ReMatching: Low-Resolution Representations for Scalable Shape Correspondence

F. Maggioli[2] and D. Baieri[1] and E. Rodolà[1] and S. Melzi[2]

[1]Sapienza - University of Rome, Italy
[2]Università di Milano Bicocca, Italy

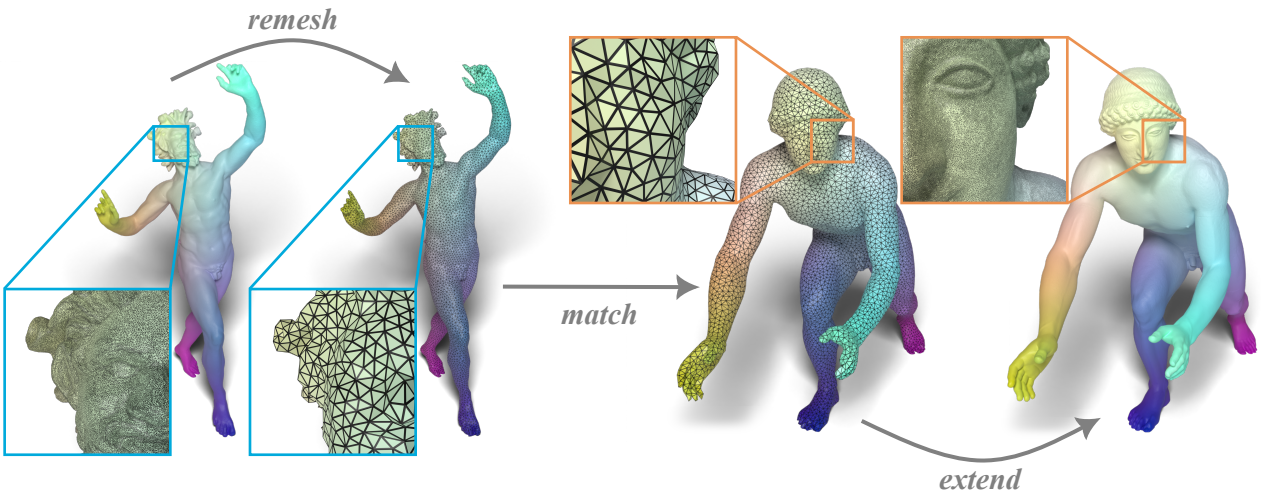

Figure 1: *Correspondence between two dense shapes computed with our method, using ZoomOut [MRR*19] on low-resolution meshes. The coordinates functions of the dancing faun statue on the left (~ 750k vertices) are transferred using a functional map to the Aphaea warrior statue on the right (~ 3.5M vertices). The colors encode the correspondence. Despite the mesh density, shown in the close-up, the computation took ~ 2 minutes.*

Abstract

We introduce ReMatching*, a novel shape correspondence solution based on the functional maps framework. Our method, by exploiting a new and appropriate* re*-meshing paradigm, can target shape-*matching *tasks even on meshes counting millions of vertices, where the original functional maps does not apply or requires a massive computational cost. The core of our procedure is a time-efficient remeshing algorithm which constructs a low-resolution geometry while acting conservatively on the original topology and metric. These properties allow translating the functional maps optimization problem on the resulting low-resolution representation, thus enabling efficient computation of correspondences with functional map approaches. Finally, we propose an efficient technique for extending the estimated correspondence to the original meshes. We show that our method is more efficient and effective through quantitative and qualitative comparisons, outperforming state-of-the-art pipelines in quality and computational cost.*

CCS Concepts

• ***Computing methodologies*** → ***Shape analysis;*** • ***Theory of computation*** → *Computational geometry;* • ***Mathematics of computing*** → *Functional analysis;*

1. Introduction and related work

The task of finding a semantically meaningful correspondence between discrete surfaces has always been a fundamental topic in the field of shape analysis. Researchers developed a wealth of solutions for this application, and new methods continue to be designed [DYDZ22,Sah20]. Among the various approaches, the functional maps framework [OBCS*12, OCB*16] received significant attention. Rather than finding a point-wise correspondence, the functional maps framework aims to define a correspondence between functions, encoding it into a small matrix. Following this direction, a variety of works tried to improve the actual computation of the map [DSO20, MRR*19, RPWO18] or to extend the framework to different types of bases [MRCB18, NMR*18, MMO*21].

Despite the amount of research outcomes based on the functional maps framework, both exploiting geometric properties and tools [ERGB16, BDK17, EBC17] and clever optimization techniques [NO17,RMO*20,RMWO21], the problem of computing the map is still bounded by the time complexity of sparse eigendecompositions [Kre06]. To overcome this problem, researchers started investigating scalable solutions to the Laplace-Beltrami eigenproblem. In this regard, multi-resolution techniques [VBCG10,NBH18, SVBC19,NH22] proved to be very effective for scalable computation of spectral quantities for tasks such as retrieval and mesh filtering [RWP06,LZ10], but their applicability to the functional map framework proved to require additional care [MO23]. Furthermore, recent research proposed spectral coarsening methods [LJO19] or spectral preserving simplification techniques [LLT*20]. However, these techniques usually rely on the computation of the full-resolution spectrum, making them unsuitable for large-scale applications. Gao *et al.* [GRE*23] proposed a mixed-integer programming scalable solution to the matching problem, but they specifically target sparse correspondences only. Alternatively, Marin *et al.* [MCPM24], who provide an approach for human registration that scales across large datasets, but it is unsuitable for dense shapes.

Closely related to our work is the contribution from Magnet *et al.* [MO23], which, for the first time, proposes a solution for scaling the functional maps approach to high-resolution meshes. Their idea is to reduce the dimensionality of the eigenproblem, inducing a linear relationship between the functional space on the mesh and a lower-dimension functional space on a sparse sample. The extreme sparsity of the sampling, combined with its efficient computation, leads to an algorithm that can effectively deal with high-density meshes. Nevertheless, this type of reduction does not mitigate the negative influence that small components and local details at high frequency could have on the alignment of the spectra.

Finally, our pipeline relies on building robust and sparse representations of dense meshes while still keeping an approximately bijective correspondence between the high- and low-resolution shapes, and it is worth mentioning that other works have already attempted to do so. Jiang *et al.* [JSZP20] propose building a lower resolution prismatic shell preserving bijectivity with the original surface. However, their algorithm does not scale to meshes with a high triangle count, gives no control and no guarantees on the final vertex count, and requires many assumptions on the input shape (*e.g.*, manifoldness, orientability, no self-intersections). On the other hand, Liu *et al.* [LGC*23] propose to exploit an intrinsic error metric for decimating a mesh to a given size while keeping track of a geodesic barycentric mapping of each triangle of the resulting shape to a geodesic triangle on the input surface. Despite the robustness of the method, the simplification approach makes it unsuitable for a functional maps setting, where it is required to decimate shapes from millions of vertices to a few thousand. This limitation is further discussed in Section 4.2.

We introduce a new scalable functional map pipeline that efficiently handles meshes with high vertex density (see Figure 1), yields stable results, and is not bounded by the quality of the original triangulation. To summarize:

- we translate the matching pipeline to low-resolution representations, enabling a fast and scalable computation of functional maps between dense shapes, providing the first alternative to [MO23];
- we propose a new efficient and geometry-preserving remeshing algorithm specifically designed for our pipeline;
- we exploit a fast solution for extending scalar maps from the low-resolution remeshed shape to the original surface, thus efficiently obtaining suitable bases for function transfer and shape matching.

2. Background

In the discrete setting, we represent a shape as a triangular mesh $\mathcal{M} = (V, E, T)$, where (i) $V \subset \mathbb{R}^3$ is a set of vertices; (ii) $E \subset V^2$ is a set of edges between vertices; (iii) $T \subset V^3$ is a set of triangles composing the surface.

2.1. Functional maps

We discretize a scalar function $f : \mathcal{M} \to \mathbb{R}$ as a signal defined on the vertices V and represented as a vector $f \in \mathbb{R}^{|V|}$. Hence, the Laplace-Beltrami operator $\Delta_{\mathcal{M}} : \mathcal{F}(\mathcal{M}) \to \mathcal{F}(\mathcal{M})$ is discretized as a sparse matrix $\mathbf{L}_{\mathcal{M}} \in \mathbb{R}^{|V|\times|V|}$, generally represented by means of a stiffness matrix $\mathbf{S}_{\mathcal{M}}$ and a mass matrix $\mathbf{A}_{\mathcal{M}}$ [PP93,MDSB03]. The eigendecomposition $\mathbf{S}_{\mathcal{M}}\Phi_{\mathcal{M}} = \mathbf{A}_{\mathcal{M}}\Phi_{\mathcal{M}}\Lambda_{\mathcal{M}}$ of the Laplacian yields an orthonormal basis for the functional space on the surface [Lev06,LZ10]. This basis has some analogies with the Fourier basis and is optimal to represent smooth functions when the basis is truncated [ABK15].

Given two shapes $\mathcal{M}$ and $\mathcal{N}$, respectively with m and n vertices, a point-to-point map $\Pi : \mathcal{M} \to \mathcal{N}$ can be expressed as a binary matrix $\Pi \in \{0,1\}^{m\times n}$ such that $\Pi(i,j) = 1$ if the vertex j-th of $\mathcal{N}$ corresponds to the vertex i-th of $\mathcal{M}$, and $\Pi(i,j) = 0$ otherwise. Any point-wise correspondence Π estabilishes a functional map $T_\Pi : \mathcal{F}(\mathcal{N}) \to \mathcal{F}(\mathcal{M})$ as $T_\Pi(f) = f \circ \Pi, \forall f \in \mathcal{F}(\mathcal{N})$.

As proposed in [OBCS*12], given Φ_k and Ψ_k the Laplacian bases truncated to size k respectively on $\mathcal{M}$ and $\mathcal{N}$, we can project T_Π in these bases and compactly encode the functional map in a $k \times k$ matrix $\mathbf{C} = \Phi_k^\top \mathbf{A}_{\mathcal{M}} \Pi \Psi_k$, which is the linear operator that transfers the coefficients of functions from $\mathcal{F}(\mathcal{N})$ in the ones of corresponding functions of $\mathcal{F}(\mathcal{M})$ respectively computed with Φ_k and Ψ_k.

Finally, given a functional map T_Π, or its compact representation $\mathbf{C}$, we can retrieve the correspondence Π on which the map is built by the nearest neighbor search in the embedding space of the Laplacian eigenfunctions as detailed in [OCB*16].

