POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

PC-Gau: PCA Basis of Scattered Gaussians for Shape Matching via Functional Maps

Tesi di Laurea Magistrale in
Computer Science and Engineering - Ingegneria Informatica

Author: **Michele Colombo**

Student ID: 944962
Advisor: Prof. Giacomo Boracchi
Co-advisor: Prof. Simone Melzi
Academic Year: 2021-22

Abstract

Shape matching, i.e. finding correspondences between pairs of shapes, is a key problem in geometry processing, with a wide range of applications, from texture transfer to statistical shape analysis. The functional map approach frames the problem as a correspondence between functional spaces defined on the shapes. Such correspondence admits a compact matricial representation, once a functional basis is defined on each mesh. For many applications, however, dense point-wise maps are required and lots of methods have been proposed to obtain accurate point-wise maps using functional maps. The majority of such methods employed the eigenfunctions of the Laplace-Beltrami operator as the functional basis. We found that a major limitation of the LB basis, though, is that its energy is not evenly distributed on the mesh surface, providing vertices in different areas of the mesh with an inconsistent quality of representation. Our analysis shows that this uneven distribution is a source of errors in the maps obtained. We propose, thus, a new functional basis whose energy is evenly distributed on the whole mesh. We build our basis by applying PCA to a dictionary of Gaussian functions scattered on the mesh surface. The rationale behind is to obtain a uniform representation power, by enforcing the uniform distribution of the initial dictionary. This procedure builds, by construction, an orthonormal basis that is compatible with existing functional map pipelines designed for the standard basis. Through experimental evaluation on established datasets, we show that our basis produces significantly more accurate point-wise maps — compared to the standard basis — when employed in the same shape-matching pipeline. Moreover, the benefits coming from advanced pipelines add up to the benefits given by the adoption of our basis, making our approach complementary to other improvements proposed to the functional map framework. Finally, we devise some metrics to assess the quality of vertex representation provided by our basis. Aside from evaluation, these metrics can be used to perform an accurate choice of the construction parameters, introducing adaptivity in our method.

Keywords: Shape Matching, Functional Maps, Functional Basis

Abstract in lingua italiana

L'allineamento di forme 3D, cioè il problema di stabilire corrispondenze tra di esse, è un problema chiave nell'ambito delle elaborazioni geometriche, con un'ampia gamma di applicazioni, dal trasferimento di texture all'analisi delle forme stesse. L'approccio delle mappe funzionali formula il problema come una corrispondenza tra spazi funzionali definiti sulle mesh. Questa corrispondenza ammette una rappresentazione matriciale compatta, una volta che è stata definita una base funzionale su ogni forma. Molte applicazioni, però, richiedono mappe punto-a-punto. Sono, dunque, stati proposti molti metodi per ottenere mappe puntuali accurate, usando le mappe funzionali. La maggior parte di questi metodi adotta le autofunzioni dell'operatore di Laplace-Beltrami come mappa funzionale. Riteniamo, però, la base LB possiede il grande limite di avere un'energia che non è distribuita uniformemente sulla superficie della mesh, fornendo un'inconsistente qualità di rappresentazione ai vertici in diverse aree della mesh. La nostra analisi evidenzia che questa distribuzione disuniforme è fonte di errori nelle mappe ottenute. Noi proponiamo, quindi, una nuova base funzionale, la cui energia è distribuita uniformemente su tutta la mesh. La nostra base è costruita applicando PCA a un dizionario di Gaussiane sparse sulla superficie della mesh. L'idea alla base, è quella di ottenere un potere di rappresentazione uniforme, imponendo un'uniforme distribuzione del dizionario iniziale. Da questa procedura si ottiene, per costruzione, una base ortonormale compatibile con gli attuali metodi di allineamento, pensati per la base standard LB. Attraverso la valutazione sperimentale su dataset riconosciuti, mostriamo che la nostra base produce mappe puntuali significativamente più accurate rispetto a LB, a parità di metodo di allineamento impiegato. Inoltre, i benefici provenienti dall'uso di pipeline più avanzate si somma ai benefici derivanti dall'impiego della nostra base, rendendo il nostro approccio complementare ad altre migliorie proposte per il framework delle mappe funzionali. Infine, proponiamo alcune metriche per verificare la qualità di rappresentazione dei vertici fornita dalla nostra base. Oltre che alla sua valutazione, queste metriche possono anche essere impiegate per operare una buona scelta dei parametri di costruzione, introducendo adattività nel nostro metodo.

Parole chiave: Allineamento di forme 3D, Mappe funzionali, Base funzionale

Contents

Introduction

Shape matching is a key problem in the context of computer graphics and geometry processing. From an intuitive point of view, it consists in finding correspondences between the points of two 3D shapes. Figure 1 shows an example of shape matching, where the correspondence between two points — one on the man and one on the woman — is rendered through the identity of color.

Shape matching has a wide range of applications, including texture and deformation transfer [24, 30], object retrieval [9], and statistical shape analysis [4]. It is often the preliminary step of a wider processing pipeline. It can be used, for instance, to apply data augmentation on datasets of 3D shapes in order to train better classifiers on them.

The problem is particularly challenging when the shapes in the pair are related by non-rigid deformations. This is the case, for instance, of meshes representing the same human body in different poses, different human bodies in the same pose or a combination of the two, as the example in Figure 1. In the non-rigid setting, the space of possible correspondences is exponential in size.

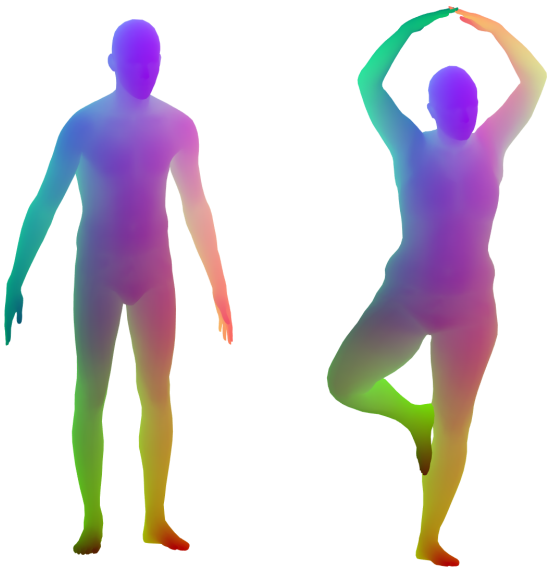


Figure 1: Example of non-rigid shape matching, rendered as a color correspondence between points.

The initial approach was either to solve the problem by heuristic search or to find the correspondence among few points and then extend the mapping to the whole shape (see [33] for a complete survey). The seminal work of Ovsjanikov *et al.* [23] represented a breakthrough in the field: instead of directly finding a point-wise map between shapes, they proposed to put in correspondence functions defined on the shapes. Given a basis for the functional space of each mesh, this functional correspondence can be represented

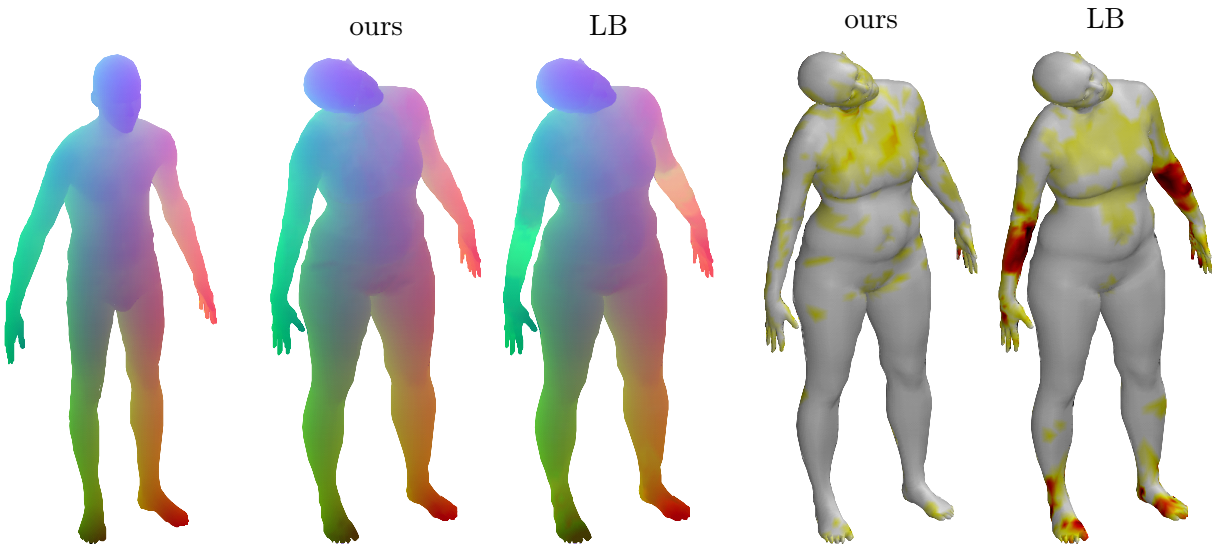
compactly as a matrix C and the optimization problem to find C has linear constraints. This approach immediately imposed itself, due to the level of accuracy reached, and the flexibility and efficiency of the method.

Functional maps provide a compact representation and a convenient way to transfer functions from one mesh to another. However, for many applications it is necessary to recover a dense point-wise map [8], for instance when transferring textures between different shapes. Recovering a point-wise map from a functional map is not a straightforward task and many works, starting from [23] itself, have proposed different methods to accomplish this. Many subsequent works tried to improve point-wise accuracy focusing mainly on improving the estimation of matrix C [16, 21, 26], or the algorithm applied to convert functional maps into point-wise maps [8, 26, 27].

As mentioned before, any method involving functional maps presupposes the use of a basis for the space of functions defined on a mesh. The choice of the basis is a critical aspect of the framework, which heavily affects the final result. However, few alternatives have been proposed in this regard, particularly with the aim of improving the quality of point-wise maps. The vast majority of methods, starting from [23] itself, have adopted the eigenfunctions of the Laplace-Beltrami operator as functional basis. Such basis, that we will refer to as LB for brevity, is the mesh equivalent of the harmonic basis [32] and is optimal for approximating smooth functions on meshes [2]. Nonetheless, LB presents a major limitation in the context of point-wise conversion: its energy is not uniformly distributed on the mesh. In other words, LB's capability of discriminating between vertices and providing them with a meaningful representation is uneven across the mesh surface. In particular, the energy is concentrated on the massive areas, at the expense of narrower extremities. As a consequence, the error of the point-wise maps obtained using LB is localized on such areas, as the example in Figure 2 qualitatively shows.

Following the approach of few other works [10, 14, 20], we propose a new basis for the functional space on a mesh. Our basis is designed to be employed in functional map pipelines and its primary characteristic is to have an energy that is evenly distributed on the mesh surface. We construct our basis by generating a set of Gaussian functions scattered on the mesh and then applying PCA to obtain a set of orthogonal generators. The idea is that PCA, by minimizing the reconstruction error on the samples, produces a basis of a subspace that reflects the even distribution of the Gaussians in the initial dictionary — and the uniform distribution of such Gaussians can be easily enforced.

The embedding space created by our basis provides vertices with a representation of sufficiently good quality in all areas of the mesh. By quality of representation, we mean



(a) target (b) point-wise maps (c) error

Figure 2: Example of shape matching using functional maps. (b) shows two different point-wise maps between the woman and the man, rendered with colors. They are obtained using our basis and the standard LB basis. (c) shows the error of the respective maps in warm colors: the darker, the higher. LB presents a high error on extremities, such as arms and feet.

the following properties of an embedding space: (i) to discriminate well between different vertices and (ii) to preserve the locality of vertices. At the same time, our basis retains good properties of LB, such as orthonormality, frequency ordering and isometry invariance, making it a viable replacement in pipelines designed for LB. This compatibility makes our approach complementary to other proposals tackling different aspects of the functional map framework, since they can be used in combination with our basis to obtain greater benefits.

Through experiments on established datasets, we show that the aforementioned properties of our basis actually result in an improvement of the quality of the final point-wise map. We compare the results of both our basis and LB in different shape-matching pipelines [16, 21] and also using functional maps computed from ground-truth correspondence. Note that the last setting is not a viable option in practical applications, but allows us to rule out the impact of the method used to estimate C from the comparison. Our basis reaches significantly higher values of point-wise accuracy in all the analyzed settings.

Outline In Chapter 1 and 2 we start by providing the necessary background knowledge on 3D shapes and shape-matching, with particular focus on the functional map framework. We also present the notation used in the rest of the Thesis. In Chapter 3 we present the procedure to build our basis, together with a brief analysis of the characteristics that descend directly from its construction. These characteristics allow us to employ our basis and LB in the same pipelines, in order to compare their results in terms of accuracy on the final point-wise maps. We perform this experimental evaluation for different pipelines in Chapter 4. In Chapter 5 we provide an analysis of the properties that make our basis good for point-wise conversion, with a focus on their energy distribution on the mesh surface. In Chapter 4, we also analyze the choice of the parameters used to build the basis, and their impact on the final result. Finally, in Chapter 6, we sum everything up, presenting some limitations of our work and some possible directions to explore in the future.

1 | 3D shapes

In this Chapter, we briefly present the models that we will adopt to represent 3D shapes. We also provide the very essential background knowledge on functional analysis in the context of 3D shapes. However, any detail that is not necessary to understand the main topic of this Thesis is avoided: for an in-depth description, see [32]. We start by considering the continuous case in Section 1.1, which provides the theoretical ground of the concepts and tools presented. Then, we switch to the discrete setting in Section 1.2 and see how such concepts and tools translate to the discrete case. In the rest of the thesis we will always consider the discrete setting, since it is the one that is used in practical implementations.

1.1. Continuous case: 2D manifolds

In the continuous (theoretical) setting, we model shapes as compact two-dimensional manifolds embedded in $\mathbb{R}^3$. The intuitive idea of manifolds is a surface in $\mathbb{R}^3$ that locally resembles a two-dimensional plane. Let us consider a manifold $\mathcal{X}$ and two scalar functions $f : \mathcal{X} \to \mathbb{R}$ and $g : \mathcal{X} \to \mathbb{R}$ defined on $\mathcal{X}$. f and g assign a value in $\mathbb{R}$ to any point of $\mathcal{X}$. We can define the inner product between f and g as:

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

and the norm of f as:

$$\|f\|^2_{\mathcal{X}} = \langle f, f \rangle_{L^2(\mathcal{X})}$$

We denote the space of square-integrable functions defined on $\mathcal{X}$ as

$$L^2(\mathcal{X}) = \left\{ f : \mathcal{X} \to \mathbb{R} \text{ s.t. } \|f\|^2_{\mathcal{X}} < \infty \right\}$$

1.1.1. Laplace Beltrami operator

The Laplace-Beltrami operator $\Delta_{\mathcal{X}}$ is a differential operator from $L^2(\mathcal{X})$ to $L^2(\mathcal{X})$, which assigns to any function $f \in L^2(\mathcal{X})$ the divergence of its gradient: $\Delta_{\mathcal{X}} f = -\operatorname{div}_{\mathcal{X}}(\nabla f)$. We omit other details on the definition of LB and we focus on its eigenfunctions, which provide the tools for spectral analysis on a manifold.

An eigenfunction of an operator is a function ϕ that is only scaled by a factor, when the operator is applied to ϕ. Referring to Laplace-Beltrami operator we have that its eigenfunctions are such that:

$$\Delta_{\mathcal{X}} \phi_i = \lambda_i \phi_i$$

The Laplace-Beltrami operator is self-adjoint, which means that

$$\langle \Delta_{\mathcal{X}} f, g \rangle_{L^2(\mathcal{X})} = \langle f, \Delta_{\mathcal{X}} g \rangle_{L^2(\mathcal{X})}$$

From this property, two important consequences follow. First, its eigenvalues λ_i are real non-negative and constitute a discrete set, therefore we can put them — and the corresponding eigenfunctions — in ascending order:

$$\lambda_1 < \lambda_2 < \ldots < \lambda_i < \ldots$$

Second, its eigenfunctions ϕ_i are orthonormal according to the inner product:

$$\begin{cases} \langle \phi_i, \phi_j \rangle_{L^2(\mathcal{X})} = 1 & \text{if } i = j \\ \langle \phi_i, \phi_j \rangle_{L^2(\mathcal{X})} = 0 & \text{if } i \neq j \end{cases} \tag{1.1}$$

Figure 1.1 shows the first eigenfunctions of the Laplace-Beltrami operator for an example manifold. Note that $\lambda = 0$ is always an eigenvalue for $\Delta_{\mathcal{X}}$, therefore ϕ_1 is always the constant function with $\|\phi_1\|_{\mathcal{X}} = 1$ (we usually choose the positive one).

1.1.2. Spectral analysis on manifolds

The eigenfunctions of Laplace-Beltrami form an orthonormal basis $\Phi_{\text{LB}} = \{\phi_i\}$ for $L^2(\mathcal{X})$. This basis is the equivalent of the Fourier basis on a mesh, and it is often referred to as *Manifold Harmonics* [32]. Given an orthonormal basis $\Psi = \{\psi_i\}$ for $L^2(\mathcal{X})$, we can

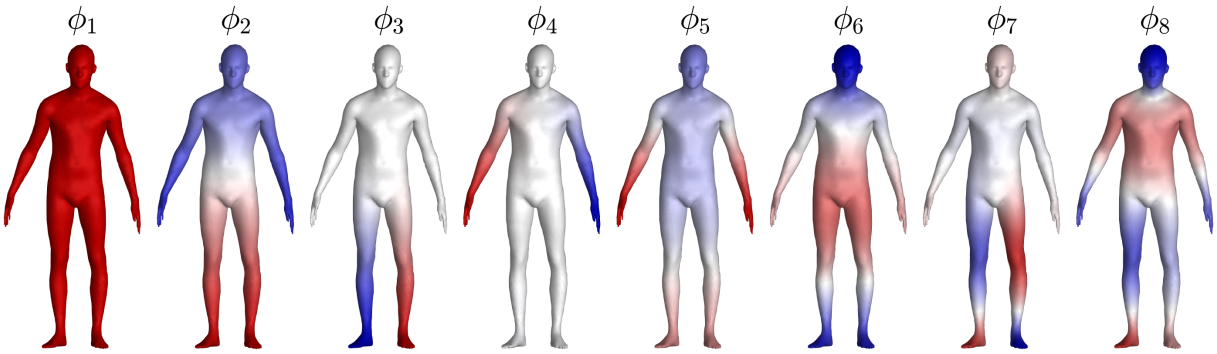


Figure 1.1: An example of the first 8 eigenfunctions of the Laplace-Beltrami operator on a manifold.

express any function $f \in L^2(\mathcal{X})$ as the series:

$$f = \sum_i \underbrace{\langle f, \psi_i \rangle_{L^2(\mathcal{X})}}_{\hat{f}_i} \psi_i \tag{1.2}$$

where $\hat{f}_i$ are the coefficients of f in Ψ. Note that the coefficients $\hat{f}_i$ are easily computed through an inner product thanks to the orthonormality of Ψ.

Dirichlet energy The Dirichlet energy is a measure of the smoothness of a function $f \in L^2(\mathcal{X})$ on a manifold. It is computed as:

$$E_d(f) = \frac{1}{2} \int_{\mathcal{X}} \|\nabla_{\mathcal{X}} f(x)\|_2^2 \; dx = \langle f, \Delta_{\mathcal{X}} f \rangle_{L^2(\mathcal{X})} \qquad (\geq 0)$$

The Dirichlet energy can also be interpreted as the frequency of a function on a mesh. This is in accordance with the intuitive idea that the constant function has zero frequency and the more a function has rapid variations, the more its frequency increases. Since $\Delta_{\mathcal{X}} \phi_i = \lambda \phi_i$ for an eigenfunction ϕ_i, the Dirichlet energy of ϕ_i coincides with its eigenvalue λ_i, thus the eigenfunctions are ordered by increasing frequency. It can be shown that Φ_{LB} is the orthonormal basis with the lowest Dirichlet energy of the atoms.

Truncated basis The series in (1.2) can be truncated at $i \leq k$, namely

$$\tilde{f} = \sum_{i=1}^{k} \underbrace{\langle f, \psi_i \rangle_{L^2(\mathcal{X})}}_{\hat{f}_i} \psi_i$$

In the this case, $\tilde{f}$ is an approximation of f and the truncated basis $\Psi_k = \{\psi_i\}_{1 \ldots k}$ is basis of a subspace $R_k \subset L^2(\mathcal{X})$.

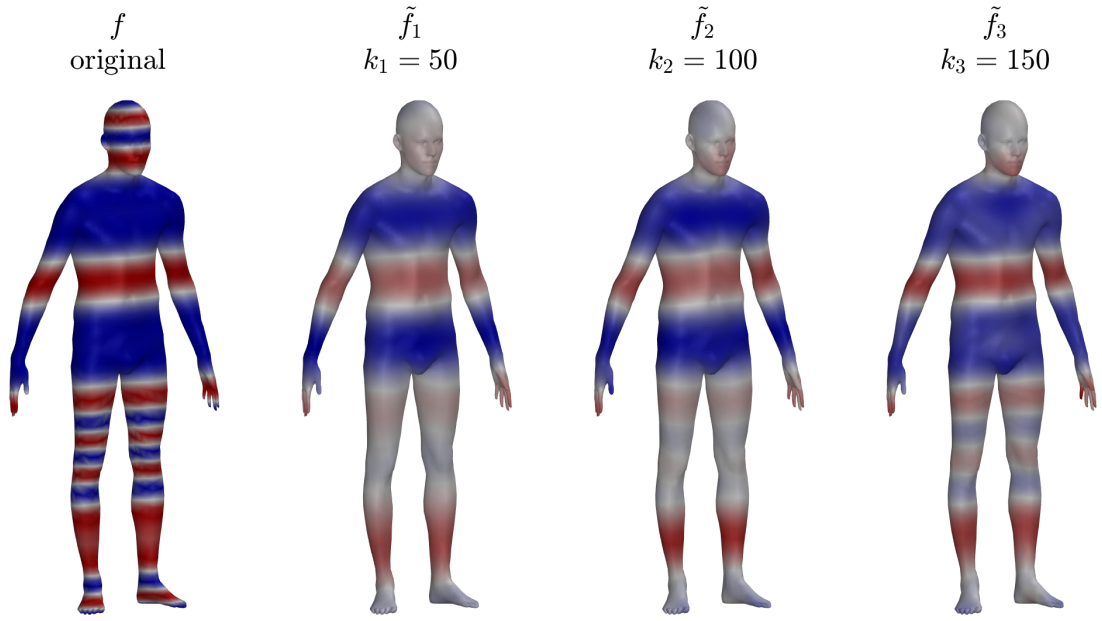


Figure 1.2: Low-pass filter approximation of a function f, for a different number k of atoms used. Smooth variations are well reconstructed already at $k = 50$, while for approximating rapid variations well (knees and head) more atoms are needed.

In the case of the LB basis Φ_{LB}, $\tilde{f}$ is a low-pass filter approximation of f, because the basis functions are ordered in increasing frequency. Figure 1.2 gives a visual idea of what we mean by low-pass filter approximation $\tilde{f}$ of a function f. It has been proven [2] that the truncated basis provided by the first k eigenfunctions of the Laplace-Beltrami operator is optimal for approximating smooth functions on manifolds.

1.2. Discrete case: 3D meshes

In the discrete setting, a manifold $\mathcal{X}$ is represented as a 3D mesh $\mathcal{M}$ $(V_{\mathcal{M}}, E_{\mathcal{M}}, F_{\mathcal{M}}, X_{\mathcal{M}})$:

- $V_{\mathcal{M}} = \{1, \dots, n\}$ is the ordered set of the n vertices, each one represented through an integer index
- $E_{\mathcal{M}} \subset (V_{\mathcal{M}} \times V_{\mathcal{M}})$ is the set of edges
- $F_{\mathcal{M}} \subset (V_{\mathcal{M}} \times V_{\mathcal{M}} \times V_{\mathcal{M}})$ is the set of triangular faces
- $X_{\mathcal{M}} \in \mathbb{R}^{n \times 3}$ is a matrix containing the Euclidean coordinates of the vertices $V_{\mathcal{M}}$ in $\mathbb{R}^3$. They are obtained by sampling n points from $\mathcal{X}$.

