DOI: 10.1111/cgf.13789
Eurographics Symposium on Geometry Processing 2019
D. Bommes and H. Huang
(Guest Editors)

Volume 38 (2019), Number 5

Hierarchical Functional Maps between Subdivision Surfaces

Meged Shoham[1] Amir Vaxman[2] Mirela Ben-Chen[1]

[1]Technion - Israel Institute of Technology
[2]Utrecht University

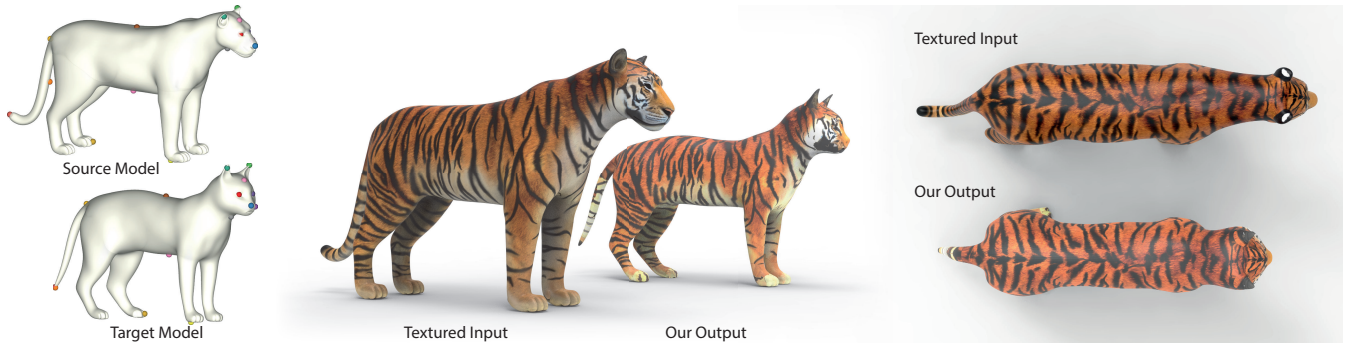

Figure 1: *We compute a detailed map between subdivision surfaces given by their control meshes using a set of input landmarks (left), and use it to accurately transport highly detailed texture images (center, right).*

Abstract

We propose a novel approach for computing correspondences between subdivision surfaces with different control polygons. Our main observation is that the multi-resolution spectral basis functions that are often used for computing a functional *correspondence can be compactly represented on subdivision surfaces, and therefore can be efficiently computed. Furthermore, the reconstruction of a pointwise map from a functional correspondence also greatly benefits from the subdivision structure. Leveraging these observations, we suggest a* hierarchical *pipeline for functional map inference, allowing us to compute correspondences between surfaces at fine subdivision levels, with hundreds of thousands of polygons, an order of magnitude faster than existing correspondence methods. We demonstrate the applicability of our results by transferring high-resolution sculpting displacement maps and textures between subdivision models.*

CCS Concepts

• ***Computing methodologies*** → ***Mesh models;***

1. Introduction

Subdivision surfaces are a popular shape representation for 3D modeling, used in the design pipeline of many artists [LVGL*13]. A common workflow [vG09]pp. 101 entails designing a polygonal, often a purely quadrangular model with a small number of polygons, subdividing it multiple times to achieve higher smoothness, and then sculpting fine details on the subdivided model. If the model is to be used in a low-resource environment, such as a game or an augmented-reality application, the geometric details are then "baked" into an image, and only the low resolution geometry is used at runtime. The details are rendered as normal maps or bump maps, and more recently, by using hardware tessellation, as displacement maps [NKF*16].

The detailing process, i.e., designing a realistic 3D model starting from a low resolution polygonal *base mesh*, is time consuming and expensive. This is evidenced, for example, by the price differences between a base mesh and a detailed model, that can reach two orders of magnitude [Tur19]. It is natural then to consider *reusing* the detailing of one model as a starting point for the detailing of a similar model. For example, if one designs a family of digital characters with similar facial details, it would be useful to design one such model, and then transfer the detailed editing to other base

meshes. Similar paradigms are often used in computer graphics, for example, deformation transfer [SP04] and style transfer [BHS*17].