2.2. Intrinsic Delaunay triangulations (IDT)

The IDT [Riv94, BS07] has been introduced for generalizing the Delaunay triangulation [D*34] to non-Euclidean metric spaces where classical algorithms [Bow81, Wat81] cannot be used. IDT relies on the duality with the Voronoi diagram [ES94, LL00], as three texels meeting at a point form a triangle whose edges cross the texels boundaries.

Definition 1. *Let $(\mathcal{M}, d_\mathcal{M})$ be a 2-dimensional metric space (with distance function $d_\mathcal{M}$), and let $S = \{p_i\}_{i=1}^s \subset \mathcal{M}$ be a set of s sample points. The* Voronoi decomposition *(VD) of $\mathcal{M}$ with respect to S is the collection $\{P_i\}_{i=1}^s$ such that*

$$P_i = \left\{ q \in \mathcal{M} \ : \ p_i = \underset{p_j \in S}{\arg\min} \left(d_\mathcal{M}(p_j, q) \right) \right\}, \qquad \bigcup_{i=1}^{s} P_i = \mathcal{M} \tag{1}$$

The points in S are called generators*, and the P_i are called* texels *(or* cells*).*

The intersection between two or more texels could be non-empty. In such a case, we define those texels as *adjacent*. A VD is *general* if the intersection of four or more texels is empty. If a VD is general, a *Voronoi vertex* is a point that belongs to three texels, and a *Voronoi edge* is a curve C connecting two Voronoi vertices and belonging to two texels.

It can be shown that there is a one-to-one correspondence between a Voronoi edge dividing P_i and P_j and a Delaunay edge connecting p_i to p_j [DZM07]. This correspondence gives rise to a necessary and sufficient condition for the IDT to be a proper triangulation (*i.e.*, it realizes a simplicial complex).

Definition 2 ([ACDL00, ES94])**.** *If S induces a general VD, the VD satisfies the* closed ball property *if:*

- *every P_i is a closed 2-ball (*i.e.*, a topological disk without holes);*
- *every $P_i \cap P_j$ is either empty or a closed 1-ball (*i.e.*, a topological segment);*
- *every $P_i \cap P_j \cap P_k$ is either empty or a closed 0-ball (*i.e.*, a point).*

Theorem 1 ([ES94, ACDL00, DZM07])**.** *If S induces a general VD, then the VD satisfies the closed ball property if and only if its dual IDT is a proper triangulation.*

3. Method

We introduce a new scalable functional map pipeline handling very dense meshes and yielding stable results independently of the input triangulation's quality.

Our method acts through three main steps. At first, we compute two low-resolution meshes that preserve the original metrics of the input shapes (Section 3.1). Then, we apply any existing pipeline, yielding a functional map between the low-resolution eigenspaces (Section 3.2). Finally, we extend the eigenfunctions to the original meshes, allowing us to use them with the functional map estimated in the previous step to compute the desired correspondence between them (Section 3.3). Our technique is very general and works with arbitrary manifold triangulations, including meshes with degenerate geometry, non-orientable surfaces, and multiple connected components. The manifoldness requirement does not provide an obstacle to the applicability of our algorithm, as it can be efficiently enforced using existing techniques [FCS*13].

Figure 2: *The iterative front propagation from samples inducing a geodesic VD.*

3.1. Intrinsic Delaunay remeshing

Front propagation and FPS. Naively computing the VD induced by a set of samples can quickly become very expensive. The exact geodesic algorithm introduced by Mitchell *et al.* [MMP87] is computationally unfeasible on large meshes. Even using faster solutions, such as the heat method [CWW13], the fast marching algorithm [KS98], or even the Dijkstra distance [Dij59], searching the closest sample for each vertex is still a $\mathcal{O}(|V|\, s)$ operation. Instead, we take inspiration from the front propagation method proposed by Peyré *et al.* [PC06]. We start with a decomposition of the mesh induced by a single sample p_1. This means that every vertex v is part of the texel P_1 and has distance $D_1[v] = d_\mathcal{M}(p_1, v)$ from the sample set. We then assume to have a decomposition $P_1, \cdots, P_{k-1}$ induced by samples $p_1, \cdots, p_{k-1}$, and a vector D_{k-1} storing the distance of each vertex from the sample set. When adding a sample p_k to the VD, we can exploit the fact that $D_k = \min(D_{k-1}, d_\mathcal{M}(p_k, V))$. We start a front from p_k and expand it, updating the distances and assigning vertices to texel P_k until we reach all vertices such that $D_{k-1}[v] < d_\mathcal{M}(p_k, v)$ (see Figure 2). By propagating the front with the fast marching or Dijkstra's algorithm, we can compute a geodesic VD with time $\mathcal{O}(|V| \log(|V|) \log(s))$.

To obtain the sample points, Peyré *et al.* [PC06] propose to use a geodesic farthest point sampling, achieving a geodesic uniform sampling of the surface.

Definition 3. *Let $(\mathcal{M}, d_\mathcal{M})$ be a 2-dimensional metric space, and let $s \in \mathbb{N}$ be an integer. A* farthest point sampling (FPS) *of size s with respect to $d_\mathcal{M}$ is a set $S_s \subset \mathcal{M}$ of s sample points in $\mathcal{M}$, incrementally built from a singleton set $S_1 = \{p_1\} \subset \mathcal{M}$ according to the following rule:*

$$S_{\ell+1} = S_\ell \cup \left\{ \underset{q \in \mathcal{M}}{\arg\max} \ \underset{p \in S_\ell}{\min} \ d_\mathcal{M}(q, p) \right\}. \tag{2}$$

However, searching for the maximum distance in Equation (2) at every new sample brings again the algorithm to a $\mathcal{O}(|V|\, s)$ complexity, really becoming a bottleneck for large meshes counting millions of vertices. In contrast, we propose to introduce a fixed-size binary max-heap in the front propagation algorithm, keeping track of the distances from each vertex to the sample set. During

Algorithm 1 Geodesic FPS and its VD.

procedure VORONOIFPS($\mathcal{M} = (V,E,T)$, s)
  $i \leftarrow$ random index in $[1,|V|]$
  $S \leftarrow \{i\}$
  $D \leftarrow d_{\mathcal{M}}(V,V_i)$
  $H \leftarrow$ max-heap initialized with D
  $P \leftarrow$ vector of length $|V|$ initialized to i
  for $h \leftarrow 2$ **to** s **do**
    $p \leftarrow$ FINDMAX(H)
    SETKEY(H, p, 0)
    $D_p \leftarrow 0$
    $S \leftarrow S \cup \{p\}$
    Propagate a front from p.
    Update D, P, and H.
  end for
end procedure

each front propagation, this heap updates the distance of each visited vertex in time $\mathcal{O}(\log(|V|))$, adding no extra complexity to the visit. Furthermore, it can gather and set to zero the farthest vertex at each iteration in just $\mathcal{O}(\log(|V|))$ time, totalling a time complexity of $\mathcal{O}(|V|\log(|V|)\log(s))$ for the entire procedure. The fast front propagation algorithm is summarized in Algorithm 1.

Flat union property. Even if we are able to compute the VD of a farthest point sampling with high efficiency, we still cannot guarantee that the dual connectivity is a proper IDT. Ensuring the closed ball property can be computationally challenging. In particular, identifying the topology of the boundary between each pair of adjacent texels can become very costly when the number of samples is large. A possibility is to add enough samples so that the surface locally behaves like a plane. This approach, proposed by Leibon *et al.* [LL00] and adopted by Peyré *et al.* [PC06], makes it easy to lose control over the number of samples, and anyway the requirements are not easy to enforce. For guaranteeing a proper IDT, while still avoiding these issues, we introduce a novel property for a VD. The proofs of our claims are provided in Appendices A and B.

Definition 4 (Flat Union Property (FUP))**.** *Let $(\mathcal{M},d_{\mathcal{M}})$ be a 2-dimensional metric space. Let then $S = \{p_i\}_{i=1}^{s} \subset \mathcal{M}$ be a set of s sample points in $\mathcal{M}$, inducing a general VD $\{P_i\}_{i=1}^{s}$. The VD induced by S satisfies the* FUP *if:*

- *every P_i is a closed 2-ball;*
- *if $P_i \cap P_j$ is not empty, then $P_i \cup P_j$ is a closed 2-ball;*
- *if $P_i \cap P_j \cap P_k$ is not empty, then $P_i \cup P_j \cup P_k$ is a closed 2-ball.*

Theorem 2. *Let $(\mathcal{M},d_{\mathcal{M}})$ be a 2-dimensional metric space. If a general VD $\{P_i\}_{i=1}^{s}$ of $\mathcal{M}$ satisfies the* FUP*, it also satisfies the closed ball property.*

The FUP is much easier to verify than the closed ball property, as we only have to ensure that regions are closed 2-balls. Indeed, the following property can be exploited in this regard.

Proposition 1. *Let $\mathcal{M} = (V,E,F)$ be a manifold polygonal mesh, then $\mathcal{M}$ is a closed 2-ball if and only if its Euler characteristic $\chi = |V| - |E| + |F|$ is 1.*

Dual mesh representation. When we compute a VD of a triangle mesh, we want to partition the vertices into disjoint sets. However, Voronoi texels are defined as regions over the surface, meaning that they should be submanifolds of $\mathcal{M}$. By only considering vertices, we likely end up having non-manifold geometries or non well-defined boundaries.

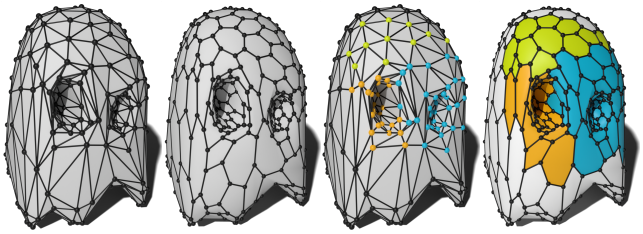

Figure 3: *Top: the primal and dual connectivities of a triangular mesh. Bottom: regions on the primal mesh are represented as sets of vertices, and the corresponding regions on the dual mesh are composed by faces.*

We consider the polygonal mesh $\tilde{\mathcal{M}} = (\tilde{V},\tilde{E},\tilde{T})$ to be the dual mesh of $\mathcal{M} = (V,E,T)$, built by placing a vertex at each face of $\mathcal{M}$ and forming a polygonal face for each triangle fan around a vertex of $\mathcal{M}$. As shown in the example from Figure 3, dual meshes allow us to define Voronoi texels by means of connected dual faces, making them actual submanifolds of the original surface. Furthermore, we can easily define boundaries between texels as paths made of dual edges.