Note that the tuple $(V_{\mathcal{M}}, E_{\mathcal{M}})$ is a graph of n vertices, denoted as mesh graph.

The mass (or area) matrix $A_{\mathcal{M}}$ is a diagonal $n \times n$ matrix containing the elements of area associated to each vertex of $\mathcal{M}$. The element of area a_{ii} associated to vertex i is computed

as:

$$a_{ii} = \frac{1}{3} \sum_{jk \text{ s.t. } (ijk) \in F_{\widetilde{M}}} A_{(ijk)}$$

where $A_{(ijk)}$ is the area of the face $(ijk) \in F_{\mathcal{M}}$

In the following, we present the discrete version of all the concept and tools presented in Section 1.1.

Functions A function f defined on a discrete mesh $\mathcal{M}$ assigns a value in $\mathbb{R}$ to each vertex of $\mathcal{M}$: $f : V_{\mathcal{M}} \to \mathbb{R}$. Since the order of the vertices in $V_{\mathcal{M}}$ is fixed, we can represent f as a column vector of real values: $f \in \mathbb{R}^n$. We denote the space of real valued functions defined on $\mathcal{M}$, or *functional space* of $\mathcal{M}$, as $\mathcal{F}(\mathcal{M}, \mathbb{R})$. In the discrete setting the functional space has dimension equal to the number of vertices n. A functional basis for $\mathcal{F}(\mathcal{M}, \mathbb{R})$ is a linearly independent set of n generators belonging to $\mathcal{F}(\mathcal{M}, \mathbb{R})$. As done in the continuous setting, we can also consider a truncated basis of size k, in this case the associated functional space is a k-dimensional subspace $R \subset \mathcal{F}(\mathcal{M}, \mathbb{R})$.

We can store an (ordered) set of functions as column vectors of a matrix of size $n \times h$, where h is the size of the set. For instance, we will often consider functional basis as matrices in $\mathbb{R}^{n \times k}$, where $k \leq n$ is the size of the basis.

Inner Product The inner product of two functions $f \in \mathcal{F}(\mathcal{M}, \mathbb{R})$ and $g \in \mathcal{F}(\mathcal{M}, \mathbb{R})$ is computed using the usual scalar product between vectors, but weighted for the mass matrix, in order to account for possible differences in vertex density [29]:

$$\langle f, g \rangle_{\mathcal{M}} = f^T \cdot A_{\mathcal{M}} \cdot g \tag{1.3}$$

The norm of the function f is defined accordingly:

$$\|f\|^2_{\mathcal{M}} = \langle f, f \rangle_{\mathcal{M}} = f^T \cdot A_{\mathcal{M}} \cdot f$$

Orthonormality In the discrete setting, the orthonormality of a basis $\Phi = \{\phi_i\}$ is defined similarly to the continuous case (see (1.1)), but using the discrete inner product:

$$\begin{cases} \langle \phi_i, \phi_j \rangle_{\mathcal{M}} = \phi_i^T \cdot A_{\mathcal{M}} \cdot \phi_j = 1 & \text{if } i = j \\ \langle \phi_i, \phi_j \rangle_{\mathcal{M}} = \phi_i^T \cdot A_{\mathcal{M}} \cdot \phi_j = 0 & \text{if } i \neq j \end{cases} \tag{1.4}$$

Equivalently, we can also express orthonormality by considering the matricial form of a basis $\Phi \in \mathbb{R}^{n \times k}$. Φ is thus orthonormal if:

$$\Phi^T A_{\mathcal{M}} \Phi = I_k$$

where I_k is the identity matrix of size k and k, with $1 \leq k \leq n$, is the size of the (truncated) basis.

The vector of coefficients $\hat{f}$, projection of a function f on an orthonormal basis Φ, is computed, again, using the discrete inner product in (1.3):

$$\hat{f} = \Phi^T A_{\mathcal{M}} f \tag{1.5}$$

The inverse operation, which is the reconstruction (synthesis) of function $\tilde{f}$ from its spectral coefficients $\hat{f}$ is performed as:

$$\tilde{f} = \Phi \cdot \hat{f} \tag{1.6}$$

Note that, in general, $\tilde{f} = \Phi\Phi^T A_{\mathcal{M}} f$ is a rank-k approximation of the original f when the basis is to truncated to the first k atoms, and $\tilde{f} = f$ when the basis is complete.

1.2.1. Discrete Laplace-Beltrami operator

The Laplace-Beltrami operator can be discretized as a $n \times n$ matrix $L_{\mathcal{M}} = A_{\mathcal{M}}^{-1} W$. W is the stiffness matrix, a sparse matrix that encodes information about the local geometry of $\mathcal{M}$. The elements in W are computed using the cotangent scheme [18, 25] as:

$$w_{ij} = \begin{cases} \dfrac{\cot \alpha_{ij} + \cot \beta_{ij}}{2} & \text{if } (i,j) \in E_{\mathcal{M}} \\ -\displaystyle\sum_{k \neq i} w_{ik} & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$

where α_{ij} and βij are the angles insisting on edge (i,j), belonging to the two triangles sharing edge (i,j). The eigenfunctions of the Laplace-Beltrami operator are computed as solutions of the following generalized eigenvalue problem [29, 32]:

$$W\phi_i = \lambda_i A_{\mathcal{M}} \phi_i$$

The set of eigenfunctions $\Phi_{\text{LB}} = \{\phi_i\}$, with real eigenvalues $\{\lambda_1 \leq \lambda_2 \leq \dots\}$, forms an orthonormal basis for $\mathcal{F}(\mathcal{M}, \mathbb{R})$, where orthonormality is defined as in (1.4). Similarly to

the continuous case, the atoms of Φ_{LB} are ordered in frequency and provide a low-pass filter approximation when truncated to k. The analysis and synthesis operations for a function $f \in \mathcal{F}(\mathcal{M}, \mathbb{R})$ are performed using (1.5) and (1.6).

2 | Shape Matching

In this chapter we present in a formal way the problem we are addressing, namely shape matching, and we define part of the notation that will be used throughout the thesis. We also present in detail the functional map approach for shape matching, to which our proposal fully belongs. Together with the original formulation of functional maps [23], we also present subsequent works that are more related to our proposal.

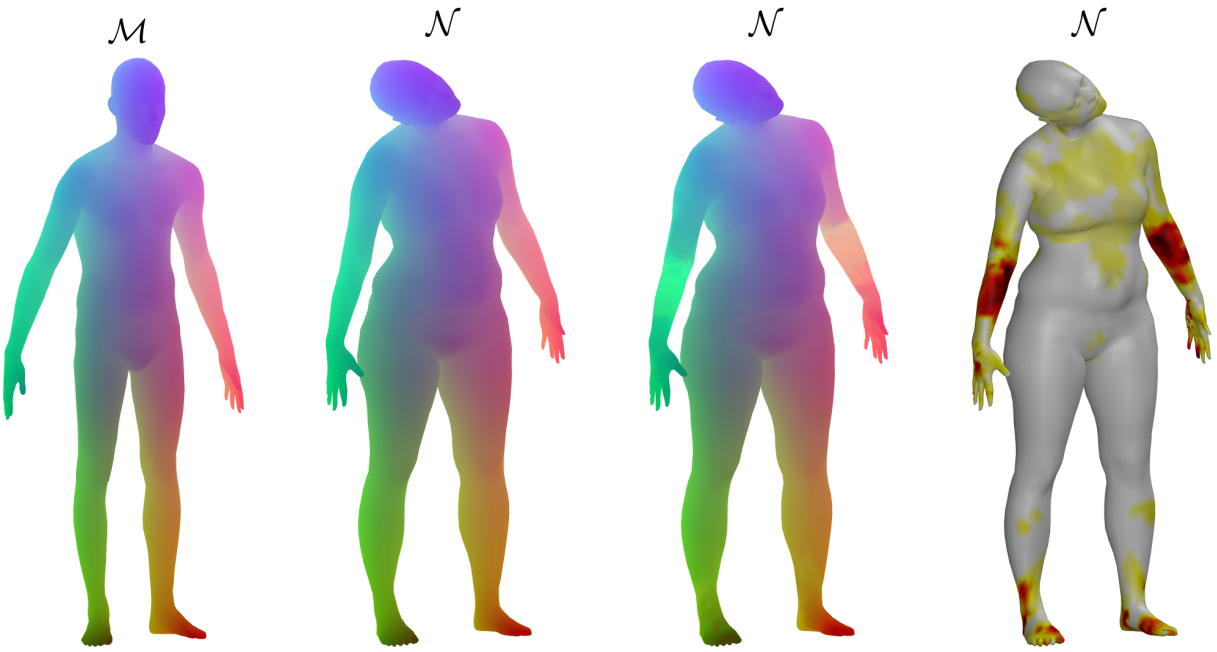
2.1. Problem formulation

The input of shape matching are two meshes $\mathcal{M}$ and $\mathcal{N}$, with sets of vertices $V_\mathcal{M}$ and $V_\mathcal{N}$ respectively. We assume $\mathcal{M}$ and $\mathcal{N}$ to have some sort of similarity, which means that an unknown correspondence $T : V_\mathcal{N} \to V_\mathcal{M}$ exists between the vertices of $\mathcal{N}$ and $\mathcal{M}$. Note that we are considering the case of non-rigid matching, so the transformation between $\mathcal{M}$ and $\mathcal{N}$ is not restricted to a particular class. To picture an example of this kind of relation, we can consider the relation that exists between different human bodies, or between different poses of the same human body or a combination of the two.

The goal of shape matching is to estimate the unknown map T, therefore its output is a function $\bar{T} : V_\mathcal{N} \to V_\mathcal{M}$. We will refer to $\bar{T}$ as point-wise map and we can represent it either as a vector of vertex indices of size $n = |V_\mathcal{N}|$ or as a matrix Π s.t. $\Pi_{ij} = 1$ if $\bar{T}(i) = j$ and 0 otherwise. Figure 2.1 shows an example of this setting: the two meshes, $\mathcal{M}$ and $\mathcal{N}$, and the two maps, T and $\bar{T}$. Point-wise maps are rendered as color correspondences: a vertex on $\mathcal{N}$ is in correspondence with a vertex on $\mathcal{M}$ if and only if they have the same color.

We want $\bar{T}$ to be as close as possible to T, which means that $\bar{T}$ should assign to each vertex $x \in V_\mathcal{N}$ a vertex on $\mathcal{M}$ geodesically close — ideally coincident — to the one assigned by T. Assuming T to be provided, we evaluate the mapping error of each vertex $x \in \mathcal{N}$ as the geodesic distance between its estimated and true image vertices:

$$e(x) = \text{GeoDist}_\mathcal{M}(\bar{T}(x), T(x)) \qquad \forall x \in V_\mathcal{N}$$



(a) target (b) unknown map T (c) estimated map $\bar{T}$ (d) geodesic error

Figure 2.1: Example of shape matching. The point-wise maps between $\mathcal{N}$ and $\mathcal{M}$ are represented through color correspondence. (d) shows the geodesic error of the estimated map (c)

Note that the geodesic error $e(x)$ is a function defined on the mesh $\mathcal{N}$ and it can be visualized through color, as done in Figure 2.1(d). In this example, the error is higher on the forearms and feet, where $\bar{T}$ is assigning wrong vertices (see Figure 2.1(c)). To asses the global *accuracy* of a point-wise map we take the average of the geodesic error on all the vertices of $\mathcal{N}$, computing the Average Geodesic Error (AGE). We will present in more detail the evaluation of point-wise maps in Section 4.3.

In some of the experiment of Chapter 4, we will also assume that a small (~ 6) set of landmarks is provided. A landmark is a couple of points $(x \in V_{\mathcal{N}}, y \in V_{\mathcal{M}})$ in known correspondence, namely $T(x) = y$. Note that landmarks are not necessary to build the functional basis (either our basis or LB), but are only required to apply specific matching pipelines.

2.2. Functional maps

The space of possible point-wise maps is exponential in size, and this makes the problem of non-rigid shape-matching very difficult. Trying to optimize directly the point-wise map is unfeasible. Even restricting the space of possible maps to isometric correspondences (see Section 3.2.3) and optimizing for the map that best preserves geodesic distances has been

shown to be a NP-hard problem [6]. The usual approach has been to find correspondences between a small set of points and then extend them to a dense correspondence. Refer to [33] for a complete survey.

The Functional Map framework was proposed by Ovsjanikov *et al.* in [23] and represented a breakthrough in the approach to the shape-matching problem. Let us consider again meshes $\mathcal{M}$ and $\mathcal{N}$ and the unknown point-wise map $T : V_{\mathcal{N}} \to V_{\mathcal{N}}$. Instead of estimating T directly, the functional map approach proposes to solve for the correspondence between functions defined on $\mathcal{M}$ and $\mathcal{N}$ first and then to extract from this functional correspondence a point-wise map.

This approach is based on the observation that the point-wise map T induces a *linear* operator $T_F : \mathcal{F}(\mathcal{M}, \mathbb{R}) \to \mathcal{F}(\mathcal{N}, \mathbb{R})$ that maps functions from $\mathcal{M}$ to $\mathcal{N}$ via the composition:

$$T_F(f) = f \circ T \qquad \forall f \in \mathcal{F}(\mathcal{M}, \mathbb{R}) \tag{2.1}$$

where

$$g = f \circ T \iff g(x) = f(T(x)) \quad \forall x \in V_{\mathcal{N}}$$

Let us suppose that both $\mathcal{M}$ and $\mathcal{N}$ are equipped with *orthonormal* functional bases $\Phi = \{\phi\}_i$ and $\Psi = \{\psi\}_j$, for $\mathcal{F}(\mathcal{M}, \mathbb{R})$ and $\mathcal{F}(\mathcal{N}, \mathbb{R})$ respectively. By extension, we denote with Φ and Ψ also their matricial form, where atoms are represented as column vectors. As shown in Section 1.2, any function $f \in \mathcal{F}(\mathcal{M}, \mathbb{R})$ can be written as

$$f = \sum_i \langle f, \phi_i \rangle_{\mathcal{M}} \cdot \phi_i = \sum_i a_i \phi_i$$

Due to the linearity of T_F, $T_F(f)$ can be written as:

$$T_F(f) = T_F\left(\sum_i a_i \phi_i\right) = \sum_i a_i T_F(\phi_i) \tag{2.2}$$

Recall that $\phi_i \in \mathcal{F}(\mathcal{M}, \mathbb{R})$ and thus, due to (2.1), $T_F(\phi_i)$ is a function in $\mathcal{F}(\mathcal{N}, \mathbb{R})$, which can be expressed in the basis Ψ:

$$T_F(\phi_i) = \sum_j \underbrace{\langle T_F(\phi_i), \psi_j \rangle_{\mathcal{N}}}_{c_{ji}} \psi_j$$

Substituting in (2.2) we obtain:

$$T_F(f) = \sum_i a_i \sum_j c_{ji}\psi_j = \sum_{ji} a_i c_{ji}\psi_j = \sum_j b_j \psi_j \tag{2.3}$$

where $b_j = \sum_i a_i c_{ji}$ are the coefficients of $T_F(f)$ in Ψ. Recalling from Section 1.2 the projection over an orthonormal basis, we can also write (2.3) in matricial form:

$$T_F(f) = \Psi \mathbf{b} = \Psi C \mathbf{a} = \Psi C \Phi^T A_{\mathcal{M}} f \tag{2.4}$$

where $\mathbf{a} = [a_i]$, $\mathbf{b} = [b_j]$ and $C = [c_{ji}]$. Note that coefficients c_{ji}, and thus C, depend only on Φ, Ψ and T_F. Therefore, given Φ and Ψ, we can transfer any function by knowing a matrix C that maps the atoms of Φ on Ψ. Therefore, in the functional map framework, matching two shapes resorts to finding the change-of-basis matrix C between the functional bases defined on the meshes.

Truncated basis In practice, we never use bases of the complete functional space $\mathcal{F}(\cdot, \mathbb{R})$, which has dimension equal to the number of vertices n of the mesh. The bases Φ and Ψ are truncated, instead, to the first k atoms, with $k \ll n$. This corresponds to truncating the series in (2.2) and (2.3) to k. This rank-k approximation of the functional correspondence is good as long as:

1. the subspace $R_{\mathcal{M}} \subset \mathcal{F}(\mathcal{M}, \mathbb{R})$ generated by the first k atoms of Φ provides a good approximation of the functions in $\mathcal{F}(\mathcal{M}, \mathbb{R})$
2. the functional spaces spanned by the first k atoms of Φ and Ψ, respectively $R_{\mathcal{M}} \subset \mathcal{F}(\mathcal{M}, \mathbb{R})$ and $R_{\mathcal{N}} \subset \mathcal{F}(\mathcal{N}, \mathbb{R})$, are reasonably aligned, even when the relation between $\mathcal{M}$ and $\mathcal{N}$ drifts from pure isometry.

These two properties of the bases are respectively called *compactness* and *stability* [23]. The standard choice of functional basis for the functional map framework, already considered in [23], is to use the first k eigenfunctions of the Laplace-Beltrami operator. We will denote the standard basis as "LB" in the rest of the Thesis. Some alternative bases have been proposed, we will briefly present them and compare to our work in Section 2.3.1.

For the sake of simplicity of this presentation of [23], we have assumed that both meshes have n vertices and both truncated basis have size k. Note that these assumptions can be easily removed, and indeed our implementation does not use them.

2.2.1. Finding C

In the truncated formulation, C is a compact matrix of size $k \times k$, independently of the size of the meshes. It is computed through an optimization problem, as the matrix that best preserve (in the least square sense), some constraints. The majority of these constraints, moreover, are linear in C, therefore the optimization problem can be solved efficiently. In the following, we briefly present various types of constraints, both as formulated in [23] and in some subsequent works.

Functional constraints

Let $\{f_i\}$ and $\{g_i\}$ be two sets of m functions defined on $\mathcal{M}$ and $\mathcal{N}$ in known pair-wise correspondence, namely:

$$g_i = T_F(f_i) \quad \forall i \in \{1, \dots, m\} \qquad \text{with } f_i \in \mathcal{F}(\mathcal{M}, \mathbb{R}), g_i \in \mathcal{F}(\mathcal{N}, \mathbb{R}) \tag{2.5}$$

where T_F depends on the unknown map T. Let us denote with $F = [f_i]$ and $G = [g_i]$ the matrices containing the functions in $\{f_i\}$ and $\{g_i\}$ as column vectors. F and G can be projected on two given (truncated) functional bases Φ and Ψ, for $\mathcal{M}$ and $\mathcal{N}$ respectively, as $\hat{F} = \Phi^T A_{\mathcal{M}} F$ and $\hat{G} = \Psi^T A_{\mathcal{N}} G$. Recalling (2.4), the Equation (2.5) can be formulated in terms of an unknown functional map C:

$$\hat{G} = C \cdot \hat{F} \tag{2.6}$$

and the corresponding least-square minimization problem:

$$\min \left\| C\hat{F} - \hat{G} \right\|_2^2 \tag{2.7}$$

As we mentioned before, functional constraints are linear in C.

Types of functional constraints Functional constraints in (2.7) can model the following constraints as special cases:

- Descriptor preservation: descriptors are family of functions designed to characterize each point of a mesh as uniquely as possible and to be preserved under near-isometric transformations. They usually exploit some isometry-invariant property, like heat diffusion [31] or quantum mechanical properties of the shapes [3].
- Landmark correspondence: a landmark ($x \in \mathcal{M}, y \in \mathcal{N}$) can be transformed in two corresponding functions f_i and g_i by considering the distance function from x and

y or functions centered in x and y, such as Gaussians or Heat Kernels.

- Segment correspondence: segments can be modelled as (smooth) indicator functions or a distance function from the segment.

Operator Commutativity

C can be required to commute with two discrete operators $B_{\mathcal{M}}$ and $B_{\mathcal{N}}$ defined on $\mathcal{F}(\mathcal{M}, \mathbb{R})$ and $\mathcal{F}(\mathcal{N}, \mathbb{R})$ respectively. This constraint is framed as

$$B_{\mathcal{N}}C = CB_{\mathcal{M}} \implies \min \|B_{\mathcal{N}}C - CB_{\mathcal{M}}\|_2^2 \tag{2.8}$$

(2.8) leads, again, to a linear constraint on C. The typical application of this constraint is to enforce the commutativity with the Laplace-Beltrami operator.

In [21] operator commutativity has also been used to enforce the preservation of point-wise products of functional constraints, through the definition of an operator for each functional constraint. The point-wise product of two functions $f(x), g(x) \in \mathcal{F}(\mathcal{M}, \mathbb{R})$ is defined as $f(x)g(x) \forall x \in V_{\mathcal{M}}$. Using the point-wise products of functional constraints allows to extract more information from such constraints and thus to solve a better-conditioned optimization problem, even when the number of linearly independent constraints is not high. In Section 4.5, we will use [21] to estimate C as a possible setting for the experimental evaluation of our basis.

Regularization constraints

When the unknown map T underlying C is assumed to be an isometry, C is orthogonal [23]. This requirement can be framed as

$$C^T = C \implies \min \left\|C^T - C\right\|_2^2$$

and is used as a regularization constraint.

In [26] an additional regularization term has been proposed to promote the preservation of the orientation in the obtained map C.

2.2.2. Conversion to point-wise map

To completely solve the shape matching problem, as we formulated it in Section 2.1, the functional map C from $\mathcal{M}$ to $\mathcal{N}$ needs to be converted to a point-wise map $\bar{T} : V_{\mathcal{N}} \to V_{\mathcal{M}}$. First, note that the knowledge of T_F contains the knowledge of T, as we could, in principle,

transfer Delta functions via T_F and put in correspondence the peaks on $\mathcal{M}$ and $\mathcal{N}$. This method, however, is very inefficient and error prone, and never used in practice.

Embedding We define the *embedding* of a vertex $x \in \mathcal{M}$, for a given basis Φ, as the vector of values assumed by basis functions in x: $\text{Emb}(x) = [\phi_i(x)]$. For a truncated basis of size k, the embedding is thus a vector in $\mathbb{R}^k$. $\text{Emb}(x)$ corresponds also to the projection coefficients, in the basis Φ, of a Delta function centered in the vertex x [23]. Thus Φ^T contains the coefficients of all the Delta functions of $\mathcal{M}$ (one for each vertex) as column vectors. Note that embeddings offers a different representation of vertices [29], in a space that is independent of translations, rotations and isometric pose changes of the mesh (as long as Φ is isometry-invariant). We denote this k-dimensional space as *embedding space*. Note that similarity in the embedding space has always to be intended regarding the Euclidean distance between embedding vectors, and by embedding distance of two vertices x and y we mean $\|\text{Emb}(x) - \text{Emb}(y)\|_2$.

Original method A simple and efficient method for converting C, from $\mathcal{M}$ to $\mathcal{N}$, into a point-wise map, proposed already in [23], consists in finding, for each column of $\Phi_{\mathcal{N}}^T$, the nearest neighbour in the columns of $C\Phi_{\mathcal{M}}^T$. The output is a map $\bar{T} : \mathcal{N} \to \mathcal{M}$. This method corresponds to transferring embeddings of vertices from $\mathcal{M}$ to $\mathcal{N}$ through C and putting in correspondence similar points in the embedding space. This method is based on the Plancherel's theorem:

Theorem 2.1. *Given two functions f_1 and f_2 defined on a manifold $\mathcal{M}$, with spectral coefficients $\mathbf{a_1}$ and $\mathbf{a_2}$, it holds that:*

$$\|\mathbf{a_1} - \mathbf{a_2}\|_2^2 = \int_{\mathcal{M}} (f_1(x) - f_2(x))^2 \mu(x) \tag{2.9}$$

If f_1 and f_2 are two functions centered in two vertices x and y, we can assume that their L^2 difference (right-hand side in (2.9)) is strictly related to the geodesic distance between x and y. Recalling that embeddings are the spectral coefficients of Delta functions, this theorem provides a relation between the distance of two vertices in the embedding space (left-hand side in (2.9)) and their geodesic distance.