To enable such an application, a detailed *correspondence* is required between two subdivision surfaces described by different control polygons. Despite a very large body of work dedicated to computing correspondences between triangle meshes [VKZHCO11], there exists, to the best of our knowledge, no method that is applicable to subdivision surfaces. Attempting to compute the correspondence on the subdivided mesh at a high resolution leads to extremely long run times, as the meshes reach hundreds of thousands of polygons. Alternatively, computing a correspondence on a low resolution mesh and subdividing it, is similarly ineffective, since the *semantics* of the geometry is not, in general, conserved by the subdivision operation. In other words, a semantically meaningful high-resolution map, that puts in correspondence related features on both shapes, is not necessarily the exact refinement of a semantic low-resolution map.

We propose a novel approach for computing a detailed, high resolution correspondence between two subdivision surfaces given by different control polygons. Our method is a generalization of the *functional map* framework [OBCS*12, OCB*17], which is a flexible approach for inferring correspondences, agnostic to the underlying geometry representation. The main required components are a *basis* for scalar functions defined on the surface, and a set of linear functional *constraints*, where both are often given in terms of the spectral decomposition of the *Laplace-Beltrami operator* (LB). Recently, a novel approach, denoted as Subdivision Exterior Calculus [dGDMD16], uses the subdivision structure to compute an accurate discretization of the LB operator on polygonal meshes. This discretization is a key component in our hierarchical approach.

It is well known, see e.g. [VL08], that the eigenfunctions of the LB operator have a *multi-resolution* nature, where functions with higher eigenvalues are more oscillatory than functions with lower eigenvalues. A similar property holds for an often used functional *descriptor*, the Heat Kernel Signature [SOG09] (HKS). This implies a perfect fit between subdivision surfaces and a *hierarchical* functional framework: the correspondence at a low resolution can be represented using a small subset of low eigenfunctions, and as the mesh resolution increases more basis eigenfunctions and descriptors can be computed and used.

We design the components for constructing efficiently a hierarchical functional map inference scheme. These include computing a hierarchical spectral basis, posing linear constraints and hierarchically optimizing for a functional correspondence. We additionally show how to leverage the subdivision structure to speed up the reconstruction of a pointwise map from the functional correspondence, an important and time consuming step. Our scheme computes high quality correspondences between subdivided meshes of hundreds of thousands of polygons, at computation times that are an order of magnitude smaller than existing correspondence approaches. We apply our detailed computed maps for transferring high resolution geometry edits, as well as texture images, showing the potential of our approach for 3D modeling applications.

1.1. Related Work

Our main goal is computing a correspondence between two subdivision surfaces, given by their base polygonal meshes. Despite the abundant amount of work on shape correspondence, to the best of our knowledge, there does not currently exist an algorithm that targets this application. Therefore, we focus our literature review on subdivision surfaces, on correspondence in general, on existing approaches for multi-resolution geometry processing, and on methods targeting our application, namely detail transfer.

Subdivision Surfaces. Subdivision surfaces are a widely used tool for animation, rendering and modeling of smooth surfaces [DKT98, ZS00, WW01, LJG14], by recursively refining a control base mesh with subdivision rules such as Catmull-Clarck [CC78] and Loop [Loo87], to name just a few. They are also used for simulation of fluids [Sta03], surface deformation [GKS02, TWS06] and architectural geometry [LPW*06].

Shape Correspondence. The literature on shape correspondence is vast, and a complete review is beyond our scope. We refer the reader to recent state-of-the-art reviews [TCL*13, LI15, Lag18] for an introduction to the topic. Most, if not all, of the shape correspondence approaches use triangle meshes or point clouds as input data, which is motivated by the need to register scanned 3D data. We, on the contrast, are interested in matching models designed by artists, which are given as polygonal (often quad) meshes.

To the best of our knowledge, there exist a very small number of papers that address the problem of correspondences between quad meshes. Eppstein et al. [EGKT08] have investigated the exact topological matching of parts of quadrangular meshes. They show that an exact solution is NP-Hard and provide an approximate greedy approach. Our goal is different, as we do allow varying quad topology, and rely on the geometry instead to supply the correspondence information. Alternatively, subdivision surface *fitting* can also generally be considered a correspondence method. Classic approaches for subdivision fitting were suggested by Litke et al. [LLS01] for Catmull-Clark subdivision, by Marinov et al. [MK05] for Loop subdivision, and many other, more recent, fitting approaches exist. It is worth noting that for subdivision fitting the base mesh is initially *extrinsically* aligned with a target triangular mesh, whereas in our case the input base meshes are general, and can be extrinsically and intrinsically different. Recently, Estellers et al. [ESC18b] suggested a robust fitting approach that takes into consideration outliers. They use a decimated version of the input mesh as the base mesh for the subdivision surface, which is extrinsically aligned to the input mesh and thus inappropriate for our application.

