obtaining $b_i^{1'}$ and $b_i^{2'}$. The symmetry index is computed as the minimal distance norm between $b_i$, $b_i^{1'}$ and between $b_i$, $b_i^{2'}$. Then, following the direction $w_k$ which scores the lower symmetry index, the corners are placed at distance $\frac{m_i}{2}$ (or $\frac{n_i}{2}$), centred around $w_k$.

An illustrative scheme of these steps is summarized in Figure 7.4. In Fig. 7.4 (1) a synthetic example of a patch $S_i$ and its boundary $b_i$ is shown. The boundary set $B_i$ composed of five pieces $C_{ij}$ is sampled by $c_{ij}(t_k)$, thus producing the set $b_i$. The PCA vectors $w_1$ and $w_2$ associated to $b_i$ are illustrated in Fig 7.4 (2) and the corner placement process in Fig. 7.4 (3).




(1) (2) (3)

Fig. 7.4 Illustration of the inner steps in Step 2 described in Section 7.3.2 from left to right: $b_i$ approximation of $B_i$ (1), PCA vectors (2)-(3), and placement of the corners (4).

## 7.3.3 Step 3: Construction of the topology-skeleton S

In this step we complete the association of the rectangular-topology grid to each $S_i$, $i = 1, \ldots, K$. In particular, the boundary $b_i$ of each $S_i$ is already known, thus, the inner nodes of $S_i$ are obtained solving the following boundary value problem with Dirichlet boundary condition $b_i$:

$$\begin{cases} \Delta_{\mathcal{M}} S_i(x) &= 0 \\ \partial S_i(x) &= b_i \ . \end{cases} \tag{7.2}$$

The Laplace operator is discretized using the finite volume scheme described in the numerical section 7.4.2.

The discretization of (7.2) based on finite volumes leads to the solution of a sparse system of linear equations $LX = b$ for the inner nodes of $S_i$. The system dimensions for every $S_i$ are given by the patch type, namely $(m-1) \times (n-1)$ for $M_i^1$ and $m \times n + 2k(m+n-2)$ for $M_i^2$.

This construction guarantees that two neighbouring patches $S_i$ and $S_j$ share the same boundary. Moreover, it allows us to further evolve each $S_i$ separately, and to join them together at the end of the evolution step described in Section 7.4 or, alternatively, to glue each $S_i$ together to form the topology skeleton $S$ and to evolve it as whole.

So far, the constructed topological skeleton discretized by pure quad mesh has at most $\#EV = \#P + 4K$ extraordinary vertices in valence, that will not increase any more during the final evolution step.

## 7.4 Skeleton Evolution

In this Section we describe the evolution of the topology-skeleton S towards vertices in $M_\triangle$ to obtain a pure quadrilateral approximation $M_\square$ to the given shape.

To that aim we will apply Lagrangian type evolution, which sustain the problem dimension. However, since an unsupervised evolution could lead to intersection of the evolving vertices, we derive an area-oriented tangential redistribution. Alternatively to the area-oriented tangential redistribution, which controls the area globally, we can apply also a local tangential redistribution based on the Laplace operator.

### 7.4.1 Lagrangian Evolution

Let us consider a family of (open) parametric surfaces $\mathcal{S} = \{\boldsymbol{x}(t,u,v) | \, (u,v) \in [0,1]^2 \, , t \in [t_0, t_{end}]\}$ obtained by evolving in time $t$ an initial surface $\mathcal{S}(t_0,\cdot,\cdot) = S$ where $S$ is the topology skeleton.

The evolution is driven by the following partial differential equation

$$\begin{aligned} \frac{\partial \mathbf{x}}{\partial t} &= \mathbf{V_N} + \mathbf{V_T} = \beta \mathbf{N} + \mathbf{V_T} \\ \mathbf{x}(0;\cdot) &= S \end{aligned} \tag{7.3}$$

where $\mathbf{V_T}$ is the evolution in tangential direction along the surface defined as

$$\mathbf{V_T}(u,v) = \alpha(u,v)\mathbf{x}_u + \lambda(u,v)\mathbf{x}_v \ , \tag{7.4}$$

with

$$\mathbf{x}_u = \frac{\partial \mathbf{x}(t,u,v)}{\partial u} \ , \qquad \mathbf{x}_v = \frac{\partial \mathbf{x}(t,u,v)}{\partial v} \ , \tag{7.5}$$

representing the partial derivatives of $\mathbf{x}(t,u,v)$, which span the tangent plane; and $\mathbf{V_N}$ represents the evolution with speed $\beta$ in the outward unit normal direction $\mathbf{N}$ to the surface computed as

$$\mathbf{N} = \frac{\boldsymbol{x}_u \times \boldsymbol{x}_v}{|\boldsymbol{x}_u \times \boldsymbol{x}_v|} \ , \tag{7.6}$$

which affects the surface image.

In the continuous settings $\mathbf{V_T}$ has no impact on the surface image, however, it has been shown in literature [83] that this kind of splitting of the movement into two orthogonal directions is very useful especially in case of a Lagrangian-type evolutions. At a specific time $t$, the result of (7.3) is a 2-manifold $\mathcal{S} = \mathbf{x}(t,\cdot,\cdot)$ and at the final time $\mathbf{x}(t_{end},\cdot,\cdot) = M_{\square}$, the desired approximation to the given shape $M_{\triangle}$.

### Normal-direction evolution

Our aim is to prescribe the normal-direction evolution $\mathbf{V_N}$ in (7.3) in such way the evolving surface will be moving towards the given shape. An intuitive way would be to set an advection term

$$\mathbf{V_N} = -(\nabla d(\mathbf{x}) \cdot \mathbf{N})\mathbf{N} \ ,$$

where $d(\mathbf{x})$ represents the signed distance function to the given shape.

In case the distance function is not smooth, to achieve a smooth evolution, we add additional term depending on the mean curvature of $\mathcal{S}$

$$\mathbf{V_N} = H\mathbf{N} - (\nabla d(\mathbf{x}) \cdot \mathbf{N})\mathbf{N}$$

where the mean curvature vector $H\mathbf{N}$ can be expressed in terms of Laplace-Beltrami operator as

$$H\mathbf{N} = \Delta_{\mathbf{x}}\mathbf{x} \ .$$

However, if such evolution is not controlled, it can result in a surface poorly approximating the point cloud. In order to control the trade-off between the advection and

diffusion term, we introduce two function parameters $\epsilon(d(\mathbf{x}))$ and $\eta(d(\mathbf{x}))$ depending on the signed distance function $d(\cdot)$ at vertex $\mathbf{x}$

$$\mathbf{V_N} = \epsilon(d(\mathbf{x}))\Delta_{\mathbf{x}}\mathbf{x} + \eta(d(\mathbf{x}))\mathbf{N} \,. \tag{7.7}$$

For $\epsilon(d(\mathbf{x}))$ in the diffusion term we would like to obtain stronger smoothing of the evolution surface in case we are far from $M_\triangle$. Thus we define $\epsilon(d(\mathbf{x}))$ as

$$\epsilon(d(\mathbf{x})) = c_1 \left(1 - e^{-\frac{d(\mathbf{x})^2}{c_2}}\right) \tag{7.8}$$

where $c_1$ and $c_2$ are parameters controlling the shape of $\epsilon(\cdot)$. In Fig. 7.5 we plot the function $\epsilon(t)$ for different values of $c_2$, which controls the width of the function. On the other hand, parameter $c_1$ controls the amplitude.

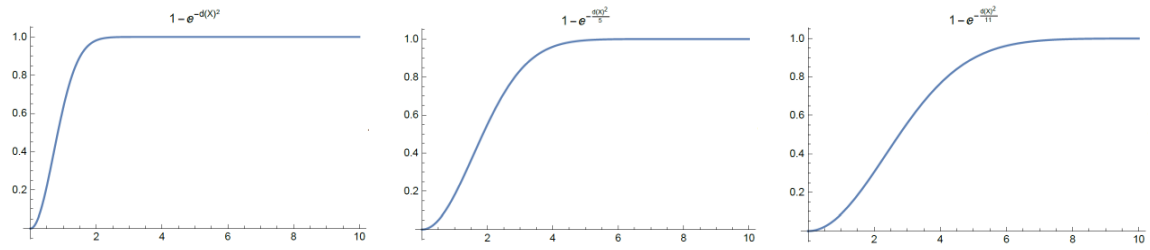


Fig. 7.5 Function $\epsilon(t)$ with $c_1 = 1$ and different $c_2$.

Using a parameter dependent on the distance function results in an evolution smoothed by a mean curvature in the points distanced from the point cloud [132].

On the other hand, function $\eta(d(\mathbf{x}))$ is composed of two parts. The first one represents the advection itself, which could be insufficient in cases the normal vector $\mathbf{N}$ and $\nabla d(\mathbf{x})$ are perpendicular. The second part represents a variation of the so-called 'balloon force', which deals with this problem and helps to enlarge the volume of the evolving surface $\mathcal{S}(t, \cdot)$. The function $\eta(d(\mathbf{x}))$ is defined as

$$\eta(d(\mathbf{x})) = -c_3 d(\mathbf{x}) \left(\nabla d(\mathbf{x}) \cdot \mathbf{N} + \sqrt{\nabla d(\mathbf{x}) \cdot \nabla d(\mathbf{x}) - (\nabla d(\mathbf{x}) \cdot \mathbf{N})^2}\right) \tag{7.9}$$

where $-d(\mathbf{x})\nabla d(\mathbf{x}) \cdot \mathbf{N}$ is the projection of the distance function gradient onto normal direction, and the rest in (7.9) is the variation of the balloon force. It is designed in a way that it vanishes when the normal vector $\mathbf{N}$ is pointing in the direction of $-\nabla d(\mathbf{x})$, while when these vectors a perpendicular, it is proportional to the distance $d(\mathbf{x})$. The parameter $c_3$ in some cases helps to correct the advection evolution depending on the

position of $\mathcal{S}(t,\cdot)$ w.r.t. $M_\triangle$. In case the surface is enclosed by $M_\triangle$ and near to its border, but $-\nabla d(\mathbf{x})$ is pointing the opposite direction of the outward unit normal $\mathbf{N}$, that means even though it seems we are close to $M_\triangle$, our evolving surface $\mathcal{S}(t,\cdot)$ is located at the opposite side of the given shape. Thus in this case, enlarging $c_3$ 'outperforms' the advection force and the surface will evolve in direction $\mathbf{N}$, towards the opposite side of $M_\triangle$. Moreover the sign of the signed distance function $\text{sign}(d(\mathbf{x}))$ provides the information whether $\mathcal{S}(t,\cdot)$ is outside from $M_\triangle$. In this case, the we should shrink the surface onto $M_\triangle$, thus flipping the sign of the balloon force.

Summarizing, the evolution in the normal direction in (7.3) is driven by (7.7) consisting of an advection force enriched by the balloon force (7.9) and a smoothing diffusion force (7.8) controlling the mean curvature flow.

### Tangential Redistribution

In this section we provide a theoretical background for the control of the area of the quads that will be performed in the discrete settings through the control of the finite volumes areas.

Let us introduce the area-density $g$ of the surface defined as

$$g = |\boldsymbol{x}_u \times \boldsymbol{x}_v| = \sqrt{\det \begin{bmatrix} \mathbf{x}_u \cdot \mathbf{x}_u & \mathbf{x}_u \cdot \mathbf{x}_v \\ \mathbf{x}_v \cdot \mathbf{x}_u & \mathbf{x}_v \cdot \mathbf{x}_v \end{bmatrix}}. \tag{7.10}$$

We are interested in its evolution and the dependence of $g$ on the normal and tangential evolution of a surface in form (7.3).

This dependence is given in Lemma 3.

**Lemma 3** *Let $\mathbf{x}(t,\cdot)$ be a smooth parametric surface representing the embedding of an evolving 2-manifold in $\mathbb{R}^3$ in time $t$ which evolution can be expressed as (7.3). Moreover, let $\mathbf{V_T} = \alpha\mathbf{x}_u + \lambda\mathbf{x}_v$. Then the local area density $g$ of $\mathbf{x}(\cdot)$, evolves as*

$$g_t = gH\beta + g\nabla_{\mathbf{x}} \cdot (\alpha, \lambda) \tag{7.11}$$

*where $H$ is the mean curvature scalar field and $\nabla_{\mathbf{x}} \cdot (\alpha, \lambda)$ is the surface divergence of the vector field $(\alpha, \lambda)$.*

The proof is postponed in the Appendix, as well as the surface divergence definition, see formulas (C.3)-(C.4) in the Appendix.

We prescribe a desired evolution of $g$ for asymptotically uniform area density in order to obtain the immersion of the surface with uniform area density all over it.

If we want the surface area density $g$ to converge to an area density $c$ during the evolution we want to prescribe the change of $g$ w.r.t. the area of the surface $A$ as

$$\left(\frac{g}{A}\right)_t = \left(\frac{c}{A} - \frac{g}{A}\right)\omega \tag{7.12}$$

where the scalar parameter $\omega$ controls the speed of convergence. Let us notice that for $\omega = 0$, there is no change, thus the density remains preserved as it was at the start of the evolution. The formula (7.12) allows to manipulate the local density $g$ via the prescribed value $c$ to be asymptotically reached for $t \to \infty$. For the uniform density distribution we can set $c = \frac{\iint_U g \,\mathrm{d}u\mathrm{d}v}{\iint_U \mathrm{d}u\mathrm{d}v}$, i.e. $g$ will converge to the average of $g$.

A relation between $\mathbf{V_T}$ and $g$ is summarized in Theorem 5 in terms of divergence operator.

**Theorem 5** *Let $\mathbf{x}(t,\cdot)$ be a smooth parametric surface representing the embedding of an evolving 2-manifold in $\mathbb{R}^3$ in time $t$, which evolution can be expressed as (7.3). Moreover, let $\mathbf{V_T} = \alpha\mathbf{x}_u + \lambda\mathbf{x}_v$. Then the tangential vector field $\mathbf{V_T}$, leading to a surface with area density $c$, is related to the surface area density $g$, mean curvature $H$ and the normal direction evolution speed $\beta$ as*

$$\nabla_{\mathcal{S}} \cdot \mathbf{V}_T = \nabla_{\mathbf{x}} \cdot (\alpha, \lambda) = H\beta - \frac{1}{A}\iint_{\mathcal{S}} H\beta \,\mathrm{d}X + \left(\frac{c}{g} - 1\right)\omega \,. \tag{7.13}$$

The integral in (7.13) is related to the surface $\mathcal{S}$, which for a general function $f(u,v)$ defined on $\mathcal{S}$ can be expressed as

$$\iint_{\mathcal{S}} f \,\mathrm{d}X = \int_0^1 \int_0^1 f(u,v) g \,\mathrm{d}u\mathrm{d}v \,. \tag{7.14}$$

Corollary 1 provides a unique solution to the formula in Theorem 5 in terms of potential field $\varphi$.

**Corollary 1** *Assuming* $(\alpha, \lambda)$ *is a gradient vector of a potential field* $\varphi$*, the equation (7.13) attains a unique solution for a potential field problem*

$$\Delta_{\mathbf{x}}\varphi = \nabla_{\mathbf{x}} \cdot \nabla_{\mathbf{x}}\varphi = H\beta - \frac{1}{A}\iint_{\mathcal{S}} H\beta \, \mathrm{d}X + \left(\frac{c}{g} - 1\right)\omega \,, \tag{7.15}$$

*if we prescribe the value of* $\varphi$ *at least in one point* $x$ *of the surface, or imposing the Neumann boundary conditions in case of open surface.*

### 7.4.2 Numerical scheme

In this section we derive the numerical scheme to obtain the solution of the model in (7.3) and the potential field $\varphi$ in (7.15). Due to the presence of non-linear operators in the equations, we will adapt the semi-implicit scheme in time to linearize the Laplace-Beltrami operator. For the spatial discretization, we adapt the finite volume approach for the surface $\mathcal{S}$ introduced in Section 4.2.

In the preliminary step, we approximate the distance function $d(\mathbf{x})$ on a voxelized grid enclosing the input triangulation $M_\triangle$ by the Fast Sweeping Algorithm [131] and modified it to a signed distance function by a simple flooding algorithm. Let us note that in case an evolving vertex is localized in the same voxel as a part of $M_\triangle$, we proceed by computing the true signed distance to $M_\triangle$.