Locality preservation For truncated bases, the equality in (2.9) does not hold precisely and depends on the specific basis adopted. The more this relation between embedding and geodesic distances from x is preserved among the neighbors of x, the more we say that a certain basis is *locality preserving* for x. In the opposite case, when the relation

between geodesic distances and embedding distances is not preserved well, we say that the embedding space produces a *distortion* of geodesic distances. In this unfortunate case, vertices that are far on the mesh may get similar embeddings, or vice versa. Since the conversion is performed through a nearest neighbor search in the embedding space, we are interested, in particular, that the ordering between geodesic distances and embedding distances, from a given vertex x, is preserved. We will define proper metrics to evaluate locality preservation quantitatively in Section 5.1.2.

2.3. Relation to previous work

In this section we briefly analyze some extensions of the original framework [23] and we discuss their relation to our work. We have already seen some of these [21, 26] in Section 2.2.1.

Point-wise conversion Some works have proposed alternative methods, with respect to [23], for extracting better point-wise maps from functional maps. In [27, 28] they propose to frame the problem of converting to point-wise map as the problem of finding the point-wise map that maximizes the posterior probability of a particular probability distribution. The probability distribution is defined over points in the embedding space, thus locality preservation, as defined in Section 2.2.2, still plays an important role. In [8] they introduce a smoothness prior on the point-wise map to regularize the conversion and obtain better maps. A similar purpose is targeted in [26], where they propose a complex, multi-step pipeline to promote continuity, coverage and bijectivity of the obtained map.

Iterative methods [27] can also act as refinement of an existing map $\bar{T}$. In this case, the given $\bar{T}$ is converted into a functional map and then back to a point-wise map with [27]. The ICP refinement proposed in [23] is based on the similar idea of alternating conversions to and from point-wise map.

ZoomOut [16] further expands the usage of this technique — of alternating conversions to and from point-wise maps — devising an iterative procedure which allows to extend an initial functional map of size $k' \times k'$ to a much bigger map $k \times k$. The key point is that at each iteration i the size k_i of the correspondence can be increased by converting the point-wise map back to a functional map of size $k_{i+1} > k_i$.

Relation to our work All these methods share with our work the general purpose of improving the quality of shape matching via functional maps. However, they focus on different steps of the general pipeline, namely on the estimation of C [16, 21, 26] or on

the algorithm for converting it to a point-wise map [8, 26, 27]. As we saw in Section 2.2, any method based on functional maps needs to define a functional basis on each mesh. The methods presented here, despite being agnostic to the functional basis adopted, make the implicit assumption, ubiquitous from [23] on, to be using the eigenfunctions of the Laplace-Beltrami operator [11, 32]. We, instead, propose to use a different functional basis. Note, as well, that our basis is agnostic to the methods used for computing the map C and converting it to point-wise maps. This two-way compatibility makes the two approaches, of using a different basis and using an improved shape-matching pipeline, complementary.

2.3.1. Alternative bases

Other works followed an approach more similar to ours and proposed new bases for the functional space of a mesh. However, they either target specific tasks, not directly related to our work, or they present some limitations. Examples of specific tasks for which alternative bases have been proposed include transfer of tessellation structure [17] and transfer of step functions [13] from one mesh to another. [20] promotes the sparsity of C, without bringing improvements to point-wise accuracy. Kovnatsky *et al.* [10] built a coupled pair of bases by joint diagonalization, in order to overcome the instability of LB in non-isometric pairs. With this method, the functional basis depends also on the other mesh in the pair and on the landmarks used. With a purpose more similar to ours, Melzi *et al.* [14] proposed a basis that extends LB in specific areas of the mesh. Their approach is, however, different: the atoms of their bases are localized in predefined regions, which must be provided as input. This is not trivial and presupposes the knowledge, at least roughly, of the correspondence between the shapes. Our basis, on the contrary, does not require any input other than the mesh itself. Finally, [22] and [12] showed that a functional basis can be extended by considering also point-wise products of basis atoms. This method provides great benefits to the obtained point-wise maps and can be applied, in principle, to any basis, including ours. [22] and [12] are thus complementary to our proposal.

2.4. Limits of LB

LB has been proven to be optimal for approximating functions with bounded variations on meshes [2]. This fact makes it a compact basis, according to the definition of compactness given in Section 2.2. It is also isometry invariant and moderately stable under non-isometries (see again Section 2.2 for a definition of stability). These characteristics make

it an appropriate basis for functional maps and, indeed, from [23] on, LB has been the ubiquitous choice of functional basis in this framework.

Despite its many strengths, LB presents an important limitation: the basis energy is not evenly distributed on the surface of the mesh, but is concentrated in the massive areas, leaving narrower extremities less covered. To define the energy of a basis, we consider two properties of the embedding space generated by the basis:

- discrimination power between different vertices
- locality preservation, as defined in Section 2.2.2

We claim that a lack of these properties in some areas of the mesh produces bad assignments in the point-wise map, affecting its overall accuracy. It is thus desirable, for a functional basis, to have the aforementioned properties evenly distributed on the mesh surface.

We will present evidences for this claims in Chapter 5, when we will assess the energy distribution of LB basis and compare it to the energy distribution of our basis.

3 | Proposed Solution: PC-Gau

In this chapter we present the core contribution of this Thesis: the procedure for constructing PC-Gau, a new basis for the space of real-valued functions defined on a mesh. Such basis is designed to be used as a truncated basis in functional map pipelines for shape matching. It works as a replacement of LB, with the final purpose of obtaining more accurate point-wise maps. Figure 3.1 shows a complete shape-matching pipeline with functional maps, employing PC-Gau: it shows the most relevant steps of the building procedure (colored boxes), the computed functional map C and the final point-wise map obtained.

In Section 3.1 we present in detail the procedure to build our basis. As the name suggests, PC-Gau is obtained by taking the Principal Components of a dictionary of Gaussian functions. The primary characteristic of our basis is to have an energy which is evenly distributed on the whole mesh surface. As a consequence, PC-Gau induces an embedding space more suited to point-wise conversion, producing benefits in the accuracy of the point-wise maps obtained. At the same time, PC-Gau shares many of the good properties of LB, such as orthonormality, frequency ordering and isometry invariance. This fact makes our basis a suitable replacement of LB in existing functional map pipelines, with virtually no modification needed. We analyse these properties, following directly from the construction process, in Section 3.2.

3.1. Building Procedure

The core idea of our basis is quite simple and actually has been long employed in signal processing: to represent signals, or functions, through a dictionary of Gaussian functions, see for instance [7]. Instead of using the dictionary as it is, though, we reduce the dimensionality of its space through PCA. PCA produces an orthonormal set of generators for the functional space spanned by the Gaussians. Generators are ordered according to their capability of extracting information from the initial space, and thus their energy is distributed in such a way as to approximate the initial dictionary as good as possible. The idea is to obtain an even distribution of basis energy by controlling the uniformity of

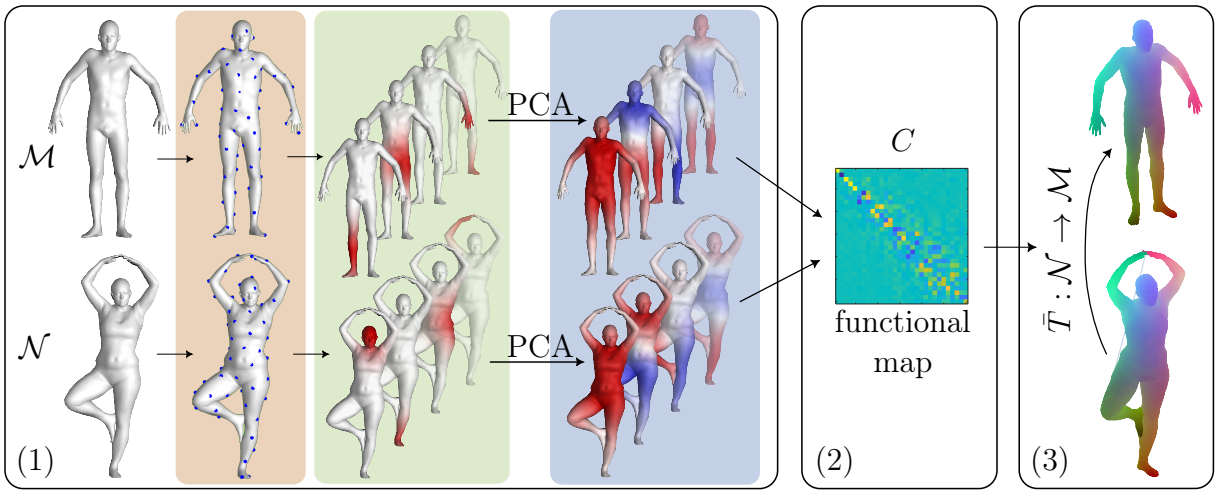


Figure 3.1: Complete shape matching pipeline with functional maps: (1) definition of a functional basis, (2) estimation of C and (3) conversion to a point-wise map. Here, step (1) shows the building procedure of PC-Gau: selection of a subset of vertices (orange box), construction of the dictionary of Gaussian functions (green box) and dimensionality reduction through PCA (blue box).

scattering of the Gaussians in the initial dictionary, since such uniform scattering can be easily enforced.

Algorithm 3.1 presents the step-by-step procedure to build PC-Gau for a mesh $\mathcal{M}$ with n vertices. In the description, we also refer to Figure 3.1 as a visual reference.

Algorithm 3.1 Construction of PC-Gau

input: $\mathcal{M}$, $V_{\mathcal{M}}$, $A_{\mathcal{M}}$, σ, q, normalize, k
$Q = \text{FPS}(\mathcal{M}, q)$
for $i \in V_{\mathcal{M}}$, $j \in Q$ **do**
    $D_{ij} = \text{GeoDist}_{\mathcal{M}}(V_{\mathcal{M}}(i), Q(j))$
    $G_{ij} = \exp\left(-D_{ij}^2/\sigma\right)$
end for
if normalize **then**
    for $i \in V_{\mathcal{M}}$, $j \in Q$ **do**
        $G_{ij} = G_{ij} / \sqrt{G_{*j}^T A_{\mathcal{M}} G_{*j}}$
    end for
end if
$P = \text{PCA}\left(G^T, \text{variableWeights}{=}A_{\mathcal{M}},\ \text{center}{=}\text{false}\right)$
$P_k = P(:, 1{:}k)$
output: P_k

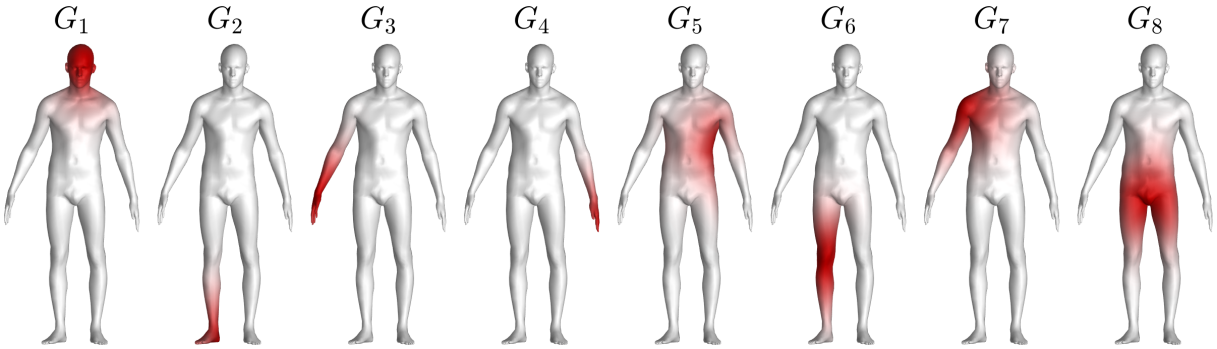


Figure 3.2: Example of some Gaussian functions G_i, generated with $\sigma = 0.05$.

3.1.1. Selection of a subset of vertices

We start by selecting a subset Q of q vertices uniformly scattered on the mesh (line 2 of Algorithm 3.1). For this purpose, we use Farthest Point Sampling (FPS) [19] with Euclidean distance. We want them to be evenly distributed on the surface in order to induce an even distribution of the dictionary of Gaussians. This, in turn, translates into an even distribution of of the final basis, thanks to the properties of PCA. We can see an example of sampling in the orange box of Figure 3.1.

The final result, though, is pretty robust to the specific method employed, as long as it provides a sufficiently even distribution of Q on the mesh. In Section 5.2.1 we will test the use of random sampling for this phase. Even though random sampling depends on the density of vertices on the mesh surface, it still provides good results, often equivalent to FPS.

In our tests we used $q = 1000$ for all datasets, as this parameter is independent from the total number of vertices. See Section 5.2.1 for an analysis on the choice of q.

3.1.2. Dictionary of Gaussian functions

Then, we compute q Gaussian functions, each one centered in a vertex of Q (lines 3-6). To do so, we start by computing the geodesic distance from each vertex $x \in Q$ to any other vertex $i \in V_{\mathcal{M}}$ (lines 3-4). For performance reasons, the geodesic distance is approximated with the shortest path along the edges of the mesh, which can be easily computed with the Dijkstra algorithm. We obtain a matrix D, of size $n \times q$, in which each element D_{ij} is the distance between the j-th vertex in Q and the i-th vertex on the mesh. Then, we apply

$$G_{ij} = exp\Big(- \frac{D_{ij}^2}{\sigma} \Big) \tag{3.1}$$

and obtain a matrix $G = [G_{ij}]$, still of size $n \times q$, in which each column is a Gaussian function centered in a vertex of Q (line 5). The parameter σ in 3.1 is arbitrarily chosen and sets the amplitude of the Gaussians. In our tests we used $\sigma = 0.05$ for all datasets, since the meshes where normalized to unitary area and thus their scale was similar across different datasets. See Section 5.2.2 for an analysis on the impact of the value of σ on the basis and the point-wise maps obtained.

We can optionally normalize each column G_j of G for its norm (lines 7-11), computed as $\|G_j\|_{\mathcal{M}} = \sqrt{G_j^T A_{\mathcal{M}} G_j}$, where $A_{\mathcal{M}}$ is the matrix of area elements associated to the vertices of $\mathcal{M}$. In our experiments we used normalization. An example of some of the obtained Gaussians can be seen in the green box in Figure 3.1 and in Figure 3.2.

3.1.3. Dimensionality reduction and orthogonalization

We compute the PCA of G^T (line 12), meaning that each Gaussian function is considered a sample and each vertex of the mesh a variable. We do not center the variables, but we weight them for the element of area associated to each vertex. The result of PCA is a set of q vectors of size n, called Principal Components (PCs). They can be interpreted as q generators of the functional space S spanned by the Gaussian dictionary. Since $q < n$, S is a q-dimensional subspace of $\mathcal{F}(\mathcal{M}, \mathbb{R})$ (assuming the Gaussians are all linearly independent).

Finally, we select the first k PCs to form our basis and we store them in a matrix P_k of size $n \times k$ (line 13). They are a truncated basis of $\mathcal{F}(\mathcal{M}, \mathbb{R})$ or, equivalently, a basis of a k-dimensional subspace $R \subset S \subset \mathcal{F}(\mathcal{M}, \mathbb{R})$. In particular, for the properties of PCA, the first k PCs form the set of the k orthonormal generators with the lowest reconstruction error on the initial samples [1]:

$$P_k = \underset{P_k \in \mathbb{R}^{n \times k}}{\operatorname{argmin}} \left\{ \sum_{i=1}^{q} \left\| G_i - P_k P_k^T A_{\mathcal{M}} G_i \right\|_2^2 \right\} \text{ s.t. } \quad P_k^T A_{\mathcal{M}} P_k = I_k \tag{3.2}$$

where G_i is the i-th column of G, namely the i-th Gaussian function. Note that orthonormality and projection in (3.2) are expressed with respect to the inner product defined on the mesh (see Section 1.2). Our method uses property (3.2) to distribute evenly the expressive power of the basis, starting from a set of sample functions $\{G_j\}$ that is evenly distributed on the mesh. Intuitively, this is because, if the Gaussians are evenly distributed, each variable (vertex of the mesh) presents a similar amount of variance in the set G. Therefore, each variable (vertex) requires a similar expressive power of the basis to be represented.

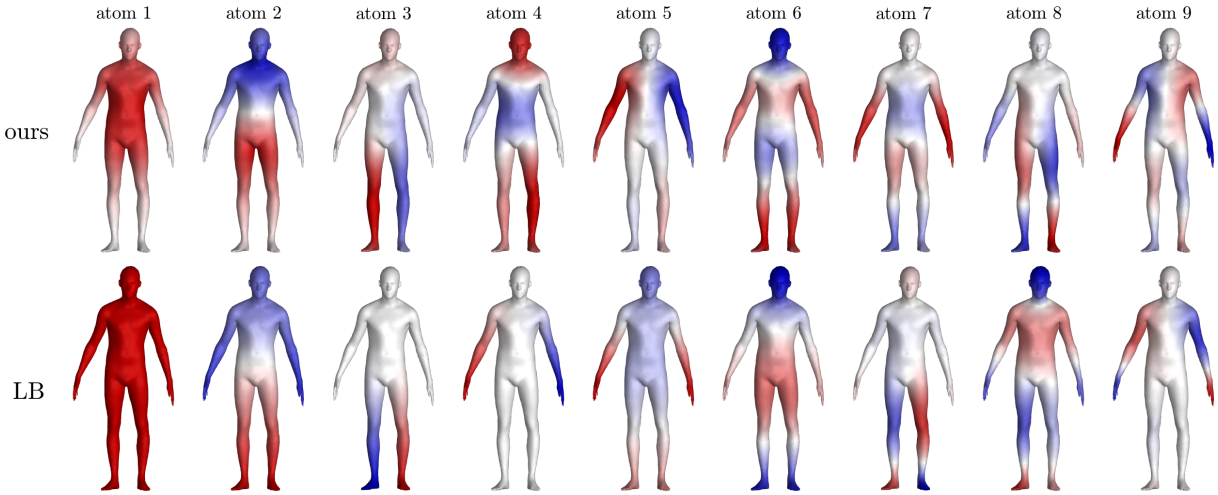


Figure 3.3: Visual representation of the atoms of PC-Gau (top) and LB (bottom). Function values are rendered through colors, positive values are red and negative values are blue.

We can see an example of the resulting basis atoms in the blue box in figure 3.1. Figure 3.3 presents a visual comparison between the first 9 atoms of PC-Gau and LB, for an example mesh.

3.2. Properties of PC-Gau

In this Section, we analyze some of the basic properties of PC-Gau, that follow directly from its construction. These properties make our basis compatible with functional map pipelines designed for LB, with no or little modification required. We will analyze the properties that differentiate PC-Gau from LB, making it more suited to obtain better point-wise maps, in Chapter 5.

3.2.1. Orthonormality

As we mentioned in Section 3.1.3, our basis is orthonormal according to the inner product of mesh $\mathcal{M}$:

$$P_k^T A_{\mathcal{M}} P_k = I_k$$

In general PCA produces a set of Principal Components that is orthonormal according to the usual scalar product between vectors. In order to enforce orthonormality according to the inner product of the mesh $\mathcal{M}$, we weighted the variables (the vertices) for the corresponding element of area. In our notation, these are the elements on the diagonal of $A_{\mathcal{M}}$.

The orthonormal property, common to LB, is useful because it makes the projection of a

function f on the basis particularly simple and efficient, as we saw in Section 1.2:

$$\hat{f} = P_k^T \cdot A_{\mathcal{M}} \cdot f \tag{3.3}$$

(3.3) is used also when converting a given point-wise map $\Pi : \mathcal{N} \to \mathcal{M}$, represented in matricial form, to a functional map C:

$$C = P_{\mathcal{N}}^T \cdot A_{\mathcal{N}} \cdot \Pi \cdot P_{\mathcal{M}} \tag{3.4}$$

$\Pi P_{\mathcal{M}}$ are the atoms of the basis of $\mathcal{M}$ transferred on $\mathcal{N}$ through the ground truth correspondence, which are then projected on the basis of $\mathcal{N}$.

3.2.2. Frequency ordering

Figure 3.4 shows the Dirichlet energy of each atom of PC-Gau, comparing ours to LB, computed on an example mesh. As we saw in Section 1.1.2, Dirichlet energy measures the smoothness of a function f and can be interpreted as the frequency of f. Note that LB is perfectly ordered in frequency because it is, by construction, the orthonormal basis with the lowest Dirichlet energy. We observe that, despite not perfect, our basis presents an approximate frequency ordering of the atoms. This means that the truncated PC-Gau provides a low-pass filter approximation of a function, similarly to LB (see Section 1.1.2). This fact, again, reinforces the choice to use the first k Principal Components as truncated basis, which was introduced in 3.1.3. Qualitatively, we can asses the frequency ordering, and also the similarity with LB, in Figure 3.3.

We also observe that the frequency of the atoms of PC-Gau rises much more rapidly than LB from the 30-th atom. This fact suggests that some higher-frequency information about the shape of the mesh is incorporated already in the first Principal Components. This claim needs further investigation, which is out of the scope of this Thesis, but also sheds the light on possible further application of PC-Gau, for instance in the approximation of coordinate-related signals (i.e. functions) on meshes. We will briefly present the results of some tests on signal approximation and transfer in Section 5.4.

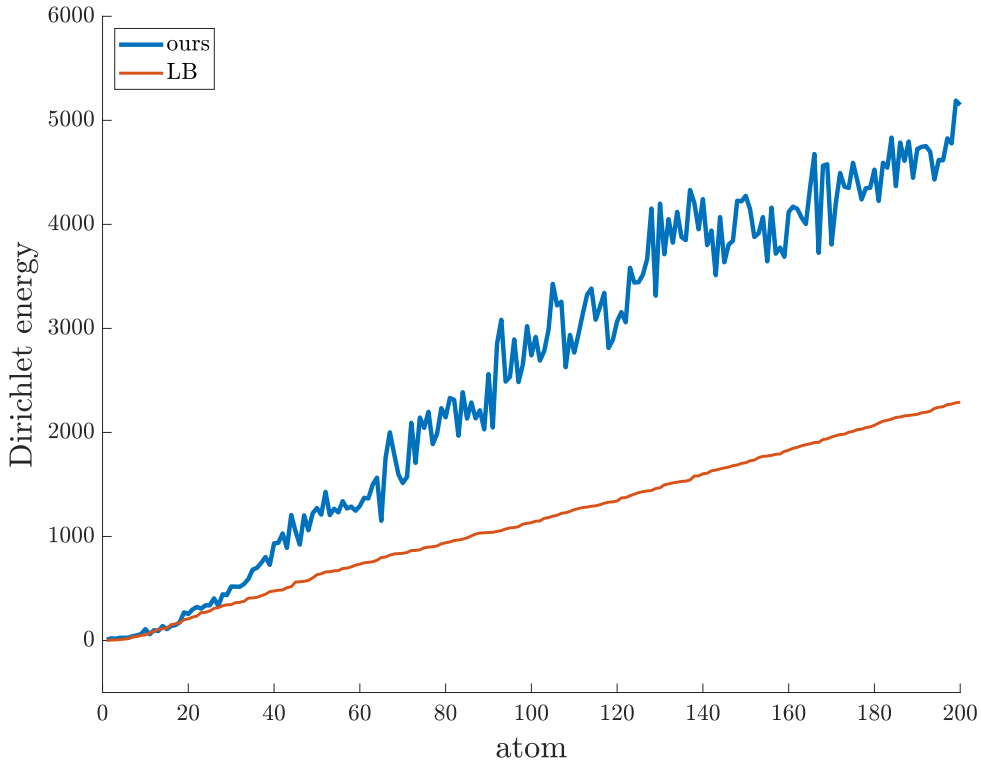


Figure 3.4: Values of the Dirichlet energy for each atom of PC-Gau and LB, computed on a sample mesh. PC-Gau is approximately ordered in increasing frequency and presents, as expected, Dirichlet energies higher than LB.

3.2.3. Isometry invariance

Isometries

Two meshes $\mathcal{M}$ and $\mathcal{N}$ are said to be isometric, or related by an isometry, if the underlying $T : \mathcal{N} \to \mathcal{M}$ preserves the geodesic distance between any couple of vertices:

$$\text{GeoDist}_{\mathcal{N}}(x, y) = \text{GeoDist}_{\mathcal{M}}(T(x), T(y)) \qquad \forall x, y \in V_{\mathcal{N}} \tag{3.5}$$

This concept, which should be Boolean in principle, has often been associated with a degree, starting from [23] itself. So, we can say that two meshes are near-isometric or related by a near-isometry if the distances in (3.5) are only approximately preserved. The two meshes become less and less isometric as the difference between such distances increases.