While it is possible to triangulate any polygonal mesh, the resulting triangle meshes will have non-optimal elements, which might degrade the differential operators that are used in computations. Alternatively, it is possible to *remesh* a quadrangular mesh using uniform triangular elements, however, that might lead to loss of prominent features if the remeshing is too coarse. Furthermore, the triangle meshes have to be very fine to enable the transport of highly detailed edits or texture. Hence, mapping approaches that are designed for triangle meshes [AL16] can potentially be used by remeshing the input quads to a very fine refinement of the subdivi-

sion surface, however, this leads to computation times which are an order of magnitude larger than ours, see Figure 13.

Functional Correspondence. The functional map framework [OBCS*12] is a general approach for computing correspondences, which is agnostic to the underlying geometric representation. As it relies on a reduced basis for scalar functions, it can be applied to any shape representation where such a basis can be computed, e.g., point clouds [HCO18]. We are not aware of an existing work that uses functional maps for mapping between quadrangular meshes. The framework has been used for *computing* approximately consistent quadrangulations of triangle meshes [ACBCO17], yet there the functional map was given as input. Recently, an interactive approach to map computation has been introduced [GBKS18], where a functional map is quickly computed using user-placed curves. To transfer texture, the authors extract a point-to-point map in a post-processing slower step, which is not interactive. Our approach, in contrast, leverages the subdivision structure to efficiently compute *both* the functional map and the point-to-point map for meshes refined to hundreds of thousands of polygons, albeit not at interactive rates. Other recent functional map regularizations, constraints and priors [ERGB16, VLB*17, NO17, RPWO18] are complementary to our method, as they can be applied at the coarsest level instead of the basic functional map method that we used. In [NMR*18] the authors suggested an approach to transfer high frequency functions and improve existing functional maps via product preservation. Since this step comes on top of an existing functional map it is complementary to our approach, and may serve as an additional improvement. Finally, the point-to-point reconstruction step has been addressed as a separate problem in the functional framework [RMC15, EBC17, ESBC19, NMR*18], and some of these methods provide a vertex-to-point-in-triangle map as output, which can be used for transferring smooth textures. Note, though, that the meshes still need to be very fine, in order to support non-linear texture deformation, leading to long running times and large memory consumption.

Multi Resolution Spectral Geometry Processing. Beyond subdivision surfaces, other classical approaches include, for example, multi resolution through smoothing [GSS99] and multi resolution through remeshing [BK04]. More recently, Vaxman et al. [VBCG10] used a multi-resolution remeshing based approach to compute the Heat Kernel Signature. We, on the other hand, provide a full shape correspondence pipeline for subdivision surfaces, and in addition, provide bounds on the representation error of subdivided functions in the refined basis. Our work is based on the recently proposed Subdivision Exterior Calculus (SEC) [dGDMD16], that builds differential operators which use the geometry of a refinement of the base mesh for geometry processing on the base mesh. While they define the discrete operators, such as the Laplacian, the authors did not provide an analysis of the spectral decomposition of the Laplacian at different subdivision levels as we do, nor did they address computing spectral descriptors. Estellers et al. [ESC18b] use the subdivision basis functions and quadrature rules for computing the eigenfunctions of the Laplacian and spectral descriptors. Our approach, on the other hand, does not require numerical integration, and we additionally supply bounds on the representation error using the hierarchically computed eigenvectors. Recently Nasikun et al. [NBH18] proposed a fast approximation for the lowest part of the Laplacian spectrum of large meshes. They construct a subspace of local basis functions around sampled points and then solve a restricted, simpler, eigenproblem. The constructed Laplacian basis does not rely on or inherit any property from the subdivision structure of the shape, and thus is different from our approach. Interestingly, [LV18] proposed a method to spectrally approximate large graphs with smaller graphs, not necessarily given by a polygonal mesh, and showed a relation between the spectra of the corresponding graph Laplacians, providing a probabilistic bound. Our approach, in contrast, is based on refinement rather than coarsening, and leverages the subdivision structure and the mesh geometry. It is interesting, yet out of scope for this paper, to further investigate the relation between our bound and the bound they provide for regular graphs. Finally, a few methods exist for computing a localized basis at different scales, e.g. [Rus11, MRCB18], yet these are all computed on a single surface, for, e.g., partial shape correspondence. We, on the other hand, compute *global* basis functions, at multiple subdivision levels.