#### Finite Volume Scheme for the Model

We integrate (7.3) over the finite volume $V_i$ where we replaced (7.7) for $\mathbf{V_N}$

$$\int_{V_i} \partial_t \mathbf{x} \, \mathrm{d}S = \int_{V_i} \epsilon(d(\mathbf{x}))\Delta_s \mathbf{x} \, \mathrm{d}S + \int_{V_i} \eta(d(\mathbf{x}))\mathbf{N} \, \mathrm{d}S + \int_{V_i} \mathbf{V}_T \, \mathrm{d}S \,. \tag{7.16}$$

Assuming the functions $\epsilon(\cdot)$ in (7.8) and $\eta(\cdot)$ in (7.9) obtain a constant values over the control volume $V_i$, i.e. $\epsilon_i = \epsilon(d(\mathbf{x}))|_{V_i}$, $\eta_i = \eta(d(\mathbf{x}))|_{V_i}$; and applying the Green's theorem we can rewrite (7.16) as

$$\int_{V_i} \partial_t \mathbf{x} \, \mathrm{d}S = \epsilon_i \int_{\partial V_i} \nabla_s \mathbf{x} \cdot \mathbf{n}_i \, \mathrm{d}S + \int_{V_i} \eta_i \mathbf{N} \, \mathrm{d}S + \int_{V_i} \mathbf{V}_T \, \mathrm{d}S \,. \tag{7.17}$$

The first term on the right hand side of (7.17), using the definition of finite volumes, can be rewritten as

$$\epsilon_i \int_{\partial V_i} \nabla_s \mathbf{x} \cdot \mathbf{n}_i \, \mathrm{d}s = \epsilon_i \sum_{j \in N_\square(\mathbf{x}_i)} \int_{e_j} \nabla_s \mathbf{x} \cdot \mathbf{n}_j \, \mathrm{d}s = \epsilon_i \sum_{j \in N_\square(\mathbf{x}_i)} \int_{e_j} \frac{\partial \mathbf{x}}{\partial \mathbf{n}_j} \, \mathrm{d}s \tag{7.18}$$

where $e_j$ is an edge of the finite volume $V_i$ on a quad $Q_j$, that consists of two edges $e_j^1$ and $e_j^3$, and $\mathbf{n}_j$ denotes the normal on $e_j$.

Moreover, $e_j^1$ and $e_j^3$ can be parametrized as

$$e_j^1(\rho) = (\mathbf{r}_j^2 - \mathbf{r}_j^1)\rho + \mathbf{r}_j^1 \ , \quad \rho \in (0,1) \ ,$$
$$e_j^3(\phi) = (\mathbf{r}_j^2 - \mathbf{r}_j^3)\phi + \mathbf{r}_j^3 \ , \quad \phi \in (0,1) \ .$$

Thus, we can divide each term of the sum (7.18) into sum of integrals along $e_j^1$ and $e_j^3$, $j = 1, \ldots, m_i$ as

$$\epsilon_i \int_{\partial V_i} \nabla_s \mathbf{x} \cdot \mathbf{n} \, \mathrm{d}s = \epsilon_i \sum_{j \in N_\square(\mathbf{x}_i)} \int_{e_j^1} \frac{\partial \mathbf{x}}{\partial \mathbf{n}_j^1} \, \mathrm{d}s + \int_{e_j^3} \frac{\partial \mathbf{x}}{\partial \mathbf{n}_j^3} \, \mathrm{d}s \ . \tag{7.19}$$

The normal derivatives are approximated using bilinear interpolation and the detailed formulas are derived in the Appendix. Using the formulas from Appendix, we can approximate (7.19), thus (7.18) as

$$\begin{aligned} \epsilon_i \int_{\partial V_i} \nabla_{\mathbf{x}} \mathbf{x} \cdot \mathbf{n} \, \mathrm{d}s \approx \epsilon_i \sum_{j \in N_\square(\mathbf{x}_i)} & \frac{m(e_j^1)}{|\mathbf{N}_j^1|} \left[ \frac{1}{4} \left( -3\mathbf{x}_j^0 + 3\mathbf{x}_j^1 - \mathbf{x}_j^3 + \mathbf{x}_j^2 \right) - \frac{a_j^1}{2} \left( -\mathbf{x}_j^0 - \mathbf{x}_j^1 + \mathbf{x}_j^3 + \mathbf{x}_j^2 \right) \right] \\ & + \frac{m(e_j^3)}{|\mathbf{N}_j^3|} \left[ \frac{1}{4} \left( -3\mathbf{x}_j^0 - \mathbf{x}_j^1 + 3\mathbf{x}_j^3 + \mathbf{x}_j^2 \right) - \frac{a_j^3}{2} \left( -\mathbf{x}_j^0 + \mathbf{x}_j^1 - \mathbf{x}_j^3 + \mathbf{x}_j^2 \right) \right] \end{aligned} \tag{7.20}$$

where $\mathbf{N}_j^1$, $\mathbf{N}_j^3$ are defined in (C.12)-(C.13) and $m(e_j^k) = |\mathbf{r}_j^2 - \mathbf{r}_j^k|$ denotes the Euclidean measure of edge $e_j^k$.

The second term on the right-hand-side of (7.17) can be rewritten as

$$\eta_i \int_{V_i} \mathbf{N} \, \mathrm{d}S = \eta_i \sum_{j \in N_\square(\mathbf{x}_i)} \left( \mathbf{r}_j^1 - \mathbf{r}_j^0 \right) \times \left( \mathbf{r}_j^2 - \mathbf{r}_j^0 \right) + \left( \mathbf{r}_j^2 - \mathbf{r}_j^0 \right) \times \left( \mathbf{r}_j^3 - \mathbf{r}_j^0 \right) \ , \tag{7.21}$$

since $\eta_i$ is considered a constant value on the finite volume $V_i$.

Finally, the left-hand side term in (7.17) is approximated by means of the forward finite difference and the mid-point quadrature formula

$$\int_{V_i} \partial_t \mathbf{x} \, \mathrm{d}S = m(V_i) \left( \frac{\mathbf{x}_i^{t+1} - \mathbf{x}_i^t}{\Delta t} \right) \tag{7.22}$$

where the measure of the finite volume $V_i$ is given by

$$m(V_i) = \frac{1}{2}\left| \sum_{j \in N_\square(\mathbf{x}_i)} \left(\mathbf{r}_j^1 - \mathbf{r}_j^0\right) \times \left(\mathbf{r}_j^2 - \mathbf{r}_j^0\right) + \left(\mathbf{r}_j^2 - \mathbf{r}_j^0\right) \times \left(\mathbf{r}_j^3 - \mathbf{r}_j^0\right) \right| .$$

The third term on the right-hand-side in (7.17) will be discretized in the following Section.

### Numerical scheme for the tangential redistribution

We recall that in order to obtain $\mathbf{V}_T$, or equivalently the third term on the right-hand-side in (7.17), we first have to obtain the potential field $\varphi$ in (7.15). Integrating (7.15) over the finite volume $V_i$ we get

$$\int_{V_i} \Delta_x \varphi \, \mathrm{d}S = \int_{V_i} H\beta \, \mathrm{d}S - \int_{V_i} \frac{1}{A} \iint_{\mathcal{S}} H\beta \, \mathrm{d}X \mathrm{d}S + \int_{V_i} \left(\frac{c}{g} - 1\right) \omega \, \mathrm{d}S \, . \tag{7.23}$$

The left-hand-side term in (7.23) leads to the same coefficient matrix as (7.20) for $\epsilon_i = 1$.

The first term on the right-hand side in (7.23) can be approximated again utilizing the relation (C.16) from Normal Derivatives Approximation in the Appendix, since $\Delta_{\mathbf{x}}\mathbf{x} = H\mathbf{N}$. Scalar $\beta_i$ at vertex $\mathbf{x}_i$ represents the normal direction evolution speed, and is defined as

$$\beta_i = \epsilon_i H + \eta_i \, .$$

The second term on the right-hand side in (7.23) represents the mean value of $H\beta$ along the surface and can be approximated as

$$\int_{V_i} \frac{1}{A} \iint_{\mathcal{S}} H\beta \, \mathrm{d}X \mathrm{d}\mathcal{S} \approx \frac{m(V_i)}{A} \sum_{i=1}^{n} m(V_i) H_i \beta_i \tag{7.24}$$

where the area $A \approx \sum_{i=1}^{n} m(V_i)$.

Regarding the third term in (7.23), if we want in the discrete setting every finite volume to have the same area, we would set $c = A$, however, in our case we want every quad to have the same area, so $c$ should depend on number of quads which form the finite volume. The value of $c$ is approximated as

$$c = \frac{\iint_U g \, dudv}{\iint_U \, dudv} \approx \frac{A}{\sum_i^n |N_\square(X_i)|} = \frac{A}{\sum_i^n m_i}$$

and the value of $g$ is approximated by $g \approx \frac{m(V_i)}{m_i}$.

Thus the last term of (7.23) can be approximated as

$$\int_{V_i} \left(\frac{c}{g} - 1\right) \omega \, \mathrm{d}S \approx \left(\frac{\frac{A}{\sum_i^n m_i}}{\frac{m(V_i)}{m_i}} - 1\right) \omega \, m(V_i) = \left(\frac{A m_i}{\sum_i^n m_i} - m(V_i)\right) \omega \, . \tag{7.25}$$

Replacing the integral approximations into (7.23), we obtain a sparse linear system of $n$ equations for the unknown values of potential $\varphi_i$, $i = 1, ..., n$ at vertices $\mathbf{x}_i$. The system is solved at every time step $t$.

Knowing the values of potential $\varphi$, we now describe how to compute $\mathbf{V}_T$ as a gradient of $\varphi$ on a surface that appears in the evolution model (7.3). It can be computed using the following identity [41]

$$\int_{V_i} \mathbf{V}_T \, \mathrm{d}\mathcal{S} = \int_{V_i} \nabla_{\mathbf{x}} \varphi \, \mathrm{d}\mathcal{S} = \int_{\partial V_i} \varphi \mathbf{n} \, \mathrm{d}s - \int_{V_i} \varphi k \mathbf{N} \, \mathrm{d}\mathcal{S} \, . \tag{7.26}$$

The boundary integral on the right hand side is approximated using the bilinear interpolation

$$\begin{aligned}
\int_{\partial V_i} \varphi \mathbf{n} \, \mathrm{d}s &= \sum_{j \in N_\square(\mathbf{x}_i)} \int_{e_j^1} \varphi \mathbf{n}_j^1 \, \mathrm{d}s + \int_{e_j^3} \varphi \mathbf{n}_j^3 \, \mathrm{d}s \\
&\approx \sum_{j \in N_\square(\mathbf{x}_i)} \int_{e_j^1} \varphi \frac{\mathbf{N}_j^{1,t+1}}{\left|\mathbf{N}_j^{1,t}\right|} \, \mathrm{d}s + \int_{e_j^3} \varphi \frac{\mathbf{N}_j^{3,t+1}}{\left|\mathbf{N}_j^{3,t}\right|} \, \mathrm{d}s \\
&\approx \sum_{j \in N_\square(\mathbf{x}_i)} m(e_j^1) \left(\frac{3}{8}\varphi_i + \frac{3}{8}\varphi_j^1 + \frac{1}{8}\varphi_j^2 + \frac{1}{8}\varphi_j^3\right) \frac{\mathbf{N}_j^{1,t+1}}{\left|\mathbf{N}_j^{1,t}\right|} \\
&\qquad + m(e_j^3) \left(\frac{3}{8}\varphi_i + \frac{3}{8}\varphi_j^2 + \frac{1}{8}\varphi_j^1 + \frac{1}{8}\varphi_j^3\right) \frac{\mathbf{N}_j^{3,t+1}}{\left|\mathbf{N}_j^{3,t}\right|} \, .
\end{aligned} \tag{7.27}$$

The second integral term is approximated by taking $\varphi$ constant on the finite volume and by applying the Green's theorem we obtain

$$\begin{aligned}\int_{V_i} \varphi k\mathbf{N}\,\mathrm{d}\mathcal{S} = \varphi_i \int_{V_i} k\mathbf{N}\,\mathrm{d}\mathcal{S} = \varphi_i \int_{\partial V_i} \frac{\partial \mathbf{x}}{\partial \mathbf{n}_i}\,\mathrm{d}s = \varphi_i \int_{\partial V_i} \mathbf{n}_i\,\mathrm{d}s \\ \approx \varphi_i \left( \sum_{j\in N_\square(\mathbf{x}_i)} \int_{e_j^1} \frac{\mathbf{N}_j^{1,t+1}}{|\mathbf{N}_j^{1,t}|}\mathrm{d}s + \int_{e_j^3} \frac{\mathbf{N}_j^{3,t+1}}{|\mathbf{N}_j^{3,t}|}\mathrm{d}s \right) \\ \approx \varphi_i \left( \sum_{j\in N_\square(\mathbf{x}_i)} m(e_j^1)\frac{\mathbf{N}_j^{1,t+1}}{|\mathbf{N}_j^{1,t}|} + m(e_j^3)\frac{\mathbf{N}_j^{3,t+1}}{|\mathbf{N}_j^{3,t}|} \right) . \end{aligned} \tag{7.28}$$

Putting together (7.27) and (7.28), the last term on the right-hand side of (7.17) is approximated as

$$\begin{aligned}\int_{V_i} \mathbf{V}_T\,\mathrm{d}\mathcal{S} \approx \sum_{j\in N_\square(\mathbf{x}_i)} m(e_j^1)\left(-\frac{5}{8}\varphi_i + \frac{3}{8}\varphi_j^1 + \frac{1}{8}\varphi_j^2 + \frac{1}{8}\varphi_j^3\right)\frac{\mathbf{N}_j^{1,t+1}}{|\mathbf{N}_j^{1,t}|} \\ + m(e_j^3)\left(-\frac{5}{8}\varphi_i + \frac{3}{8}\varphi_j^2 + \frac{1}{8}\varphi_j^1 + \frac{1}{8}\varphi_j^3\right)\frac{\mathbf{N}_j^{3,t+1}}{|\mathbf{N}_j^{3,t}|} . \end{aligned} \tag{7.29}$$

In addition, since the relations (C.12), (C.13) for $\mathbf{N}_j^1$ and $\mathbf{N}_j^3$ hold in every time step $t$, the vector equation (7.29) can be added implicitly to the system, which is then solved.

In many applications, especially FEA, the requirements on the resulting mesh elements' shape properties are the uniformity of quad areas, edge lengths and right angles, i.e. elimination of degenerated quads. The tangential velocity $\mathbf{V_T}$ aims to uniform the quad areas, however, in order to satisfy the quad shape requirements, we have to consider either the edge lengths or the angles in addition to the area-oriented redistribution.

Therefore, we decide to incorporate an additional ad-hoc naive angle-oriented velocity $v_T$ at each $V_i$ that is implicitly added to the linear system for $\mathbf{x}_i$ in order to create an evolution preserving both the area and angle distribution of the quads. The velocity $v_T$ is computed as

$$v_T = \frac{m(V_i)}{\sum_{j\in N_\square(\mathbf{x}_i)} a_j^1 + a_j^3} \sum_{j\in N_\square(\mathbf{x}_i)} a_j^1(\mathbf{x}_j^1 - \mathbf{x}_j^0) + a_j^3(\mathbf{x}_j^3 - \mathbf{x}_j^0)$$

where $a_j^k$, $k \in \{1,3\}$ equals

$$a_j^k = \frac{1 + |u_j^k| \cos \frac{u_j^1 \cdot u_j^3}{|u_j^1||u_j^3|}}{|u_j^1||u_j^3|} ,$$

for $u_j^k = \mathbf{x}_j^k - \mathbf{x}_j^0$ being the quad edge vector.

With this choice of weights, $v_T$ direction dictates the movement of $\mathbf{x}_i$ in such way that obtuse angles at $\mathbf{x}_i$ are reduced causing enlargement of the acute angles around $\mathbf{x}_i$.

Putting together the equations (7.19), (7.21), (7.22) and (7.29), we get coefficients of the surface evolution linear system approximating solution of (7.3) at time $t$ for the unknown position vector coordinates $x_i^{t+1}$. The system matrix is highly sparse, in general non-symmetric. The solution for $x_i^{t+1}$ is obtained via sparse BiCGSTAB solver.

The evolution is stopped when the following stopping criterion is satisfied

$$\|d(\mathbf{x})\|_\infty \leq d_{th}$$

where $d_{th}$ is given threshold on the distance function values from the evolving surface $\mathcal{S}$ to $M_\triangle$.

## 7.5 Numerical experiments

We preliminary tested our approach on several input meshes. In general, we propose two approaches to the Skeleton Evolution Phase.

The first proposal `ALG_1` evolves each $S_i$ separately, imposing Dirichlet boundary conditions at the common boundaries. This strategy turns out to be useful when specific parts of the shape have to be modified while maintaining the prescribed boundaries. This is the case, for example, of boundary representation (BRep) of objects in solid modeling.

The second proposal `ALG_2` evolves the Topology-skeleton $S$ as whole. Naturally, in this case we glue all parts $S_i$ together, allowing the common boundaries to be moved, thus, producing visually appealing results with better vertex distribution around each $S_i$ boundary. Alternatively, the same strategy can be applied post-process to `ALG_1`.

In every example we combined the global area-oriented (GA) redistribution with the implicit local angle-oriented one.

For all the experiments we use reported edge length $h$, we fixed the parameters $c_1 = 0.03$, $c_2 = 0.002$ for the MCF weighting function $\epsilon$ in (7.8). The advection term parameter $c_3$ in function $\eta$ in (7.9) was set $c_3 = 2$ and the time step $\tau = 0.1$.

For the redistribution speed $\omega$ we chose the following strategy. At the beginning, we use $\omega = 1$ what results in favouring the local angle-oriented redistribution and after a few evolution iterations, we increase $\omega = 10$, thus, obtaining a good compromise between the mesh quad areas and their angles.