This construction has the positive side effect to ensure that every VD is general. Indeed, the only place where three or more texels can meet is a dual vertex. Since texels are made of dual faces (*i.e.*, primal vertices), and the primal mesh is triangular, there is no dual vertex where more than three faces can meet.

Given the correspondence between primal and dual geometric elements, we can verify if a region is a closed 2-ball with Proposition 1 without explicitly constructing the dual mesh. Exploiting efficient data structures, like hash-maps and hash-sets, we can verify the topology of each texel (or union of two or more texels) with a single pass over vertices, edges and triangles. After the initial FPS, we further add samples to enforce the flat union property: if two or more adjacent texels do not form a closed 2-ball, we break the connection adding samples at the Voronoi vertex or along the Voronoi edge; if a texel is not a closed 2-ball, we add a sample along its boundary to reduce its coverage.

3.2. Low-resolution matching

Let $\mathcal{M}$ and $\mathcal{N}$ be two triangular meshes, and let $\hat{\mathcal{M}}$ and $\hat{\mathcal{N}}$ be their low-resolution counterparts computed with the algorithm presented above (or any other alternative algorithm). Furthermore, let $\hat{\Phi}$ be a basis for the functional space over $\hat{\mathcal{M}}$, and $\hat{\Psi}$ a basis for the functional space over $\hat{\mathcal{N}}$. For example these bases can be the truncated subset of the eigenvectors of the LBO as proposed in [OBCS*12]. We assume to have some pipeline yielding a functional map $\mathbf{C}$, such that

$$\hat{\Psi}\,\mathbf{C} \;=\; \hat{\Pi}\,\hat{\Phi}, \tag{3}$$

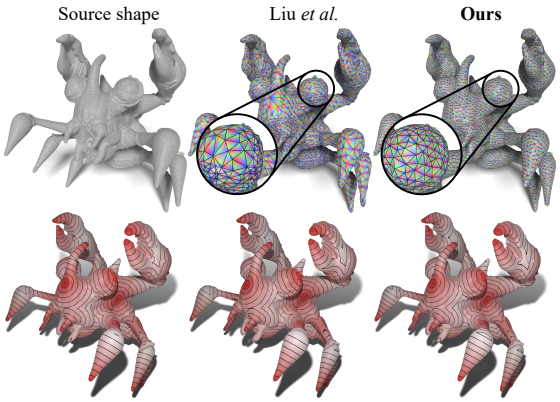

Figure 4: *Top: remeshing and mapping of the original vertices with the solution from Liu* et al. *[LGC*23] and our pipeline. Bottom: comparison of ground truth geodesic distances from multiple vertices and extended with both methods.*

where $\hat{\Pi} : \hat{\mathcal{M}} \to \hat{\mathcal{N}}$ is a correspondence between the vertices of $\hat{\mathcal{M}}$ and the vertices of $\hat{\mathcal{N}}$. The adopted functional maps solution can be any of the available alternatives, ranging from the original one [OBCS*12] to the most recent [DCMO22]. In our experiments, we consider two of the widely adopted solutions: constrained optimization with product preservation [NO17] and the iterative procedure ZoomOut [MRR*19]. The first is representative of the possible optimization strategies, while the second is probably the most efficient refinement technique for functional maps estimation. Both these methods have been efficiently and effectively applied on meshes with a limited number of vertices (*i.e.*, up to 10 thousands vertices), which is exactly the setting we are in after the proposed remeshing step.

3.3. Extending the correspondence

To extend the basis from the remeshing to the original surface, we need a mapping that transports scalar fields from the low-resolution mesh $\hat{\mathcal{M}}$ to the high-resolution shape $\mathcal{M}$. A possible solution would be to map each triangle of $\hat{\mathcal{M}}$ to a geodesic triangle in $\mathcal{M}$, and then computing the geodesic barycentric coordinates of each vertex inside the triangle [LGC*23]. Unfortunately, this approach is very costly, as it requires to compute exact geodesic paths onto $\mathcal{M}$ [Rus10]. Instead, we approximate this mapping by projecting each vertex of $\mathcal{M}$ onto the closest surface point of $\hat{\mathcal{M}}$. Despite this approach not being accurate in general, every triangle in $\hat{\mathcal{M}}$ corresponds to a geodesic triangle (homeomorphic to a disk) on $\mathcal{M}$ with the same vertices. Furthermore, starting from a farthest point sampling gives evenly spaced samples, mitigating the geometric complexity of geodesic triangles.

A surface point of $\hat{\mathcal{M}}$ is either a vertex, or a weighted average of two (if on an edge) or three vertices (if on a triangle). We build a linear map $\mathbf{U}_{\mathcal{M}} \in \mathbb{R}^{m \times s}$ representing the projection of each vertex $v_i \in \mathcal{M}$ as a linear combination of at most three vertices in $\hat{\mathcal{M}}$ (*i.e.*, the vertices encoding the closest surface point). Figure 4 shows a comparison between our pipeline and the simplification method proposed by Liu *et al.* [LGC*23]. The top row shows the different geometries produced by the two methods in reducing the mesh size by 90% and how our mapping compares to the approximate geodesic barycentric coordinates. In the bottom row we compare the two methods in approximating a multiple source geodesic distance field, showing that they do not present appreciable differences. More discussion is provided in Appendix E.

Scalar functions can be evaluated at any surface point by interpolating the values at the vertices. Since every vertex of $\mathcal{M}$ is associated to a point on the surface of $\hat{\mathcal{M}}$, we can use the linear map $\mathbf{U}_{\mathcal{M}}$ for extending scalar functions from the s vertices of $\hat{\mathcal{M}}$ to the m vertices of $\mathcal{M}$.

We extend the bases $\hat{\Phi}$ and $\hat{\Psi}$ to the full-resolution meshes as $\Phi = \mathbf{U}_{\mathcal{M}}\hat{\Phi}$ and $\Psi = \mathbf{U}_{\mathcal{N}}\hat{\Psi}$. Then, in a similar fashion as done in Equation (3), we search for a correspondence $\Pi : \mathcal{M} \to \mathcal{N}$ that maps vertices of $\mathcal{M}$ to vertices of $\mathcal{N}$ as

$$\mathbf{U}_{\mathcal{N}}\,\hat{\Psi}\,\mathbf{C} \;=\; \Psi\,\mathbf{C} \;=\; \Pi\,\Phi \;=\; \Pi\,\mathbf{U}_{\mathcal{M}}\,\hat{\Phi}, \tag{4}$$

where, for the sake of clarity, we explicitly write the equation in terms of the bases $\hat{\Phi}$ and $\hat{\Psi}$. Given the extended bases Φ and Ψ, and the same functional map $\mathbf{C}$ estimated to solve Equation (3), we obtain the correspondence Π via a nearest neighbor search in the spectral space.

We notice that nearest neighbor algorithms become very slow on large meshes, effectively becoming a possible bottleneck for the pipeline. Our experiments presented in Figure 10 show that, to the scope of this paper, this solution provides satisfying performance. However, we acknowledge that establishing a relationship between (3) and (4) could lead to a more efficient expression of Π in terms of $\hat{\Pi}$, $\mathbf{U}_{\mathcal{M}}$, and $\mathbf{U}_{\mathcal{N}}$. This avenue warrants future exploration.

4. Results

Our goal is to evaluate not only the matching accuracy of our method but also its time performance. For this task, we test it on datasets for dense correspondences, such as the SHREC19 dataset [MMR*19] and the TOSCA dataset [BBK08]. We also introduce a novel dataset, BadTOSCA, obtained by randomly altering the vertex positions of the TOSCA meshes, to ensure that our method is stable even under circumstances where strong isometry cannot be assumed†.

Our pipeline relies on an efficient remeshing algorithm that can quickly and drastically reduce the size of a mesh by orders of magnitude while still preserving the underlying geometry and metric. In principle, any remeshing algorithm could be adopted in our pipeline, as the map for extending scalar fields discussed in Section 3.3 only requires having two input meshes. We compare our solution against well-established isotropic and anisotropic remeshing algorithms (respectively, IRM [HDD*93, BPR*06] and ARM [NLG15]), as well as the scalable functional maps (SFM)

† Details on the dataset generation are provided in Appendix D.

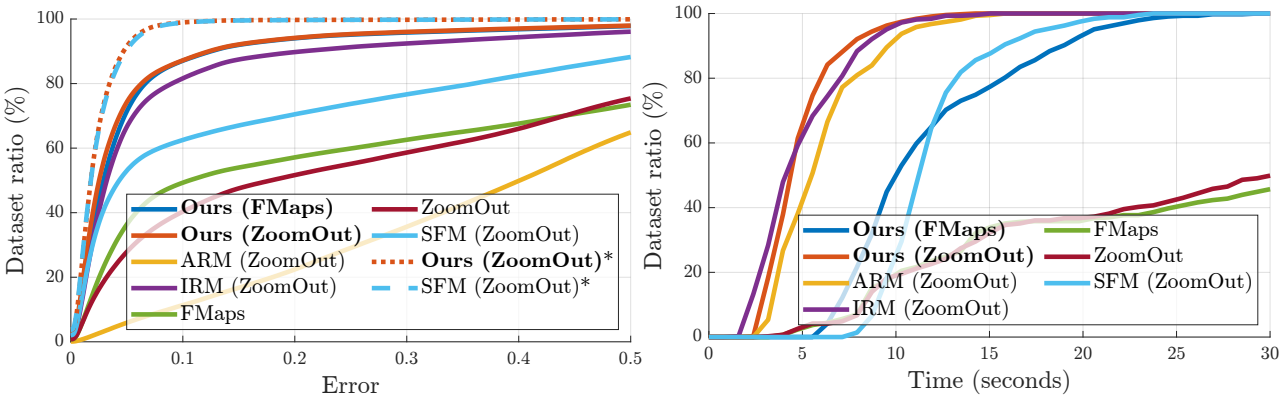

Figure 5: *Left: Accuracy curves on the SHREC19 challenge pairs for the tested methods. We also consider ideal approaches where the initial functional map for ZoomOut is given ground truth dashed lines). Right: Cumulative curves of the execution time for evaluated methods. For all methods, we also consider the time required to run the remeshing or resampling step(s).*

approach proposed by Magnet *et al.* [MO23], which produces a subsampling of the vertices in place of a low-resolution mesh.

We implemented our remeshing algorithm in C++, using Eigen [GJ*10] and libigl [JP*18], and the matching pipeline in MATLAB.