Note that discretization itself introduces some level of non-isometry. If we consider, for instance, a continuous manifold $\mathcal{X}$ discretized independently into two meshes $\mathcal{M}$ and $\mathcal{N}$, we have that, in general, $\mathcal{M}$ and $\mathcal{N}$ will not be perfectly isometric. We can intuitively see this, by considering that edge lengths are not preserved and so it is not preserved the

geodesic distance between neighbor vertices. As we consider couple of vertices that are more and more far apart, though, the relative effect of discretization on their geodesic distance will be more and more negligible, so we can still say that the meshes are near-isometric.

Note also that deformations of real objects are not, in general, perfect isometries. It is the case, for instance, of pose changes in a human body, because of the elasticity of human tissues. We can still consider them, though, as near-isometries with reasonable approximation.

In the following, we will often neglect non-isometries introduced by discretization and pose change and we will refer to such relations, for simplicity, as isometries.

Isometry invariance of our basis

Since our basis is constructed purely on geodesic distances, if geodesic distances are preserved between two meshes, so it is the resulting basis, except for possible sign swaps in the atoms. Isometry invariance implies that the functional map C between perfectly isometric meshes is a diagonal matrix with ± 1 elements on the diagonal. For near-isometric pairs this is not exactly true, but the energy of C is still heavily concentrated on the diagonal. For less-isometric pairs, the energy spreads away from the diagonal, especially for higher frequencies, making C funnel-shaped. This behaviour is similar to LB, and is due to the fact that lower frequencies contain more global information about the shapes, that is preserved also for mild non-isometries [23]. You can see a comparison with LB on a near-isometric pair and on a non-isometric pair in Figure 3.5.

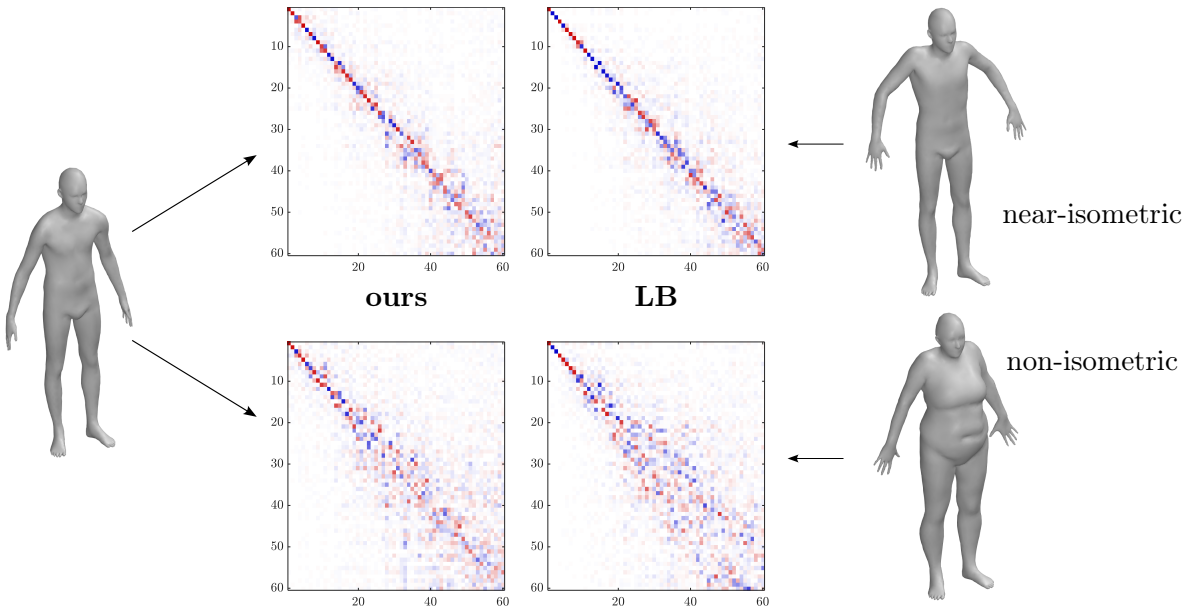


Figure 3.5: Examples of functional maps between a near-isometric and a non-isometric pair of meshes, both with PC-Gau and LB. Note the difference in the distribution of the energy in C: the matrices are nearly diagonal for near-isometries and funnel-shaped for non-isometries.

4 | Experimental Evaluation

In this chapter, we apply different functional map pipelines — using both PC-Gau and LB as functional basis — to pairs of meshes extracted from established datasets. Since the ultimate goal of PC-Gau is to improve the quality of shape matching, with respect to the currently widely used LB, the experimental evaluation of the accuracy of the obtained maps constitutes a crucial assessment for our basis. We will analyze the relation between these results and the specific properties of PC-Gau in Chapter 5, to give evidence of our claim that the improvement on quality of maps comes from the even distribution of the basis energy.

In this chapter, in particular, we compare the performance of our basis and LB in four different settings:

- In Section 4.4 we compute a functional map C from the ground-truth correspondence and then convert it to point-wise maps with the method shown in [23] and briefly explained in Section 2.2.2.
- In Section 4.5 we estimate a functional map C from an optimization problem, using [21], and then convert it to a point-wise map using again the method explained in Section 2.2.2.
- In Section 4.6 we compute a small functional map C_{ini} with [21] and then extend it using ZoomOut [16], which produces a point-wise map at each iteration.
- In Section 4.7 we use a slightly modified procedure for the conversion to point-wise maps, with respect to the one explained in Section 2.2.2, and we apply it both to ground-truth C and inside ZoomOut [16].

Let us recall, once again, that we are able to test PC-Gau and LB in the exact same setting because of their compatibility. This makes the results from the two bases comparable without further considerations.

In all the settings the metrics and the dataset considered are the same, providing comparability also between different settings. Metrics are described in Section 4.3 and datasets

in Section 4.1. In Section 4.2, we define the global experimental setting, with the common parameters used in evaluation, and we make a few remarks on the implementation used in tests.

4.1. Datasets

The experimental evaluation has been carried out on the following established datasets. We normalized all the meshes to unitary surface area, in order to make the error comparable between different datasets.

FAUST [4] is a dataset of 100 meshes, composed by 10 people in 10 poses each (the same across the meshes). Each mesh was independently remeshed to 5k vertices, in order to avoid to introduce implicit knowledge from the common connectivity structure. Unless otherwise specified, we randomly selected 200 pairs from this dataset. Due to remeshing, the ground-truth correspondence between meshes is only partial and represented as a (partial) relation $\tilde{T}$ (see Section 4.3.1).

MWG is a dataset of 24 meshes, composed by 5 meshes of a gorilla, 7 meshes of a man and 12 meshes of a woman. Each class has a different number of vertices. We denote with **MWG iso** the dataset with only man and woman meshes. We randomly selected 200 pairs from MWG and 120 pairs from MWG iso. The ground-truth correspondence is provided as a partial relation $\tilde{T}$, with $|\tilde{T}| = 900$.

TOSCA [5] is a dataset of human bodies and animals in different poses. We used 5 meshes from the same person, totaling 20 pairs, in our tests. The meshes are high resolution (∼50k vertices) and share the connectivity (vertices are in 1:1 correspondence), therefore, this dataset provides a complete ground-truth correspondence T between meshes.

SHREC19 [15] is a dataset composed by 44 meshes taken from different sources. Meshes in this dataset present wide variations in the discretization, both in the number of vertices and in the regularity of the tessellation. They are also present differences in the type of model: computer generated, elaborated from real scans, puppets, real bodies. All these facts provide additional challenges to shape matching and makes this a challenging dataset. We randomly selected 200 pairs for our tests. The ground-truth correspondence is provided as a partial relation $\tilde{T}$, with $|\tilde{T}| = 6890$.

4.2. Implementation

We implemented the procedure to build PC-Gau and the pipelines to perform the evaluation it in Matlab. The code for ZoomOut [16] and [21] is provided by the authors and available online. In Table 4.1, we recap the parameters used throughout all the tests, when not specified otherwise. Refer to Section 3.1 for a detailed explanation of their meaning. In Section 5.2 we present a brief analysis on the selection of their values.

parameter	value used
q:	1000
Q sampling method:	FPS (Euclidean distance)
σ:	0.05
normalization:	true
k:	60

Table 4.1: Parameters used in experiments

Computation time

Before presenting the core evaluation, we remark that the computation times of PC-Gau and LB are comparable. We do not consider computation time as one of the main parameters of evaluation, but, since we are claiming that our basis is a suitable replacement for LB, it is important from a practical perspective that the two bases are computable in similar times. Table 4.2 presents the average computation time of PC-Gau and LB on two datasets with very different mesh resolutions. To compute PC-Gau, we explicitly stored the mesh graph for each mesh and then used the internal function of Matlab to compute distances on a graph. For LB, we used the cotangent weight scheme [25], as explained in Section 1.2.1. Note that, in our implementation, the computation time of PC-Gau is independent of the number of atoms produced.

Note also that the computation of PC-Gau does not require to compute the distances between any pair of vertices on the mesh, but it only needs the distances from any vertex to the vertices in the subset Q. Since the size q of Q is independent of the mesh size n, the memory requirement is linear in the number of vertices n, and not quadratic.

dataset	ours [s]	LB(60 atoms) [s]	LB(200 atoms) [s]
FAUST (remeshed at 5k vertices)	1,06	0,29	1,17
TOSCA (∼50k vertices)	8,58	3,05	10,01

Table 4.2: Computation times for PC-GAU and LB. For commonly used number of atoms ($60 \sim 200$), times are comparable. The computation time of our basis is independent of the number of atoms.

4.3. Evaluation metrics

In this section, we present the metrics adopted in the following sections to asses the quality of point-wise maps.

4.3.1. Geodesic Error

The geodesic error, as briefly mentioned in Section 2.1, is the main metric to asses the *accuracy* of a point-wise map. Let us consider two meshes $\mathcal{M}$ and $\mathcal{N}$ and a point-wise map $\bar{T} : \mathcal{N} \to \mathcal{M}$. $\bar{T}$ associates to each vertex of $\mathcal{N}$ a vertex on $\mathcal{M}$. We also assume to be provided with a ground-truth point-wise map $T : \mathcal{N} \to \mathcal{M}$. For the moment, we assume T to be a complete function, then we will relax this constraint. We define the Geodesic Error $e(x)$ for each vertex $x \in V_{\mathcal{N}}$ as the distance between the image point of x on $\mathcal{M}$ through $\bar{T}$ and the ground-truth image point provided by T:

$$e(x) = \text{GeoDist}_{\mathcal{M}}\left(\bar{T}(x), T(x)\right) \qquad \forall x \in V_{\mathcal{N}} \tag{4.1}$$

For some datasets, the provided ground-truth correspondence is not a complete function but a relation $\tilde{T} \subset V_{\mathcal{N}} \times V_{\mathcal{M}}$. This means that $\tilde{T}$ may not be defined for each vertex of $\mathcal{N}$ and also that $\tilde{T}$ may associate two or more different vertices on $\mathcal{M}$ to the same vertex $x \in V_{\mathcal{N}}$. We can easily extend equation 4.1 to this case, simply by assigning an index i to each pair of vertices in $\tilde{T}$ and associating the error to the each.

$$e(i) = \text{GeoDist}_{\mathcal{M}}\left(\bar{T}(x), y\right) \qquad \forall (x, y) \in \tilde{T}$$

The resulting $e = [e(i)]$ is no longer a function defined on the mesh, but it is a generic vector of geodesic errors.

Average Geodesic Error

In order to have a unique value to assess the global quality of the point-wise map, we can compute the average on the values of $e(i)$ and obtain the Average Geodesic Error (or AGE):

$$\mathrm{AGE}(\bar{T}) = \operatorname*{Avg}_{i} \{e(i)\} \tag{4.2}$$

This works both in the case the ground truth correspondence is a total function T, in that case $i \in V_{\mathcal{N}}$, and in the case it is a relation $\tilde{T}$, in that case $i \in \{1 \dots |\tilde{T}|\}$. When evaluating multiple pairs taken from a dataset, we can simply consider the mean of the AGE of the single pairs.

Mean Relative Error When we are comparing two point-wise maps T_1 and T_2, in addition to the absolute value of $\mathrm{AGE}(T_1)$ and $\mathrm{AGE}(T_2)$, we can compute the relative error of T_1 with respect to T_2:

$$\mathrm{RE} = \frac{\mathrm{AGE}(T_1) - \mathrm{AGE}(T_2)}{\mathrm{AGE}(T_2)}$$

In our tests, in particular, we are interested in assessing the relative error of a point-wise map computed with PC-Gau with respect to point-wise maps computed with LB, for the same setting. When evaluating multiple pairs, we can just compute the mean of the relative error of each pair of meshes. We call this quantity Mean Relative Error, which, in our case, reads:

$$\mathrm{MRE} = \operatorname*{Avg}_{p \in \mathrm{pairs}} \frac{\mathrm{AGE}(T_{p,\mathrm{ours}}) - \mathrm{AGE}(T_{p,\mathrm{LB}})}{\mathrm{AGE}(T_{p,\mathrm{LB}})} \tag{4.3}$$

A negative value of MRE means that, on average, PC-Gau is performing better than LB in terms of Average Geodesic Error. Note that, in general, the MRE does not coincide with computing the relative difference, between PC-Gau and LB, of the mean AGE on the dataset.

Cumulative Geodesic Error

AGE is good for assessing the global quality of a point-wise map with a unique value. If we want more information on the composition of the error, though, we can consider the curve of the Cumulative Geodesic Error. This metric associates to any given error threshold t the percentage of correspondences d with a geodesic error lower that t. By varying t and plotting d, we obtain an increasing function that shows the percentage of correct correspondences for a given tolerance. In this case a higher value of the curve is

better. In the ideal case, it should coincide with a step function in zero. You can find an example of such plots in Figure 4.2.

Error localization

Instead of computing the average of the geodesic error on all the vertices of the mesh, as we did to compute the AGE in (4.2), we can represent the error in (4.1) as a function on the mesh. In this way we can asses the *localization*, or *spatial distribution* of the error, which in turn means to visually evaluate which are the areas where the error is higher.

If we are considering multiple pairs from a dataset, we can visualize on a mesh $\mathcal{Y}$ the geodesic error averaged on all the pairs. We just need to align the error functions using the ground truth correspondence. For instance, given a pair $(\mathcal{M},\mathcal{N})$ with ground-truth map $T_1 : \mathcal{N} \to \mathcal{M}$, and the ground-truth map $T_2 : \mathcal{Y} \to \mathcal{N}$, we can associate to each vertex x of $\mathcal{Y}$ the error referred to the map $\bar{T} : \mathcal{N} \to \mathcal{M}$ as:

$$e(x) = \text{GeoDist}_{\mathcal{M}}\left(\bar{T}(T_2(x)), T_1(T_2(x))\right) \qquad \forall x \in V_{\mathcal{Y}} \tag{4.4}$$

In this way, by considering a sufficiently large number of pairs, we can assess if the error presents a systematic spatial distribution pattern and, if yes, where the error is localized.

If the ground-truth correspondence is provided as a relation, we can still align errors across different pairs. We just need to make sure that T_1 is defined for some image value of T_2. In datasets, when the correspondence is not a total function, it is usually provided as a relation for the same subset of vertices on all the meshes, therefore the previous condition is met for all the vertices in the relation. Thus errors can be easily aligned on different meshes. Note that (4.4) does not provide a complete function in this case. In order to visualize it on a mesh, we interpolate values to fill in the holes.

4.3.2. Distortion

Smoothness Another desirable quality of a point-wise map $\bar{T}$, not necessarily related to the global value of accuracy, is smoothness. For instance, smoothness is particularly desirable in texture transfer and deformation transfer, because it ensures that there are no gaps or bumps in the transferred texture or deformation. From an intuitive point of view, smoothness means that vertices that are close to each other, are mapped by $\bar{T}$ to vertices that are still close to each other.
[16] proposes a metric to assess the smoothness of a point-wise map $\bar{T} : \mathcal{N} \to \mathcal{M}$. It consists in transferring the coordinate functions X,Y,Z from $\mathcal{M}$ to $\mathcal{N}$ with $\bar{T}$ and computing

the mean of their Dirichlet energies on $\mathcal{N}$. [26] proposes a metric for smoothness, too: for each edge on $\mathcal{N}$ it evaluates the ratio between the geodesic distance of the mapped endpoints on $\mathcal{M}$ and the length of the edge itself. These metrics are usually called distortion and are inversely related to smoothness: a low distortion means a better smoothness.

Coverage Both of the metrics presented above would assign zero distortion to a trivial map $\bar{T} : \mathcal{N} \to \mathcal{M}$ that maps each vertex of $\mathcal{N}$ to the same vertex on $\mathcal{M}$. This is, of course, not a desirable behavior. To fix this behaviour, these smoothness metrics need to be considered in conjunction with a coverage metric, which is usually assessed [26] as the ratio between the area on $\mathcal{M}$ covered by $\bar{T}$ and the total surface of $\mathcal{M}$. Where by area covered by $\bar{T}$ on $\mathcal{M}$ we mean the sum of the area elements associated to vertices on $\mathcal{M}$ which are image, via $\bar{T}$, of at least one vertex on $\mathcal{N}$.

Our distortion metric In order to consider a unique metric, we propose to use a distortion metric that assess smoothness while penalizing for bad coverage. It is a simple modification of the metric presented in [26] and consists in assessing, for each edge on $\mathcal{N}$, the absolute difference between its length and the geodesic distance between the images of its endpoints. By using the absolute difference, instead of the ratio, we are penalizing the case in which two vertices $x \in \mathcal{N}$ and $y \in \mathcal{N}$ are mapped on the same vertex $z \in \mathcal{M}$. In this case, indeed, the edge (x, y) would get a penalty equal its own length. This metric can still be defined as a distortion, therefore the lower its value, the smoother the map and the better.

4.4. I setting: C computed from ground-truth

As experimental setting, we start by considering the case in which the functional map C between $\mathcal{M}$ and $\mathcal{N}$ is computed from a provided ground-truth correspondence $T : \mathcal{N} \to \mathcal{M}$. In the following, we will refer to a C computed in this way as *ground-truth* C for brevity. This is, of course, a pure theoretical setting, as we are never provided with a ground-truth correspondence in real applications. Despite its ideal connotation, we focus on this setting because it has the great advantage of making the evaluation independent of the technique chosen to estimate C. This experimental setting is equivalent to assessing the quality of the point-wise map under the assumption that we are able to estimate the best possible functional map C between two given basis $\Phi_{\mathcal{M}}$ and $\Phi_{\mathcal{N}}$.
The obtained point-wise map, though, still depends on the algorithm used to convert C into a point-wise map. We chose to use the original algorithm, proposed in [23] and explained in Section 2.2.2, both for its simplicity and for its wide use in real applications.

4.4.1. Technicalities

If a dataset provides a complete ground-truth map T between each pair of meshes, the optimal functional map C can be computed, for both PC-Gau and LB, using Equation (3.4). If, instead, the ground-truth correspondence is provided as an incomplete relation $\tilde{T}$, we cannot transfer function using $\tilde{T}$ and thus (3.4) cannot be applied. In this case we can still recover a functional map C from $\tilde{T}$. Given two bases Φ and Ψ, for $\mathcal{M}$ and $\mathcal{N}$ respectively, we proceed as following:

1. we assign an index i to each couple $(x, y) \in \tilde{T} \subset (V_{\mathcal{N}} \times V_{\mathcal{M}})$. We refer to the i-th couple in $\tilde{T}$ as $\tilde{T}_i$
2. For each i, we insert the x-th row of Φ as the i-th row of $\tilde{\Phi}$ and the y-th row of Ψ as the i-th row of Ψ, where $(x, y) = \tilde{T}_i$. The rows of $\tilde{\Phi}$ and $\tilde{\Psi}$ are now in correspondence according to $\tilde{T}$.
3. we compute $C = \tilde{\Psi}^{\dagger} \cdot \tilde{\Phi}$, where $\dagger$ denotes the Moore-Penrose pseudoinverse.

4.4.2. Results

Table 4.3 presents the experimental results on the accuracy of the point-wise maps obtained. In this setting PC-Gau outperforms LB, consistently providing lower errors. Note that, among the 620 pairs considered across 5 datasets, PC-Gau is better in 97% of the cases.

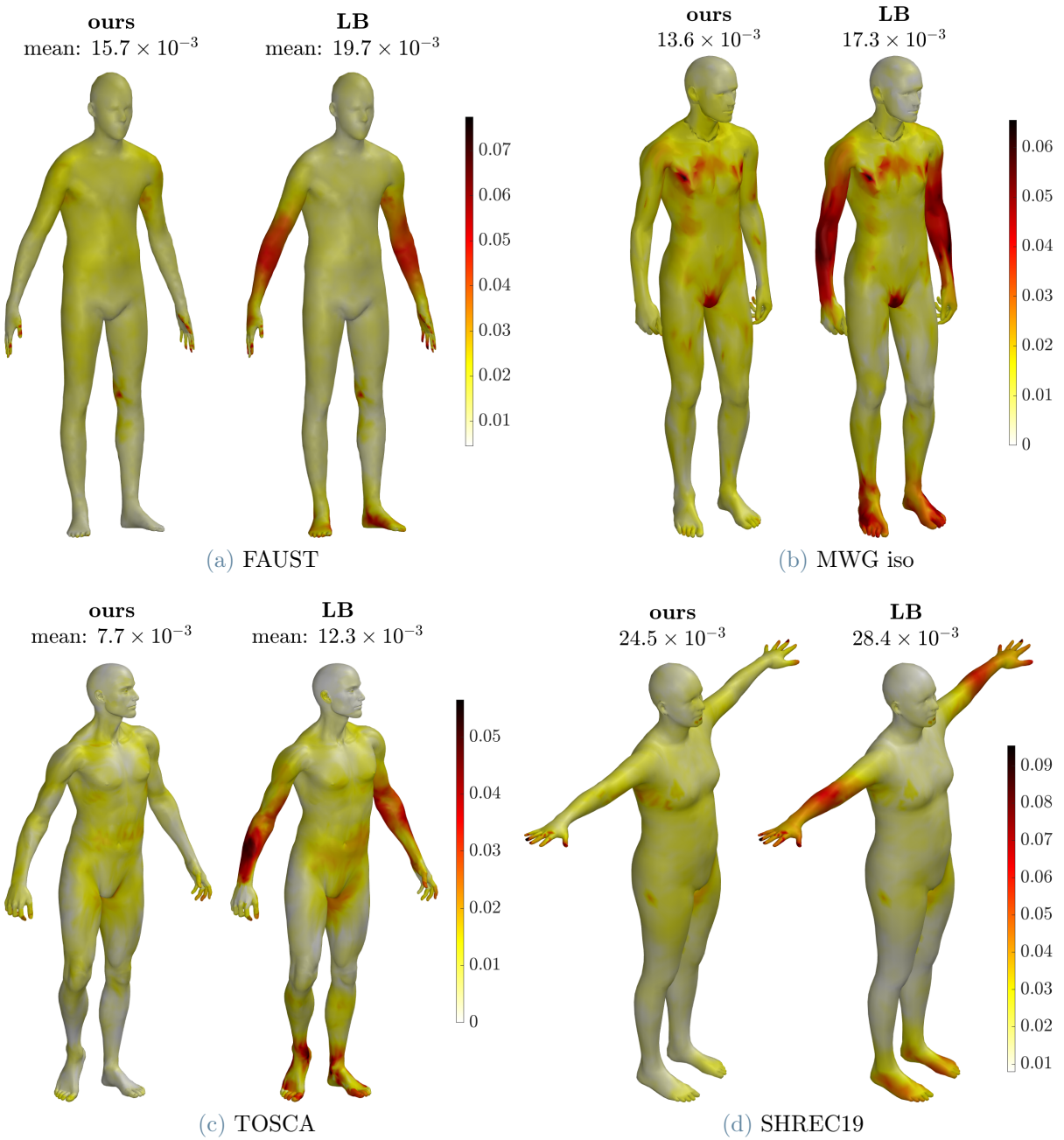
From the curves in Figure 4.2, we observe that PC-Gau and LB perform similarly regarding the correspondences with a low geodesic error. For thresholds below 0.015, the percentage of correspondences is very similar in most of the datasets. LB, instead, presents more vertices with a larger error. We see this from the fact that PC-Gau approaches 100% much faster than LB after the 0.02 threshold. This means that, while a portion of the vertices is correctly matched using both bases, there is a set of vertices which is badly matched using LB.