Table 7.1 Overview on the input triangular meshes and the output quadrangulations.

| | $M_\triangle$ | | | $M_\square$ | | | |
|---|---|---|---|---|---|---|---|
| | $\#V_{in}$ | $\#T_{in}$ | $K$ | $h$ | $\#V_{out}$ | $\#Q_{out}$ | $\#EV$ |
| `dolphin` | 7573 | 15142 | 13 | 0.040 | 2880 | 2878 | 74 |
| `fertility` | 19994 | 40000 | 39 | 0.025 | 6266 | 6272 | 228 |
| `hand` | 6607 | 13210 | 10 | 0.040 | 3792 | 3790 | 55 |
| `horse_2` | 8078 | 16152 | 11 | 0.030 | 4124 | 4122 | 62 |
| `plane` | 7470 | 14936 | 12 | 0.040 | 3402 | 3400 | 63 |
| `pliers` | 5110 | 10216 | 4 | 0.030 | 2414 | 2412 | 20 |
| `teddy` | 9548 | 19092 | 8 | 0.040 | 4281 | 4279 | 44 |
| `teddy_2` | 1029 | 2056 | 8 | 0.040 | 2640 | 2640 | 44 |
| `wolf` | 4712 | 9420 | 17 | 0.030 | 9798 | 9796 | 91 |

Informations on the input/output meshes considered in the examples are reported in Table 7.1. From the second to fourth column we report the number of vertices, triangles and number of patches $K$ produced in Phase 1 of the input mesh $M_\triangle$, The rest of the columns give insight on the output mesh $M_\square$ giving the information about the selected edge length $h$, corresponding number of vertices, quads and the number of extraordinary vertices $\#EV$.

### 7.5.1 Example 1: Mesh Quadrangulation

In this experiment, we preliminary tested our algorithm on different meshes, investigating the quality of the result in terms of uniformity and closeness to the input shape $M_\triangle$. Given as input the desired edge length $h$, validation of the results has been assessed on the following measures:

- $\bar{h}$ – mean edge length of the output mesh
- $\sigma_{\bar{h}}$ – standard deviation from $\bar{h}$
- $\bar{d} = \frac{1}{n}\sum_{i=1}^{n} |d(\mathbf{x}_i)|$ – the mean value of the distance function in points of $M_\square$, because when $\mathbf{x}_i$ lies in the same voxel as part of $M_\triangle$, $d(\mathbf{x}_i)$ is computed as the exact distance to $M_\triangle$.
- $\sigma_{\bar{d}}$ – standard deviation from $\bar{d}$
- $\bar{Q} = \frac{1}{n_Q}\sum_{i=1}^{n_Q} |Q_i|$ – mean area of the quads
- $\sigma_{\bar{Q}}$ – standard deviation from $\bar{Q}$

Table 7.2 Statistics of the preliminary results on the edge length, quad size, angle size and distance function values at mesh vertices.

| | $h$ | $\bar{h}$ | $\sigma_{\bar{h}}$ | $\bar{Q}$ | $\sigma_{\bar{Q}}$ | $\bar{\measuredangle}$ | $\sigma_{\bar{\measuredangle}}$ | $\bar{d}$ | $d_\infty$ |
|---|---|---|---|---|---|---|---|---|---|
| dolphin_ALG1 | 0.040 | $3.22e-02$ | $8.67e-03$ | $8.78e-04$ | $3.66e-05$ | 89.80 | 22.48 | $1.33e-05$ | $6.91e-04$ |
| dolphin_ALG2 | 0.040 | $3.22e-02$ | $8.59e-03$ | $9.09e-04$ | $4.13e-05$ | 89.87 | 20.04 | $1.52e-05$ | $1.16e-03$ |
| fertility_ALG1 | 0.025 | $1.40e-02$ | $3.69e-03$ | $1.61e-04$ | $1.01e-05$ | 89.63 | 24.68 | $1.55e-05$ | $2.89e-03$ |
| fertility_ALG2 | 0.025 | $1.45e-02$ | $4.37e-03$ | $1.74e-04$ | $1.14e-05$ | 89.67 | 26.27 | $-1.09e-05$ | $1.05e-02$ |
| hand_ALG1 | 0.040 | $3.10e-02$ | $7.38e-03$ | $8.46e-04$ | $1.83e-05$ | 89.90 | 18.31 | $-2.63e-05$ | $8.17e-03$ |
| hand_ALG2 | 0.040 | $3.02e-02$ | $7.54e-03$ | $8.23e-04$ | $1.71e-05$ | 89.90 | 16.50 | $-2.11e-05$ | $1.14e-02$ |
| horse_2_ALG1 | 0.030 | $2.65e-02$ | $7.72e-03$ | $5.92e-04$ | $2.06e-05$ | 89.73 | 20.61 | $-2.52e-05$ | $8.51e-03$ |
| horse_2_ALG2 | 0.030 | $2.63e-02$ | $6.80e-03$ | $6.05e-04$ | $1.65e-05$ | 89.73 | 20.54 | $-2.78e-05$ | $6.47e-03$ |
| plane_ALG1 | 0.040 | $2.81e-02$ | $8.24e-03$ | $6.70e-04$ | $2.89e-05$ | 89.86 | 18.65 | $2.89e-05$ | $1.62e-02$ |
| plane_ALG2 | 0.040 | $2.88e-02$ | $1.03e-02$ | $7.01e-04$ | $3.59e-05$ | 89.86 | 16.82 | $2.65e-05$ | $1.57e-02$ |
| pliers_ALG1 | 0.030 | $3.01e-02$ | $1.40e-02$ | $6.68e-04$ | $1.79e-05$ | 89.88 | 21.63 | $-1.13e-05$ | $5.60e-03$ |
| pliers_ALG2 | 0.030 | $3.04e-02$ | $1.43e-02$ | $6.80e-04$ | $1.90e-05$ | 89.90 | 20.67 | $-1.07e-05$ | $7.66e-03$ |
| teddy_ALG1 | 0.040 | $3.95e-02$ | $1.03e-02$ | $1.33e-03$ | $4.52e-05$ | 89.78 | 23.98 | $-7.63e-07$ | $1.17e-03$ |
| teddy_ALG2 | 0.040 | $3.97e-02$ | $1.09e-02$ | $1.36e-03$ | $4.36e-05$ | 89.74 | 21.83 | $-1.73e-04$ | $1.24e-02$ |
| wolf_ALG1 | 0.030 | $1.96e-02$ | $6.03e-03$ | $3.02e-04$ | $1.05e-05$ | 89.75 | 24.13 | $-9.75e-06$ | $7.30e-03$ |
| wolf_ALG2 | 0.030 | $1.88e-02$ | $5.57e-03$ | $3.04e-04$ | $9.28e-06$ | 89.81 | 19.97 | $-9.15e-06$ | $8.60e-03$ |

- $\bar{\measuredangle}$ – mean angle between edges over the mesh in degrees
- $\sigma_{\bar{\measuredangle}}$ – standard deviation from $\bar{\measuredangle}$

We created the topology-skeleton $S$ using the reported edge length $h$ an we run both the proposed algorithms `ALG_1` and then `ALG_2`. In Fig. 7.6 we report sample of the reconstructed meshes in alphabetical order as listed in Table 7.2. In the first column of Fig. 7.6 we visualize the input topological skeleton $S$ together with an outline of the input mesh $M_\triangle$. Each patch $S_i$ of $S$ is coloured in a different colour. The middle column illustrate the output of `ALG_1` coloured in false colours in range [blue, red] according to the quad areas of the mesh. One can notice that sometimes, the boundaries of the object lead to slightly different mean quad areas among the patches. Nevertheless, in the third column of Fig. 7.6 that illustrates the output of `ALG_2`, the reader can appreciate the uniform colour distribution for the quad areas, which was obtained also thanks to the movement of the patch boundaries, since all the patches $S_i$ were glued together. Reconstructions of more complicated meshes in terms of level of details and higher genus are represented by meshes `fertility, horse` and `wolf`. From a visual inspection we can observe that our approach successfully reconstructed the mouth and ears in `wolf` mesh as well as a higher-genus mesh `fertility`.

In addition to the qualitative evaluation illustrated in Fig. 7.6, we report in Table 7.2 indicators of the quad mesh properties.

### 7.5.2 Example 2: Different resolutions

In this experiment, we produced the topology-skeletons $S_i$ for `teddy` mesh using four different resolutions controlled by decreasing mesh size $h$ in the range $\{0.05, 0.04, 0.025, 0.01\}$ and we evolved the associated skeletons. Then we have decreased the input $M_\triangle$ mesh resolution density down to 10%, see `teddy_2` in Table 7.1, and repeated the evolutions. The corresponding results of `ALG_2` are reported in Figure 7.7 in the first and last column. Each resolution result in Fig. 7.7 is accompanied with a detail of the arm together with the original triangulation coloured in blue. From the reported results we can state that our algorithm is robust to the input-output mesh resolution, while its smoothness can be controlled by the parameters in (7.8) relative to the diffusion term in the evolution model.

Fig. 7.6 Example 1: Reconstruction results of different meshes. from left to right: input skeleton $S$ with outline of $M_{\triangle}$ in transparent, result of `ALG_1`, result of `ALG_2`.

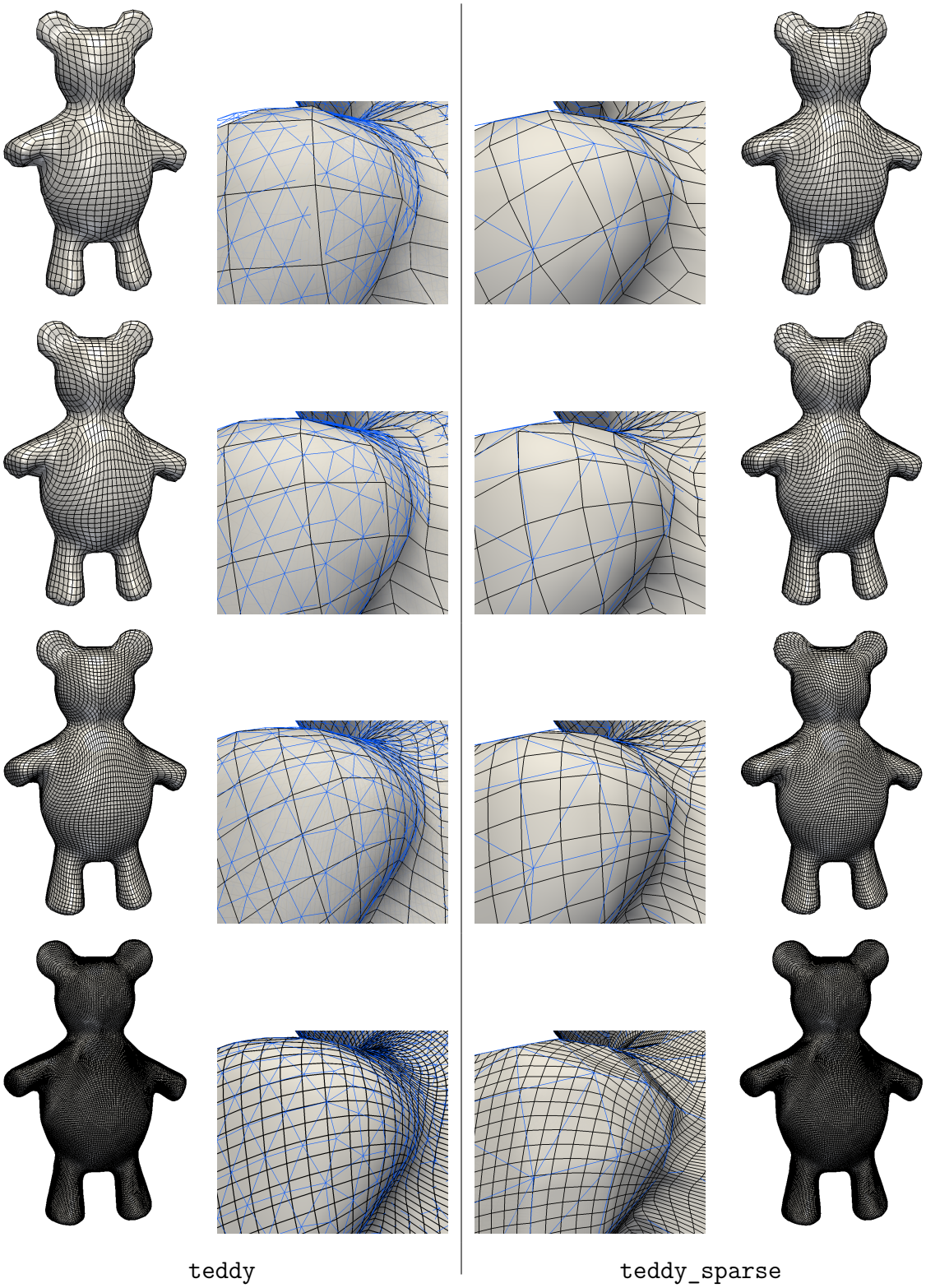



Fig. 7.7 Example 2: Different resolution results for `teddy` mesh (left) and its 90% down-sample (right). From top to bottom the results for chosen parameter $h = \{0.05, 0.04, 0.025, 0.1\}$ respectively together with zoom to the reconstructed mesh produced by `ALG_2` with original triangulation over-imposed in blue.

# Chapter 8

# Conclusions

In this work, we studied problems in the field of Geometry Processing. In particular, we focused on the 2-manifold partitioning which is rather a complex task. Preliminary to the object decomposition itself, a shape analysis that drives the variational partitioning problem is needed.

To that aim, we demonstrated as a simple particle flow can be efficiently used for the shape diameter analysis of a cloud of points representing the boundary of an object. Therefore, the proposed algorithm for SDF computation in Section 5.1 represents a valid alternative to the original ray-tracing-based proposal [108], which requires to be necessarily applied to meshes. We validated the proposed SDF Flow algorithm both in terms of accuracy and in terms of efficiency. The quality of the results are comparable to the ray-tracing based proposals.

Next, we proposed in Section 5.2 a novel definition of the affinity matrix depending on the mean curvature of the object considered, which eigen-decomposition provides us the spectral coordinates of the object's vertices. In particular, the eigenvectors corresponding to the first few smallest non-zero eigenvalues, represent the slowly-varying functions over the studied surface, capturing its main features and details.

Moreover, we proposed a sparsity-promoting variational method (Section 5.3) to produce compressed functions $L_p$CMs, which are quasi-eigenfunctions of the Laplacian operator. We proved that the generated functions are highly localized in space and the size of their support depends on the sparsity parameter $p$ and on the penalty parameter $\mu$. An Augmented Lagrangian method was applied to solve this non-convex non-differentiable optimization problem, yielding an iterative algorithm with efficient solutions to sub-problems. Also this compact support basis proves to be very useful for spectral shape processing.

Apart from the latter two methods of shape analysis, which lead to a partitioning mimicking the human perception; Shape Diameter Function gives rise to the thickness-based partitioning. The main goal of thickness-based partitioning is to construct a map of thickness of the object based on the SDF values.

Using the above-mentioned single-/multi-channel functions as a shape attribute, we presented different proposals for the variational partitioning of an object, represented by a triangular mesh, into $K$ separated regions. Our work is based on recent advances in sparsity-inducing penalties that have been successfully applied in image processing [69]. The multi-channel object partitioning framework in Section 6.1 is based on a variational formulation where we introduced a novel shape metric, allowing the capture of more subtle details of the segmented boundaries than the traditional $L_1$ metric. The sparsity imposed in the variational formulation represents the key aspect for a successful shape partitioning. In order to deal with function inhomogeneity the functional has been enriched with a smooth regularizer. Thus the resulting variational models hold the potential both for piecewise-constant and piecewise-smooth representations of the partitioning function. We propose a fast iterative algorithm to accurately approximate the minimizer of the partitioning functional. The mathematical framework is robust and efficient, however, the $L_p$ norm term in the functional leads to a non-convex optimization problem which solution can be stalled in local minima.

Therefore, we explored the replacement of the $L_p$ penalty term with another sparsity inducing, non-convex, but parametric penalty function, so that to control the convexity of the overall functional and benefit of the convex optimization tools. In particular, we presented a new variational CNC model in Section 6.3 for multiphase segmentation of real-valued functions on manifolds, where a non-convex regularization term allows for penalizing the non-smoothness of the inner segmented parts and preserves the length of the boundaries. The CNC strategy is generalized for unstructured three dimensional meshes/graphs, and theoretical local conditions of convexity are derived for the proposed model where we explicitly used the manifold's geometry. This result allows to benefit from the maintenance of the strict (or, better, strong) convexity of the optimization problem to be solved.

Towards the solutions of the non-convex non-smooth and CNC functionals, we propose a fast iterative algorithm to accurately approximate the local minimizer of the partitioning functional as well as an ad hoc iterative ADMM-based algorithm together with a specific multivariate proximal map to solve the minimization problem.