4.1. Quality of matching

To evaluate our shape matching pipeline, we tested our method on the SHREC19 dataset [MMR*19]. We can use different shape correspondence techniques to test the entire pipeline. In particular, we compute the functional map on the low-resolution representation using both the widely adopted approach with products preservation (FMaps) [NO17] and ZoomOut [MRR*19]. We compare our method with the same approaches on the full-resolution meshes, as well as against SFM. For a fair comparison, we remesh all the shapes to 3k vertices with all methods, as Magnet *et al.* [MO23] state that this produces the best performance with SFM. To further ease the alignment of the Laplacian spectra, we also post-process the remeshed shapes (produced with IRM, ARM, and our algorithm) to remove small disconnected components made of few triangles before computing the extension map discussed in Section 3.3. This post-processing step cannot be performed for SFM, due to the nature of its sampling strategy. The benefits of this post-processing will be discussed in Section 4.3.

We summarize the results of our experiment in Figure 5, where we show the accuracy curves for all methods on the SHREC19 connectivity track benchmark, following the paradigm proposed by Kim *et al.* [KLF11]. To prove the time efficiency of our pipeline, we also measure the time taken by every method on each pair of shapes in the benchmark (including the remeshing step), and we show the cumulative curves of the execution times over the entire dataset in a similar way with respect to the accuracy curves.

We see that all the low-resolution approaches provide better time and quality performance than the full-resolution methods, as the simpler geometry makes it easier and faster to align the Laplacian eigenfunctions. ARM is the only exception, since the anisotropic

Method	AGE ($\cdot 10^{-2}$) ↓	AUC ↑
Ours (ZoomOut)*	***2.40***	***95.12%***
*SFM**	*2.46*	*95.22%*
Ours (FMaps)	6.23	88.21%
Ours (ZoomOut)	**5.87**	**88.82%**
SFM	16.84	70.19%
ARM (ZoomOut)	41.59	30.34%
IRM (ZoomOut)	8.30	84.62%
FMaps	28.31	56.15%
ZoomOut	29.36	52.02%

Table 1: *Average geodesic error and area under the accuracy curve for each tested method on the SHREC19 challenge pairs. The top rows (denoted by $*$) show the performance of our algorithm and SFM under the assumption of having a ground truth functional map for initializing ZoomOut.*

remeshing produces many skew triangles and a singular Laplacian matrix for all the shapes. Furthermore, our technique (with both the FMaps and ZoomOut backend) outperforms SFM and achieves better results than using the IRM remeshing strategy. This difference can be better appreciated in Table 3, where we report the average geodesic error (normalized with respect to the shape diameter) and the area under the accuracy curves. In both Figure 5 and Table 3 we have also considered the idealized scenario where our method with ZoomOut as backend and SFM are initialized with a ground truth functional map. In this case, the two approaches obtain comparable results.

We also compare our method with the other approaches on the TOSCA and BadTOSCA datasets, summarizing the results in Figure 6. In the figure, we show the accuracy curves for the tested methods on both datasets, also providing the average geodesic error normalized by the shape diameter. We see that under the favorable and unrealistic conditions offered by TOSCA, where shapes of the same class are almost perfectly isometric, SFM achieves the best results. However, when we consider the slightly altered geometry

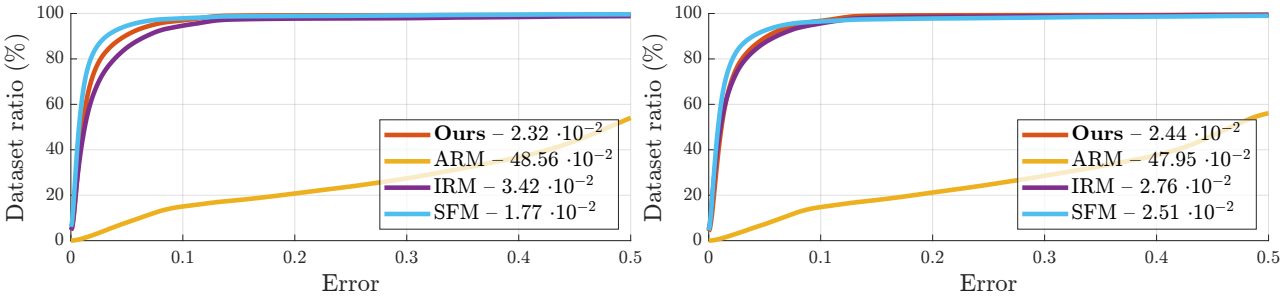

Figure 6: *Accuracy curves and average geodesic error on the TOSCA dataset (left) and the BadTOSCA dataset (right) for the tested methods.*

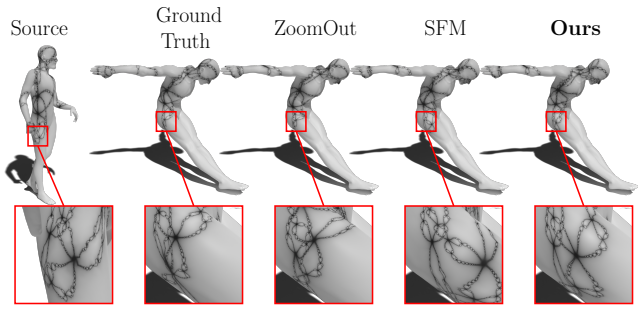

Figure 7: *Coordinate transfer between non-isometric shapes. The coordinates are used to generate highly complex patterns, whose sensitivity to the input enhances the transfer errors.*

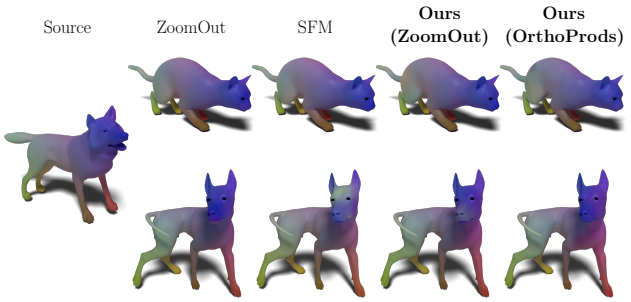

Figure 8: *Inter-class coordinate transfer from the wolf model to a cat and a dog. Our pipeline is not constrained by the standard Laplacian basis and can be used with other approaches as well.*

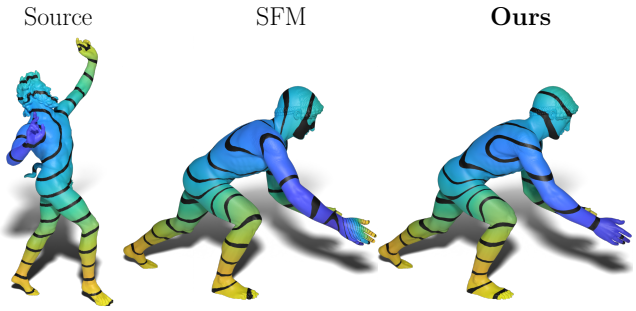

Figure 9: *Comparison of function transfer between very high-resolution models with SFM and our approach.*

of BadTOSCA, SFM suffers a performance drop of about 50%, while our method shows more stability.

A significant benefit of the functional map approach is its independence from the difference in resolution between the source and target shape. Transferring a function with functional maps yields a much smoother and neat result than directly a point-wise correspondence. In this regard, we test our method in transferring coordinate functions between isometric and non-isometric shapes with substantial differences in resolution and triangulation. In the example from Figure 7 we generated a complex procedural fractal pattern as a function of the coordinates [MBMR22]. This pattern is very sensitive to the input, so even slight errors are highly enhanced. We see that SFM cannot precisely map the coordinates of the source mesh, shifting around and distorting the details of the pattern.

Furthermore, the example in Figure 8 shows that we can also deal with inter-class function transfer with our pipeline, outperforming SFM in a visual comparison. In the example, we also showcase that we should not necessarily rely on the standard Laplacian basis as a backend. Instead of using ZoomOut to transfer functions between the low-resolution meshes, we rely on the orthogonalized eigenproducts basis introduced by Maggioli *et al.* [MMO*21], extending it with the same technique described in Section 3.2.

Finally, to ensure that our method can scale to high-resolution meshes, we tested it using two very high-resolution 3D scans of real statues: a dancing faun counting ∼750k vertices and a warrior from the temple of Aphaea counting ∼3.5M vertices (see Figures 1 and 9). Since SFM can also deal with dense shapes, we compare its results with the ones of our method. Figure 9 shows an example of the transfer of a geodesic distance function. While our method can faithfully transfer the function, SFM produces incoherent distances (*e.g.*, wrong isolines on the head and near the shoulder) and evident artefacts (*e.g.*, the wrong distance on the hand).

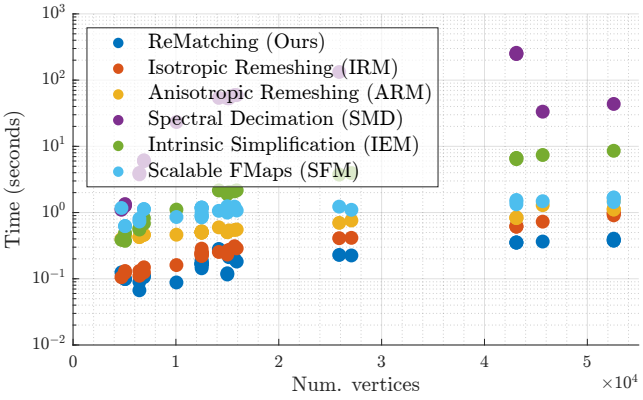

Figure 10: *Execution time (logarithmic scale on the y-axis) of different remeshing algorithms plotted against the number of vertices of the input shape (x-axis). For SFM we show the time taken for generating the sampling.*

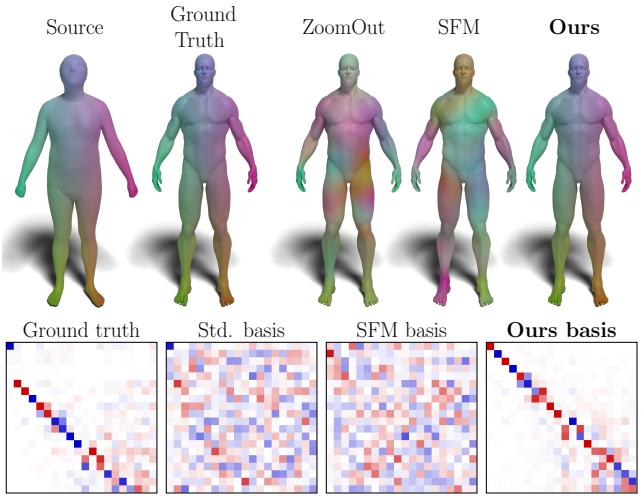

Figure 11: *Top row: coordinate transfer between a pair from SHREC19 using ZoomOut on full-resolution meshes, SFM and our approach. Bottom row: The ground truth* 20×20 *functional map, compared with the alignment computed with FMaps [NO17] from the full Laplacian eigenbasis, the SFM extended basis and our extended basis.*

4.2. Scalable performance

In Section 4.1 we saw that general-purpose remeshing solutions are not an optimal choice for the matching task. However, Lescoat *et al.* [LLT*20] provided a specialized mesh decimation algorithm (SMD) that reduces the complexity of the mesh while still preserving the Laplacian eigenfunctions. Furthermore, Liu *et al.* [LGC*23] proposed an elegant solution for simplifying a shape based on intrinsic error metric (IEM), which also builds a map for extending scalar functions using geodesic barycentric coordinates. We perform the evaluation on the SHREC19 dataset [MMR*19], remeshing the surfaces to 3k vertices with all the methods.