Detail transfer. An early example of detail transfer for subdivision surfaces was presented by Biermann et al. [BMBZ02], where parts of the surface were parametrized to the plane to allow for copy-paste operations. Other multi resolution modeling approaches are discussed in the SIGGRAPH course dedicated to the topic [Zor06]. These techniques were also generalized to triangle meshes [SBSCO06, SS10a, TSS*11], and developed into a highly successful mesh editing tool, known as MeshMixer [SS10b]. While related, our approach is different than these methods, in that it aims for a *global* correspondence between two subdivision surfaces, that allows to transfer detailed displacement maps and texture images.

1.2. Contributions

Given two input base polygonal meshes, and a set of user specified landmarks, we compute a multi-level map between the refinements of the base meshes. The main contributions of our approach are:

- We show the relation between the eigenfunctions of the SEC Laplace-Beltrami operator at different subdivision levels.
- We develop a Hierarchical Functional Maps (HFM) scheme for subdivision surfaces that is efficient and accurate, allowing us to compute maps for refinements with hundreds of thousands of polygons in a few minutes.
- We apply the computed correspondence for detailed displacement maps and texture image transfer between subdivision surfaces with different base polygon meshes.

2. Background

2.1. Functional Maps

We give here a brief overview of the functional map framework, to make the paper self contained. More details can be found in the paper that introduced the concept [OBCS*12], and in the recent SIGGRAPH course dedicated to the topic [OCB*17].

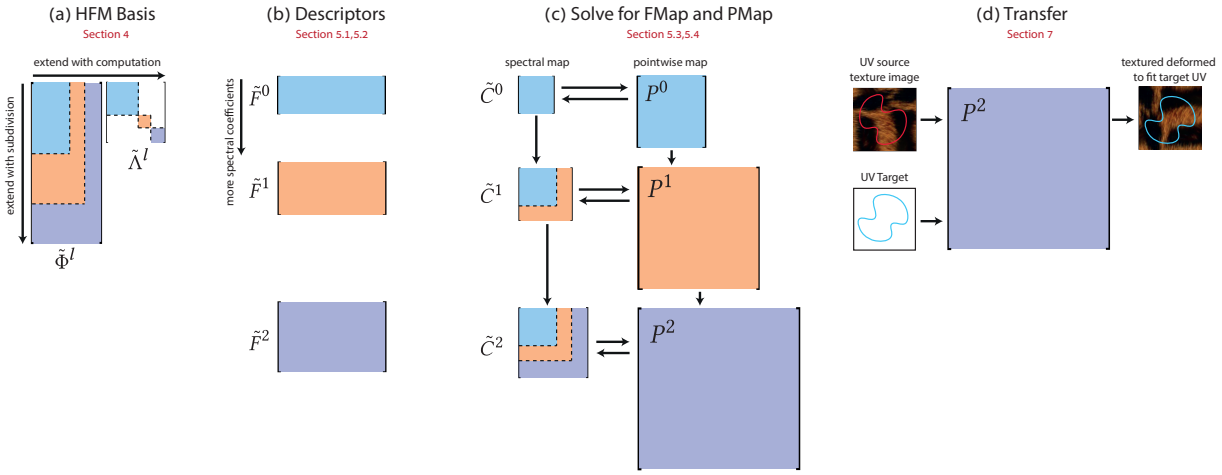

Figure 2: *Our pipeline: (a) compute a hierarchical basis, (b) set up linear constraints from descriptors, (c) solve hierarchically for the spectral functional map and the corresponding pointwise map at all levels, and (d) use the final fine pointwise map for texture transfer.*