An advantage of these proposals with respect to other methods is that the solution of the partitioning optimization problem is independent on the number of partitions $K$

required, which is only exploited in the post-processing second step. However, the decomposition step is currently implemented as a naive $K$-means-alike clusterization algorithm which coincides with thresholding in case of single channel shape attribute. Therefore, we would like to investigate a more convenient, convex, thresholding formulation.

From the computational point of view, the efficiency can be definitely improved by CPU/GPU parallelization, and utilization of register-type variables when possible. In case of the SDF map, exploitation of greedy strategies like for example the ASDF approach proposed in [64] is in consideration.

The abilities of the proposed algorithms for object partitioning have been shown for the shape diameter attribute and for spectral decomposition. However, our formulations are quite general, and other surface attributes can be used instead for a suitable initialization of the variational problem, thus obtaining a generalized partitioning framework. The idea of generalized framework point in two directions for the future research:

- Shape Analysis – exploit different shape attributes and their robustness in shape partitioning.
- Shape Partitioning – study of the partitioning model and its limitations, e.g. incorporate into the model more sophisticated edge-detection functions based both on the geometry and function edges, thus, control better the effect of smooth regularizer for the inner segmented parts.

We concluded this work with a possible application of the shape partitioning in area of the surface-patch quadrangulation. To that aim, we utilize the 0-genus partitioning and supervised Lagrangian-type evolution model to approximate the original shape. The evolution model is robust, however, in order to preserve a uniform quad distribution among the partitioning patches, we can consider just 0-genus partitions of one boundary. Consideration of a cylindrical 0-genus patches, produced by a possibly more sophisticated partitioning method, is left for further investigation. The resulting meshes obtained by our algorithm can be useful for FEM computations, since we aim for a compromise between the uniform quad areas and the right angles among the mesh. On the other hand, a mesh that may be used for CAD purposed leads to a curvature-oriented redistribution of the points, with strict reduction of the extraordinary vertices.

# References


[1] Agathos, A., Pratikakis, I., Perantonis, S., Sapidis, N., and Azariadis, P. (2007). 3d mesh segmentation methodologies for cad applications. *Computer-Aided Design and Applications*, 4(6):827–841.

[2] Arthur, D. and Vassilvitskii, S. (2007). K-means++: The advantages of careful seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '07, pages 1027–1035, Philadelphia, PA, USA. Society for Industrial and Applied Mathematics.

[3] Asafi, S., Goren, A., and Cohen-Or, D. (2013). Weak Convex Decomposition by Lines-of-sight. *Computer Graphics Forum*.

[4] Attene, M., Katz, S., Mortara, M., Patane, G., Spagnuolo, M., and Tal, A. (2006). Mesh segmentation - a comparative study. In *Proceedings of the IEEE International Conference on Shape Modeling and Applications 2006*, SMI '06, pages 7–, Washington, DC, USA. IEEE Computer Society.

[5] Au, O. K.-C., Zheng, Y., Chen, M., Xu, P., and Tai, C.-L. (2011). Mesh segmentation with concavity-aware fields. *IEEE Transactions on Visualization & Computer Graphics*, 18:1125–1134.

[6] Barekat, F. (2014). On the consistency of compressed modes for variational problems associated with the schrödinger operator. *SIAM Journal on Mathematical Analysis*, 46(5):3568–3577.

[7] Barekat, F., Caflisch, R., and Osher, S. (2017). On the support of compressed modes. *SIAM J. Math. Analysis*, 49(4):2573–2590.

[8] Beck, A. and Teboulle, M. (2009a). A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM J. Img. Sci.*, 2(1):183–202.

[9] Beck, A. and Teboulle, M. (2009b). Gradient-based algorithms with applications to signal recovery. *Convex optimization in signal processing and communications*, pages 42–88.

[10] Benhabiles, H., Lavoué, G., Vandeborre, J.-P., and Daoudi, M. (2011). Learning boundary edges for 3d-mesh segmentation. *Computer Graphics Forum*, 30(8):2170–2182.

[11] Berger, M., Tagliasacchi, A., Seversky, L. M., Alliez, P., Levine, J. A., Sharf, A., and Silva, C. (2014). State of the art in surface reconstruction from point clouds. *Eurographics STAR (Proc. of EG'14)*.

[12] Bertsekas, D. (1996). *Constrained Optimization and Lagrange Multiplier Methods*. Athena scientific series in optimization and neural computation. Athena Scientific.

[13] Blake, A. and Zisserman, A. (1987). *Visual Reconstruction*. MIT Press, Cambridge, MA, USA.

[14] Boier-Martin, I., Rushmeier, H., and Jin, J. (2004). Parameterization of triangle meshes over quadrilateral domains. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, SGP '04, pages 193–203, New York, NY, USA. ACM.

[15] Bommes, D., Lévy, B., Pietroni, N., Puppo, E., Silva, C., Tarini, M., and Zorin, D. (2013). Quad-mesh generation and processing: A survey. *Comput. Graph. Forum*, 32(6):51–76.

[16] Bommes, D., Zimmer, H., and Kobbelt, L. (2009). Mixed-integer quadrangulation. *ACM Trans. Graph.*, 28(3):77:1–77:10.

[17] Botsch, M., Kobbelt, L., Pauly, M., Alliez, P., and Levy, B. (2010). *Polygon Mesh Processing*. Ak Peters Series. Taylor & Francis.

[18] Boyd, S., Parikh, N., Chu, E., Peleato, B., and Eckstein, J. (2011). Distributed optimization and statistical learning via the alternating direction method of multipliers. *Found. Trends Mach. Learn.*, 3(1):1–122.

[19] Boyd, S. and Vandenberghe, L. (2004). *Convex Optimization*. Cambridge University Press, New York, NY, USA.

[20] Bredies, K. (2009). A forward–backward splitting algorithm for the minimization of non-smooth convex functionals in banach space. *Inverse Problems*, 25(1):015005.

[21] Bronstein, A. M., Choukroun, Y., Kimmel, R., and Sela, M. (2016). Consistent discretization and minimization of the L1 norm on manifolds. *CoRR*, abs/1609.05434.

[22] Cai, X., Chan, R., and Zeng, T. (2013). A two-stage image segmentation method using a convex variant of the mumford–shah model and thresholding. *SIAM Journal on Imaging Sciences*, 6(1):368–390.

[23] Candès, E. J., Wakin, M. B., and Boyd, S. P. (2008). Enhancing sparsity by reweighted $\ell_1$ minimization. *Journal of Fourier Analysis and Applications*, 14(5):877–905.

[24] Casciola, G., Lazzaro, D., Montefusco, L. B., and Morigi, S. (2006). Shape preserving surface reconstruction using locally anisotropic radial basis function interpolants. *Comput. Math. Appl.*, 51(8):1185–1198.

[25] Catmull, E. and Clark, J. (1998). Seminal graphics. chapter Recursively Generated B-spline Surfaces on Arbitrary Topological Meshes, pages 183–188. ACM, New York, NY, USA.

[26] Chambolle, A. and Pock, T. (2011). A first-order primal-dual algorithm for convex problems with applications to imaging. *Journal of Mathematical Imaging and Vision*, 40(1):120–145.

[27] Chan, T. F., Esedoglu, S., and Nikolova, M. (2006). Algorithms for finding global minimizers of image segmentation and denoising models. Technical report, SIAM Journal On Applied Mathematics.

[28] Chan, T. F. and Mulet, P. (1999). On the convergence of the lagged diffusivity fixed point method in total variation image restoration. *SIAM J. Numer. Anal.*, 36(2):354–367.

[29] Chan, T. F. and Vese, L. A. (2001). Active contours without edges. *Trans. Img. Proc.*, 10(2):266–277.

[30] Chen, P. Y. and Selesnick, I. W. (2014). Group-sparse signal denoising: Non-convex regularization, convex optimization. *IEEE Transactions on Signal Processing*, 62(13):3464–3478.

[31] Chen, X., Golovinskiy, A., and Funkhouser, T. (2009). A benchmark for 3d mesh segmentation. *ACM Trans. Graph.*, 28(3):73:1–73:12.

[32] Chouzenoux, E., Jezierska, A., Pesquet, J.-C., and Talbot, H. (2013). A majorize-minimize subspace approach for \ell_2-\ell_0 image regularization. *SIAM Journal on Imaging Sciences*, 6(1):563–591.

[33] Cignoni, P., Callieri, M., Corsini, M., Dellepiane, M., Ganovelli, F., and Ranzuglia, G. (2008a). MeshLab: an Open-Source Mesh Processing Tool. In Scarano, V., Chiara, R. D., and Erra, U., editors, *Eurographics Italian Chapter Conference*. The Eurographics Association.

[34] Cignoni, P., Callieri, M., Corsini, M., Dellepiane, M., Ganovelli, F., and Ranzuglia, G. (2008b). MeshLab: an Open-Source Mesh Processing Tool. In Scarano, V., Chiara, R. D., and Erra, U., editors, *Eurographics Italian Chapter Conference*. The Eurographics Association.

[35] Cohen-Steiner, D., Alliez, P., and Desbrun, M. (2004). Variational shape approximation. *ACM Trans. Graph.*, 23(3):905–914.

[36] Combettes, P. L. and Wajs, V. R. (2005). Signal recovery by proximal forward-backward splitting. *Multiscale Modeling & Simulation*, 4(4):1168–1200.

[37] Daniels, J., Silva, C., and Cohen, E. (2009). Semi-regular quadrilateral-only remeshing from simplified base domains. *Computer Graphics Forum*, 28(5):1427–1435.

[38] Daniels, J., Silva, C. T., Shepherd, J., and Cohen, E. (2008). Quadrilateral mesh simplification. *ACM Trans. Graph.*, 27(5):148:1–148:9.

[39] Dong, S., Bremer, P.-T., Garland, M., Pascucci, V., and Hart, J. C. (2006). Spectral surface quadrangulation. *ACM Trans. Graph.*, 25(3):1057–1066.

[40] Du, Q., Faber, V., and Gunzburger, M. (1999). Centroidal voronoi tessellations: Applications and algorithms. *SIAM Rev.*, 41(4):637–676.

[41] Dziuk, G. and Elliott, C. M. (2013). Finite element methods for surface pdes. *Acta Numerica*, 22:289–396.

[42] Eck, M., DeRose, T., Duchamp, T., Hoppe, H., Lounsbery, M., and Stuetzle, W. (1995). Multiresolution analysis of arbitrary meshes. In *Proceedings of the 22Nd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '95, pages 173–182, New York, NY, USA. ACM.

[43] Floater, M. S. and Hormann, K. (2005). *Surface Parameterization: a Tutorial and Survey*, pages 157–186. Springer Berlin Heidelberg, Berlin, Heidelberg.

[44] Glowinski, R., Osher, S. J., and Yin, W. (2017). *Splitting Methods in Communication, Imaging, Science, and Engineering*. Springer Publishing Company, Incorporated, 1st edition.

[45] Goldstein, T. and Osher, S. (2009). The split bregman method for l1-regularized problems. *SIAM Journal on Imaging Sciences*, 2(2):323–343.

[46] Gorodnitsky, I. F. and Rao, B. D. (1992). A new iterative weighted norm minimization algorithm and its applications. In *[1992] IEEE Sixth SP Workshop on Statistical Signal and Array Processing*, pages 412–415.

[47] Gower, J. and Dijksterhuis, G. (2004). *Procrustes Problems*. Oxford Statistical Science Series. OUP Oxford.

[48] Gray, A. (1997). *Modern Differential Geometry of Curves and Surfaces with Mathematica*. CRC Press, Inc., Boca Raton, FL, USA, 2nd edition.

[49] Gu, X. and Yau, S.-T. (2003). Global conformal surface parameterization. In *Proceedings of the 2003 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, SGP '03, pages 127–137, Aire-la-Ville, Switzerland, Switzerland. Eurographics Association.

[50] Guskov, I., Vidimče, K., Sweldens, W., and Schröder, P. (2000). Normal meshes. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '00, pages 95–102, New York, NY, USA. ACM Press/Addison-Wesley Publishing Co.

[51] Heider, P., Pierre-Pierre, A., Li, R., Mueller, R., and Grimm, C. (2012). Comparing local shape descriptors. *The Visual Computer*, 28(9):919–929.

[52] Hesse, R. and Luke, D. R. (2013). Nonconvex notions of regularity and convergence of fundamental algorithms for feasibility problems. *SIAM Journal on Optimization*, 23(4):2397–2419.

[53] Huang, J., Zhang, M., Ma, J., Liu, X., Kobbelt, L., and Bao, H. (2008). Spectral quadrangulation with orientation and alignment control. *ACM Trans. Graph.*, 27(5):147:1–147:9.

[54] Hunter, D. R. and Lange, K. (2004). A tutorial on mm algorithms. *The American Statistician*, 58(1):30–37.

[55] Huska, M., Lanza, A., Morigi, S., and Sgallari, F. (2017a). Convex non-convex segmentation over surfaces. In Lauze, F., Dong, Y., and Dahl, A. B., editors, *Scale Space and Variational Methods in Computer Vision: 6th International Conference, SSVM 2017, Kolding, Denmark, June 4-8, 2017, Proceedings*, pages 348–360, Cham. Springer International Publishing.

[56] Huska, M., Lazzaro, D., and Morigi, S. (2018). Shape partitioning via $\mathrm{l}_p$ compressed modes. *to appear in Journal of Mathematical Imaging and Vision*.

[57] Huska, M., Medla, M., Mikula, K., and Morigi, S. (2017b). Patch-surface quadrangulation. in preparation.

[58] Huska, M. and Morigi, S. (2017a). A meshless strategy for shape diameter analysis. *The Visual Computer*, 33(3):303–315.

[59] Huska, M. and Morigi, S. (2017b). Sparsity-inducing variational shape partitioning. *Electronic Transactions on Numerical Analysis journal*, 46:36–54.

[60] Jakob, W., Tarini, M., Panozzo, D., and Sorkine-Hornung, O. (2015). Instant field-aligned meshes. *ACM Trans. Graph.*, 34(6):189:1–189:15.

[61] Kaick, O. V., Fish, N., Kleiman, Y., Asafi, S., and Cohen-OR, D. (2014). Shape segmentation by approximate convexity analysis. *ACM Trans. Graph.*, 34(1):4:1–4:11.

[62] Kalogerakis, E., Hertzmann, A., and Singh, K. (2010). Learning 3d mesh segmentation and labeling. *ACM Trans. Graph.*, 29(4):102:1–102:12.

[63] Katz, S. and Tal, A. (2003). Hierarchical mesh decomposition using fuzzy clustering and cuts. *ACM Trans. Graph.*, 22(3):954–961.

[64] Kovacic, M., Guggeri, F., Marras, S., and Scateni, R. (2010). Fast approximation of the shape diameter function. In *Proceedings Workshop on Computer Graphics, Computer Vision and Mathematics (GraVisMa)*, volume 5, pages 65–72.

[65] Lai, R. and Osher, S. (2014). A splitting method for orthogonality constrained problems. *Journal of Scientific Computing*, 58(2):431–449.

[66] Lai, Y. K., Jin, M., Xie, X., He, Y., Palacios, J., Zhang, E., Hu, S. M., and Gu, X. (2010). Metric-driven rosy field design and remeshing. *IEEE Transactions on Visualization and Computer Graphics*, 16(1):95–108.

[67] Lanza, A., Morigi, S., Selesnick, I., and Sgallari, F. (2017). Nonconvex nonsmooth optimization via convex—nonconvex majorization—minimization. *Numer. Math.*, 136(2):343–381.

[68] Lanza, A., Morigi, S., and Sgallari, F. (2015). *Convex Image Denoising via Non-Convex Regularization*, pages 666–677. Springer International Publishing, Cham.

[69] Lanza, A., Morigi, S., and Sgallari, F. (2016a). Constrained $\mathrm{tv}_p$–$\ell_2$ model for image restoration. *J. Sci. Comput.*, 68(1):64–91.

[70] Lanza, A., Morigi, S., and Sgallari, F. (2016b). Convex image denoising via non-convex regularization with parameter selection. *Journal of Mathematical Imaging and Vision*, 56(2):195–220.

[71] Lee, A. W. F., Sweldens, W., Schröder, P., Cowsar, L., and Dobkin, D. (1998). Maps: Multiresolution adaptive parameterization of surfaces. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '98, pages 95–104, New York, NY, USA. ACM.

[72] Lellmann, J. and Schnörr, C. (2011). Continuous multiclass labeling approaches and algorithms. *CoRR*, abs/1102.5448.

[73] Lévy, B. and Liu, Y. (2010). Lp centroidal voronoi tessellation and its applications. *ACM Trans. Graph.*, 29(4):119:1–119:11.

[74] Lévy, B. and Zhang, H. R. (2010). Spectral mesh processing. In *ACM SIGGRAPH 2010 Courses*, SIGGRAPH '10, pages 8:1–8:312, New York, NY, USA. ACM.

[75] Lewis, A. S., Luke, D. R., and Malick, J. (2009). Local linear convergence for alternating and averaged nonconvex projections. *Foundations of Computational Mathematics*, 9(4):485–513.