This analysis is compatible with the spatial distribution of the error shown in Figure 4.1. From there, we observe that the error of LB is localized in arms, particularly forearms, and feet. The error of PC-Gau, on the contrary, presents minimal variations across the mesh surface. The spatial distribution confirms what we observed in Figure 4.2: LB is not slightly worse than PC-Gau on all the mesh, uniformly. There are areas in which the erorr of LB is (much) higher than PC-Gau, arms and feet, and areas in which it is similar or even slightly lower. We will put explicitly in relation the distribution of the error and the distribution of basis energy, both for LB and PC-Gau, in Section 5.1.

For the distortion of the point-wise maps, we refer to the first two columns of Table 4.7. We used the metric presented in Section 4.3.2 to simultaneously assess smoothness and coverage of maps. In this setting, our basis and LB perform similarly in terms of distortion.

dataset	ours ($\times 10^{-3}$)	LB ($\times 10^{-3}$)	MRE ($\times 10^{-2}$)
FAUST	**15,7**	19,7	**-20,3**
MWG	**20,8**	24,9	**-20,2**
MWG iso	**13,6**	17,3	**-25,5**
TOSCA	**7,7**	12,3	**-39,6**
SHREC19	**24,5**	28,4	**-13,9**

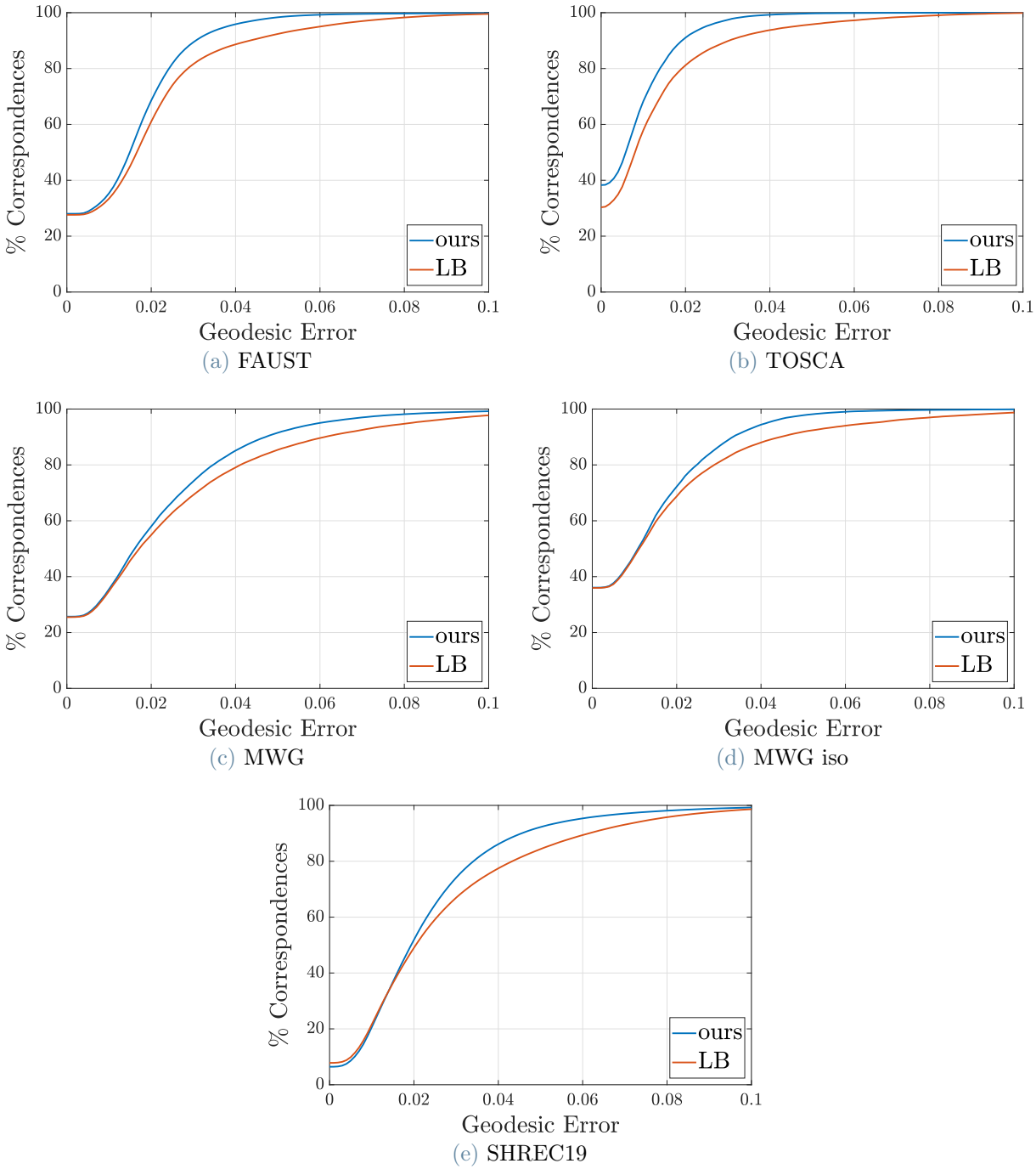
Table 4.3: Average Geodesic Error and Mean Relative Error of point-wise maps computed from ground-truth C. In this setting, PC-Gau outperforms LB in all the datasets.



(a) FAUST (b) MWG iso

(c) TOSCA (d) SHREC19

Figure 4.1: Spatial distribution of geodesic error of maps obtained from ground-truth C. Average on multiple pairs from different datasets. The error of LB is localized in forearms and feet. PC-Gau, instead, presents minimal variations across the mesh surface.



(a) FAUST (b) TOSCA

(c) MWG (d) MWG iso

(e) SHREC19

Figure 4.2: Cumulative Geodesic Error of point-wise maps converted from ground-truth C, for different datasets. Using LB, there is a considerable subset of vertices getting a large geodesic error.

4.5. II setting: C estimated with product preservation

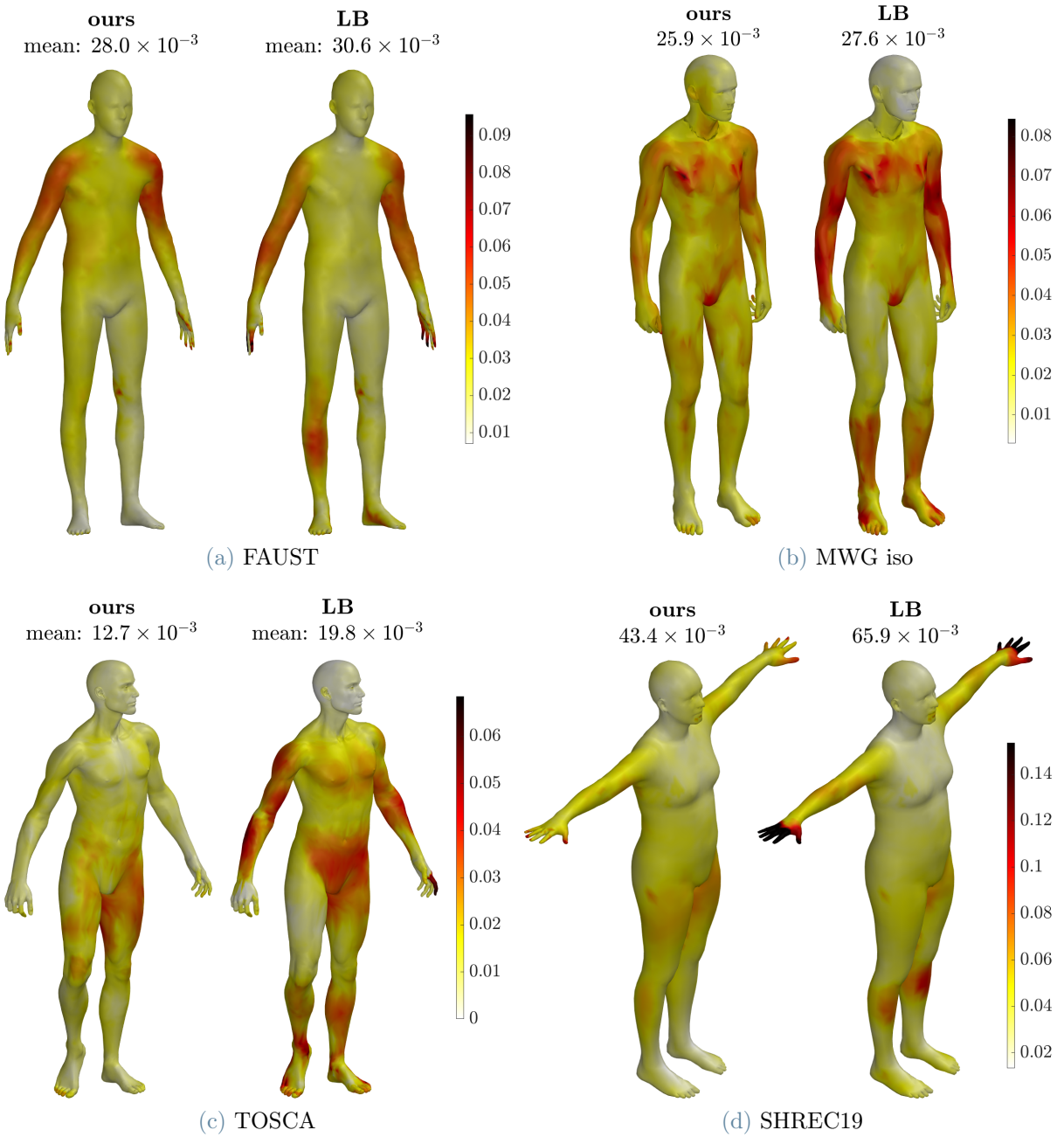
In real applications we cannot compute C from the ground truth correspondence, as we did in Section 4.4, but we need to define some constraints on C and solve the resulting optimization problem, as described in Section 2.2.1. In this section we test PC-Gau and LB in combination with the method proposed in [21]. This method uses the preservation of point-wise products of functional constraints to extract more information from the set of functional constraints available and thus set a better-conditioned optimization problem. In our implementation we used functional constraints based on 6 landmarks (head, chest, hands, feet) and WKS [3] as descriptor. For the conversion to point-wise maps we still use the original method proposed in [23] and presented in Section 2.2.2.

4.5.1. Results

Table 4.4 shows the Average Geodesic Errors of the point-wise maps obtained using PC-Gau and LB in this setting, and the Mean Relative Error of PC-Gau with respect to LB. We observe that, even though to a lesser degree, PC-Gau still performs better than LB in almost all datasets. In MWG they have an equivalent accuracy, on average. Note that the fact that MRE is anyway negative for MWG, indicates that PC-Gau performs better on nearly-isometric pairs, for which the error is low, and worse on strongly non-isometric pairs. This is confirmed by the fact that, by removing gorilla meshes from the dataset (see MWG iso row in Table 4.4), the advantage of PC-Gau is back.

Figure 4.4 shows the curves of the cumulative geodesic error. In this setting, the margin of PC-Gau is thinner, in general, with the only exception of TOSCA. Note, however, that meshes in this dataset present a 1:1 correspondence of vertices and share the connectivity: this fact is probably implicitly advantaging PC-Gau.

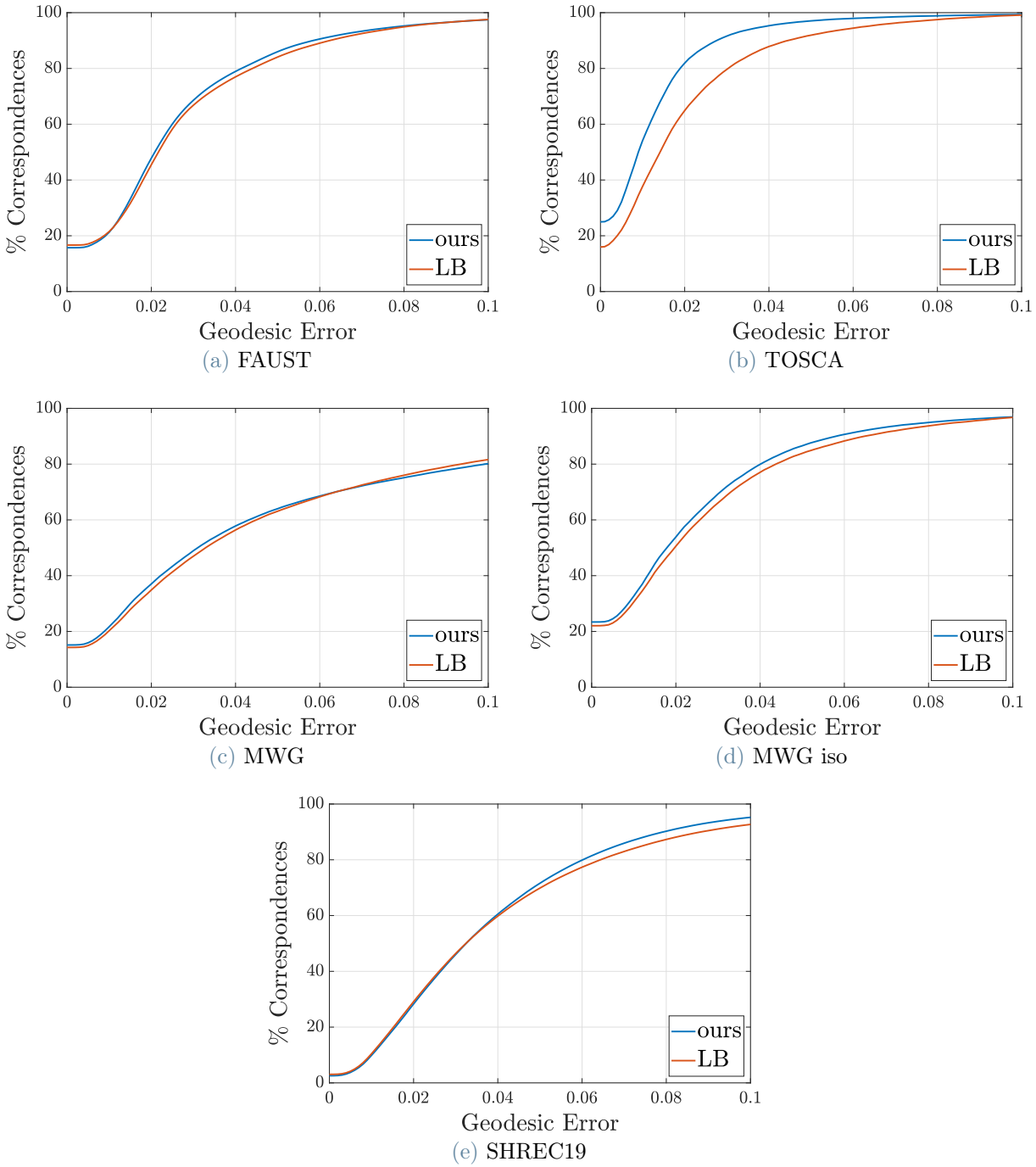
Figure 4.3 shows the spatial distribution of the error, averaged on the considered pairs from each dataset. In this setting, the estimation of C introduces additional noise to the final point-wise map. However, we can still observe that LB almost always presents a high error on forearms, lower legs and feet. In addition, some error is present in other areas, such as hip and fingers, probably due to errors in the estimation of C. The error distribution of PC-Gau is affected by the estimation of C as well. However, the error distribution of PC-Gau is still more uniform and does not show particular areas with systematic errors across different datasets.



(a) FAUST (b) MWG iso

(c) TOSCA (d) SHREC19

Figure 4.3: Spatial distribution of geodesic error of maps obtained estimating C with product preservation [21]. Average on multiple pairs from different datasets. In this setting LB still presents areas with systematically high error values, such as forearms, lower legs and feet. The error of PC-Gau is, globally, more evenly distributed.



(a) FAUST (b) TOSCA

(c) MWG (d) MWG iso

(e) SHREC19

Figure 4.4: Cumulative Geodesic Error of point-wise maps converted from C estimated with product preservation [21], for different datasets. The advantage of PC-Gau is lower in this setting, but still present.

dataset	ours ($\times 10^{-3}$)	LB ($\times 10^{-3}$)	MRE ($\times 10^{-2}$)
FAUST	**28,0**	30,6	**-4,9**
MWG	**58,7**	**58,7**	**-8,4**
MWG iso	**25,9**	27,6	**-10,1**
TOSCA	**12,7**	19,8	**-39,3**
SHREC19	**56,2**	78,5	**-14,8**

Table 4.4: Average Geodesic Error and Mean Relative Error for point-wise maps computed from a C estimated with product preservation [21].

4.6. III setting: C estimated with ZoomOut

In this section we evaluate our basis when used in the iterative method presented in [16]. This method, as described in Section 2.3, relies heavily on conversion to point-wise maps. For this reason, we expect that PC-Gau will perform particularly well compared to LB. For the initial map we used a map C_{ini} estimated with [21] and the same functional constraints as in Section 4.5. The size of C_{ini} varies in different datasets between 16×16 and 20×20. The size of the final C, though, is always 60×60 as in previous settings.

4.6.1. Results

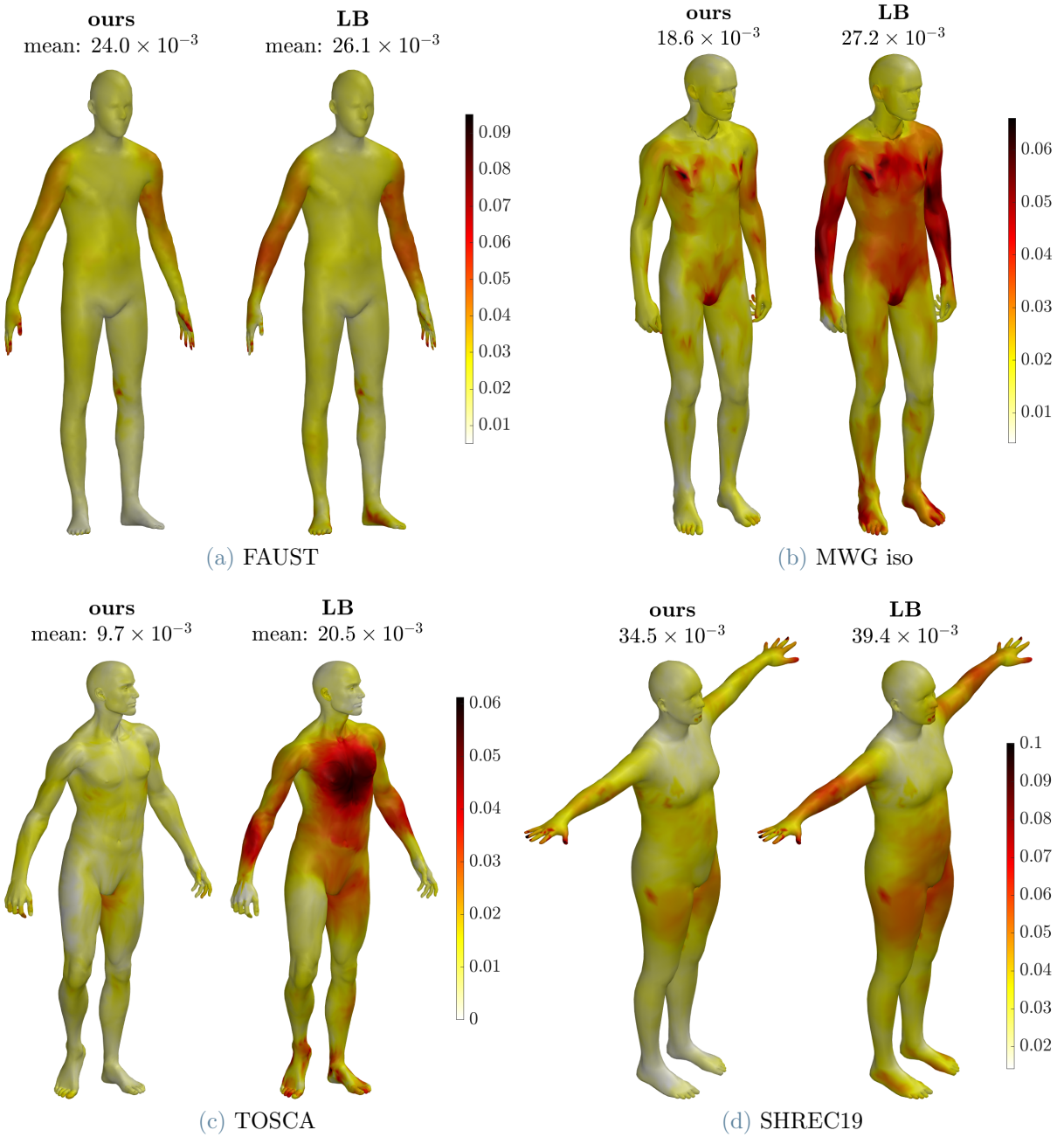
Table 4.5 shows the Average Geodesic Errors of the point-wise maps obtained using PC-Gau and LB in this setting, and the Mean Relative Error of PC-Gau with respect to LB. In this setting PC-Gau performs substantially better than LB in all datasets. More interestingly, by comparing Table 4.5 and Table 4.4, we observe that the use of ZoomOut in combination with PC-Gau brings benefit to the point-wise accuracy for all the datasets. This is not always true for LB, which is more susceptible to instability in the estimation of the small initial C_{ini}. This observation supports our claim that PC-Gau is not only compatible with pipelines designed for LB, but it is complementary to other approaches aimed at improving shape matching. As we can see from the results in this setting, the benefits from ZoomOut and PC-Gau adds up in the final accuracy.

From Figure 4.5 we observe that the maps obtained with LB presents systematical errors on forearms and feet. The error of the maps obtained with PC-Gau, instead, is distributed evenly on the mesh surface, with minimal variations, similarly to the results in Section 4.4. The high-error region on the chest for MWG iso and TOSCA, for maps obtained with LB, is due to front/back swaps in the initial map. Our basis is clearly less sensible to such problems.