2.1.1. Notation

We work with a polygonal mesh $\mathcal{M} = (\mathcal{V}, \mathcal{F}, \mathcal{E})$, given by its vertices, faces and edges, respectively. We denote $|\mathcal{V}| = n$ and $|\mathcal{F}| = m$, and further denote by $X \in \mathbb{R}^{n\times 3}$ the embedding of its vertices in $\mathbb{R}^3$. We consider piecewise linear (PL) functions $g : \mathcal{M} \to \mathbb{R}$, given by their values on the vertices $\mathcal{V}$, and thus $g \in \mathbb{R}^n$. The mass-weighted inner product of functions on $\mathcal{M}$ is given by $\langle g,h\rangle_M = g^T Mh$, with the corresponding norm $\|g\|_M^2 = g^T Mg$. For matrices G,H with functions as columns we set $\langle G,H\rangle_M = G^T MH$. Following the formulation of Alexa and Wardetzky [AW11], the Laplace-Beltrami operator is discretized by the Laplacian matrix $L = M^{-1}W$, where M is a diagonal mass matrix for the vertices, and W is the integrated Laplacian, e.g., the cotangent Laplacian for triangle meshes. Further, $\Lambda \in \mathbb{R}^{k\times k}$ is a diagonal matrix of the eigenvalues of L, sorted from small to large, and $\Phi \in \mathbb{R}^{n\times k}$ has the eigenvectors of L as columns in the same order, such that $W\Phi = M\Phi\Lambda$. The eigenfunctions are M-orthonormal, namely $\langle \Phi,\Phi\rangle_M = I_{k\times k}$. We denote the pseudo-inverse of Φ by $\Phi^\dagger \in \mathbb{R}^{k\times n}$, and note that $\Phi^\dagger = \Phi^T M$. When more than one mesh is discussed, we denote it with a subscript, e.g. L_i is the Laplacian matrix of the mesh $\mathcal{M}_i$, for $i \in \{1,2\}$.

2.1.2. Basics

The basic idea of the functional map framework is to generate a map that puts in correspondence *functions* instead of points. Specifically, for every map $T_{12} : \mathcal{V}_1 \to \mathcal{V}_2$, from the vertices of $\mathcal{M}_1$ to the vertices of $\mathcal{M}_2$, there exists a corresponding functional map (FMap) P_{12} that maps PL functions on $\mathcal{M}_2$ to PL functions on $\mathcal{M}_1$. It is given by $(P_{12}(g_2))(v_1) = g_2(T_{12}(v_1))$, for all vertices $v_1 \in \mathcal{V}_1$, and functions $g_2 \in \mathbb{R}^{n_2}$. It is easy to check that P_{12} is a linear operator, and thus can be described by a matrix $P_{12} \in \mathbb{R}^{n_1\times n_2}$.

The main strength of the functional map framework comes from working with a *spectral* basis for functions, usually taken to be Φ, namely the lower eigenvectors of the LB operator. In this setup, the spectral functional map $C_{12} \in \mathbb{R}^{k_1\times k_2}$ maps functions in the image of Φ_2, represented by their basis coefficients, to functions in the image of Φ_1, and is thus given by $C_{12} = \Phi_1^\dagger P_{12}\Phi_2$.

2.1.3. Inference

To compute a functional map C_{12} between two meshes $\mathcal{M}_1, \mathcal{M}_2$, we first design a set of *linear constraints*. The map is computed by solving a linear least squares optimization problem, where the constraints are *weakly* enforced, i.e., a constraint $Ax = b$ is reformulated into the objective $\|Ax - b\|^2$.

Two often used linear constraints are (1) *descriptor* constraints of the form $C_{12}F_2 = F_1$, where $F_i \in \mathbb{R}^{k_i\times d}$ and (2) *commutativity* constraints of the form $C_{12}\mathcal{O}_2 = \mathcal{O}_1 C_{12}$, where $\mathcal{O}_i \in \mathbb{R}^{k_i\times k_i}$ is a linear operator on $\mathcal{M}_i$. Both the descriptors, F_i, and the operators, $\mathcal{O}_i$, are given through their projection on the spectral bases Φ_i. This framework is quite general, and there are many other ways of computing a functional map, see e.g. [OCB*17] and citations within. We limit ourselves to these cases as they are most common.