[76] Lien, J.-M. and Amato, N. M. (2007). Approximate convex decomposition of polyhedra. In *Proceedings of the 2007 ACM Symposium on Solid and Physical Modeling*, SPM '07, pages 121–131, New York, NY, USA. ACM.

[77] Lin, J., Jin, X., Fan, Z., and Wang, C. C. L. (2008). Automatic polycube-maps. In *Proceedings of the 5th International Conference on Advances in Geometric Modeling and Processing*, GMP'08, pages 3–16, Berlin, Heidelberg. Springer-Verlag.

[78] Liu, R. and Zhang, H. (2004). Segmentation of 3d meshes through spectral clustering. In *Pacific Conference on Computer Graphics and Applications*, pages 298–305. IEEE Computer Society.

[79] Liu, R. and Zhang, H. (2007). Mesh segmentation via spectral embedding and contour analysis. *Computer Graphics Forum*, 26(3):385–394.

[80] Lu, Z. (2012). Iterative Reweighted Minimization Methods for *l_p* Regularized Unconstrained Nonlinear Programming. *ArXiv e-prints*.

[81] MacKay, D. J. C. (2002). *Information Theory, Inference & Learning Algorithms*. Cambridge University Press, New York, NY, USA.

[82] Meyer, M., Desbrun, M., Schröder, P., and Barr, A. H. (2003). *Discrete Differential-Geometry Operators for Triangulated 2-Manifolds*, pages 35–57. Springer Berlin Heidelberg, Berlin, Heidelberg.

[83] Mikula, K., Remešíková, M., Sarkoci, P., and Ševčovič, D. (2014). Manifold evolution with tangential redistribution of points. *SIAM Journal on Scientific Computing*, 36(4):A1384–A1414.

[84] Morigi, S. and Rucci, M. (2014). Multilevel mesh simplification. *The Visual Computer*, 30(5):479–492.

[85] Morigi, S., Rucci, M., and Sgallari, F. (2011). Nonlocal surface fairing. In *Scale Space and Variational Methods in Computer Vision - Third International Conference, SSVM 2011, Ein-Gedi, Israel, May 29 - June 2, 2011, Revised Selected Papers*, pages 38–49.

[86] Mumford, D. and Shah, J. (1989). Optimal approximations by piecewise smooth functions and associated variational problems. *Comm. on Pure and Applied Mathematics*, 42(5):577–685.

[87] Nesterov, Y. (1983). A method of solving a convex programming problem with convergence rate O(1/sqr(k)). *Soviet Mathematics Doklady*, 27:372–376.

[88] Neumann, T., Varanasi, K., Theobalt, C., Magnor, M., and Wacker, M. (2014). Compressed manifold modes for mesh processing. *Comput. Graph. Forum*, 33(5):35–44.

[89] Nikolova, M. (1998). Estimation of binary images by minimizing convex criteria. In *Proceedings 1998 International Conference on Image Processing. ICIP98 (Cat. No.98CB36269)*, volume 2, pages 108–112 vol.2.

[90] Nikolova, M. and Chan, R. H. (2007). The equivalence of half-quadratic minimization and the gradient linearization iteration. *IEEE Trans. Image Processing*, 16(6):1623–1627.

[91] Nikolova, M., Ng, M. K., and Tam, C. (2010). Fast nonconvex nonsmooth minimization methods for image restoration and reconstruction. *IEEE Trans. Image Processing*, 19(12):3073–3088.

[92] Ozoliņš, V., Lai, R., Caflisch, R., and Osher, S. (2013). Compressed modes for variational problems in mathematics and physics. *Proceedings of the National Academy of Sciences*, 110(46):18368–18373.

[93] Panozzo, D., Lipman, Y., Puppo, E., and Zorin, D. (2012). Fields on symmetric surfaces. *ACM Transactions on Graphics*, 31(4).

[94] Parikh, N. and Boyd, S. (2014). Proximal algorithms. *Found. Trends Optim.*, 1(3):127–239.

[95] Patanè, G., Spagnuolo, M., and Falcidieno, B. (2004). Para-graph: Graph-based parameterization of triangle meshes with arbitrary genus. *Comput. Graph. Forum*, 23(4):783–797.

[96] Patanè, G., Spagnuolo, M., and Falcidieno, B. (2007). Families of cut-graphs for bordered meshes with arbitrary genus. *Graphical Models*, 69(2):119–138.

[97] Pietroni, N., Tarini, M., and Cignoni, P. (2010). Almost isometric mesh parameterization through abstract domains. *IEEE Transaction on Visualization and Computer Graphics*, 16(4):621–635.

[98] Pinkall, U. and Polthier, K. (1993). Computing discrete minimal surfaces and their conjugates. *Experiment. Math.*, 2(1):15–36.

[99] Pock, T., Chambolle, A., Cremers, D., and Bischof, H. (2009). A convex relaxation approach for computing minimal partitions. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 810–817.

[100] Reuter, M., Biasotti, S., Giorgi, D., Patanè, G., and Spagnuolo, M. (2009). Discrete laplace-beltrami operators for shape analysis and segmentation. *Computers & Graphics*, 33(3):381–390.

[101] Rodriguez, P. and Wohlberg, B. (2006). An iteratively reweighted norm algorithm for total variation regularization. In *2006 Fortieth Asilomar Conference on Signals, Systems and Computers*, pages 892–896.

[102] Rolland-Nevière, X., Doërr, G., and Alliez, P. (2013). Robust diameter-based thickness estimation of 3d objects. *Graphical Models*, 75(6):279 – 296.

[103] Rudin, L. I., Osher, S., and Fatemi, E. (1992). Nonlinear total variation based noise removal algorithms. *Phys. D*, 60(1-4):259–268.

[104] Schonemann, P. H. (1985). On the formal differentiation of traces and determinants. *Multivariate Behavioral Research*, 20(2):113–139. PMID: 26771405.

[105] Sebastien Valette, Remy Prost, J. M. C. (2007). Generic remeshing of 3d triangular meshes with metric-dependent discrete voronoi diagrams. *IEEE Transactions on Visualization & Computer Graphics*, 14:369–381.

[106] Selesnick, I. W. and Bayram, I. (2014). Sparse signal estimation by maximally sparse convex optimization. *Trans. Sig. Proc.*, 62(5):1078–1092.

[107] Shamir, A. (2008). A survey on Mesh Segmentation Techniques. *Computer Graphics Forum*.

[108] Shapira, L., Shamir, A., and Cohen-Or, D. (2008). Consistent mesh partitioning and skeletonisation using the shape diameter function. *Vis. Comput.*, 24(4):249–259.

[109] Shor, N. Z., Kiwiel, K. C., and Ruszcayǹski, A. (1985). *Minimization Methods for Non-differentiable Functions*. Springer-Verlag New York, Inc., New York, NY, USA.

[110] Steiner, D. and Fischer, A. (2002). Cutting 3d freeform objects with genus-n into single boundary surfaces using topological graphs. In *Proceedings of the Seventh ACM Symposium on Solid Modeling and Applications*, SMA '02, pages 336–343, New York, NY, USA. ACM.

[111] Strekalovskiy, E., Chambolle, A., and Cremers, D. (2012). A convex representation for the vectorial mumford-shah functional. In *CVPR*, pages 1712–1719. IEEE Computer Society.

[112] Strong, D. and Chan, T. (2003). Edge-preserving and scale-dependent properties of total variation regularization. *Inverse Problems*, 19(6):S165.

[113] Tarini, M., Hormann, K., Cignoni, P., and Montani, C. (2004). Polycube-maps. *ACM Trans. Graph.*, 23(3):853–860.

[114] Tarini, M., Pietroni, N., Cignoni, P., Panozzo, D., and Puppo, E. (2010). Practical quad mesh simplification. *Computer Graphics Forum (Special Issue of Eurographics 2010 Conference)*, 29(2):407–418.

[115] Theologou, P., Pratikakis, I., and Theoharis, T. (2017). Unsupervised spectral mesh segmentation driven by heterogeneous graphs. *IEEE transactions on pattern analysis and machine intelligence*, 39(2):397–410.

[116] Tong, Y., Alliez, P., Cohen-Steiner, D., and Desbrun, M. (2006). Designing Quadrangulations with Discrete Harmonic Forms. In Sheffer, A. and Polthier, K., editors, *Symposium on Geometry Processing*. The Eurographics Association.

[117] Tournois, J., Wormser, C., Alliez, P., and Desbrun, M. (2009). Interleaving delaunay refinement and optimization for practical isotropic tetrahedron mesh generation. *ACM Trans. Graph.*, 28(3):75:1–75:9.

[118] Vallet, B. and Levy, B. (2008). Spectral Geometry Processing with Manifold Harmonics. *Computer Graphics Forum*.

[119] Vese, L. A. and Guyader, C. L. (2015). *Variational Methods in Image Processing*. Chapman & Hall/CRC.

[120] Villa, S., Salzo, S., Baldassarre, L., and Verri, A. (2013). Accelerated and inexact forward-backward algorithms. *SIAM Journal on Optimization*, 23(3):1607–1633.

[121] Vogel, C. R. and Oman, M. E. (1996). Iterative methods for total variation denoising. *SIAM J. SCI. COMPUT*, 17:227–238.

[122] Ševčovič, D. (2016). *Direct Lagrangian Methods for Solving Mean Curvature Flows and their Applications*. IRIS – Vydavateľstvo a tlač, s.r.o., Bratislava, Slovakia.

[123] Wang, Y., Yin, W., and Zeng, J. (2015). Global Convergence of ADMM in Nonconvex Nonsmooth Optimization. *ArXiv e-prints*.

[124] Wolke, R. and Schwetlick, H. (1988). Iteratively reweighted least squares: Algorithms, convergence analysis, and numerical comparisons. *SIAM Journal on Scientific and Statistical Computing*, 9(5):907–921.

[125] Wu Leif Kobbelt, J. (2005). Structure recovery via hybrid variational surface approximation. *Computer Graphics Forum*, 24(3):277–284.

[126] Xia, J., Garcia, I., He, Y., Xin, S.-Q., and Patow, G. (2011). Editable polycube map for gpu-based subdivision surfaces. In *Symposium on Interactive 3D Graphics and Games*, I3D '11, pages 151–158, New York, NY, USA. ACM.

[127] Yan, D.-M., Levy, B., Liu, Y., Sun, F., and Wang, W. (2009). Isotropic Remeshing with Fast and Exact Computation of Restricted Voronoi Diagram. *Computer Graphics Forum*.

[128] Yan, D.-M., Wang, W., Liu, Y., and Yang, Z. (2012). Variational mesh segmentation via quadric surface fitting. *Comput. Aided Des.*, 44(11):1072–1082.

[129] Yin, K. and Osher, S. (2013). On the completeness of the compressed modes in the eigenspace. Technical report, CALIFORNIA UNIV LOS ANGELES DEPT OF MATHEMATICS.

[130] Zhang, J., Zheng, J., Wu, C., and Cai, J. (2012). Variational mesh decomposition. *ACM Trans. Graph.*, 31(3):21:1–21:14.

[131] Zhao, H. (2005). A fast sweeping method for eikonal equations. *Math. Comput.*, 74(250):603–627.

[132] Zhao, H.-K., Osher, S., and Fedkiw, R. (2001). Fast surface reconstruction using the level set method. In *Proceedings IEEE Workshop on Variational and Level Set Methods in Computer Vision*, pages 194–201.

[133] Zuo, W., Meng, D., Zhang, L., Feng, X., and Zhang, D. (2013). A generalized iterated shrinkage algorithm for non-convex sparse coding. In *The IEEE International Conference on Computer Vision (ICCV)*.

# Appendix A

# Proofs in Section 5.3 on $L_p$ Compressed Manifold Modes

**Proof of Theorem 1.**

*Proof.* The proof is decomposed in four steps.

**Step 1:**

We first derived the Euler-Lagrange equations for (5.17). For any function $u$, let s(u) denote an element of subdifferential of $|u|^p$, that is:

$$s(u) = sign(u) \cdot p \cdot |u|^{p-1} . \tag{A.1}$$

The solutions of (5.17) are weak solutions of the following system of nonlinear boundary value problem:

$$\frac{1}{\mu} s(\psi_i) - 2\lambda_{ii}\psi_i - \Delta\psi_i - \sum_{j\neq i} \lambda_{ij}\psi_j = 0, \quad i = 1, .., N \text{ on } \Omega \tag{A.2}$$

where $\lambda_{ij}$, with $\lambda_{ij} = \lambda_{ji}$ are Lagrange multipliers corresponding to orthonormality constraints:

$$\int_\Omega \psi_i^2 dx = 1 \quad \text{and} \quad \int_\Omega \psi_i\psi_j dx = 0, \quad \text{for} \quad i, j = 1, ..., N, \ j \neq i \ . \tag{A.3}$$

**Step 2:** Upper bounds for $\lambda_{ii}$, $\|\psi_i\|_p^p$ and $\|\nabla\psi_i\|_2$

For each $i$ multiply both sides of equation (A.2) by $\psi_i(x)$ and integrate over domain $\Omega$:

$$\int_\Omega \frac{1}{\mu} s(\psi_i)\psi_i dx - 2\lambda_{ii}\int_\Omega \psi_i\psi_i dx - \int_\Omega \Delta\psi_i\psi_i dx - \sum_{j\neq i}\lambda_{ij}\int_\Omega \psi_i\psi_j dx = 0 \ . \tag{A.4}$$

By using orthonormality conditions (A.3), we can rewrite the above equation as:

$$\int_\Omega \frac{1}{\mu} s(\psi_i)\psi_i dx - 2\lambda_{ii} - \int_\Omega \Delta\psi_i\psi_i dx = 0 \tag{A.5}$$

that, using integration by parts and zero boundary conditions on $\Omega$, implies that

$$\int_\Omega \frac{1}{\mu} s(\psi_i)\psi_i dx - 2\lambda_{ii} + \int_\Omega |\nabla\psi_i|^2 \, dx = 0 \tag{A.6}$$

and then

$$\lambda_{ii} = \frac{1}{2\mu}\int_\Omega s(\psi_i)\psi_i dx + \frac{1}{2}\int_\Omega |\nabla\psi_i|^2 \, dx \; . \tag{A.7}$$

By using definition (A.1), relation (A.7) can be reformulated as:

$$\lambda_{ii} = \frac{1}{2\mu}\int_\Omega p\,|\psi_i|^p \, dx + \frac{1}{2}\int_\Omega |\nabla\psi_i|^2 \, dx \; . \tag{A.8}$$

From Proposition 4, we know that the first compressed mode $\psi$ has support whose volume satisfy (5.18). It follows that for $\mu$ sufficiently small and $0 < p \leq 1$, the $N$ disjoint copies (i.e. translates) of $\psi$ can be placed in $\Omega$, and these $N$ functions are a solution for problem (5.17). Therefore, in view of Proposition 4, there exist $\mu_0$ (depending on values of $p$,$N$, and $d$) such that for $\mu < \mu_0$:

$$\sum_{i=1}^{N}\int_\Omega \frac{1}{\mu}|\psi_i|^p \, dx + \sum_{i=1}^{N}\frac{1}{2}\int_\Omega |\nabla\psi_i|^2 \, dx \leq m(\Omega)^{\frac{1}{p}-1} C_1 N \mu^{-\frac{4}{4+d}} \; . \tag{A.9}$$

Because each of the summands in the left hand side of above inequality is positive, there exist constant $C_2$ (depending on $d$ and $N$) such that for $\mu < \mu_0$

$$\int_\Omega \frac{1}{\mu}|\psi_i|^p \, dx \leq m(\Omega)^{\frac{1}{p}-1} C_2 \mu^{-\frac{4}{4+d}} \quad \text{and} \quad \int_\Omega |\nabla\psi_i|^2 \, dx \leq m(\Omega)^{\frac{1}{p}-1} C_2 \mu^{-\frac{4}{4+d}} \; . \tag{A.10}$$

Moreover, replacing the above inequalities into (A.8), it follows that there exist a constant $C_3$ (depending on $d$, $N$ and $p$), such that for $\mu < \mu_0$

$$|\lambda_{ii}| < p\frac{C_2}{2}\mu^{-\frac{4}{4+d}} m(\Omega)^{\frac{1}{p}-1} + p\frac{C_2}{2}\mu^{-\frac{4}{4+d}} m(\Omega)^{\frac{1}{p}-1} < C_3\mu^{-\frac{4}{4+d}} m(\Omega)^{\frac{1}{p}-1}. \tag{A.11}$$

**Step 3:** Upper bounds for $\lambda'_{ij}s$.
Fix $i$. For $k \neq i$, multiply both sides of equation (A.2) by $\psi_k(x)$ and integrate over $\Omega$:

$$\int_\Omega \left( \frac{1}{\mu} s(\psi_i)\psi_k - 2\lambda_{ii}\psi_i\psi_k - \Delta\psi_i\psi_k - \sum_{j\neq i} \lambda_{ij}\psi_j\psi_k \right) dx = 0 \,, \tag{A.12}$$

which, using orthonormality condition (A.3) and integration by parts, implies that:

$$\frac{1}{\mu}\int_\Omega s(\psi_i)\psi_k dx + \int_\Omega (\nabla\psi_i)(\nabla\psi_k)dx - \lambda_{ik} = 0 \,. \tag{A.13}$$