In the first two columns of Table 4.9 we show the distortion of the obtained maps. Note that, in this setting, our basis performs similarly or better than LB in terms of smoothness and coverage.

dataset	ours ($\times 10^{-3}$)	LB ($\times 10^{-3}$)	MRE ($\times 10^{-2}$)
FAUST	**24,0**	26,1	**-7,5**
MWG	**51,2**	70,6	**-26,9**
MWG iso	**18,6**	27,2	**-28,3**
TOSCA	**9,7**	20,5	**-49,9**
SHREC19	**34,5**	39,4	**-10,3**

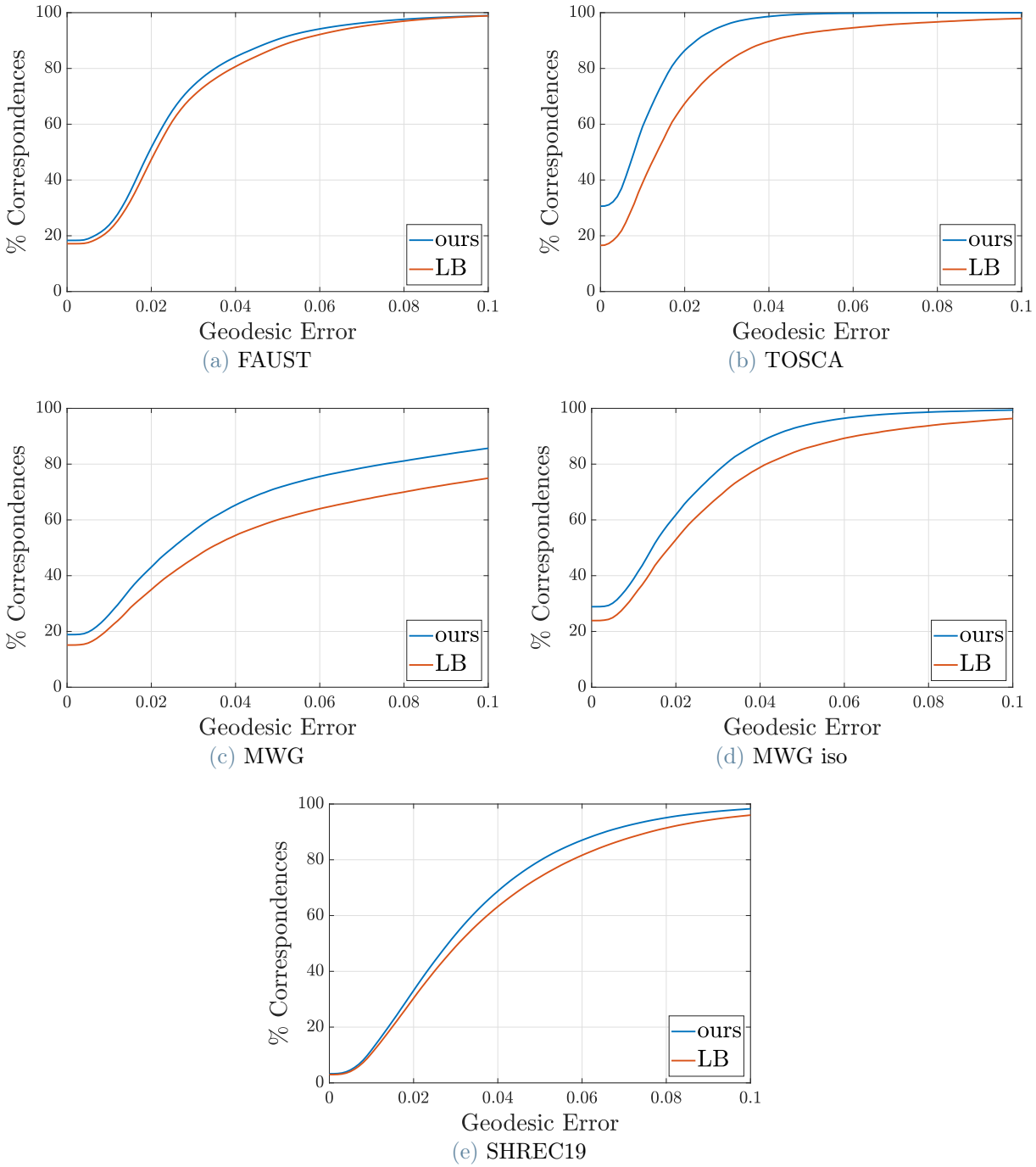
Table 4.5: Average Geodesic Error and Mean Relative Error of point-wise maps computed with ZoomOut [16]. In this setting, PC-Gau performs better than LB.



(a) FAUST (b) MWG iso

(c) TOSCA (d) SHREC19

Figure 4.5: Spatial distribution of geodesic error of maps obtained with ZoomOut [16]. Average on multiple pairs from different datasets. LB presents presents a systematic error on forearms and feet. The error of PC-Gau is evenly distributed. The high error on the chest for LB in (b) and (c) is due to front/back swaps in the initial map



(a) FAUST (b) TOSCA

(c) MWG (d) MWG iso

(e) SHREC19

Figure 4.6: Cumulative Geodesic Error of point-wise maps obtained with ZoomOut [16]. PC-Gau has a sensible margin on LB, especially when considering a high error thresholds. This means that a considerable amount of vertices is badly matched by LB.

4.7. IV setting: Point-wise conversion with generalized embeddings

In Section 2.2.2 we presented a simple method, taken from [23], to convert a functional map C form $\mathcal{M}$ to $\mathcal{N}$ to a point-wise map $T : \mathcal{N} \to \mathcal{M}$. We now consider a small modification of this method, which has already been adopted in previous works like [22], and we test it applied to our basis and LB. This modification consists in using smooth functions, instead of delta functions, to compute the embedding of vertices. Let us explain it in more detail.
In Section 2.2.2 we defined the embedding of the vertex x, given a basis $\Phi = \{\phi_i\} \in \mathbb{R}^{n \times k}$, as the vector of values assumed by basis functions in x: $\text{Emb}(x) = [\phi_i(x)] \in \mathbb{R}^k$. $\text{Emb}(x)$ corresponds to the coefficients of a delta function centered in the vertex x [23]. Similarly to what has been done in [22], we extend the concept of embedding of vertex x to the coefficients, in the truncated basis Φ, of a generic function γ_x centered in x. We denote this embeddings as *generalized embeddings*.

Embedding functions

The class of the embedding functions γ can be arbitrarily chosen. It is sufficient that γ_x is a function localized in the neighborhood of a vertex x and centered in x. For our tests we considered three classes of embedding functions:

Gaussian: Gaussian functions of the geodesic distance from vertex x. They are built in the same way as described in Section 3.1.2.

Sparse Frame (SF): approximation of Gaussian functions, computed as shown in [13], except for the binarization.

Heat Kernels (HK): approximation of heat diffusion from vertex x. They are computed using the eigenfunctions of the Laplace-Beltrami operator, as shown in [31].

All these classes of embedding functions require a parameter that sets the amplitude of the functions. We chose the values of such parameters in order to produce functions with similar Dirichlet energies between the different classes. Figure 4.7 shows an example for each class, centered in a vertex on the chest.

Computation

While for delta-embeddings we can get the embedding of each vertex simply by transposing the basis matrix Φ, in order to compute generalized embeddings we need an extra step. Let

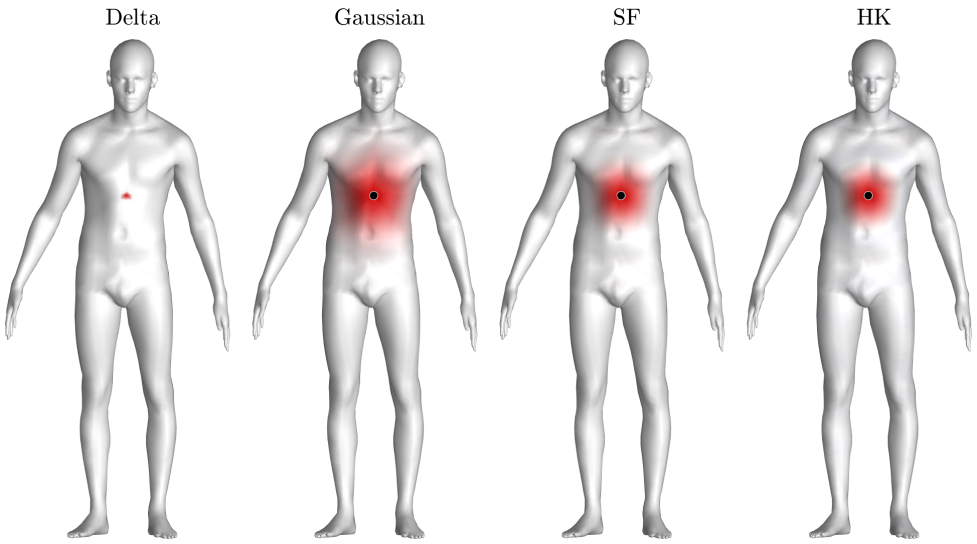


Figure 4.7: Example of different classes of embedding functions. The γ_x corresponding to vertex x is centered and localized in x, which is highlighted as a black circle in the example.

us consider a mesh $\mathcal{M}$ with n vertices. Let us suppose that $\Gamma_\mathcal{M}$ contains the embedding functions as columns, one centered in each vertex, so $\Gamma_\mathcal{M}$ has size $n \times n$. We can project them on an orthogonal basis $\Phi_\mathcal{M} \in \mathbb{R}^{n \times k}$ by matrix multiplication (see Section 3.2.1):

$$\Sigma_\mathcal{M} = \Phi_\mathcal{M}^T \cdot A_\mathcal{M} \cdot \Gamma_\mathcal{M}$$

$\Sigma_\mathcal{M}$ contains the coefficient vectors as columns, which are also the embeddings of each vertex of $\mathcal{M}$. We can do similarly for $\mathcal{N}$ and obtain $\Sigma_\mathcal{N}$. Then, the procedure is identical to delta-embeddings: for each column of $\Sigma_\mathcal{N}$ we find the nearest neighbour in $C\Sigma_\mathcal{M}$.

Despite the extra computation required, which can be costly for large meshes, generalized embeddings provide some benefits to the point-wise conversion and allow to obtain better point-wise maps for the same C. Such maps generally have a lower Average Geodesic Error and a lower distortion.

4.7.1. Ground-truth C

We applied this modified conversion with generalized embeddings to functional maps computed with the provided ground-truth correspondence. We start with this setting because generalized embeddings only affect the conversion to point-wise map, therefore we want to evaluate the benefit on the best possible C.

Table 4.6 shows the Average Geodesic Error of the obtained point-wise maps. We observe that Gaussians and Heat Kernels bring a sensible benefit to the conversion. More importantly, we observe that this benefit is added up to the benefit brought by using our basis.

This confirms the claim that our basis is complementary to other approaches, such this enhanced conversion, in improving the final point-wise maps.

Table 4.7 shows the distortion of the same point-wise maps. Here we can observe that the benefit of using generalized embeddings and our basis is more than additional, as the benefit on smoothness is much more evident with PC-Gau than with LB. Let us consider Gaussian functions, for instance. On average, we have an improvement on distortion of 23% on the different datasets with ours basis. With LB, the average improvement is only 6%.

4.7.2. ZoomOut

We applied generalized embeddings also to ZoomOut [16]. Note that in ZoomOut the conversion to point-wise map is performed at each iteration (see Section 2.3) and we used generalized embeddings for all these conversions. We used only Gaussian functions, since they provided good and stable results with ground-truth C.

Tables 4.8 and 4.9 present the average geodesic error and the distortion in this setting. They confirm what we observed before:

- The benefit provided by generalized embeddings adds up to the benefit given by our basis, keeping a substantial advantage of the accuracy of PC-Gau on all the datasets.
- Generalized embeddings greatly improve the smoothness and coverage of point-wise maps obtained with our basis, much more than they do with LB.

dataset	Delta		Gaussian		SF		HK	
	ours $\times 10^{-3}$	**LB** $\times 10^{-3}$	**ours** $\times 10^{-3}$	**LB** $\times 10^{-3}$	**ours** $\times 10^{-3}$	**LB** $\times 10^{-3}$	**ours** $\times 10^{-3}$	**LB** $\times 10^{-3}$
FAUST	**15,7**	19,7	**14,1**	19,0	**14,4**	18,6	**14,3**	19,4
MWG	**20,8**	24,9	**20,1**	24,7	**19,4**	23,8	**18,7**	23,8
MWG iso	**13,6**	17,3	**12,4**	16,4	**12,1**	16,5	**11,9**	16,7
TOSCA	**7,7**	12,3	**6,3**	8,7	**6,6**	10,3	**6,9**	11,0
SHREC19	**24,5**	28,4	**22,1**	27,5	**38,8**	39,6	**22,9**	28,0

Table 4.6: Average Geodesic Error of point-wise maps obtained using different functions for the embedding of vertices, converted from ground-truth C. Using smooth functions provides better results in most of the cases. PC-Gau always performs better than LB for the same embedding function.

dataset	Delta		Gaussian		SF		HK	
	ours $\times 10^{-3}$	**LB** $\times 10^{-3}$	**ours** $\times 10^{-3}$	**LB** $\times 10^{-3}$	**ours** $\times 10^{-3}$	**LB** $\times 10^{-3}$	**ours** $\times 10^{-3}$	**LB** $\times 10^{-3}$
FAUST	**7,3**	8,4	**5,7**	8,1	**6,3**	8,5	**6,0**	8,4
MWG	**9,9**	11,1	**8,3**	10,6	**9,1**	11,4	**7,7**	10,6
MWG iso	**8,1**	9,7	**6,4**	8,9	**7,0**	9,7	**6,4**	9,5
TOSCA	**3,3**	3,7	**2,1**	3,3	**2,1**	3,4	**2,1**	3,6
SHREC19	7,8	**7,5**	**6,2**	7,4	**6,3**	8,5	**5,4**	7,2

Table 4.7: Distortion of point-wise maps obtained using different functions for the embedding of vertices, converted from ground-truth C. Using smooth functions in combination with our basis greatly improves smoothness and coverage, increasing the advantage over LB.

dataset	Delta		Gaussian	
	ours $\times 10^{-3}$	**LB** $\times 10^{-3}$	**ours** $\times 10^{-3}$	**LB** $\times 10^{-3}$
FAUST	**24,0**	26,1	**22,7**	24,4
MWG	**51,2**	70,6	**39,2**	57,4
MWG iso	**18,6**	27,2	**16,8**	24,3
TOSCA	**9,7**	20,5	**11,5**	39,4
SHREC19	**34,5**	39,4	**33,7**	36,7

Table 4.8: Average Geodesic Error of point-wise maps obtained using ZoomOut [16] and Gaussian-embeddings. Using a smooth function, such as Gaussians, always improves the conversion for PC-Gau.

dataset	Delta		Gaussian	
	ours $\times 10^{-3}$	**LB** $\times 10^{-3}$	**ours** $\times 10^{-3}$	**LB** $\times 10^{-3}$
FAUST	**7,5**	8,6	**5,8**	8,1
MWG	**10,4**	21,2	**7,2**	13,8
MWG iso	9,1	**9,0**	**6,2**	8,4
TOSCA	**3,3**	4,0	**2,3**	3,9
SHREC19	**6,8**	7,4	**5,8**	7,2

Table 4.9: Distortion of point-wise maps obtained using Gaussian-embeddings inside ZoomOut [16]. Our basis particularly benefits from the usage of a smooth function for embeddings, reaching a considerably lower distortion than LB.

5 | Analysis

In this chapter, we further analyze the characteristics of PC-Gau. In Section 5.1 we analyze in greater detail the results obtained in the experimental evaluation, highlighting their relation with the core characteristic of PC-Gau, namely the even distribution of its energy on the mesh surface. In Section 5.2 we analyze the choice of parameters for the construction of PC-Gau, showing their impact on the final results, and we devise a simple method to select good values of such parameters in real applications. In Section 5.3 we briefly discuss the choice for the size of the basis k adopted throughout all this Thesis. Finally, in Section 5.4, we consider different applications for our basis, other than shape matching, shedding light on some limitations of PC-Gau but also on interesting future directions to explore.

5.1. Spatial distribution of basis energy

In this Section, we consider the spatial distribution of the energy of the bases on the mesh surface. Intuitively, by energy of a basis in the vertex x, we mean the expressive power and the quality of the embedding space induced by the basis in a neighborhood of x. More precisely, as we stated in Section 2.4, we consider two properties to define the quality of the embedding space: (i) the discrimination power between different vertices and (ii) the preservation of locality among vertices.

5.1.1. Discrimination power

We start by considering how the discrimination power of LB and PC-Gau is distributed on the surface of a mesh. We define discrimination power as the capability of a basis to assign sufficiently different embeddings to different vertices. Quantitatively, we assess discrimination power with the following metric.

Metric

Let us consider a mesh $\mathcal{M}$ with n vertices and the matrix of its embeddings $\Sigma_{\mathcal{M}}$. We could use also generalized embeddings (see Section 4.7), but here we consider delta-embeddings because of their simplicity and because they are the most common choice, therefore $\Sigma_{\mathcal{M}} = \Phi_{\mathcal{M}}^T$. Note that $\text{Emb}(x)$ is the x-th column of this matrix, namely $\text{Emb}(x) = \Sigma_{\mathcal{M}}(:, x)$. For each vertex x we find the vertex y which has the most similar embedding, in terms of Euclidean distance, to the embedding of x. We associate to x the euclidean distance between $\text{Emb}(x)$ and $\text{Emb}(y)$. We want such distance to be sufficiently large, in order to allow a proper discrimination between the vertices, once the embeddings are transferred with C (see Section 2.2.2). In order to make the metric independent of the vertex density, we normalize the euclidean distance by the geodesic distance between x and y. Note that y is not, in general, the closest vertex on the mesh and that this normalization rewards the choice of a y which is actually close in the geodesical sense. Summarizing, we propose and use the following metric to assess the discrimination power of a basis in the vertex x:

$$\text{Dis}(x) = \frac{\|\text{Emb}(x) - \text{Emb}(y)\|_2}{\text{GeoDist}_{\mathcal{M}}(x, y)} \qquad \text{with } y = \underset{z \in V_{\mathcal{M}} \setminus \{x\}}{\text{argmin}} \{\|emb(z) - emb(x)\|_2\} \tag{5.1}$$

$\text{Dis}(x)$ is a function defined on the mesh and we can render it with colors, in order to visually assess how the discrimination power of a basis is spatially distributed on a mesh. In order to make the assessment more general, we average the value of Dis on all the meshes of a dataset. To do this, we transfer the function Dis computed on different meshes of the dataset to a unique mesh $\mathcal{M}$, using the provided ground-truth correspondence, and then compute the average on $\mathcal{M}$. This is also possible when the provided ground-truth correspondence is not complete: in this case we fill in missing values by interpolation, in order to have a smoother visualization.

Results

In Figures 5.1a and 5.2a, we compare the spatial distribution of discrimination power between PC-Gau and LB on FAUST and SHREC19. We chose to show FAUST and SHREC19 as example datasets in this section, but the results are similar all the other datasets. We observe that for PC-Gau the value of Dis is evenly distributed on the mesh and there are no areas with particularly low values. LB, on the contrary, presents some areas (arms and legs) with noticeably lower values. This means that vertices in these areas are not well discriminated in the embedding space created by LB. We also observe that the metric value is slightly lower for PC-Gau in some areas of the body. However, the global value is considerably higher for PC-Gau. This means that our basis is capable

of redistributing evenly the energy on the whole mesh, including extremities, without leaving the massive areas poorly covered.

5.1.2. Locality Preservation

We now analyze the locality preservation of PC-Gau, compared to LB. In Section 2.2.2 we gave a definition of locality preservation and an intuitive explanation of its role in the conversion to point-wise maps. In this section we present two metrics to evaluate quantitatively the locality preservation of PC-Gau and LB, both in terms of overall value and distribution on the mesh surface. We show that the overall improvement in locality preservation of PC-Gau with respect to LB comes, again, from a more evenly distribution on mesh.

Metrics

We start by presenting the two metrics used to assess locality preservation.

Embedding/Geodesic Distance Correlation (EGDC) For each vertex x, we compute the correlation between the euclidean distance of the s embeddings closest to $\text{Emb}(x)$ and the geodesic distances of the corresponding vertices. In formulas, we can write:

$$\text{EGDC}(x) = \text{corr}\left(\text{GeoDist}_{\mathcal{M}}(x, y), \|\text{Emb}(x) - \text{Emb}(y)\|_2\right)_{y \in S} \tag{5.2}$$

where S is the set of s vertices with the closest embedding to $\text{Emb}(x)$. The idea, here, is to use correlation to evaluate how much the ordering between geodesic distances and embedding distances is preserved. Of course, the higher the correlation, the better. The choice to limit the computation to the lowest s embedding distances is to restrict the evaluation in a neighborhood of the embedding. After a certain point, indeed, embedding distances reach a plateau, which is trivially uncorrelated to the geodesic distances. In the current evaluation, we used $s = 80$.

Mean Geodesic Distance (MGD) For each vertex x we find, again, the t closest embeddings. Then we compute the mean of the geodesic distances, from x, of the corresponding vertices. Finally, we normalize the result by the mean geodesic distance of the actual t closest vertices to x (in the geodesic sense). In formulas, we can write:

$$\text{MGD}(x) = \frac{\text{Avg}_{y \in R}\left\{\text{GeoDist}_{\mathcal{M}}(x, y)\right\}}{\text{Avg}_{z \in \bar{R}}\left\{\text{GeoDist}_{\mathcal{M}}(x, z)\right\}} \tag{5.3}$$

where R is the set of t vertices with the lowest embedding distance to $\text{Emb}(x)$ and $\bar{R}$ is the set of t vetices with the lowest geodesic distance from x. Here the idea is that, if locality is preserved, the closest embeddings should correspond also to the closest vertices in the geodesical sense. In this case the value of MGD gets lower and approaches to 1. Therefore, lower is better for MGD. We used a smaller scale for this metric, choosing $t = 10$.

Results

Note that both EGDC and MGD are functions defined on the mesh. To asses their overall value we compute the average on the vertices of a mesh and then average this value across multiple meshes from a dataset. Alternatively, we can visualize the value of these metrics as a function on the mesh to assess their spatial distribution. As usual, we can average the value of EGDC and MGD across multiple meshes, using the provided ground-truth correspondence, as briefly described in Section 5.1.1.

We present the overall value of EGDC and MGD for different datasets in Table 5.1. Note that our basis performs better than LB in both metrics and in all datasets. Note also that the overall values of MGD are more stable across different datasets, compared to LB.

dataset	EGDC		MGD	
	ours $\times 10^{-2}$	**LB** $\times 10^{-2}$	**ours**	**LB**
FAUST	**79,7**	73,8	**1,11**	1,34
MWG	**83,8**	78,1	**1,12**	1,36
TOSCA	**84,1**	79,0	**1,12**	1,40
SHREC19	**81,9**	68,8	**1,14**	1,86

Table 5.1: Overall values of EGDC and MGD. Average on the meshes of different datasets. Our basis performs better than LB in both metrics, in all datasets.

Figures 5.1b, 5.1c, 5.2b and 5.2c show the average distribution of MGD and EGDC on the meshes of FAUST and SHREC19. The distribution shown gives us an insight on the higher overall results of EGDC and MGD: LB presents areas with extremely bad values (whether high or low depends on the metrics). The values for PC-Gau, instead, are more evenly distributed on the mesh. For PC-Gau, some areas have slightly worse values compared to LB, similarly to what happened with discrimination power in Section 5.1.1. However, the overall effect of the even distribution is an increase of the global quality of the basis. This is an important strength of PC-Gau: to spread the basis energy over the whole mesh surface, without penalizing (too much) any area and reaching an overall

improvement.

In Section 5.2 we use the metrics defined in the current section to perform parameter selection at test time. As explained in Section 5.2.3, the predictive power of EGDC and MGD on the error of the final maps, for different values of the parameters, is a further confirmation of their capability to capture properties of the embedding space that are actually relevant to point-wise conversion.