To achieve the desired scalability, the remeshing step in our pipeline must be fast enough to maintain the improvement obtained by reducing the size of the eigenproblem. In this regard, Figure 10 shows that our technique achieves better time performance than the other remeshing algorithms and the sampling strategy of SFM. In contrast, SMD and IEM are decimation algorithms that iteratively reduce the mesh size, making them orders of magnitude slower and unsuitable for large meshes. Moreover, SMD requires a precomputation of the Laplacian eigenbasis on the original surface to guide the decimation process and preserve the spectrum, introducing additional time complexity.

4.3. Benefits of remeshing

In Section 4.1, we discussed how we remove small disconnected components after the remeshing step. The example in Figure 11 shows a comparison of mapping between a low-resolution mesh (Source), with less than 7k vertices, to a detailed mesh where eyes are represented with two disconnected components, each composed of 462 vertices out of the 27k of the entire mesh (∼2% of the total vertices). This is enough geometry to attract energy at lower frequencies, and when we try to align the first 20 eigenfunctions with FMaps [NO17], the resulting functional map turns out to be mostly noise. Initializing ZoomOut with such a map leads to meaningless correspondence, and SFM does not mitigate this issue, as it tries to preserve the original spectrum as much as possible. In contrast, by removing these components and mapping them to their closest surface points, we introduce a small error in the final correspondence, but at the same time, we ease the alignment of the initial eigenbases, resulting in more accurate and meaningful mapping. This is evident in the bottom row of Figure 11, where the ground truth functional map presents a gap in the alignment of the eigenfunctions due to the impossibility of mapping the disconnected eyes of the target shape in any point of the source shape. Using the real eigenfunctions or a close approximation yields an intense noise in the map, producing the meaningless mapping of ZoomOut and SFM. Instead, by mapping the eyes to their nearest point on the head, we can meaningfully align our extended basis.

5. Conclusions

We presented a new pipeline for scalable shape matching that relies on a low-resolution representation to compute a functional map between dense shapes efficiently. A core part of this pipeline is our novel remeshing algorithm, which efficiently computes an intrinsic Delaunay triangulation of uniform surface sampling. This procedure is applied to reduce the computational cost of the functional maps framework and make it efficient also on very dense meshes. Furthermore, we use a fast and intuitive technique for extending scalar functions from low-resolution meshes to their dense counterparts, extending the computed correspondence to the original very high-resolution shapes. Our experimental evaluation proved that our procedure is effective on various types of shapes at different resolutions, outperforming other state-of-the-art solutions and

solving topological issues related to the alignment of the Laplacian eigenbasis.

While using a triangulated mesh as a low-resolution representation could, in principle, be exploited with other types of shape-matching pipelines, we only tested it within the functional maps framework, using the Laplace-Beltrami eigenbasis or bases derived from it. Studying the behaviour of our method with different "backends", as well as exploring the relationships between the low-resolution and the high-resolution correspondences, would be an interesting matter for future investigations.

References

[ABK15] AFLALO Y., BREZIS H., KIMMEL R.: On the optimality of shape and data representation in the spectral domain. *SIAM Journal on Imaging Sciences 8*, 2 (2015), 1141–1160. 2

[ACDL00] AMENTA N., CHOI S., DEY T. K., LEEKHA N.: A simple algorithm for homeomorphic surface reconstruction. In *Proceedings of the sixteenth annual symposium on Computational geometry* (2000), pp. 213–222. 3

[BBK08] BRONSTEIN A., BRONSTEIN M., KIMMEL R.: *Numerical Geometry of Non-Rigid Shapes*. Springer, New York, NY, 2008. 5, 11

[BDK17] BURGHARD O., DIECKMANN A., KLEIN R.: Embedding shapes with green's functions for global shape matching. *Computers & Graphics 68* (2017), 1–10. 2

[Bow81] BOWYER A.: Computing Dirichlet tessellations*. *The Computer Journal 24*, 2 (01 1981), 162–166. 3

[BPR*06] BOTSCH M., PAULY M., ROSSL C., BISCHOFF S., KOBBELT L.: Geometric modeling based on triangle meshes. In *ACM SIGGRAPH 2006 Courses*. 2006, pp. 1–es. 5

[BS07] BOBENKO A. I., SPRINGBORN B. A.: A discrete laplace–beltrami operator for simplicial surfaces. *Discrete & Computational Geometry 38*, 4 (2007), 740–756. 3

[CWW13] CRANE K., WEISCHEDEL C., WARDETZKY M.: Geodesics in heat: A new approach to computing distance based on heat flow. *ACM Transactions on Graphics (TOG) 32*, 5 (2013), 1–11. 3

[D*34] DELAUNAY B., ET AL.: Sur la sphere vide. *Izv. Akad. Nauk SSSR, Otdelenie Matematicheskii i Estestvennyka Nauk 7*, 793-800 (1934), 1–2. 3

[DCMO22] DONATI N., CORMAN E., MELZI S., OVSJANIKOV M.: Complex functional maps: A conformal link between tangent bundles. In *Computer Graphics Forum* (2022), vol. 41, Wiley Online Library, pp. 317–334. 5

[Dij59] DIJKSTRA E. W.: A note on two problems in connexion with graphs. *Numerische Mathematik 1* (1959), 269–271. 3

[DSO20] DONATI N., SHARMA A., OVSJANIKOV M.: Deep geometric functional maps: Robust feature learning for shape correspondence. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2020), pp. 8592–8601. 2

[DYDZ22] DENG B., YAO Y., DYKE R. M., ZHANG J.: A survey of non-rigid 3d registration. In *Computer Graphics Forum* (2022), vol. 41, Wiley Online Library, pp. 559–589. 2

[DZM07] DYER R., ZHANG H., MÖLLER T.: Voronoi-delaunay duality and delaunay meshes. In *Proceedings of the 2007 ACM symposium on Solid and physical modeling* (2007), pp. 415–420. 3

[EBC17] EZUZ D., BEN-CHEN M.: Deblurring and denoising of maps between shapes. In *Computer Graphics Forum* (2017), vol. 36, Wiley Online Library, pp. 165–174. 2

[ERGB16] EYNARD D., RODOLA E., GLASHOFF K., BRONSTEIN M. M.: Coupled functional maps. In *2016 Fourth International Conference on 3D Vision (3DV)* (2016), IEEE, pp. 399–407. 2

[ES94] EDELSBRUNNER H., SHAH N. R.: Triangulating topological spaces. In *Proceedings of the tenth annual symposium on Computational geometry* (1994), pp. 285–292. 3

[FCS*13] FEI Y., CHEN S., SU D., LUO J., LI M.: A new algorithm for repairing non-manifold surfaces. In *2013 IEEE 10th International Conference on High Performance Computing and Communications & 2013 IEEE International Conference on Embedded and Ubiquitous Computing* (2013), IEEE, pp. 1704–1708. 3

[GJ*10] GUENNEBAUD G., JACOB B., ET AL.: Eigen v3. http://eigen.tuxfamily.org, 2010. 6

[GRE*23] GAO M., ROETZER P., EISENBERGER M., LÄHNER Z., MOELLER M., CREMERS D., BERNARD F.: Sigma: Scale-invariant global sparse shape matching. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2023), pp. 645–654. 2

[HDD*93] HOPPE H., DEROSE T., DUCHAMP T., MCDONALD J., STUETZLE W.: Mesh optimization. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques* (1993), pp. 19–26. 5

[JP*18] JACOBSON A., PANOZZO D., ET AL.: libigl: A simple C++ geometry processing library, 2018. https://libigl.github.io/. 6

[JSZP20] JIANG Z., SCHNEIDER T., ZORIN D., PANOZZO D.: Bijective projection in a shell. *ACM Transactions on Graphics (TOG) 39*, 6 (2020), 1–18. 2

[KLF11] KIM V. G., LIPMAN Y., FUNKHOUSER T.: Blended intrinsic maps. *ACM transactions on graphics (TOG) 30*, 4 (2011), 1–12. 6

[Kre06] KRESSNER D.: *Numerical Methods for General and Structured Eigenvalue Problems*. Lecture Notes in Computational Science and Engineering. Springer Berlin Heidelberg, 2006. 2

[KS98] KIMMEL R., SETHIAN J. A.: Computing geodesic paths on manifolds. *Proceedings of the national academy of Sciences 95*, 15 (1998), 8431–8435. 3

[Lev06] LEVY B.: Laplace-beltrami eigenfunctions towards an algorithm that "understands" geometry. In *IEEE International Conference on Shape Modeling and Applications 2006 (SMI'06)* (2006), pp. 13–13. 2

[LGC*23] LIU H.-T. D., GILLESPIE M., CHISLETT B., SHARP N., JACOBSON A., CRANE K.: Surface simplification using intrinsic error metrics. *ACM Trans. Graph. 42*, 4 (jul 2023). URL: https://doi.org/10.1145/3592403, doi:10.1145/3592403. 2, 5, 8, 12, 13

[LJO19] LIU H.-T. D., JACOBSON A., OVSJANIKOV M.: Spectral coarsening of geometric operators. *ACM Transactions on Graphics (TOG) 38*, 4 (jul 2019). 2

[LL00] LEIBON G., LETSCHER D.: Delaunay triangulations and voronoi diagrams for riemannian manifolds. In *Proceedings of the sixteenth annual symposium on Computational geometry* (2000), pp. 341–349. 3, 4

[LLT*20] LESCOAT T., LIU H.-T. D., THIERY J.-M., JACOBSON A., BOUBEKEUR T., OVSJANIKOV M.: Spectral mesh simplification. In *Computer Graphics Forum* (2020), vol. 39, Wiley Online Library, pp. 315–324. 2, 8