Descriptors are often defined through a function of the eigenvalues $\rho : \mathbb{R}^+ \to \mathbb{R}^+$, and can be classified as *signatures* and *landmarks*. Signatures, e.g., the Heat Kernel Signature [SOG09] and the Wave Kernel Signature [ASC11], do not require any prior knowledge on the correspondence, and can be generally defined as the diagonal of the matrix $\Phi\rho(\Lambda)\Phi^T$, where ρ is applied entry-wise to the diagonal of Λ. Landmarks, on the other hand, require the knowledge of two corresponding vertices $v_i \in \mathcal{V}_i$ per landmark. Given a vertex $v \in \mathcal{V}$, these are computed by $\Phi\rho(\Lambda)\Phi^T\delta_v$, where $\delta_v \in \mathbb{R}^n$ is a vector of zeros with a single 1 at the vertex v. The descriptors are then projected on Φ_i to get the matrices F_i that are used in the linear optimization.

Commutativity operators arise as priors on the expected correspondence. For example, the Laplacian operators of two surfaces which are nearly isometric are expected to commute with the output map. Similarly, if the surfaces exhibit intrinsic symmetry, the symmetry maps are expected to commute with the output map as well. Finally, descriptor constraints can be formulated equivalently as operator commutativity constraints, leading to better maps [NO17].

2.1.4. Point-to-point Map Reconstruction

Once a spectral functional map C_{12} has been computed, it can be used as-is to transfer functions from $\mathcal{M}_2$ to $\mathcal{M}_1$. However, it is often beneficial to extract a *full* functional map, represented as a permutation matrix $P_{12} \in \mathbb{R}^{n_1 \times n_2}$, from which a vertex-to-vertex map, T_{12}, can be extracted. Quite a few methods exist that achieve this, e.g., [RMC15, EBC17], yet they are mostly variations on the following approach: use the map C_{12} as an initial solution for the ICP algorithm [BM92] for rigid alignment in the spectral domain. Specifically, the objective $\|C_{12}\Phi_2^T - \Phi_1^T P_{12}\|^2$ is alternately minimized for P_{12} and C_{12} under the constraints that P_{12} is a permutation matrix, and C_{12} is an orthonormal matrix.

2.2. Subdivision Exterior Calculus

2.2.1. Notation

We work with a polygonal base mesh and its refinements, up to the finest subdivision level, denoted by f. We distinguish between meshes at different subdivision levels with a superscript. Thus, we have a set of meshes $\mathcal{M}^l = (\mathcal{V}^l, \mathcal{F}^l, \mathcal{E}^l)$, with $l \in \{0, ..f\}$, where $\mathcal{M}^0$ is the base mesh. Following de Goes et al. [dGDMD16], we define a subdivision matrix $S^l \in \mathbb{R}^{n^{l+1} \times n^l}$ for the vertices at level l. Hence, the embedding of $\mathcal{V}^{l+1}$ is given by $\mathbb{X}^{l+1} = S^l \mathbb{X}^l$, for $l > 0$, taking $\mathbb{X}^0 = X^0$. We accumulate the subdivision of multiple levels by multiplying the corresponding subdivision matrices, namely $\mathbb{S}^{fl} = S^{f-1} S^{f-2} \cdots S^{l+1} S^l$ for $0 \le l < f$. We use Loop subdivision for triangle meshes, and Catmull-Clark for quad meshes.

2.2.2. Discrete Differential Operators

The geometry of the subdivided meshes $\mathcal{M}^l$ changes significantly from the control mesh after a few subdivision levels. The methodology proposed in SEC is to use the *mass matrices* of the finest subdivision level for computing the differential operators of all levels, by defining a subdivision operator that commutes with the discrete exterior derivative. It is straight forward to show that we can compute the SEC unweighted Laplace-Beltrami operator by $\mathbb{W}^l = (\mathbb{S}^{fl})^T W^f \mathbb{S}^{fl}$. Similarly, the SEC mass matrix for the vertices is given by $\mathbb{M}^l = (\mathbb{S}^{fl})^T M^f \mathbb{S}^{fl}$. Note, that since the finest subdivision level f is assumed to be constant, we remove it from the notation for clarity. Figure 3 illustrates the construction of the operators at the different levels.