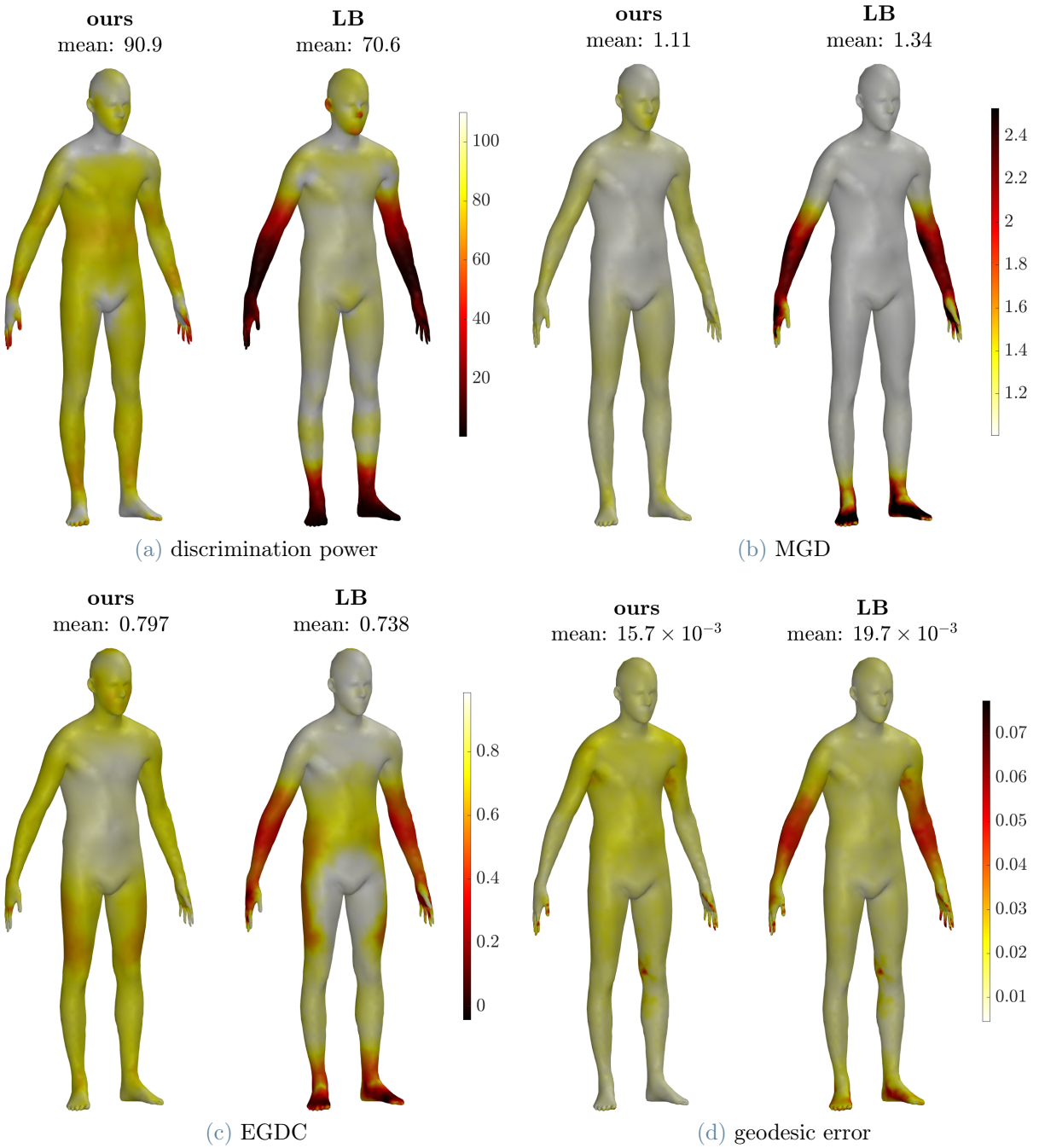
5.1.3. Relation to point-wise error

One of the major claims of this work is that the energy of LB is not (sufficiently) evenly distributed on the mesh surface and that this causes a lack of accuracy in the conversion to point-wise maps. We have, thus, proposed a basis that has energy distributed (more) evenly on the mesh surface, in order to overcome this issue and obtain better point-wise maps. We showed in the experimental evaluation in Chapter 4 that our basis actually reaches better accuracy than LB in many different settings. In the current chapter, we also showed that the energy of our basis is evenly distributed on the mesh, according to metrics for discrimination power and locality preservation that we defined. At the same time, we showed that the energy of LB presents areas with a lower quality of embedding space. In order to bridge the gap between basis energy and point-wise error, and support their causal relation, we now explicitly compare the spatial distribution of the error in point-wise maps with the spatial distribution of the basis energy, for PC-Gau and LB. If our reasoning is correct, the error should be localized where the embedding space of a given basis is lacking in quality.

Figures 5.1 and 5.2 compare at one glance the spatial distribution of Dis (5.1), MGD (5.3), EGDC (5.2) and geodesic error for FAUST and SHREC19. The functions depicted are averaged on multiple meshes or multiple pairs extracted from the dataset. The geodesic error is computed on point-wise maps converted from grond-truth C, exactly as in Section 4.4. Figures 5.1d and 5.2d are indeed taken from Figure 4.1. For this comparison, we chose the setting with ground-truth C, in order to be as independent as possible from possible errors in the estimation of C.

We can immediately observe that the error distribution closely resembles the energy distribution of the correspondent bases. Point-wise maps obtained using LB present a high error on the arms (particularly around the elbow) and feet. These areas are included, and almost coincide, in the areas in which the quality of the embedding space, evaluated as discrimination power and locality preservation, is lower. The error distribution for PC-Gau, instead, is much more even on the mesh, and so is its energy distribution.

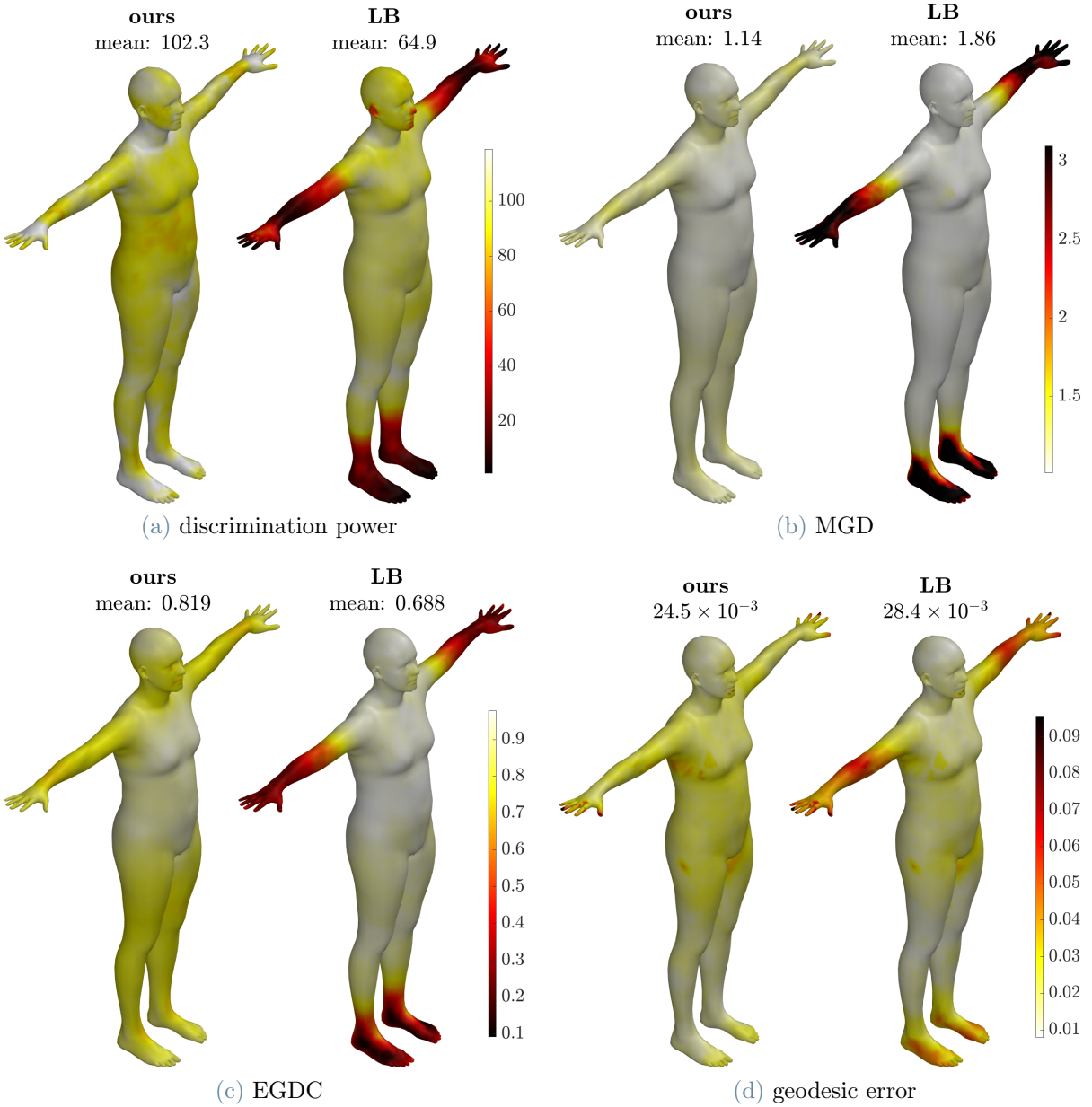
PC-Gau gets slightly worse values of error on the massive areas of the body, such as trunk and thighs, but the overall value is still better than LB. Note, finally, that isolated spikes, common for *ours* and LB, are probably due to some error in the ground-truth correspondence provided.



(a) discrimination power (b) MGD

(c) EGDC (d) geodesic error

Figure 5.1: Average spatial distribution of basis energy (measured through discrimination power, MGD, EGDC) and geodesic error on meshes from FAUST. Darker is worse in all cases. The error of LB is localized where the quality of the embedding space is lower. PC-Gau presents a uniform distribution both of basis energy and geodesic error.



(a) discrimination power (b) MGD

(c) EGDC (d) geodesic error

Figure 5.2: Average spatial distribution of basis energy (measured through discrimination power, MGD, EGDC) and geodesic error on meshes from SHREC19. Darker is worse in all cases. The error of LB is localized where the quality of the embedding space is lower. PC-Gau presents a uniform distribution both of basis energy and geodesic error.

5.2. Parameter Selection

In this Section we analyze more in detail the choice of the parameters used to build our basis, and their impact on the resulting point-wise maps. Note that, while LB does not require any parameter for its computation, PC-Gau depends on some parameters:

- the number of vertices q (and Gaussian functions) in the subset Q
- the amplitude σ of the Gaussian functions

However, we show that the value of such parameters can be chosen using the metrics presented in Section 5.1 and 5.1.2, which do not require any ground truth correspondence to be computed. This fact makes the accurate choice of the parameters possible also at test time, in real applications.

Note that in this section we consider only parameters strictly related to the computation of the basis itself. Instead, we address the choice of the number of basis atoms k in Section 5.3.

5.2.1. Number of vertices q

As shown in Section 3.1, the Gaussian functions in the dictionary, to which is applied PCA, are centered in q vertices scattered on the mesh surface. Thus, q determines also the number of Gaussians in the dictionary. In the experimental section we used $q = 1000$ in all the settings and for all datasets. Now, we show why this is a reasonable choice.

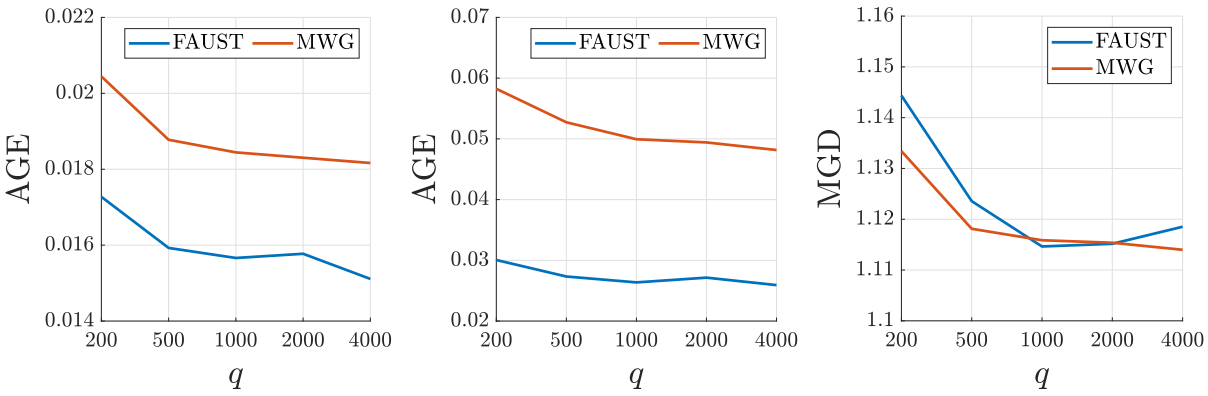
Figure 5.3 presents the average geodesic error of point-wise maps obtained with our basis, computed using different values of q. For each value of q the basis is recomputed and then tested both as shown in Section 4.4 (C computed from ground-truth) and 4.5 (C estimated using product preservation [21]). These tests have been performed on 30 pairs from FAUST and MWG.

We observe that the error decreases as q increases, with less noticeable effects after $q = 1000$. However, the time needed to compute our basis increases with q, more noticeably when q is high. Consider, for instance, that computing PC-Gau on a mesh of FAUST takes around 1 second for $q = 1000$ and more than 35 seconds for $q = 4000$. Therefore, $q = 1000$ is a good trade-off between accuracy and computation time, since it provides almost optimal results for much less computation time.

The evaluation we presented until now, however, is possible only when a ground-truth correspondence is provided to assess the average geodesic error on the final point-wise

map. In a real shape-matching scenario, this is not possible, but we can use the average value of the metrics proposed in Section 5.1.2 to choose a proper value of q. Note that these metrics asses the quality of the embedding space on a single mesh and, therefore, do not require any ground-truth correspondence to be computed. In particular, here we consider the use of the Mean Geodesic Distance (MGD, see Section 5.1.2) for the purpose, since it is faster to compute than EGDC (see Section 5.1.2). Figure 5.3c shows the average value of MGD computed on meshes from MWG and FAUST, for instances of PC-Gau computed with different values of q. We observe that MGD is a good predictor for the error of the final point-wise map and therefore it can be used to select the value of q. In the cases of the example, we would have selected $q = 1000$ as the value with the better trade-off between quality and computation performance.

We also mention that the distribution of Q is more dependent on the vertex density when q is close to the total number of vertices n. This can easily be seen by considering the limit case in which $q = n$: in this case, Q contains all the vertices and the Gaussians are concentrated where the vertex density is higher. Therefore, it makes additional sense not to use too high values of q.



(a) point-wise error (GT C) (b) point-wise error (NO17 C) (c) locality preservation

Figure 5.3: Comparison of point-wise error (a)(b) and locality preservation (c) among different values of q.

Q sampling method

We saw in previous sections that the most important strength of PC-Gau is the even distribution of its energy on the mesh surface. We saw in Section 3.1 that we induce this characteristic by choosing a subset of vertices Q that is evenly distributed on the mesh. In Section 3.1 we used Farthest Point Sampling (FPS) [19] with Euclidean metric to sample Q, but in principle we could use any method that provides a even scattering of Q. In this section, we test the use of random sampling of q vertices among the n vertices of the

mesh for sampling Q.

Figure 5.4 presents a comparison between FPS and random sampling in terms of spatial distribution of basis energy (through Dis and MGD metrics) and geodesic error of the obtained point-wise maps. For the geodesic error both its localization and its mean value (AGE) are shown. We used FAUST and SHREC19 for this experiment: while FAUST has a regular tessellation on all meshes, SHREC19 presents some meshes with a heavily irregular one. However, we observe that for $q = 1000$ the two methods are practically equivalent, both regarding the energy distribution and the geodesic error.

Note that we still prefer FPS to random sampling as it is, in general, more stable and safer to use:

1. because it is insensible to a difference of vertex density on the mesh. We see this in Figure 5.4a, where the distribution is slightly more uneven for random sampling. Consider, for instance, the higher difference in values between hand and chest.
2. because with low values of q there is a chance that some parts of the mesh are left uncovered by random sampling.

5.2.2. Amplitude of Gaussian functions σ

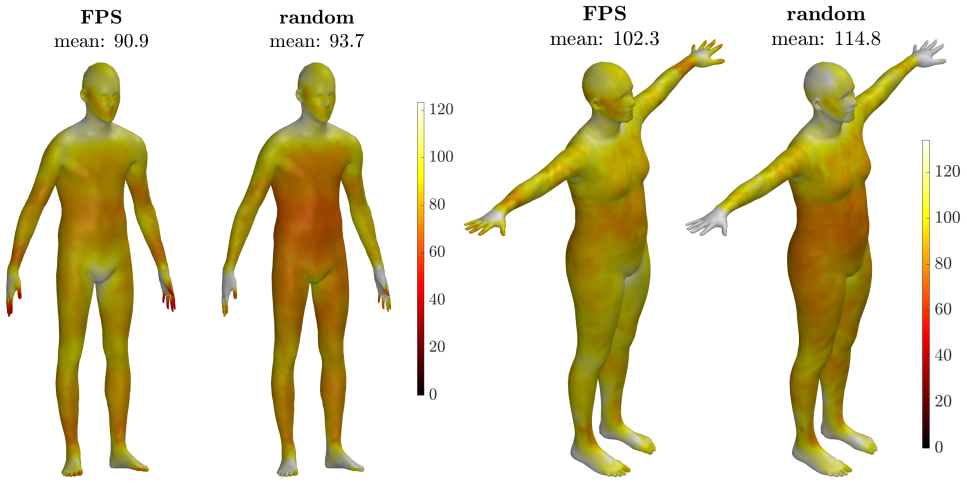
Let us recall from Section 3.1 that the Gaussian centered in the j-th vertex of Q is computed as

$$G(i) = \exp\left(-\frac{\text{GeoDist}^2_{\mathcal{M}}(V_{\mathcal{M}}(i), Q(j))}{\sigma}\right) \quad \forall i \in V_{\mathcal{M}}$$

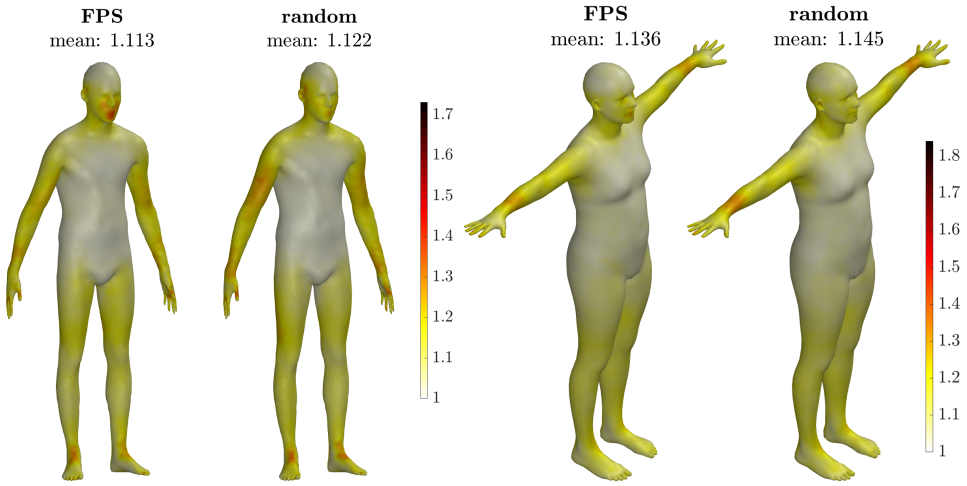
We observe that the parameter σ sets the amplitude of the Gaussian. The smaller is σ and the more the Gaussian is localized around the vertex $Q(j)$, as shown in Figure 5.5.

The choice of σ has a sensible impact on the quality of our basis and of the final point-wise map. Here, we present a brief analysis that justifies our choice to use $\sigma = 0.05$. As we did for q in Section 5.3, we start by presenting the results on the geodesic error of the point-wise map $\bar{T}$ obtained for different values of σ. Then, we show that a similar choice of σ can be performed also in real scenarios by considering the value of MGD (see Section 5.1.2).

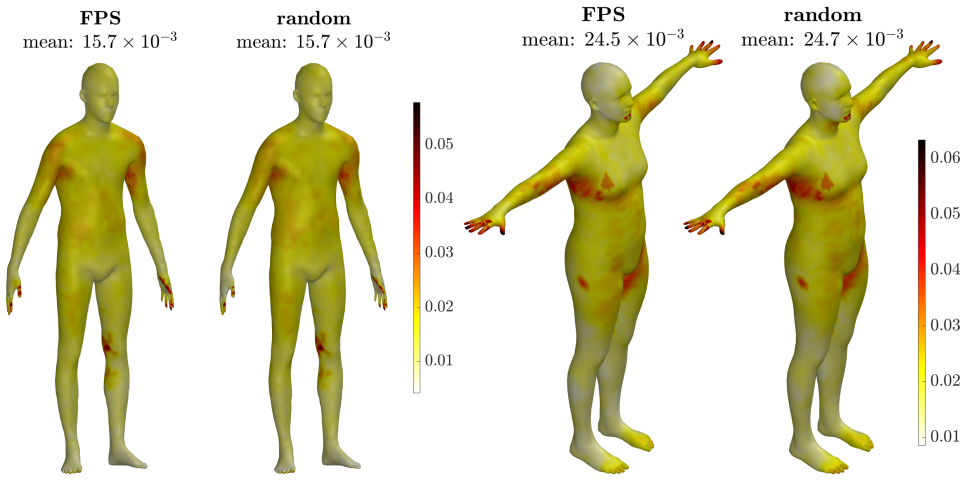
Figure 5.6 presents the average geodesic error on point-wise maps obtained from our basis computed using different values of σ. The point-wise maps are extracted using both a C computed from the ground truth correspondence (a), as done in Section 4.4, and a C estimated using [21] (b), as done in Section 4.5. The tests have been performed on 30 random pairs from FAUST and MWG. We observe that the error presents a minimum for



(a) discrimination power



(b) MGD



(c) geodesic error

Figure 5.4: Comparison between random sampling and Farthest Point Search in the selection of Q on FAUST (first column) and SHREC19 (second column). We consider different metrics to perform the comparison, both as distribution and as average value: discrimination power (a), MGD (b) and geodesic error (c). The error is computed on point-wise maps obtained from a ground-truth C. Darker is worse in all cases. The results show that even a random sampling is a viable option for sampling Q.

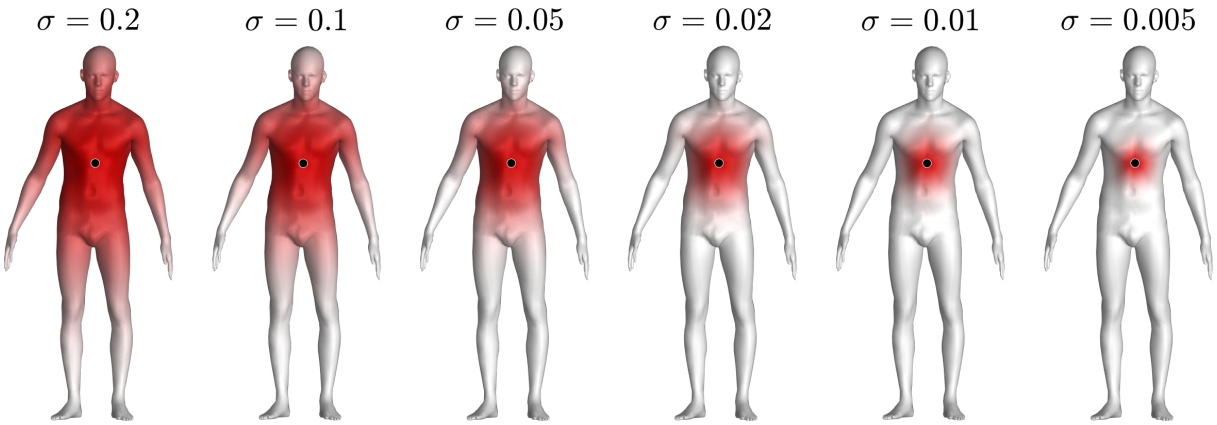


Figure 5.5: Example of Gaussian functions obtained for different values of σ. The lower σ, the more localized the Gaussian around the vertex x (marked here with a black circle).

$0.05 > \sigma > 0.02$, meaning that Gaussians that are both too wide and too narrow produce sub-optimal results. The choice of σ, thus, is slightly more difficult than the choice of q.

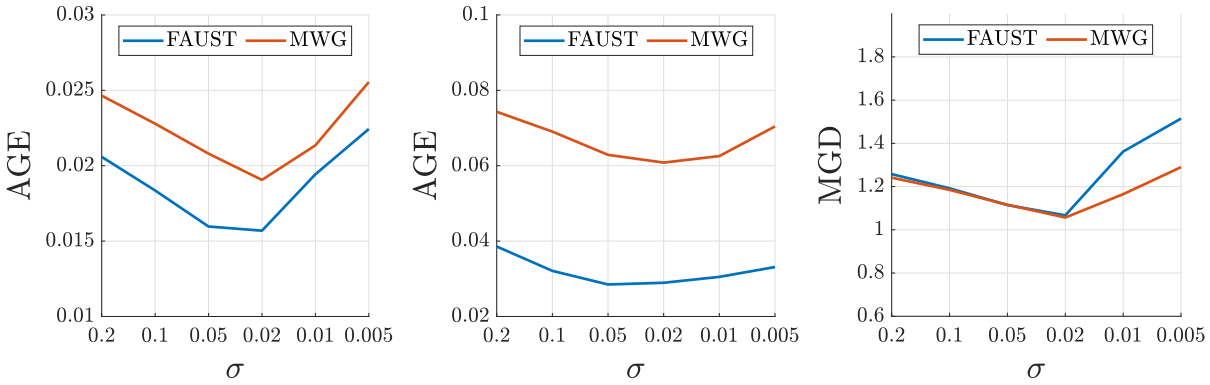
In a real scenario, where no ground-truth correspondence is given, we can use the value of MGD to choose the best value of σ. This is similar to what we did in Section 5.3 for q. Again, we prefer MGD over EGDC because MGD is faster to compute, but they both yield to the same choice of σ. Figure 5.6c shows the global value of MGD, computed for instances of our basis obtained for different values of σ. In the chart is averaged on 30 meshes from FAUST and 24 from MWG.