Therefore

$$\lambda_{ik} = \frac{1}{\mu}\int_\Omega s(\psi_i)\psi_k \, dx + \int_\Omega (\nabla\psi_i)(\nabla\psi_k) \, dx \,. \tag{A.14}$$

By using relation (A.1), we have

$$\left| \frac{1}{\mu}\int_\Omega s(\psi_i)\psi_k \, dx \right| \leq \frac{1}{\mu}\int_\Omega |s(\psi_i)\psi_k| \, dx = \frac{p}{\mu}\int_\Omega |\psi_i|^{p-1} \, |\psi_k| \, dx = \frac{p}{\mu}\int_\Omega |\psi_i|^p \, \frac{|\psi_k|}{|\psi_i|} dx \,. \tag{A.15}$$

Since $|\psi_i|^p$ does not change sign on $\Omega$, by the First Mean Value Theorem for Integrals, there exists $\xi \in \Omega$, with $\psi_i(\xi) \neq 0$, such that, if we set $M = \frac{|\psi_k(\xi)|}{|\psi_i(\xi)|}$, it follows that

$$\frac{p}{\mu}\int_\Omega |\psi_i|^p \, \frac{|\psi_k|}{|\psi_i|} dx = M\frac{p}{\mu}\int_\Omega |\psi_i|^p \, dx \,.$$

Making use of (A.10), we conclude that

$$\left| \frac{1}{\mu}\int_\Omega s(\psi_i)\psi_k dx \right| \leq pMm(\Omega)^{\frac{1}{p}-1} C_2 \mu^{-\frac{4}{4+d}} . \tag{A.16}$$

Finally, using Cauchy-Schwarz and equation (A.10),

$$\left| \int_\Omega (\nabla\psi_i)(\nabla\psi_k)dx \right| \leq \left( \int_\Omega |\nabla\psi_i|^2 \, dx \right)^{\frac{1}{2}} \left( \int_\Omega |\nabla\psi_k|^2 \, dx \right)^{\frac{1}{2}} < (m(\Omega)^{\frac{1}{p}-1} C_2 \mu^{-\frac{4}{4+d}})^{\frac{1}{2}} (m(\Omega)^{\frac{1}{p}-1} C_2 \mu^{-\frac{4}{4+d}})^{\frac{1}{2}} = m(\Omega)^{\frac{1}{p}-1} C_2 \mu^{-\frac{4}{4+d}} . \tag{A.17}$$

Substituting the two upper bounds given in (A.16) and (A.17) into equation (A.14), we

have for $\mu < \mu_0$

$$|\lambda_{ik}| < pMm(\Omega)^{\frac{1}{p}-1}C_2\mu^{-\frac{4}{4+d}} + m(\Omega)^{\frac{1}{p}-1}C_2\mu^{-\frac{4}{4+d}} = \\ m(\Omega)^{\frac{1}{p}-1}C_2\mu^{-\frac{4}{4+d}}(pM+1) < C_4 m(\Omega)^{\frac{1}{p}-1}\mu^{-\frac{4}{4+d}} \quad \text{(A.18)}$$

where $C_4$ depends on $N$, $M$, $p$ and $\mu$.

**Step 4:** Bounding the volume of the support $\psi_i$'s
For each $i$ multiply both sides of equation (A.2) by $\frac{1}{p}sign(\psi_i)\,|\psi_i|^{1-p}$ and integrate over domain $\Omega$:

$$\frac{1}{\mu}\,|supp(\psi_i)| - \frac{2}{p}\lambda_{ii}\int_\Omega |\psi_i|^{2-p}\,dx - \frac{1}{p}\int_\Omega \Delta\psi_i sign(\psi_i)\,|\psi_i|^{1-p}\,dx - \frac{1}{p}\sum_{j\neq i}\lambda_{ij}\int_\Omega \psi_j sign(\psi_i)\,|\psi_i|^{1-p}\,dx = 0\,, \quad \text{(A.19)}$$

namely

$$\frac{1}{\mu}\,|supp(\psi_i)| = \\ \left|\frac{2}{p}\lambda_{ii}\int_\Omega |\psi_i|^{2-p}\,dx + \frac{1}{p}\int_\Omega \Delta\psi_i sign(\psi_i)\,|\psi_i|^{1-p}\,dx + \frac{1}{p}\sum_{j\neq i}\lambda_{ij}\int_\Omega \psi_j sign(\psi_i)\,|\psi_i|^{1-p}\,dx\right| \leq \\ \left|\frac{2}{p}\lambda_{ii}\int_\Omega |\psi_i|^{2-p}\,dx + \frac{1}{p}\int_\Omega \Delta\psi_i sign(\psi_i)\,|\psi_i|^{1-p}\,dx\right| + \left|\frac{1}{p}\sum_{j\neq i}\lambda_{ij}\int_\Omega \psi_j sign(\psi_i)\,|\psi_i|^{1-p}\,dx\right|. \quad \text{(A.20)}$$

Define

$$\Omega^+ = \{x \in \Omega : \psi_i(x) > 0\}$$

and

$$\Omega^- = \{x \in \Omega : \psi_i(x) < 0\}\,.$$

According to Green's formula

$$\int_{\Omega^+}\Delta\psi_i dx = \int_{\partial\Omega^+}\frac{\partial\psi_i}{\partial\nu}dS \leq 0$$

where $\nu$ is outward pointing unit normal vector along $\partial\Omega^+$. Since $\psi$ is positive in $\Omega^+$ and becomes zero on $\partial\Omega^+$, the right-hand side of the above expression is not positive. With a similar argument, we have that

$$\int_{\Omega^-}\Delta\psi_i dx = \int_{\partial\Omega^-}\frac{\partial\psi_i}{\partial\nu}dS \geq 0\,.$$

Hence, since $|\psi_i|^{1-p} \geq 0 \ \forall i$, it follows that:

$$\int_\Omega \Delta\psi_i sign(\psi_i)\,|\psi_i|^{1-p}\,dx = \int_{\Omega^+} \Delta\psi_i\,|\psi_i|^{1-p}\,dx - \int_{\Omega^-} \Delta\psi_i\,|\psi_i|^{1-p}\,dx \leq 0\,. \tag{A.21}$$

Using inequality (A.21), (A.20) can be rewritten as:

$$\begin{aligned}\frac{1}{\mu}\,|supp(\psi_i)| \leq \left|\frac{2}{p}\lambda_{ii}\int_\Omega |\psi_i|^{2-p}\,dx\right| + \left|\frac{1}{p}\sum_{j\neq i}\lambda_{ij}\int_\Omega |\psi_j|\,|\psi_i|^{1-p}\,dx\right| \leq \\ \frac{2}{p}|\lambda_{ii}|\int_\Omega |\psi_i|^{2-p}\,dx + \frac{1}{p}\sum_{j\neq i}|\lambda_{ij}|\int_\Omega |\psi_j|\,|\psi_i|^{1-p}\,dx \leq \\ \frac{2}{p}|\lambda_{ii}|\int_\Omega |\psi_i|\,|\psi_i|^{1-p}\,dx + \frac{1}{p}\sum_{j\neq i}|\lambda_{ij}|\int_\Omega |\psi_j|\,|\psi_i|^{1-p}\,dx\,.\end{aligned} \tag{A.22}$$

We set $\tilde{p} = 1-p$, $0 < \tilde{p} < 1$ , for $0 < p < 1$

$$\frac{1}{\mu}\,|supp(\psi_i)| \leq \frac{2}{p}|\lambda_{ii}|\int_\Omega |\psi_i|\,|\psi_i|^{\tilde{p}}\,dx + \frac{1}{p}\sum_{j\neq i}|\lambda_{ij}|\int_\Omega |\psi_j|\,|\psi_i|^{\tilde{p}}\,dx\,. \tag{A.23}$$

Since $|\psi_i|^{\tilde{p}}$ does not change sign on $\Omega$, by the First Mean Value Theorem for Integrals, there exist $\xi, \eta \in \Omega$ such that, if we set $\bar{M} = |\psi_i(\xi)|$ and $\tilde{M} = |\psi_j(\eta)|$, relation (A.23) can be rewritten as

$$\frac{1}{\mu}\,|supp(\psi_i)| \leq \frac{2}{p}\bar{M}|\lambda_{ii}|\int_\Omega |\psi_i|^{\tilde{p}}\,dx + \frac{1}{p}\sum_{j\neq i}\tilde{M}|\lambda_{ij}|\int_\Omega |\psi_i|^{\tilde{p}}\,dx. \tag{A.24}$$

By using (A.10), (A.11),(A.18), then (A.24) becomes

$$\begin{aligned}\frac{1}{\mu}\,|supp(\psi_i)| \leq \frac{2}{p}\bar{M}C_3 m(\Omega)^{\frac{1}{p}-1}\mu^{-\frac{4}{4+d}}m(\Omega)^{\frac{1}{1-p}-1}C_2\mu^{-\frac{4}{4+d}+1} + \\ \frac{1}{p}(N-1)\tilde{M}C_4 m(\Omega)^{\frac{1}{p}-1}\mu^{-\frac{4}{4+d}}m(\Omega)^{\frac{1}{1-p}-1}C_2\mu^{-\frac{4}{4+d}+1} \\ \leq C_5\mu^{-\frac{8}{4+d}+1}m(\Omega)^{\frac{1}{p(1-p)}-2}\end{aligned} \tag{A.25}$$

where $C_5$ depends on $N$ and $p$. □

# Appendix B

# Proofs in Section 6.3 on the CNC Approach in Segmentation Over Surfaces

**Proof of Lemma 2.**

*Proof.* Let $x := (x_0, x_1, \ldots, x_v)^T \in \mathbb{R}^{v+1}$, be the vector of neighbors associated with a generic vertex in $V$ of valence $v$. Then, function $f_v(\,\cdot\,; \lambda, \eta, a)$ in (6.22) can be rewritten in more compact form as follows:

$$f_v(x; \lambda, \eta, a) \;=\; \frac{\lambda}{2}\, x^T D x \;+\; \frac{\eta}{2}\, x^T Q x \;+\; \phi\Big(\sqrt{x^T Q\, x}\,;\, a\Big)\;, \tag{B.1}$$

with the arrowhead and diagonal matrices $Q$ in (6.26) and $D \in \mathbb{R}^{(v+1)\times(v+1)}$ defined as

$$D = \operatorname{diag}\left(\left\{\frac{1}{v_j + 1}\right\}_{j=0}^{v}\right), \tag{B.2}$$

respectively. Let $\tilde{v} := \max_j v_j$ and $\widetilde{D} := \frac{1}{\tilde{v}+1}\, I$, with $I$ denoting the $(v+1) \times (v+1)$ identity matrix. Then, since the matrix $D - \widetilde{D}$ is positive semi-definite, a sufficient condition for function $f_v$ in (B.1) to be strictly convex is that the function $\tilde{f}_v$ defined by

$$\begin{aligned} \tilde{f}_v(x; \lambda, \eta, a) \;&=\; \frac{\lambda}{2}\, x^T \widetilde{D} x \;+\; \frac{\eta}{2}\, x^T Q x \;+\; \phi\Big(\sqrt{x^T Q\, x}\,;\, a\Big) \\ &=\; \frac{\lambda}{2(1+\tilde{v})}\, x^T x \;+\; \frac{\eta}{2}\, x^T Q x \;+\; \phi\Big(\sqrt{x^T Q\, x}\,;\, a\Big)\;, \end{aligned} \tag{B.3}$$

is strictly convex.

The matrix $Q$ in (6.26) is real, symmetric, arrowhead and positive semi-definite with the following eigenvalue decomposition:

$$Q \;=\; V\Lambda\, V^T\,, \quad \Lambda = \mathrm{diag}(0, \lambda_1, \lambda_2, \ldots, \lambda_v)\,, \quad VV^T = V^TV = I\,, \tag{B.4}$$

where orthogonality of the modal matrix $V$ in (B.4) follows from symmetry of the matrix $Q$ and the nonzero eigenvalues satisfy $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_v > 0$. Then, we decompose the diagonal eigenvalues matrix $\Lambda$ in (B.4) as follows:

$$\Lambda = Z\widetilde{\Lambda}Z\,, \quad Z = \mathrm{diag}(1, \sqrt{\lambda_1}, \sqrt{\lambda_2}, \ldots, \sqrt{\lambda_v})\,, \quad \widetilde{\Lambda} = \mathrm{diag}(0, \underbrace{1, 1, \ldots, 1}_{v \text{ entries}})\,. \tag{B.5}$$

Substituting (B.5) into (B.4), then (B.4) into into (B.3), we obtain the following equivalent expression for the function $\tilde{f}_v$:

$$\tilde{f}_v(x; \lambda, \eta, a) \;=\; \frac{\lambda}{2(1+\tilde{v})}\, x^Tx \;+\; \frac{\eta}{2}\, x^TVZ\widetilde{\Lambda}Z\,V^T\,x \;+\; \phi\Big(\sqrt{x^TVZ\widetilde{\Lambda}Z\,V^T\,x}\,;\, a\Big)\,. \tag{B.6}$$

Recalling that the property of convexity for a function is invariant under non-singular linear application of its domain, we introduce the following one for the domain $\mathbb{R}^{v+1}$ of function $\tilde{f}_v$ above:

$$x \;=\; Ty\,, \quad T := VZ^{-1} \in \mathbb{R}^{(v+1)\times(v+1)}\,, \tag{B.7}$$

which is non-singular due to $V$ and $Z$ being non singular matrices. By defining as $\tilde{f}_v^T := \tilde{f}_v \circ T$ the function $\tilde{f}_v$ in the $T$-transformed domain, we have:

$$\tilde{f}_v^T(y; \lambda, \eta, a) \;=\; \frac{\lambda}{2(1+\tilde{v})}\, y^TZ^{-2}y \;+\; \frac{\eta}{2}\, y^T\widetilde{\Lambda}\,y \;+\; \phi\Big(\sqrt{y^T\widetilde{\Lambda}\,y}\,;\, a\Big)\,. \tag{B.8}$$

Recalling the definitions of $Z$ and $\widetilde{\Lambda}$ in (B.5), we can write (B.8) in explicit form:

$$
\begin{aligned}
\tilde{f}_v^{\,T}(y;\lambda,\eta,a) \;&=\; \frac{\lambda}{2(1+\tilde{v})}\left(y_0^2+\frac{y_1^2}{\lambda_1}+\frac{y_2^2}{\lambda_2}+\ldots+\frac{y_v^2}{\lambda_v}\right)\\
&+\; \frac{\eta}{2}\left(y_1^2+y_2^2+\ldots+y_v^2\right)\;+\;\phi\Big(\sqrt{y_1^2+y_2^2+\ldots+y_v^2}\,;\,a\Big)\\
&=\; \frac{\lambda}{2(1+\tilde{v})}\left[y_0^2+\sum_{j=1}^{v}\left(\frac{1}{\lambda_j}-\frac{1}{\lambda_1}\right)y_j^2\right]+\frac{\lambda}{2(1+\tilde{v})\lambda_1}\left(y_1^2+y_2^2+\ldots+y_v^2\right)\\
&+\; \frac{\eta}{2}\left(y_1^2+y_2^2+\ldots+y_v^2\right)\;+\;\phi\Big(\sqrt{y_1^2+y_2^2+\ldots+y_v^2}\,;\,a\Big)\\
&=\; \frac{\lambda}{2(1+\tilde{v})}\left[y_0^2+\sum_{j=1}^{v}\left(\frac{1}{\lambda_j}-\frac{1}{\lambda_1}\right)y_j^2\right]\\
&+\; \frac{1}{2}\left(\eta+\frac{\lambda}{(1+\tilde{v})\lambda_1}\right)\left(y_1^2+y_2^2+\ldots+y_v^2\right)\;+\;\phi\Big(\sqrt{y_1^2+y_2^2+\ldots+y_v^2}\,;\,a\Big)\\
&=\; \frac{\lambda}{2(1+\tilde{v})}\left[y_0^2+\sum_{j=1}^{v}\Big(\underbrace{\frac{1}{\lambda_j}-\frac{1}{\lambda_1}}_{>0}\Big)y_j^2\right]\;+\;g_v(y_1,\ldots,y_v;\lambda,\eta,a)\,, \qquad \text{(B.9)}
\end{aligned}
$$

where the function $g_v$ in (B.9) is defined in (6.24). Since the first term in (B.9) is (quadratic) convex, a sufficient condition for the function $\tilde{f}_v^{\,T}$ in (B.9) to be strictly convex is that the function $g_v$ in (6.24) is strictly convex. This concludes the proof after recalling that the function $f_v$ is strictly convex if the function $\tilde{f}_v$ is strictly convex and that the function $\tilde{f}_v$ is strictly convex if and only if the function $\tilde{f}_v^{\,T}$ is strictly convex. □

**Proof of Proposition 5.**

*Proof.* The functional $\mathcal{J}(\,\cdot\,;\lambda,\eta,a)$ in (6.21) is clearly proper. Moreover, since the functions $\phi(\,\cdot\,;a)$ and $\|\cdot\|_2$ are both continuous and bounded from below by zero, $\mathcal{J}$ is also continuous and bounded from below by zero. The second and third terms of our functional $\mathcal{J}$ in (6.21) are not coercive. However, since the first term (namely, the fidelity term) is quadratic and strictly convex, hence coercive, and the second and third terms are bounded from below by zero, $\mathcal{J}$ is coercive.