[LZ10] LÉVY B., ZHANG H.: Spectral mesh processing. In *ACM SIGGRAPH 2010 Courses*. 2010, pp. 1–312. 2

[MBMR22] MAGGIOLI F., BAIERI D., MELZI S., RODOLÀ E.: Newton's fractals on surfaces via bicomplex algebra. In *ACM SIGGRAPH 2022 Posters*. 2022, pp. 1–2. 7

[MCPM24] MARIN R., CORONA E., PONS-MOLL G.: Nicp: Neural icp for 3d human registration at scale. In *European Conference on Computer Vision* (2024), Springer. 2

[MDSB03] MEYER M., DESBRUN M., SCHRÖDER P., BARR A. H.: Discrete Differential-Geometry Operators for Triangulated 2-Manifolds. In *Visualization and mathematics III*. Springer, New York, NY, 2003, pp. 35–57. 2

[MMO*21] MAGGIOLI F., MELZI S., OVSJANIKOV M., BRONSTEIN M. M., RODOLÀ E.: Orthogonalized fourier polynomials for signal approximation and transfer. In *Computer Graphics Forum* (2021), vol. 40, Wiley Online Library, pp. 435–447. 2, 7

[MMP87] MITCHELL J. S., MOUNT D. M., PAPADIMITRIOU C. H.: The discrete geodesic problem. *SIAM Journal on Computing 16*, 4 (1987), 647–668. 3

[MMR*19] MELZI S., MARIN R., RODOLÀ E., CASTELLANI U., REN J., POULENARD A., WONKA P., OVSJANIKOV M.: Shrec 2019: Matching humans with different connectivity. In *Eurographics Workshop on 3D Object Retrieval* (2019), vol. 7, The Eurographics Association, p. 3. 5, 6, 8

[MO23] MAGNET R., OVSJANIKOV M.: Scalable and efficient functional map computations on dense meshes. In *Computer Graphics Forum* (2023), vol. 42, Wiley Online Library, pp. 89–101. 2, 6

[MRCB18] MELZI S., RODOLÀ E., CASTELLANI U., BRONSTEIN M. M.: Localized manifold harmonics for spectral shape analysis. In *Computer Graphics Forum* (2018), vol. 37, Wiley Online Library, pp. 20–34. 2

[MRR*19] MELZI S., REN J., RODOLÀ E., SHARMA A., WONKA P., OVSJANIKOV M.: Zoomout: spectral upsampling for efficient shape correspondence. *ACM Transactions on Graphics (TOG) 38*, 6 (2019), 1–14. 1, 2, 5, 6

[NBH18] NASIKUN A., BRANDT C., HILDEBRANDT K.: Fast approximation of laplace-beltrami eigenproblems. In *Computer Graphics Forum* (2018), vol. 37, Wiley Online Library, pp. 121–134. 2

[NH22] NASIKUN A., HILDEBRANDT K.: The hierarchical subspace iteration method for laplace–beltrami eigenproblems. *ACM Transactions on Graphics (TOG) 41*, 2 (2022), 1–14. 2

[NLG15] NIVOLIERS V., LÉVY B., GEUZAINE C.: Anisotropic and feature sensitive triangular remeshing using normal lifting. *Journal of Computational and Applied Mathematics 289* (2015), 225–240. 5

[NMR*18] NOGNENG D., MELZI S., RODOLA E., CASTELLANI U., BRONSTEIN M., OVSJANIKOV M.: Improved functional mappings via product preservation. In *Computer Graphics Forum* (2018), vol. 37, Wiley Online Library, pp. 179–190. 2

[NO17] NOGNENG D., OVSJANIKOV M.: Informative descriptor preservation via commutativity for shape matching. In *Computer Graphics Forum* (2017), vol. 36, Wiley Online Library, pp. 259–267. 2, 5, 6, 8

[OBCS*12] OVSJANIKOV M., BEN-CHEN M., SOLOMON J., BUTSCHER A., GUIBAS L.: Functional maps: a flexible representation of maps between shapes. *ACM Transactions on Graphics (ToG) 31*, 4 (2012), 1–11. 2, 4, 5

[OCB*16] OVSJANIKOV M., CORMAN E., BRONSTEIN M., RODOLÀ E., BEN-CHEN M., GUIBAS L., CHAZAL F., BRONSTEIN A.: Computing and processing correspondences with functional maps. In *SIGGRAPH ASIA 2016 Courses*. 2016, pp. 1–60. 2, 3

[PC06] PEYRÉ G., COHEN L. D.: Geodesic remeshing using front propagation. *International Journal of Computer Vision 69* (2006), 145–156. 3, 4, 14

[PP93] PINKALL U., POLTHIER K.: Computing Discrete Minimal Surfaces and their Conjugates. *Experimental mathematics 2*, 1 (1993), 15–36. 2

[Riv94] RIVIN I.: Euclidean structures on simplicial surfaces and hyperbolic volume. *Annals of mathematics 139*, 3 (1994), 553–580. 3

[RMO*20] REN J., MELZI S., OVSJANIKOV M., WONKA P., ET AL.: Maptree: recovering multiple solutions in the space of maps. *ACM Transactions on Graphics (TOG) 39*, 6 (2020), 264–1. 2

[RMWO21] REN J., MELZI S., WONKA P., OVSJANIKOV M.: Discrete optimization for shape matching. In *Computer Graphics Forum* (2021), vol. 40, Wiley Online Library, pp. 81–96. 2

[RPWO18] REN J., POULENARD A., WONKA P., OVSJANIKOV M.: Continuous and orientation-preserving correspondences via functional maps. *ACM Transactions on Graphics (ToG) 37*, 6 (2018), 1–16. 2

[Rus10] RUSTAMOV R. M.: Barycentric coordinates on surfaces. In *Computer Graphics Forum* (2010), vol. 29, Wiley Online Library, pp. 1507–1516. 5

[RWP06] REUTER M., WOLTER F.-E., PEINECKE N.: Laplace–beltrami spectra as 'shape-dna'of surfaces and solids. *Computer-Aided Design 38*, 4 (2006), 342–366. 2

[Sah20] SAHILLIOĞLU Y.: Recent advances in shape correspondence. *The Visual Computer 36*, 8 (2020), 1705–1721. 2

[SVBC19] SHOHAM M., VAXMAN A., BEN-CHEN M.: Hierarchical functional maps between subdivision surfaces. In *Computer Graphics Forum* (2019), vol. 38, Wiley Online Library, pp. 55–73. 2

[VBCG10] VAXMAN A., BEN-CHEN M., GOTSMAN C.: A multi-resolution approach to heat kernels on discrete surfaces. In *ACM SIGGRAPH 2010 papers*. 2010, pp. 1–10. 2

[Wat81] WATSON D. F.: Computing the n-dimensional Delaunay tessellation with application to Voronoi polytopes*. *The Computer Journal 24*, 2 (01 1981), 167–172. 3

Appendix A: Proof of Theorem 2

Theorem 2. *Let $(\mathcal{M}, d_{\mathcal{M}})$ be a 2-dimensional metric space. If a general VD $\{P_i\}_{i=1}^{s}$ of $\mathcal{M}$ satisfies the* FUP*, it also satisfies the closed ball property.*

Proof. The first condition of the closed ball property trivially holds.

Suppose $P_i \cap P_j$ is not empty, but it is not a closed 1-ball. Since $P_i \cup P_j$ is topologically flat, if their boundary is formed by two or more connected components, none of these can be a closed loop. But if the boundary between P_i and P_j is formed by two or more topological segments, then $P_i \cup P_j$ must contain a hole, which contradicts the second condition of the flat union property. Conversely, if their boundary is formed by a single connected component which is not a topological segment, then the boundary must form a loop inside $P_i \cup P_j$. Such loop would either contain P_i or P_j, forming a hole in the other texel and violating the first condition of the flat union property.

Suppose $P_i \cap P_j \cap P_k$ is not empty, but it is not a closed 0-ball. Every connected component of this intersection cannot be more than zero dimensional, so P_i, P_j, and P_k must meet at more than one point. Since we proved that $P_i \cap P_j$ is a closed 1-ball, then $P_i \cap P_j \cap P_k$ must be formed by two points which are also the endpoints of $P_i \cap P_j$. A similar argument can be made for $P_j \cap P_k$ and $P_k \cap P_i$. Since $P_i \cup P_j \cup P_k$ is a closed 2-ball that contains three curves incident on the same two points, then two of these curves must form a closed loop containing the other one. Without loss of generality, suppose $P_i \cap P_j$ and $P_i \cap P_k$ form a closed loop. Since P_i is a closed 2-ball, then the region enclosed by this loop must be P_i. But this region also contains the boundary between P_j and P_k, which contradicts the fact that P_i, P_j, and P_k are Voronoi texels. Hence, $P_i \cap P_j \cap P_k$ must be a closed 0-ball. □

Appendix B: Proof of Proposition 1

Proposition 2. *Let $\mathcal{M} = (V, E, F)$ be a manifold polygonal mesh. The mesh $\mathcal{M}$ is a closed 2-ball if and only if its Euler characteristic $\chi = |V| - |E| + |F|$ is equal to 1.*

Proof. The Euler characteristic obeys to the equation $\chi = 2 - 2g - B$, where g is the genus of the surface and B is the number of boundary components. We also know that $\mathcal{M}$ is a closed 2-ball if and only if $g = 0$ and $B = 1$, thus if $\mathcal{M}$ is a closed 2-ball, we have $\chi = 1$.

Suppose $\chi = 2 - 2g - B = 1$, then $B = 1 - 2g$, where B, g are non-negative integers. If $g = 0$, then $B = 1$, and $\mathcal{M}$ is a closed 2-ball. Since $g > 0$ implies $B < 0$, there cannot be other valid solutions, and hence $\chi = 1$ implies that $\mathcal{M}$ is a closed 2-ball. □

Appendix C: Handling large triangles

In most cases, we wish to preserve any large planar faces appearing in the original mesh, as they represent large planar regions which should be preserved in the final triangulation. However, sometimes these kind of faces could introduce distorted triangles and degenerate angles, and thus it becomes preferable to introduce extra triangulation and removing them. For this task, we introduce a resampling strategy which can be performed as a preprocessing step.