Notice the good agreement between the values of MGD and geodesic error in the different settings. In particular, we observe that the value of σ that minimizes the MGD ($\sigma = 0.02$) also leads to optimal or nearly-optimal results on the error in all cases. Remember that MGD assesses the quality of the embedding space produced by a basis on a mesh, only with reference to the mesh itself.

Note also that in our experiments we chose to use $\sigma = 0.05$, which is close to the optimal value, according to MGD, but it does not coincide with it. We justify our choice by observing that MGD increases more rapidly for $\sigma < 0.02$ than for $\sigma > 0.05$. Our choice is thus more robust to possible inaccuracies in the estimation provided by MGD.

5.2.3. Predictive power of locality-preservation metrics

As we saw qualitatively in Section 5.2.1 and 5.2.2, there is a strong agreement between the values of the metrics presented in Section 5.1.2 for an instance of PC-Gau computed with a certain choice of parameters, and the error of the point-wise map obtained by a shape-matching pipeline employing the same instance of PC-Gau.



(a) point-wise error (GT C) (b) point-wise error (NO17 C) (c) locality preservation

Figure 5.6: Comparison of point-wise error (a) (b) and locality preservation (c) for different values of σ. MGD is a good predictor of the error at test time, thus can be used to select the best value of σ.

We can assess this agreement, or predictive power, through correlation. Note that both the metrics (EGDC and MGD) and the point-wise map depend on the parameters used to build the basis. We can write MGD(σ), EGDC(σ) and AGE($\bar{T}(\sigma)$) to highlight their dependence on σ. Table 5.2 presents the value of correlation between MGD(σ) (and EGDC(σ)) and AGE($\bar{T}(\sigma)$) for σ assuming values in the set $[0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 1, 2] \times 10^{-1}$. The first entry, for instance, is computed as:

$$\text{corr}\left(\text{MGD}(\sigma), \text{AGE}(\bar{T}_{\text{GT } C}(\sigma))\right) \quad \text{with } \sigma \in [0.005, 0.01, \ldots, 0.1, 0.2]$$

where $\bar{T}$ is computed by converting a C found using the ground truth correspondence. Other entries are compute similarly. Numerical values support what we observed qualitatively in Sections 5.2.1 and 5.2.2.

dataset	GT C		NO17 C	
	MGD(σ)	EGDC(σ)	MGD(σ)	EGDC(σ)
FAUST	0,94	-0,91	0,58	-0,78
MWG	0,99	-0,82	0,84	-0,82

Table 5.2: Correlation between metrics for measuring the locality-preservation (MGD and EGDC) and the resulting point-wise errors, both computed for different values of σ.

In the previous sections, we used this predictive power to devise a method for selecting the optimal values for parameters in real application. We now devote this brief Section to show how this accordance, between metrics and error, is particularly relevant for our work. For different reasons:

1. It supports our claim that the quality of the final point-wise maps strongly depends

on the quality of the embedding space, which is a property of the single mesh. This fact, in turn, strengthens our claim that LB is particularly limited by the fact the the quality of its embedding is not even across the whole mesh.

2. It shows that we devised meaningful metrics to assess the quality the embedding space produced by a basis on a mesh.

3. it paves the way to a higher level of adaptivity for PC-Gau, in which the basis is tailored on each mesh by selecting the best parameters for that specific mesh.

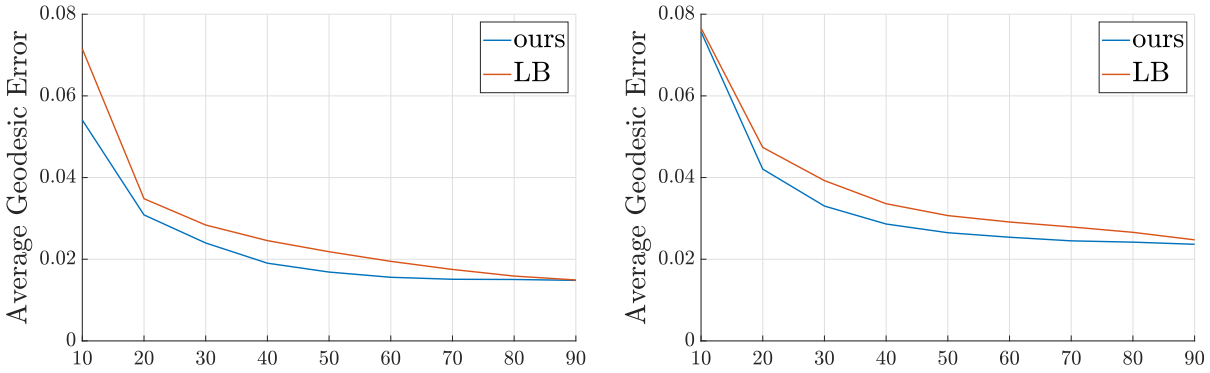
5.3. Number of atoms k

The first step of any shape-matching pipeline with functional maps is to choose a *moderately sized* functional basis. This size of the basis is given by the parameter k, which is the number of basis atoms considered. As we saw in Section 2.2, this truncation makes the correspondence compactly represented by a matrix C of size $k \times k$, which is in turn amenable to optimization. For the sake of simplicity, we made the assumption that both bases have size k. Note, however, that this assumption is not required neither by the functional map framework, nor by PC-Gau or LB. In the general case in which the two bases have different sizes $k_{\mathcal{M}}$ and $k_{\mathcal{N}}$, C is a rectangular matrix of size $k_{\mathcal{N}} \times k_{\mathcal{M}}$. In the following, though, we still consider the simplified version, for consistency with the rest of the Thesis.

In all the experiments presented in this work we used a number of atoms $k = 60$. Typical values for functional maps vary between 40 and 160, so the value we chose falls perfectly in it. Here, we evaluate briefly the use of other values of k, between 10 and 90. For each value of k, we compute C using the ground-truth correspondence, both for LB and PC-Gau. Then we convert to point-wise map, using the method presented in Section 2.2.2. The result shown in Figure 5.7 is averaged on 30 random pairs from FAUST and SHREC19. We observe that the advantage of our basis is significant for $k < 70$. This allow us to use a lower value of k for PC-Gau, in comparison to LB, and obtain the same geodesic error on the final point-wise map. For instance, using 50 atoms of PC-Gau in SHREC19 is equivalent, on average, to using 80 atoms of LB.

This analysis is consistent with the claims about the distribution of basis energy we made throughout this work. If a limit of LB is its energy distribution, with a higher number of atoms k, the energy of LB spreads over the whole mesh and its distribution gets more evenly. Therefore, the advantage provided by PC-Gau becomes less and less noticeable with increasing k. However, being able to reach similar results with less atoms is still a

valuable benefit, as the cost of solving the optimization problem to find C increases with k.



(a) FAUST (b) SHREC19

Figure 5.7: Average Geodesic Error as a function of the number of atoms k considered. Point-wise maps converted from ground-truth C. Average on 30 pairs from FAUST (a) and SHREC19 (b). The advantage of our basis is sensible for $k < 70$.

5.4. Function approximation and transfer

In this section, we evaluate the capability of PC-Gau of approximating functions on a mesh and of transferring them from one mesh to another *without recovering a point-wise map* between the two meshes. This is a related, but different application context from shape matching. As usual we compare the performance of PC-Gau with the standard basis LB.

5.4.1. Approximation

As we saw in Sections 1.1.2 and 3.2.2, when we truncate LB or PC-Gau to the first k atoms we obtain the basis of a subspace S of $\mathcal{F}(\mathcal{M}, \mathbb{R})$, in which functions from $\mathcal{F}(\mathcal{M}, \mathbb{R})$ are represented through a low-pass filter approximation. We can evaluate the quality of the approximation $\tilde{f}$ of a function $f \in \mathcal{F}(\mathcal{M}, \mathbb{R})$ by considering the normalized error with respect to the original function:

$$\varepsilon_{\text{approx}} = \frac{\left\| f - \tilde{f} \right\|_{\mathcal{M}}}{\|f\|_{\mathcal{M}}} \qquad \text{with } \tilde{f} = \Phi_{\mathcal{M}} \Phi_{\mathcal{M}}^T A_{\mathcal{M}} f \tag{5.4}$$

The norms in Equation (5.4) are computed as shown in Chapter 1:

$$\|g\|_{\mathcal{M}} = \sqrt{\langle g, g \rangle_{\mathcal{M}}} = \sqrt{g^T A_{\mathcal{M}} g} \qquad \forall g \in \mathcal{F}(\mathcal{M}, \mathbb{R})$$

In the first column of Table 5.3 we present the comparison between the approximation error of LB and PC-Gau for different classes of functions, approximated on the meshes of FAUST:

- **WKS** [3] and **HKS** [31] are two descriptors. We used 100 samples each, computed for 100 values of their parameters.
- **Cosine** are 10 sample sinusoidal functions of the coordinates.
- **Indicator** are the indicator functions corresponding to 10 disjoint segments on the mesh.
- **HK**, **Gaussian** and **Delta** are, respectively, 100 Heat Kernels, Gaussian functions and Delta functions scattered on the surface. Gaussians are computed with two different amplitudes: $\sigma = 0.05$ (same amplitude of the Gaussians used to build PC-Gau) and $\sigma = 0.005$.
- **Coordinates** are the three functions given by the X, Y, Z coordinates of each vertex.

As in the rest of the Thesis, we used a basis size of $k = 60$.

From Table 5.3, we observe that LB has, in general, an approximation error lower than PC-Gau. The only exceptions are

- Gaussian functions computed with the same σ used in the construction of PC-Gau, as we would expect from the properties of PCA (see Section 3.1.3).
- Sinusoidal functions of the coordinates and Coordinates: even though our basis provides a low-pass filter approximation, it probably retains more information about the local geometry of vertices, therefore functions of the Coordinates are better approximated. This claim is supported by the fact that the Dirichlet energy of the atoms of PC-Gau is higher than the atoms of LB, as shown in Section 3.2.2.

5.4.2. Transfer

Let us recall from Section 2.2 that in the Functional Map framework we can transfer the coefficients $\hat{f}$ of a function f from a mesh $\mathcal{M}$ to a mesh $\mathcal{N}$ simply by matrix multiplication: $\hat{g} = C\hat{f}$. Then, we can reconstruct the function on $\mathcal{N}$ from $\hat{g}$ as $g = \Phi_{\mathcal{N}}\hat{g}$. Summarizing, given two basis $\Phi_{\mathcal{M}}$ and $\Phi_{\mathcal{N}}$, and their functional correspondence C, a function $f \in \mathcal{F}(\mathcal{M}, \mathbb{R})$ is transferred on $\mathcal{N}$ as:

$$\tilde{g} = \Phi_{\mathcal{N}} C \Phi_{\mathcal{M}}^T A_{\mathcal{M}} f \tag{5.5}$$

Class of functions	approximation		transfer (GT C)		transfer (NO17 C)	
	ours $\times 10^{-2}$	**LB** $\times 10^{-2}$	**ours** $\times 10^{-2}$	**LB** $\times 10^{-2}$	**ours** $\times 10^{-2}$	**LB** $\times 10^{-2}$
WKS	27,17	**19,6**	28,41	**21,37**	31,06	**27,88**
HKS	6,28	**1,72**	6,67	**2,33**	**9,51**	9,99
Cosine	**24,02**	24,54	26,61	**26,54**	**31,85**	33,78
Indicator	30,18	**26,74**	30,91	**27,29**	32,61	**30,64**
HK	56,57	**49,19**	59,4	**52,46**	62,08	**56,94**
Gaussian ($\sigma = 0.05$)	**2,73**	5,43	**4,57**	6,19	**10,94**	13,8
Gaussian ($\sigma = 0.005$)	53,27	**46,85**	56,05	**49,83**	58,84	**54,3**
Delta	99,18	**99,16**	99,28	**99,25**	**99,36**	**99,36**
Coordinates	**5,08**	5,85	**5,83**	6,61	**11,35**	15,04

Table 5.3: Normalized error for function approximation and transfer on FAUST. Comparison between *ours* and LB. WKS, HKS, HK, gauss, delta are averaged over 100 sample functions. LB is generally better in function approximation and transfer.

We applied Equation (5.5) to each of the functions used for approximation in Section 5.4.1, using both a C computed from the ground truth correspondence (as done in Section 4.4) and a C estimated using product preservation [21] (as done in Section 4.5). The results are shown in the second and third columns of Table 5.3. The tests have been performed on 200 random pairs from FAUST. In the case of function transfer, we define the normalized error, analogously to the approximation case, as:

$$\varepsilon_{\text{transfer}} = \frac{\|g - \tilde{g}\|_{\mathcal{N}}}{\|g\|_{\mathcal{N}}} \qquad \text{with } \tilde{g} = \Phi_{\mathcal{N}} C \Phi_{\mathcal{M}}^T A_{\mathcal{M}} f$$

where $g \in \mathcal{F}(\mathcal{N}, \mathbb{R})$ is the image of $f \in \mathcal{F}(\mathcal{M}, \mathbb{R})$ computed using the ground-truth correspondence $T : V_{\mathcal{N}} \to V_{\mathcal{M}}$ provided by the dataset:

$$g(x) = f(T(x)) \qquad \forall x \in V_{\mathcal{N}}$$

Note that for this evaluation we need a dense ground-truth correspondence, defined for each vertex of $\mathcal{N}$, therefore we used the original version of FAUST, not re-meshed to 5K vertices.

From Table 5.3 we observe that the functions transferred using LB have, in general, lower error compared to PC-Gau, with the exceptions of Gaussians (with $\sigma = 0.05$) and Coordinates. This is due to the good performances of our basis for the approximation of these specific classes.

6 | Conclusions

We presented the procedure for constructing a new basis for the space of real-valued functions defined on a mesh. Compared to the eigenfunctions of the Laplace-Beltrami operator, which is currently the ubiquitous basis for functional maps, the energy of our basis is more evenly distributed on the mesh surface. As a consequence, the embedding space induced by our basis provides a more informative representation for vertices in all areas of the mesh. We proposed a few metrics to evaluate some key properties of the embedding space, namely discrimination power and locality preservation, which are particularly useful for point-wise conversion of functional maps. We also showed that these metrics can be employed in realistic settings to make an accurate choice of the parameters required to build our basis.

Our basis shares some of the good qualities of the LB basis, such as orthonormality, frequency ordering and isometry invariance. Thanks to these common properties, our basis can replace the standard LB basis in existing functional map pipelines, at no cost. Through experimental evaluation on pairs of meshes from established datasets, we showed that replacing LB with our basis actually leads to substantial improvements in the accuracy of the point-wise maps obtained, for the same shape matching pipeline. In particular, under the assumption to be able to estimate the optimal functional map C, our basis constantly outperforms LB. To give a scale, on the 620 pairs tested overall, our basis was better in 97% of cases, showing an average accuracy gain of 20% with respect to LB.

Interestingly, the benefit brought by the use of our basis adds up to other improvements tackling different parts of the functional map framework, namely the estimation of C or the conversion to point-wise map. This makes our proposal complementary to other approaches in the improvement of shape matching via functional maps.

6.1. Limitations

Our basis still presents some important limitations.

- It is generally worse than LB in signal approximation and transfer, so it may not

be an option when we want to transfer signals without recovering a point-wise map. There are, however, exceptions to this observation, which open the way to further work in this context.

- Our basis may be more difficult to align between shapes related by strong non-isometries, also depending on the method used for the estimation of C

6.2. Future work

Our work leaves room for interesting directions to explore in the future.

Dictionary composition The initial dictionary, to which PCA is applied, does not necessarily need to be composed of Gaussian functions. The use of other classes of functions, possibly incorporating information about the local geometry of the mesh, may lead to interesting properties of the basis.

Adaptability In Section 5.2 we showed that an accurate choice of parameters can be made by using the proposed metrics for locality preservation. This method could be employed to tailor the parameters mesh-by-mesh, bringing more adaptability to our basis. In addition, the parameters of the functions in the dictionary could vary function-by-function, adapting even to the local geometry of the mesh.

Signal approximation Our basis could find application in the approximation of specific classes of signals (functions) on meshes, as the results in Section 5.4 seem to suggest. In particular, the approximation of coordinates and other functions based on the local structure of the mesh could benefit from our basis.

Bibliography

[1] Y. Aflalo and R. Kimmel. Regularized principal component analysis. *Chinese Annals of Mathematics, Series B*, 38:1–12, 2017.

[2] Y. Aflalo, H. Brezis, and R. Kimmel. On the optimality of shape and data representation in the spectral domain. *SIAM J. Imaging Sci.*, 8(2):1141–1160, 2015.

[3] M. Aubry, U. Schlickewei, and D. Cremers. The wave kernel signature: A quantum mechanical approach to shape analysis. In *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pages 1626–1633. IEEE, 2011.

[4] F. Bogo, J. Romero, M. Loper, and M. J. Black. FAUST: Dataset and evaluation for 3D mesh registration. In *Proc. CVPR*, pages 3794–3801, Columbus, Ohio, 2014. IEEE.

[5] A. Bronstein, M. Bronstein, and R. Kimmel. *Numerical Geometry of Non-Rigid Shapes*. Springer, New York, NY, 2008.

[6] E. Çela. *The Quadratic Assignment Problem: Theory and Algorithms*. Kluwer Academic Publishers, 1998.

[7] G. Da Poian, R. Bernardini, and R. Rinaldo. Gaussian dictionary for compressive sensing of the ecg signal. In *2014 IEEE Workshop on Biometric Measurements and Systems for Security and Medical Applications (BIOMS) Proceedings*, pages 80–85, 2014.

[8] D. Ezuz and M. Ben-Chen. Deblurring and denoising of maps between shapes. *Computer Graphics Forum*, 36(5):165–174, 2017.

[9] D. Giorgi, S. Biasotti, and L. Paraboschi. Shape retrieval contest 2007: Watertight models track. *SHREC competition*, 8, 07 2008.

[10] A. Kovnatsky, M. Bronstein, A. Bronstein, K. Glashoff, and R. Kimmel. Coupled quasi-harmonic bases. *Computer Graphics Forum*, 32(2pt4):439–448, 2013.

[11] B. Levy. Laplace-beltrami eigenfunctions towards an algorithm that "understands"

geometry. In *IEEE International Conference on Shape Modeling and Applications 2006 (SMI'06)*, pages 13–13, 2006.

[12] F. Maggioli, S. Melzi, M. Ovsjanikov, M. Bronstein, and E. Rodolà. Orthogonalized fourier polynomials for signal approximation and transfer. In *Proceedings of Eurographics 2021*, 2021.

[13] S. Melzi. Sparse representation of step functions on manifolds. *Computers & Graphics*, 82:117–128, 2019. ISSN 0097-8493.

[14] S. Melzi, E. Rodolà, U. Castellani, and M. Bronstein. Localized manifold harmonics for spectral shape analysis. *Computer Graphics Forum*, 37(6):20–34, 2018.

[15] S. Melzi, R. Marin, E. Rodolà, U. Castellani, J. Ren, A. Poulenard, P. Wonka, and M. Ovsjanikov. SHREC 2019: Matching Humans with Different Connectivity. In *Eurographics Workshop on 3D Object Retrieval*. The Eurographics Association, 2019.

[16] S. Melzi, J. Ren, E. Rodolà, A. Sharma, P. Wonka, and M. Ovsjanikov. Zoomout: Spectral upsampling for efficient shape correspondence. *ACM Transactions on Graphics (TOG)*, 38(6):155:1–155:14, Nov. 2019. ISSN 0730-0301.

[17] S. Melzi, R. Marin, P. Musoni, F. Bardon, M. Tarini, and U. Castellani. Intrinsic/extrinsic embedding for functional remeshing of 3d shapes. *Computers & Graphics*, 88:1–12, 2020. ISSN 0097-8493.

[18] M. Meyer, M. Desbrun, P. Schröder, and A. H. Barr. Discrete Differential-Geometry Operators for Triangulated 2-Manifolds. In *Visualization and mathematics III*, pages 35–57. Springer, New York, NY, 2003.

[19] C. Moenning and N. A. Dodgson. Fast Marching farthest point sampling. Technical Report UCAM-CL-TR-562, University of Cambridge, Computer Laboratory, Apr. 2003.

[20] T. Neumann, K. Varanasi, C. Theobalt, M. Magnor, and M. Wacker. Compressed manifold modes for mesh processing. *Computer Graphics Forum*, 33(5):35–44, 2014.

[21] D. Nogneng and M. Ovsjanikov. Informative descriptor preservation via commutativity for shape matching. *Computer Graphics Forum*, 36(2):259–267, 2017.

[22] D. Nogneng, S. Melzi, E. Rodolà, U. Castellani, M. Bronstein, and M. Ovsjanikov. Improved functional mappings via product preservation. *Computer Graphics Forum*, 37(2):179–190, 2018.

[23] M. Ovsjanikov, M. Ben-Chen, J. Solomon, A. Butscher, and L. Guibas. Functional

maps: a flexible representation of maps between shapes. *ACM Transactions on Graphics (TOG)*, 31(4):30:1–30:11, 2012.

[24] D. Panozzo, Y. Lipman, E. Puppo, and D. Zorin. Fields on symmetric surfaces. *ACM Trans. Graph.*, 31(4), jul 2012. ISSN 0730-0301.

[25] U. Pinkall and K. Polthier. Computing Discrete Minimal Surfaces and their Conjugates. *Experimental mathematics*, 2(1):15–36, 1993.

[26] J. Ren, A. Poulenard, P. Wonka, and M. Ovsjanikov. Continuous and orientation-preserving correspondences via functional maps. *ACM Transactions on Graphics (TOG)*, 37(6), 2018.

[27] E. Rodolà, M. Moeller, and D. Cremers. Point-wise map recovery and refinement from functional correspondence. In *Proc. Vision, Modeling and Visualization (VMV)*, 2015.

[28] E. Rodolà, M. Möller, and D. Cremers. Regularized pointwise map recovery from functional correspondence. In *Computer Graphics Forum*, volume 36, pages 700–711. Wiley Online Library, 2017.

[29] R. M. Rustamov. Laplace-beltrami eigenfunctions for deformation invariant shape representation. In *Proc. SGP*, pages 225–233. Eurographics Association, 2007.

[30] R. W. Sumner and J. Popović. Deformation transfer for triangle meshes. In *ACM Transactions on Graphics (TOG)*, volume 23, pages 399–405. ACM, 2004.

[31] J. Sun, M. Ovsjanikov, and L. Guibas. A concise and provably informative multi-scale signature based on heat diffusion. *Computer graphics forum*, 28(5):1383–1392, 2009.

[32] B. Vallet and B. Lévy. Spectral geometry processing with manifold harmonics. *Computer Graphics Forum*, 27(2):251–260, 2008.

[33] O. Van Kaick, H. Zhang, G. Hamarneh, and D. Cohen-Or. A survey on shape correspondence. *Computer Graphics Forum*, 30(6):1681–1707, 2011.

List of Figures

List of Tables

List of Symbols

Symbol	Description
$\mathcal{X}$	2D manifold
$\mathcal{M}, \mathcal{N}$	3D meshes
$V_{\mathcal{M}}$	set of vertices of $\mathcal{M}$
$A_{\mathcal{M}}$	diagonal matrix of area elements associated to $V_{\mathcal{M}}$
$E_{\mathcal{M}}$	set of edges of $\mathcal{M}$
$F_{\mathcal{M}}$	set of faces of $\mathcal{M}$
Q	set of vertices scattered on the mesh
q	cardinality of Q
σ	amplitude of Gaussian functions
k	number of basis atoms
Φ, Ψ	(truncated) bases for the functional space of a mesh