As far as strong convexity is concerned, it follows from Definition 3 that the functional $\mathcal{J}(\,\cdot\,;\lambda,\eta,a)$ in (6.21) is $\mu$-strongly convex if and only if the functional $\widetilde{\mathcal{J}}(\,\cdot\,;\lambda,\eta,a,\mu)$

defined as

$$\widetilde{\mathcal{J}}(u;\lambda,\eta,a,\mu) := \underbrace{\frac{\lambda}{2}\|u-f\|_2^2 + \sum_{i=1}^{n}\|(\nabla_w u)_i\|_2^2 + \sum_{i=1}^{n}\phi\left(\|(\nabla_w u)_i\|_2;a_i\right)}_{\mathcal{J}(u;\lambda,\eta,a)} - \frac{\mu}{2}\|u\|_2^2$$
$$= \mathcal{A}(u) + \frac{\lambda-\mu}{2}\|u\|_2^2 + \sum_{i=1}^{n}\|(\nabla_w u)_i\|_2^2 + \sum_{i=1}^{n}\phi\left(\|(\nabla_w u)_i\|_2;a_i\right) \quad \text{(B.10)}$$

is convex, where $\mathcal{A}(u)$ is an affine function of $u$. We notice that the functional $\widetilde{\mathcal{J}}$ in (B.10) almost coincides with the original functional $\mathcal{J}$ in (6.21), the only difference being the coefficient $\lambda-\mu$ instead of $\lambda$. Hence, we can apply the results in Theorem 4 and state that $\widetilde{\mathcal{J}}$ in (B.10) is convex if condition (6.27) is satisfied with $\lambda-\mu$ in place of $\lambda$. By substituting $\lambda-\mu$ for $\lambda$ in that condition, deriving the solution interval for $\mu$ and then taking the maximum, one obtains equality (6.28). □

**Proof of Proposition 6.**

*Proof.* The demonstration of condition (6.42) for strict convexity of the function $\theta$ in (6.41) is straightforward. In fact, the function $\theta$ can be equivalently rewritten as

$$\theta(x) = \underbrace{\phi\left(\|x\|_2;a\right) + \frac{\eta+\beta}{2}\|x\|_2^2}_{\bar{\theta}(x)} + \mathcal{A}(x)\,, \quad x\in\mathbb{R}^v\,, \qquad \text{(B.11)}$$

with $\mathcal{A}(x)$ an affine function of $x$, so that a necessary and sufficient condition for $\theta$ to be strictly convex is that the function $\bar{\theta}$ in (B.11) is strictly convex. We then notice that $\bar{\theta}$ is almost identical to the function $g$ in (6.24), the only difference being the coefficient $(\eta+\beta)$ that for $g$ reads $(\eta+\lambda/\kappa)$. By setting $\eta+\beta=\eta+\lambda/\kappa \iff \lambda=\kappa\beta$, the two functions coincide. Condition for strict convexity of $g$ in (6.27) reads as $\lambda/\kappa > a-\eta$, hence by substituting $\lambda=\kappa\beta$ in it we obtain condition (6.42) for strict convexity of $\theta$.

For the proof of statement (6.44), according to which the unique solution $x^*$ of the strictly convex problem (6.43) is obtained by a shrinkage of vector $r$, we refer the reader to [70][Proposition 4.5].

We now prove statement (6.45). First, we notice that if $\|r\|_2=0$, i.e. $r$ is the null vector, the minimization problem in (6.43) with the objective function $\theta(x)$ defined in (6.41) reduces to

$$\arg\min_{x\in\mathbb{R}^v}\left\{\phi\left(\|x\|_2;a\right) + \frac{\eta+\beta}{2}\|x\|_2^2\right\}. \qquad \text{(B.12)}$$

Since both the terms of the cost function in (B.12) are monotonically increasing functions of $\|x\|_2$, the solution of (B.12) is clearly $x^* = 0$. Hence, the case $\|r\|_2 = 0$ can be easily dealt with by taking any value $\xi^*$ in formula (6.44). We included the case $\|r\|_2 = 0$ in formula a) of (6.45). In the following, we consider the case $\|r\|_2 > 0$.

Based on the previously demonstrated statement (6.44), by setting $x = \xi\, r$, $\xi \geq 0$, we turn the original unconstrained $v$-dimensional problem in (6.43) into the following equivalent constrained 1-dimensional problem:

$$\begin{aligned}
\xi^* &= \arg\min_{0\leq\xi<1} \left\{ \phi\left(\|\xi r\|_2 ; a\right) + \frac{\eta}{2} \|\xi r\|_2^2 + \frac{\beta}{2} \|\xi r - r\|_2^2 \right\} \\
&= \arg\min_{0\leq\xi<1} \left\{ \phi\left(\|r\|_2\, \xi; a\right) + \frac{\eta}{2} \|r\|_2^2\, \xi^2 + \frac{\beta}{2} \|r\|_2^2\, (\xi - 1)^2 \right\} \\
&= \arg\min_{0\leq\xi<1} \left\{ f(\xi) := \phi\left(\|r\|_2\, \xi; a\right) + \|r\|_2^2 \left( \frac{\eta+\beta}{2}\, \xi^2 - \beta\xi \right) \right\} \qquad \text{(B.13)}
\end{aligned}$$

where in (B.13) we omitted the constant terms and introduced the cost function $f : \mathbb{R}_+ \to \mathbb{R}$ for future reference. Since the penalty function $\phi$ is twice continuously differentiable on $\mathbb{R}_+$ - see assumption A1) in Section 2 - the cost function $f$ in (B.13) is also twice continuously differentiable on $\mathbb{R}_+$. Moreover, $f$ is strictly convex since it represents the restriction of the strictly convex function $\theta$ in (6.41) to the half-line $\xi\, r$, $\xi \geq 0$. Hence, a necessary and sufficient condition for an inner point $0 < \xi < 1$ to be the global minimizer of $f$ is as follows:

$$f'(\xi) = 0 \iff \|r\|_2 \left[ \phi'\left(\|r\|_2 \xi; a\right) + \|r\|_2 \Big( (\eta+\beta)\, \xi - \beta \Big) \right] = 0\,. \qquad \text{(B.14)}$$

Since $f$ is continuously differentiable and strictly convex on $\mathbb{R}_+$, the first-order derivative $f'(\xi)$ is continuous and strictly increasing in the optimization domain $0 \leq \xi \leq 1$ and at the extremes we have:

$$f'(0^+) = \|r\|_2 \Big[ \phi'(0^+; a) - \beta\, \|r\|_2 \Big]\,, \qquad f'(1) = \|r\|_2 \Big[ \phi'(\|r\|_2; a) + \eta\, \|r\|_2 \Big]\,. \qquad \text{(B.15)}$$

Since $\eta \geq 0$, $\|r\|_2 > 0$ and $\phi'(t; a) > 0$ for any $t \geq 0$ by assumption A2) in Section 2, $f'(1)$ in (B.15) is positive. Hence, we have two cases. If $f'(0^+) \geq 0$, that is $\|r\| \leq \phi'(0^+; a)/\beta = 1/\beta$, $f'(t)$ is positive in $0 < t \leq 1$, hence the function $f$ has its minimum at $\xi^* = 0$; if $f'(0^+) < 0$, that is $\|r\| > 1/\beta$, then $f$ has the minimum at its unique stationary point $0 < \xi^* < 1$, which can be obtained by solving the nonlinear equation in (B.14). The proof of statement (6.45) is thus completed. □

# Appendix C

# Proofs in Section 7.4 on the Tangential Evolution

**Proof of Lemma 3**

The local density $g$ can be controlled by the parameters $\alpha$ and $\lambda$. The relation how $g$ is changing in time depending on $\alpha$ and $\lambda$ (and prescribed $\beta$) we derive in the following.

Starting with the following relation between $g$, $\mathbf{x}_u$ and $\mathbf{x}_v$

$$g_t = |\boldsymbol{x}_u \times \boldsymbol{x}_v|_t = \frac{\boldsymbol{x}_u \times \boldsymbol{x}_v}{|\boldsymbol{x}_u \times \boldsymbol{x}_v|} \cdot (\boldsymbol{x}_u \times \boldsymbol{x}_v)_t \ , \tag{C.1}$$

the term $(\boldsymbol{x}_u \times \boldsymbol{x}_v)_t$ can be expanded as

$$\begin{aligned}(\boldsymbol{x}_u \times \boldsymbol{x}_v)_t &= (\boldsymbol{x}_u)_t \times \boldsymbol{x}_v + \boldsymbol{x}_u \times (\boldsymbol{x}_v)_t \\ &= (\boldsymbol{x}_t)_u \times \boldsymbol{x}_v + \boldsymbol{x}_u \times (\boldsymbol{x}_t)_v \\ &= (\beta \boldsymbol{N} + \alpha \boldsymbol{x}_u + \lambda \boldsymbol{x}_v)_u \times \boldsymbol{x}_v + \boldsymbol{x}_u \times (\beta \boldsymbol{N} + \alpha \boldsymbol{x}_u + \lambda \boldsymbol{x}_v)_v \\ &= (\beta \boldsymbol{N})_u \times \boldsymbol{x}_v + (\alpha \boldsymbol{x}_u + \lambda \boldsymbol{x}_v)_u \times \boldsymbol{x}_v + \boldsymbol{x}_u \times (\beta \boldsymbol{N})_v + \boldsymbol{x}_u \times (\alpha \boldsymbol{x}_u + \lambda \boldsymbol{x}_v)_v \ .\end{aligned} \tag{C.2}$$

For transparency reasons, let us further focus on (C.2) splitted into two parts by collecting the terms containing $\beta \boldsymbol{N}$ and the ones containing $\alpha \boldsymbol{x}_u + \lambda \boldsymbol{x}_v$.

Plugging the first part of (C.2) into (C.1), we obtain

$$
\begin{aligned}
&\frac{\boldsymbol{x}_u \times \boldsymbol{x}_v}{|\boldsymbol{x}_u \times \boldsymbol{x}_v|} \cdot \left( (\beta \boldsymbol{N})_u \times \boldsymbol{x}_v + \boldsymbol{x}_u \times (\beta \boldsymbol{N})_v \right) \\
&\qquad = \Big( \left( \boldsymbol{x}_u \cdot (\beta \boldsymbol{N})_u \right) \left( \boldsymbol{x}_v \cdot \boldsymbol{x}_v \right) - \left( \boldsymbol{x}_u \cdot \boldsymbol{x}_v \right) \left( \boldsymbol{x}_v \cdot (\beta \boldsymbol{N})_u \right) \\
&\qquad\quad + \left( \boldsymbol{x}_u \cdot \boldsymbol{x}_u \right) \left( \boldsymbol{x}_v \cdot (\beta \boldsymbol{N})_v \right) - \left( \boldsymbol{x}_u \cdot (\beta \boldsymbol{N})_v \right) \left( \boldsymbol{x}_u \cdot \boldsymbol{x}_v \right) \Big) \Big/ |\boldsymbol{x}_u \times \boldsymbol{x}_v| \\
&\qquad = \Big( \left( \boldsymbol{x}_{uu} \cdot \beta \boldsymbol{N} \right) \left( \boldsymbol{x}_v \cdot \boldsymbol{x}_v \right) - \left( \boldsymbol{x}_u \cdot \boldsymbol{x}_v \right) \left( \boldsymbol{x}_{vu} \cdot \beta \boldsymbol{N} \right) \\
&\qquad\quad + \left( \boldsymbol{x}_u \cdot \boldsymbol{x}_u \right) \left( \boldsymbol{x}_{vv} \cdot \beta \boldsymbol{N} \right) - \left( \boldsymbol{x}_{uv} \cdot \beta \boldsymbol{N} \right) \left( \boldsymbol{x}_u \cdot \boldsymbol{x}_v \right) \Big) \Big/ |\boldsymbol{x}_u \times \boldsymbol{x}_v| \\
&\qquad = \Big( \left( \boldsymbol{x}_{uu} \cdot \beta \boldsymbol{N} \right) \left( \boldsymbol{x}_v \cdot \boldsymbol{x}_v \right) - 2 \left( \boldsymbol{x}_u \cdot \boldsymbol{x}_v \right) \left( \boldsymbol{x}_{vu} \cdot \beta \boldsymbol{N} \right) \\
&\qquad\quad + \left( \boldsymbol{x}_u \cdot \boldsymbol{x}_u \right) \left( \boldsymbol{x}_{vv} \cdot \beta \boldsymbol{N} \right) \Big) \Big/ |\boldsymbol{x}_u \times \boldsymbol{x}_v| \\
&\qquad = \Big( \left( \boldsymbol{x}_{uu} \cdot \beta \frac{\boldsymbol{x}_u \times \boldsymbol{x}_v}{|\boldsymbol{x}_u \times \boldsymbol{x}_v|} \right) \left( \boldsymbol{x}_v \cdot \boldsymbol{x}_v \right) - 2 \left( \boldsymbol{x}_u \cdot \boldsymbol{x}_v \right) \left( \boldsymbol{x}_{vu} \cdot \beta \frac{\boldsymbol{x}_u \times \boldsymbol{x}_v}{|\boldsymbol{x}_u \times \boldsymbol{x}_v|} \right) \\
&\qquad\quad + \left( \boldsymbol{x}_u \cdot \boldsymbol{x}_u \right) \left( \boldsymbol{x}_{vv} \cdot \beta \frac{\boldsymbol{x}_u \times \boldsymbol{x}_v}{|\boldsymbol{x}_u \times \boldsymbol{x}_v|} \right) \Big) \Big/ |\boldsymbol{x}_u \times \boldsymbol{x}_v| \\
&\qquad = \Big( \left( \boldsymbol{x}_{uu} \cdot (\boldsymbol{x}_u \times \boldsymbol{x}_v) \right) \left( \boldsymbol{x}_v \cdot \boldsymbol{x}_v \right) - 2 \left( \boldsymbol{x}_u \cdot \boldsymbol{x}_v \right) \left( \boldsymbol{x}_{vu} \cdot (\boldsymbol{x}_u \times \boldsymbol{x}_v) \right) \\
&\qquad\quad + \left( \boldsymbol{x}_u \cdot \boldsymbol{x}_u \right) \left( \boldsymbol{x}_{vv} \cdot (\boldsymbol{x}_u \times \boldsymbol{x}_v) \right) \Big) \frac{|\boldsymbol{x}_u \times \boldsymbol{x}_v|}{|\boldsymbol{x}_u \times \boldsymbol{x}_v|^3} \beta \\
&\qquad = H |\boldsymbol{x}_u \times \boldsymbol{x}_v| \beta \\
&\qquad = g \, H \beta
\end{aligned}
$$

where the formula for mean curvature has been used from [48].