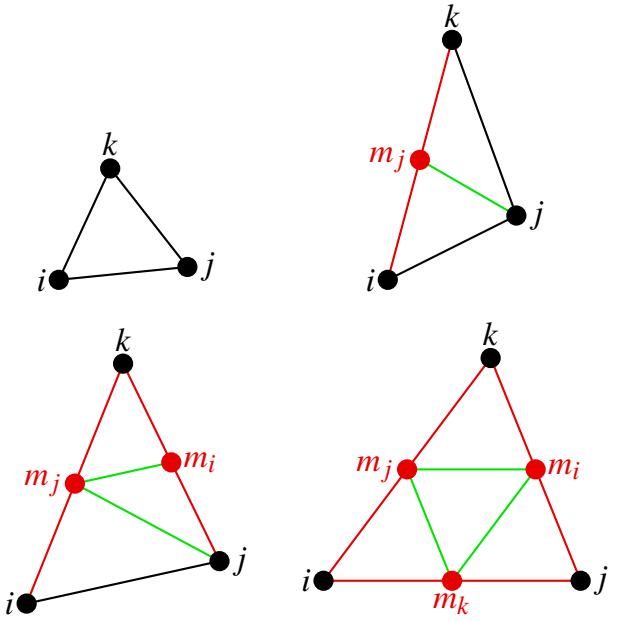

Figure 12: *Resampling scheme of the triangles, depending on the number of split edges. Red: long edges, to be split. Green: added connectivity.*

Statistically, triangular meshes have triangle count ∼2 times the number of vertices. Given the total area $A_{\mathcal{M}}$ of the original shape $\mathcal{M}$, we can estimate the average triangle area for the output mesh as $A_E \approx A_{\mathcal{M}}/2s$, with s the number of vertices of the remeshing. An equilateral triangle of area A_E has each side of length $\rho = \sqrt{2A_E/\sqrt{3}}$, so we split in half every edge with length greater than ρ and we insert a new vertex in the midpoint.

We then split mesh triangles depending on the number of incident split edges. If the triangle contains no split edges, we leave it as is. In case of a single split edge, we connect the midpoint to the opposite vertex, forming two triangles. When we have two split edges, we connect the midpoints to form a triangle and a quad. Then, to mitigate the formation of ill-shaped triangles, we divide the quad connecting the opposite vertices with the largest angle sum. Finally, if all its edges are split, we connect the midpoints to form four triangles. The process can then be iterated until no edges are oversize. The four cases for splitting a triangle are depicted in Figure 12.

Appendix D: BadTOSCA dataset

In the main manuscript, we introduced a new dataset for testing our algorithm, which we referred to as BadTOSCA. This dataset aims to introduce additional challenge to the well-known TOSCA dataset [BBK08] by altering the geometry and breaking the strong isometry between pairs of the same class. We recall the structure of the TOSCA dataset:

- the dataset is subdivided into k collections $C_1, \cdots, C_k$ of shapes;

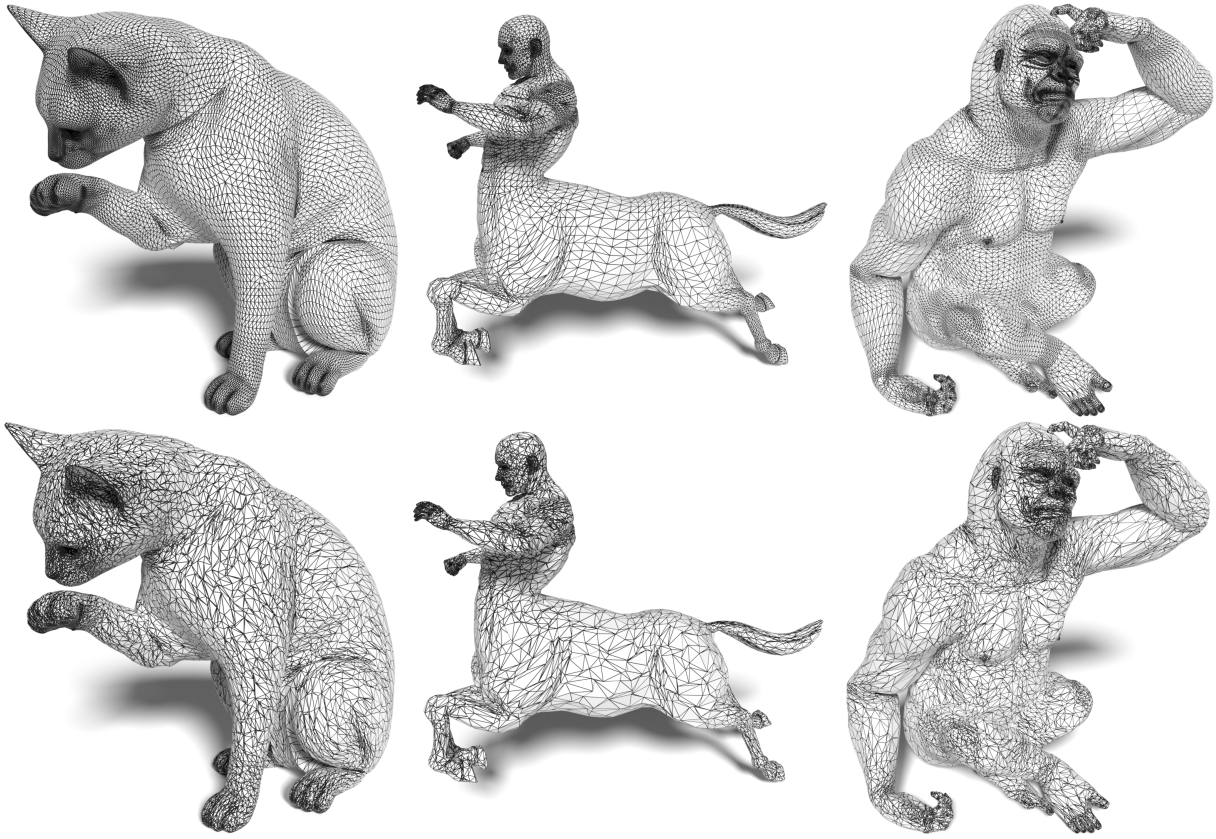

Figure 13: *Meshes from the TOSCA dataset and their altered counterparts in BadTOSCA.*

- each class C_i contains s triangular meshes $\mathcal{M}_{i_1}, \cdots, \mathcal{M}_{i_s}$ representing non-rigid deformations of the same shape;
- every two shapes $\mathcal{M}_p, \mathcal{M}_q$ belonging to the same class C_i are near perfectly isometric, have the same number of vertices, the same connectivity, and their correspondence Π is the identity matrix.

To generate the dataset, we first alter every shape independently from all the others. Given a shape $\mathcal{M} = (V, E, T)$, we generate an altered shape $\hat{\mathcal{M}} = (\hat{V}, E, T)$ by moving the position of the vertices, but preserving the connectivity. This ensures that the overall geometry is left unchanged, but at the same time that the strong isometry, the bijective correspondence, and the isomorphic connectivity cannot be exploited (see Figure 13 for a reference).

For each vertex v, we compute the set $N_T(v)$ of triangles incident on v, and we select a random triangle $t \in N_T(v)$. We then compute random barycentric coordinates $\lambda(v) = (\lambda_1(v), \lambda_2(v), \lambda_3(v))$, and place v on triangle t at $\lambda(v)$. Said $\mathbf{V} \in \mathbb{R}^{|V| \times 3}$ the matrix of the vertex positions of $\mathcal{M}$ and $\hat{\mathbf{V}} \in \mathbb{R}^{|\hat{V}| \times 3}$ the matrix of vertex positions of $\hat{\mathcal{M}}$, we can then use the barycentric coordinates of the altered vertices to build a sparse matrix $\mathbf{U} \in \mathbb{R}^{|\hat{V}| \times |V|}$ such that $\hat{\mathbf{V}} = \mathbf{U}\mathbf{V}$.

Given two meshes $\mathcal{M}_i = (V_i, E, T), \mathcal{M}_j = (V_j, E, T)$ belonging to the same class, let $\hat{\mathcal{M}}_i = (\hat{V}_i, E, T)$ and $\hat{\mathcal{M}}_j = (\hat{V}_j, E, T)$ be the corresponding altered meshes, and let $\mathbf{U}_i$ and $\mathbf{U}_j$ be the mapping of the vertices. By construction, we know that $\hat{\mathbf{V}}_i = \mathbf{U}_i \mathbf{V}_i$. Furthermore, since $\mathcal{M}_i$ and $\mathcal{M}_j$ are near perfectly isometric, the product $\mathbf{U}_j \mathbf{V}_i$ gives us the same type of altering that generated $\hat{\mathcal{M}}_j$ from $\mathcal{M}_j$, but onto the geometry of $\mathcal{M}_i$. Thus, for building the mapping Π_{ij} between $\hat{\mathcal{M}}_i$ and $\hat{\mathcal{M}}_j$ we apply a nearest neighbor search between $\mathbf{U}_i \mathbf{V}_i$ and $\mathbf{U}_j \mathbf{V}_i$.

Appendix E: Closest surface point mapping

Despite the simplicity of the method, projecting onto the closest surface point and using barycentric coordinates to interpolate values onto the original surface proves to be a valid solution, as shown by the results on the main manuscript.