Similarly, plugging the second, tangential part of (C.2) into (C.1)

$$
\begin{aligned}
&\frac{\boldsymbol{x}_u \times \boldsymbol{x}_v}{|\boldsymbol{x}_u \times \boldsymbol{x}_v|} \cdot ((\alpha \boldsymbol{x}_u \ + \lambda \boldsymbol{x}_v)_u \times \boldsymbol{x}_v + \boldsymbol{x}_u \times (\alpha \boldsymbol{x}_u + \lambda \boldsymbol{x}_v)_v) \\
&\qquad = \frac{\boldsymbol{x}_u \times \boldsymbol{x}_v}{|\boldsymbol{x}_u \times \boldsymbol{x}_v|} \cdot \Big( (\alpha_u \boldsymbol{x}_u + \alpha \boldsymbol{x}_{uu} + \lambda_u \boldsymbol{x}_v + \lambda \boldsymbol{x}_{vu}) \times \boldsymbol{x}_v \\
&\qquad\quad + \boldsymbol{x}_u \times (\alpha_v \boldsymbol{x}_u + \alpha \boldsymbol{x}_{uv} + \lambda_v \boldsymbol{x}_v + \lambda \boldsymbol{x}_{vv}) \Big) \\
&\qquad = \frac{\boldsymbol{x}_u \times \boldsymbol{x}_v}{|\boldsymbol{x}_u \times \boldsymbol{x}_v|} \cdot \Big( \alpha_u \boldsymbol{x}_u \times \boldsymbol{x}_v + \alpha \boldsymbol{x}_{uu} \times \boldsymbol{x}_v + \lambda_u \boldsymbol{x}_v \times \boldsymbol{x}_v + \lambda \boldsymbol{x}_{vu} \times \boldsymbol{x}_v \\
&\qquad\quad + \alpha_v \boldsymbol{x}_u \times \boldsymbol{x}_u + \alpha \boldsymbol{x}_u \times \boldsymbol{x}_{uv} + \lambda_v \boldsymbol{x}_u \times \boldsymbol{x}_v + \lambda \boldsymbol{x}_u \times \boldsymbol{x}_{vv} \Big) \\
&\qquad = \frac{\boldsymbol{x}_u \times \boldsymbol{x}_v}{|\boldsymbol{x}_u \times \boldsymbol{x}_v|} \cdot \Big( \alpha_u \boldsymbol{x}_u \times \boldsymbol{x}_v + \alpha \boldsymbol{x}_{uu} \times \boldsymbol{x}_v + \lambda \boldsymbol{x}_{vu} \times \boldsymbol{x}_v \\
&\qquad\quad + \alpha \boldsymbol{x}_u \times \boldsymbol{x}_{uv} + \lambda_v \boldsymbol{x}_u \times \boldsymbol{x}_v + \lambda \boldsymbol{x}_u \times \boldsymbol{x}_{vv} \Big) \\
&\qquad = \Big( \alpha_u |\boldsymbol{x}_u \times \boldsymbol{x}_v| + \alpha \frac{(\boldsymbol{x}_u \times \boldsymbol{x}_v) \cdot (\boldsymbol{x}_{uu} \times \boldsymbol{x}_v + \boldsymbol{x}_u \times \boldsymbol{x}_{uv})}{|\boldsymbol{x}_u \times \boldsymbol{x}_v|} \\
&\qquad\quad + \lambda_v |\boldsymbol{x}_u \times \boldsymbol{x}_v| + \lambda \frac{(\boldsymbol{x}_u \times \boldsymbol{x}_v) \cdot (\boldsymbol{x}_u \times \boldsymbol{x}_{vv} + \boldsymbol{x}_{vu} \times \boldsymbol{x}_v)}{|\boldsymbol{x}_u \times \boldsymbol{x}_v|} \Big) \\
&\qquad = \Big( \alpha_u |\boldsymbol{x}_u \times \boldsymbol{x}_v| + \alpha \frac{(\boldsymbol{x}_u \times \boldsymbol{x}_v) \cdot (\boldsymbol{x}_u \times \boldsymbol{x}_v)_u}{|\boldsymbol{x}_u \times \boldsymbol{x}_v|} \\
&\qquad\quad + \lambda_v |\boldsymbol{x}_u \times \boldsymbol{x}_v| + \lambda \frac{(\boldsymbol{x}_u \times \boldsymbol{x}_v) \cdot (\boldsymbol{x}_u \times \boldsymbol{x}_v)_v}{|\boldsymbol{x}_u \times \boldsymbol{x}_v|} \Big) \\
&\qquad = \Big( \alpha_u |\boldsymbol{x}_u \times \boldsymbol{x}_v| + \alpha |\boldsymbol{x}_u \times \boldsymbol{x}_v|_u + \lambda_v |\boldsymbol{x}_u \times \boldsymbol{x}_v| + \lambda |\boldsymbol{x}_u \times \boldsymbol{x}_v|_v \Big) \\
&\qquad = \frac{|\boldsymbol{x}_u \times \boldsymbol{x}_v|}{|\boldsymbol{x}_u \times \boldsymbol{x}_v|} \Big( (\alpha |\boldsymbol{x}_u \times \boldsymbol{x}_v|)_u + (\lambda |\boldsymbol{x}_u \times \boldsymbol{x}_v|)_v \Big) \\
&\qquad = g \ \nabla_x \cdot (\alpha, \lambda)
\end{aligned}
$$

where the formula for divergence of a vector field on manifold has been used in the last step. Given a function $\varphi(u, v), u \in [0, 1], v \in [0, 1]$ on a surface $\mathcal{S}$, its intrinsic gradient relative to a mapping $\mathbf{x}(t, u, v)$ is given by

$$
\nabla_{\mathbf{x}} \varphi = (\frac{1}{g}(g\varphi)_u, \frac{1}{g}(g\varphi)_v) \ . \tag{C.3}
$$

Then, the divergence of a vector $\mathbf{V}_T = \alpha\mathbf{x}_u + \lambda\mathbf{x}_v$ on a surface $\mathcal{S}$ relative to a mapping $\mathbf{x}(t,u,v)$ is defined as

$$\nabla_{\mathcal{S}} \cdot \mathbf{V}_T = \nabla_{\mathbf{x}} \cdot (\alpha, \lambda) = \frac{1}{g}((g\alpha)_u + (g\lambda)_v) \,. \tag{C.4}$$

Combining the preceding two parts together, we obtain (7.11) $\square$.

**Proof of Theorem 5**

For the time derivative of the surface area $A$ we have

$$\begin{aligned} A_t &= \left(\iint_U g \,\mathrm{d}u\mathrm{d}v\right)_t = \iint_U g_t \,\mathrm{d}u\mathrm{d}v \\ &= \iint_U g\, H\beta \,\mathrm{d}u\mathrm{d}v + \iint_U g\, \nabla_x \cdot (\alpha, \lambda)\,\mathrm{d}u\mathrm{d}v \\ &= \iint_{\mathcal{S}} H\beta \,\mathrm{d}X + \iint_{\mathcal{S}} \nabla_{\mathcal{S}} \cdot \mathbf{V}_T \,\mathrm{d}X \\ &= \iint_{\mathcal{S}} H\beta \,\mathrm{d}X + \oint_{\partial\mathcal{S}} \mathbf{V}_T \cdot \boldsymbol{n} \,\mathrm{d}S \\ &= \iint_{\mathcal{S}} H\beta \,\mathrm{d}X + 0 \end{aligned}$$

where $\mathbf{V}_T = \alpha\mathbf{x}_u + \lambda\mathbf{x}_v$ and we used Gauss's theorem. Since our surface $\mathcal{S}$ is either a closed watertight surface, or a surface with boundary which has prescribed zero Neumann boundary condition, the boundary integral in $A_t$ formulation is equal to zero.

Then, putting together the obtained expressions, we can write

$$\left(\frac{g}{A}\right)_t = \frac{g_t A - g A_t}{A^2} = \frac{(g\ H\beta + g\ \nabla_{\mathbf{x}} \cdot (\alpha, \lambda))\, A - g \iint_{\mathcal{S}} H\beta \,\mathrm{d}X}{A^2} \,, \tag{C.5}$$

and putting (C.5) into equation (7.12) we obtain (7.13) $\square$.

### Numerical Approximation of the Normal Derivatives

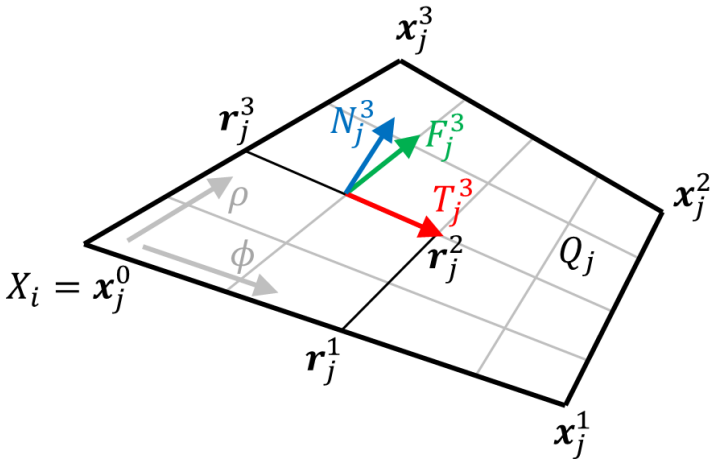



Fig. C.1 Illustration of the bilinear interpolation at one $Q_j$, $j \in N_\square(X_i)$ used in the normal derivatives approximation.

Here we describe the bilinear interpolation used to approximate normal derivatives $\frac{\partial f}{\partial \mathbf{n}_j^*}$ in (7.18), in case of LBO applied to a generic function $f$. The figure C.1 should clarify the next few paragraphs.

Let us recall the Green's theorem applied on $\int_{V_i} \Delta_s f \, \mathrm{d}S$

$$\int_{\partial V_i} \nabla_s f \cdot \mathbf{n} \, \mathrm{d}s = \sum_{j \in N_\square(X_i)} \int_{e_j^1} \frac{\partial f}{\partial \mathbf{n}_j^1} \, \mathrm{d}s + \int_{e_j^3} \frac{\partial f}{\partial \mathbf{n}_j^3} \, \mathrm{d}s \, . \tag{C.6}$$

The values of $\mathbf{x}$ on the $Q_j$, $j \in N_\square(X_i)$ can be approximated by

$$\boldsymbol{x}(\phi, \rho) = (1-\phi)(1-\rho)\mathbf{x}_j^0 + \phi(1-\rho)\mathbf{x}_j^1 + (1-\phi)\rho\mathbf{x}_j^3 + \phi\rho\mathbf{x}_j^2 \tag{C.7}$$

and similarly the values of $f$ as

$$f(\phi, \rho) = (1-\phi)(1-\rho)f_j^0 + \phi(1-\rho)f_j^1 + (1-\phi)\rho f_j^3 + \phi\rho f_j^2 \, . \tag{C.8}$$

Derivatives of $\mathbf{x}(\phi, \rho)$ are

$$\begin{aligned} \partial_\rho \boldsymbol{x}(\phi, \rho) &= -(1-\phi)\mathbf{x}_j^0 - \phi\mathbf{x}_j^1 + (1-\phi)\mathbf{x}_j^3 + \phi\mathbf{x}_j^2 \, , \\ \partial_\phi \boldsymbol{x}(\phi, \rho) &= -(1-\rho)\mathbf{x}_j^0 + (1-\rho)\mathbf{x}_j^1 - \rho\mathbf{x}_j^3 + \rho\mathbf{x}_j^2 \, . \end{aligned} \tag{C.9}$$

Non-normed tangent to the edges $e_j^1$ and $e_j^3$ are defined as

$$\begin{aligned}\boldsymbol{T}_j^1 &= \partial_\rho \boldsymbol{x}(1/2, 1/4) = -1/2\mathbf{x}_j^0 - 1/2\mathbf{x}_j^1 + 1/2\mathbf{x}_j^3 + 1/2\mathbf{x}_j^2 \,, \\ \boldsymbol{T}_j^3 &= \partial_\phi \boldsymbol{x}(1/4, 1/2) = -1/2\mathbf{x}_j^0 + 1/2\mathbf{x}_j^1 - 1/2\mathbf{x}_j^3 + 1/2\mathbf{x}_j^2 \,. \end{aligned} \tag{C.10}$$

There are another two vectors defined by the derivatives of $\boldsymbol{x}$, namely

$$\begin{aligned}\boldsymbol{F}_j^1 &= \partial_\phi \boldsymbol{x}(1/2, 1/4) = -3/4\mathbf{x}_j^0 + 3/4\mathbf{x}_j^1 - 1/4\mathbf{x}_j^3 + 1/4\mathbf{x}_j^2 \,, \\ \boldsymbol{F}_j^3 &= \partial_\rho \boldsymbol{x}(1/4, 1/2) = -3/4\mathbf{x}_j^0 - 1/4\mathbf{x}_j^1 + 3/4\mathbf{x}_j^3 + 1/4\mathbf{x}_j^2 \,. \end{aligned} \tag{C.11}$$

Using the vectors defined above, we can compute non-normed normals to the edge $e_j^1$ using the Gram-Schmidt orthogonalization process as

$$\begin{aligned}\boldsymbol{N}_j^1 &= \boldsymbol{F}_j^1 - \frac{\boldsymbol{F}_j^1 \cdot \boldsymbol{T}_j^1}{\boldsymbol{T}_j^1 \cdot \boldsymbol{T}_j^1} \boldsymbol{T}_j^1 = \partial_\phi \boldsymbol{x}(1/2, 1/4) - \frac{\partial_\phi \boldsymbol{x}(\frac{1}{2}, \frac{1}{4}) \cdot \partial_\rho \boldsymbol{x}(\frac{1}{2}, \frac{1}{4})}{\partial_\rho \boldsymbol{x}(\frac{1}{2}, \frac{1}{4}) \cdot \partial_\rho \boldsymbol{x}(\frac{1}{2}, \frac{1}{4})} \partial_\rho \boldsymbol{x}(1/2, 1/4) \\ &= \partial_\phi \boldsymbol{x}(1/2, 1/4) - a_j^1 \partial_\rho \boldsymbol{x}(1/2, 1/4) \\ &= -\frac{3}{4}\mathbf{x}_j^0 + \frac{3}{4}\mathbf{x}_j^1 - \frac{1}{4}\mathbf{x}_j^3 + \frac{1}{4}\mathbf{x}_j^2 - a_j^1 \left( -\frac{1}{2}\mathbf{x}_j^0 - \frac{1}{2}\mathbf{x}_j^1 + \frac{1}{2}\mathbf{x}_j^3 + \frac{1}{2}\mathbf{x}_j^2 \right) \end{aligned} \tag{C.12}$$

and similarly for the edge $e_j^3$ as

$$\begin{aligned}\boldsymbol{N}_j^3 &= \boldsymbol{F}_j^3 - \frac{\boldsymbol{F}_j^3 \cdot \boldsymbol{T}_j^3}{\boldsymbol{T}_j^3 \cdot \boldsymbol{T}_j^3} \boldsymbol{T}_j^3 = \partial_\rho \boldsymbol{x}(1/4, 1/2) - \frac{\partial_\rho \boldsymbol{x}(\frac{1}{4}, \frac{1}{2}) \cdot \partial_\phi \boldsymbol{x}(\frac{1}{4}, \frac{1}{2})}{\partial_\phi \boldsymbol{x}(\frac{1}{4}, \frac{1}{2}) \cdot \partial_\phi \boldsymbol{x}(\frac{1}{4}, \frac{1}{2})} \partial_\phi \boldsymbol{x}(1/4, 1/2) \\ &= \partial_\rho \boldsymbol{x}(1/4, 1/2) - a_j^3 \partial_\phi \boldsymbol{x}(1/4, 1/2) \\ &= -\frac{3}{4}\mathbf{x}_j^0 - \frac{1}{4}\mathbf{x}_j^1 + \frac{3}{4}\mathbf{x}_j^3 + \frac{1}{4}\mathbf{x}_j^2 - a_j^3 \left( -\frac{1}{2}\mathbf{x}_j^0 + \frac{1}{2}\mathbf{x}_j^1 - \frac{1}{2}\mathbf{x}_j^3 + \frac{1}{2}\mathbf{x}_j^2 \right) \,. \end{aligned} \tag{C.13}$$

Finally the derivative of $f$ in the normal direction at the edge $e_j^1$ is approximated as

$$\frac{\partial f}{\partial \mathbf{n}_j^1} \approx \frac{1}{|\mathbf{N}_j^1|} \left[ \frac{1}{4} \left( -3f_j^0 + 3f_j^1 - f_j^3 + f_j^2 \right) - \frac{a_j^1}{2} \left( -f_j^0 - f_j^1 + f_j^3 + f_j^2 \right) \right] \,, \tag{C.14}$$

and similarly for $\frac{\partial f}{\partial \mathbf{n}_j^3}$. Now we can write the approximation of Laplace-Beltrami operator applied on a generic function $f$ integrated over the finite volume $V_i$ as

$$\int_{\partial V_i} \nabla_s f \cdot \mathbf{n} \, \mathrm{d}s \approx \sum_{j \in N_\square(f_i)} \frac{m(e_j^1)}{|\mathbf{N}_j^1|} \left[ \frac{1}{4} \left( -3f_j^0 + 3f_j^1 - f_j^3 + f_j^2 \right) - \frac{a_j^1}{2} \left( -f_j^0 - f_j^1 + f_j^3 + f_j^2 \right) \right] + \frac{m(e_j^3)}{|\mathbf{N}_j^3|} \left[ \frac{1}{4} \left( -3f_j^0 - f_j^1 + 3f_j^3 + f_j^2 \right) - \frac{a_j^3}{2} \left( -f_j^0 + f_j^1 - f_j^3 + f_j^2 \right) \right] . \tag{C.15}$$

The mean curvature integral in (7.19) is approximated as

$$\int_{\partial V_i} \nabla_{\mathbf{x}} \mathbf{x} \cdot \mathbf{n} \, \mathrm{d}s \approx \sum_{j \in N_\square(\mathbf{x}_i)} \frac{m(e_j^1)}{|\mathbf{N}_j^1|} \left[ \frac{1}{4} \left( -3\mathbf{x}_j^0 + 3\mathbf{x}_j^1 - \mathbf{x}_j^3 + \mathbf{x}_j^2 \right) - \frac{a_j^1}{2} \left( -\mathbf{x}_j^0 - \mathbf{x}_j^1 + \mathbf{x}_j^3 + \mathbf{x}_j^2 \right) \right] + \frac{m(e_j^3)}{|\mathbf{N}_j^3|} \left[ \frac{1}{4} \left( -3\mathbf{x}_j^0 - \mathbf{x}_j^1 + 3\mathbf{x}_j^3 + \mathbf{x}_j^2 \right) - \frac{a_j^3}{2} \left( -\mathbf{x}_j^0 + \mathbf{x}_j^1 - \mathbf{x}_j^3 + \mathbf{x}_j^2 \right) \right] . \tag{C.16}$$