We highlight that our method can compete with the intrinsic simplification method proposed by Liu *et al.* [LGC*23] also in other settings. Figure 14 shows an example of how geodesic distances can be computed in the low-resolution surface (10k vertices for both methods, shown in the first column) and then extended to the high-resolution mesh (∼120k vertices). For this experiment, we sample $k = 30$ source points $p_1, \ldots, p_k$ from the surface and compute the exact geodesic distances $d_{\text{gt}}(p_i, v)$ from each source p_i to every vertex v of the surface, also considering the minimum distance from the sample set $\min_{p_i}(d_{\text{gt}}(p_i, v))$ (last column). We then do the same for the remeshed surface, using as source the

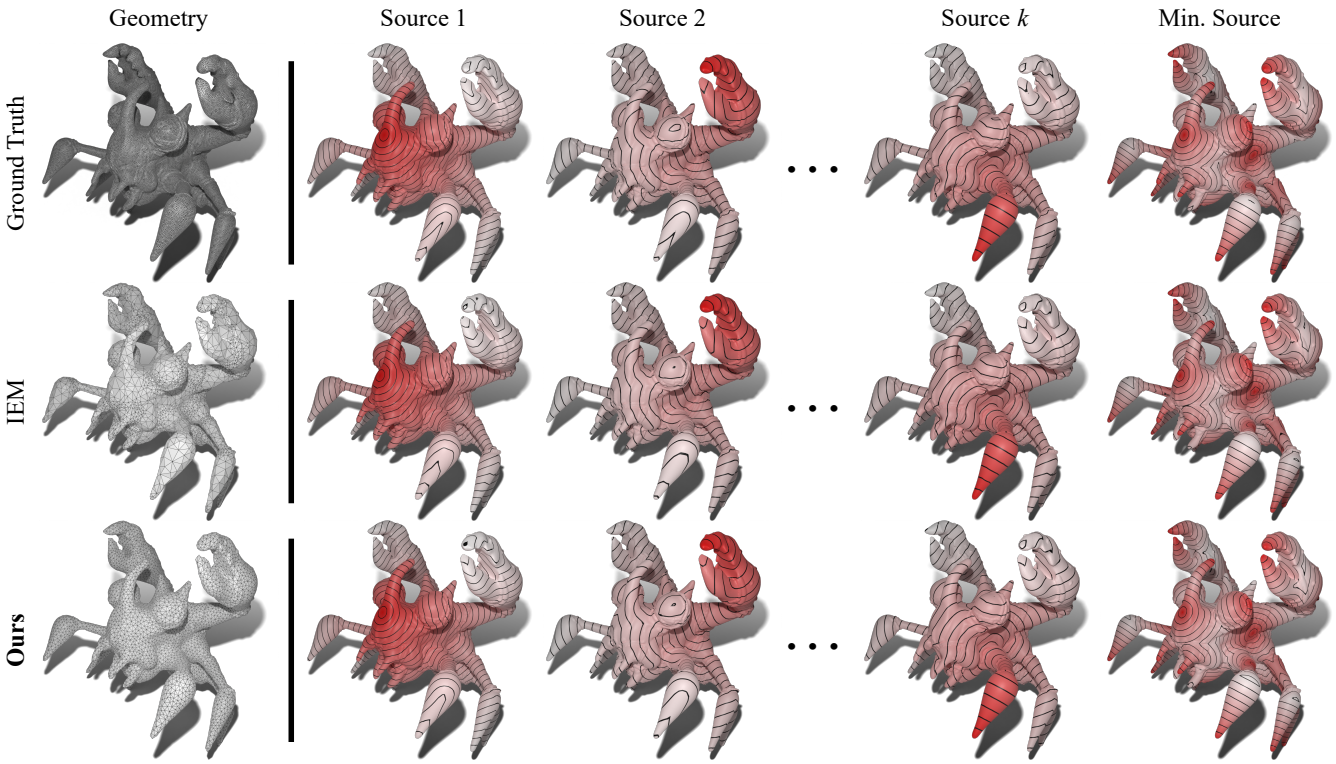

Figure 14: *Comparison between our method (last row) and IEM [LGC*23] (second row) extending geodesic distances to the high-resolution surface. The top row shows the ground truth geodesics.*

Point	Ours	IEM	Point	Ours	IEM
p_1	1.81	1.80	p_{16}	1.50	0.56
p_2	1.54	0.72	p_{17}	1.96	0.51
p_3	1.62	0.79	p_{18}	1.29	0.46
p_4	1.04	0.48	p_{19}	1.79	1.54
p_5	1.05	0.59	p_{20}	1.91	2.32
p_6	1.44	0.70	p_{21}	1.94	0.44
p_7	0.95	0.39	p_{22}	1.72	0.79
p_8	1.35	0.88	p_{23}	1.13	0.61
p_9	1.28	0.49	p_{24}	1.08	0.52
p_{10}	1.04	0.60	p_{25}	1.03	1.48
p_{11}	1.25	1.04	p_{26}	1.05	1.61
p_{12}	2.37	0.77	p_{27}	0.74	0.73
p_{13}	1.93	1.11	p_{28}	1.24	2.31
p_{14}	1.10	0.80	p_{29}	1.98	0.57
p_{15}	0.84	0.49	p_{30}	1.74	1.95
			Min. dist	2.61	3.86

Table 2: *Normalized error of approximated geodesics from each source point for both our method and IEM.*

points closest to the original samples, and extending the distance functions to the full-resolution shape using the barycentric coordinates of the closest surface point in our case, and the approximated geodesic barycentric coordinates for IEM, respectively producing approximated geodesic distances $d_{\text{ours}}(p_i, v)$ and $d_{\text{IEM}}(p_i, v)$.

For evaluating the error, for each p_i we compute the differences

$$e_{\text{ours}}(p_i, v) = \frac{d_{\text{gt}}(p_i, v) - d_{\text{ours}}(p_i, v)}{\max_v(d_{\text{gt}}(p_i, v))}, \qquad e_{\text{intrinsic}}(p_i, v) = \frac{d_{\text{gt}}(p_i, v) - d_{\text{intrinsic}}(p_i, v)}{\max_v(d_{\text{gt}}(p_i, v))}, \tag{5}$$

which are normalized by maximum distance to p_i, to ensure that each function acts at the same scale. Then, we aggregate the error by computing the norms $\|e_{\text{ours}}(p_i, \cdot)\|_{\mathcal{M}}$ and $\|e_{\text{intrinsic}}(p_i, \cdot)\|_{\mathcal{M}}$, where $\|f\|^2_{\mathcal{M}} = \int_{\mathcal{M}} f^2(x)\, \mathrm{d}x$.

In contrast to our method, IEM produces a mapping that approximates geodesic barycentric coordinates during the simplification. However, the geodesic paths between adjacent vertices in the remeshed surface are deformed to straight lines, negatively affecting the approximation of the full-resolution geodesics and producing results that are comparable to ours. The results from Table 2 shows that, while IEM obtains better results most of the time, the error is on the same scale as our solution. Furthermore, while IEM took 25.796 seconds for remeshing the surface and producing the mapping, our algorithm remeshed the surface in 1.109 seconds and produced the map in 638 milliseconds, totalling 1.747 seconds and achieving a 14.75 speedup.

Method	AGE ($\cdot 10^{-2}$) ↓	AUC ↑
Ours (ZoomOut)*	***2.40***	***95.12%***
*SFM**	*2.46*	*95.22%*
Ours (FMaps)	6.23	88.21%
Ours (ZoomOut)	**5.87**	**88.82%**
FP (FMaps)	9.11	83.19%
FP (ZoomOut)	7.90	85.29%
SFM	16.84	70.19%
ARM (ZoomOut)	41.59	30.34%
IRM (ZoomOut)	8.30	84.62%
FMaps	28.31	56.15%
ZoomOut	29.36	52.02%

Table 3: *Average geodesic error and area under the accuracy curve for each tested method on the SHREC19 challenge pairs. The top rows (denoted by $*$) show the performance of our algorithm and SFM under the assumption of having a ground truth functional map for initializing ZoomOut.*

Appendix F: Flat Union Property

Dual mesh representation algorithm

As discussed in the main text, enforcing the Closed Ball Property is more challenging than enforcing the Flat Union Property. Indeed, the Flat Union Property only requires to check that certain regions are topologically equivalent to 2-dimensional disks without holes, which can be easily verified by exploiting Proposition 2. The detailed procedure is given in Algorithm 2.

Comparison with Front Propagation

The refinement step presented in Algorithm 2 is required on top of any sampling to ensure the resulting Voronoi partitioning is the dual of a proper Intrinsic Delaunay Triangulation. In our method, we employ the Front Propagation algorithm (FP) introduced by Peyré *et al.* [PC06] to obtain an initial sampling, slightly modified for improving the computational complexity. As discussed in the original paper, the FP algorithm does not provide any guarantee of manifoldness and equivalent topology as the input. In a functional maps setting, this undesirable feature can introduce noise and error in computing the correspondence. For supporting our claim, we present in Table 3 another version of Table 1 from the main manuscript, where two additional rows report the performance of the FP algorithm on the SHREC19 dataset. While proving its effectiveness in the task, the FP algorithm achieves worse results that our method using both FMaps and ZoomOut. By adding the extra refinement step and guaranteeing manifoldness and topological equivalence, we can reduce the Average Geodesic Error (AGE) by about 30%

Algorithm 2 Flat Union Property verification.

procedure FUPCHECK($\mathcal{M} = (V, E, T), P$)
  // $R_1[i]$ is the i-th Voronoi region
  $R_1 \leftarrow$ map of Voronoi regions
  // $R_2[i,j] = R_1[i] \cup R_1[j]$, $R_3[i,j,k] = R_1[i] \cup R_1[j] \cup R_1[k]$
  $R_2 \leftarrow$ map of pairs of adjacent Voronoi regions
  $R_3 \leftarrow$ map of triplets of adjacent Voronoi regions
  // $R[i]$ contains elements of R_1, R_2, R_3 containing $R_1[i]$
  $R \leftarrow$ map of containing regions
  // Iterate over triangles to build regions and adjacencies
  for $t = (t_1, t_2, t_3) \in T$ **do**
    for vertex $v \in t$ **do**
      $R_1[P[v]] \leftarrow$ empty region
      $R[P[v]]$.`insert`$(R_1[P[v]])$
    end for
    for edge $e = (e_1, e_2) \in t$ **do**
      $R_2[P[e_1], P[e_2]] \leftarrow$ empty region
      $R[P[e_1]]$.`insert`$(R_2[P[e_1], P[e_2]])$
      $R[P[e_2]]$.`insert`$(R_2[P[e_1], P[e_2]])$
    end for
    $R_3[P[t_1], P[t_2], P[t_3]] \leftarrow$ empty region
    $R[P[t_1]]$.`insert`$(R_3[P[t_1], P[t_2], P[t_3]])$
    $R[P[t_2]]$.`insert`$(R_3[P[t_1], P[t_2], P[t_3]])$
    $R[P[t_3]]$.`insert`$(R_3[P[t_1], P[t_2], P[t_3]])$
  end for
  // Each vertex corresponds to a dual face in the regions
  for $v \in V$ **do**
    for $r \in R[P[v]]$ **do**
      r.`add_face()`
    end for
  end for
  // Each edge corresponds to a dual edge in the regions
  for $e \in E$ **do**
    for distinct vertex $v \in e$ **do**
      for $r \in R[P[v]]$ **do**
        r.`add_edge()`
      end for
    end for
  end for
  // Each triangle corresponds to a dual vertex in the regions
  for $t \in T$ **do**
    for distinct vertex $v \in t$ **do**
      for $r \in R[P[v]]$ **do**
        r.`add_vertex()`
      end for
    end for
  end for
  // Exploit Proposition 2
  for $r \in R_1 \cup R_2 \cup R_3$ **do**
    $\xi \leftarrow$ r.`num_vertices()` $-$ r.`num_edges()` $+$ r.`num_faces()`
    if $\xi = 1$ **then**
      Mark r as closed 2-ball
    end if
  end for
end procedure