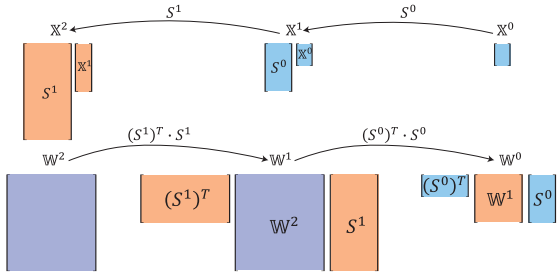

Figure 3: *Construction of refined geometries (top), and SEC operators (bottom). We map between geometries at the finest level.*

3. Hierarchical Functional Maps (HFM)

Notation. We use a combined notation of SEC and FMaps, with a superscript to denote the subdivision level, and a subscript to denote the mesh. For example, $\mathbb{W}_2^0 \in \mathbb{R}^{n_2^0 \times n_2^0}$ denotes the unweighted Laplacian of the second base mesh.

Our goal is to compute a correspondence between two subdivision surfaces, given by their control meshes $\mathcal{M}_i^0$, with $i \in \{1,2\}$. To this end, we design an *efficient* and *accurate* functional map inference scheme for subdivision surfaces by leveraging the subdivision structure. Figure 2 illustrates our pipeline.

In the following, we describe our sub-goals for each FMap component, and how we achieve them.

4. Spectral Functional Basis

The spectral functional basis is the main ingredient in the functional map approach, and it greatly contributes to its effectiveness. To achieve similar effectiveness, we pose the following requirements on the HFM basis.

4.1. Requirements

Multi-scale. The number of basis functions required to represent a function should be correlated with its oscillatory nature, or, more precisely, with the norm of its gradient. This property allows us to control the "resolution" of the computed correspondence through the number of basis functions, and thus the dimensions of the functional map matrix.

Scalable. The basis should be efficiently computable, even on meshes at high subdivision levels. As our goal is to transfer detailed displacement and texture maps, we need the HFM to be applicable to fine mesh resolutions on which detailed displacement maps can be resolved.

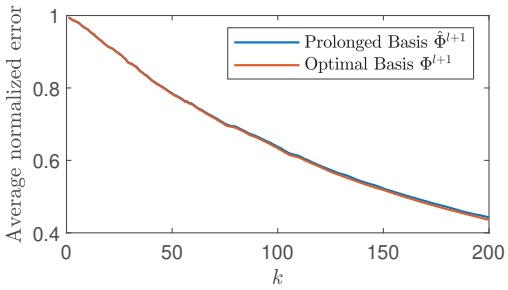

Figure 4: *Average representation error for* 1000 *random functions in the image of S^l, normalized by the functions' squared norm, as a function of the number of eigenvectors k. We compare using the prolonged eigenvectors $\hat{\Phi}^{l+1}$, with the optimal representation error achieved by using Φ^{l+1}. Note that while the exact eigenfunctions achieve a slightly better representation error for larger k values, the graphs are almost indistinguishable.*

Complete. As we tailor a spectral basis, it is imperative that our basis fully spans the space of functions we want to represent, i.e. functions up to some oscillation resolution.

Orthonormal. An orthonormal basis, with respect to the inner product with the vertex mass matrix, allows for fast inversion of the basis, without requiring the computation of the pseudo-inverse.

4.2. SEC Laplacian Eigenvectors

The SEC Laplacian operator of subdivision level $l \in \{0,..,f\}$, given by $\mathbb{L}^l = (\mathbb{M}^l)^{-1}\mathbb{W}^l$ is positive semi-definite [dGDMD16], and thus we can compute its lowest k eigenvectors and eigenvalues, given by $\Phi^l \in \mathbb{R}^{n^l \times k}$ and $\Lambda^l \in \mathbb{R}^{k \times k}$, respectively. By definition, these fulfill:

$$\mathbb{W}^l\Phi^l - \mathbb{M}^l\Phi^l\Lambda^l = 0, \quad \langle \Phi^l, \Phi^l \rangle_{\mathbb{M}^l} = I. \tag{1}$$

It is natural to consider the relation between the SEC Laplacian operators on multiple levels. First, note that for all levels $l \in \{0,..,f\}$, the operator $\mathbb{L}^l$ depends on the geometry of the finest level f, given by M^f, W^f, and on the multilevel subdivision operator $\mathbb{S}^{fl}$, which in turn depends only on the *connectivity* of the control mesh. Therefore, all Laplacians derive from *the same geometry*, and it is expected that there will be a well defined relation between Φ^l, Λ^l and $\Phi^{l+1}, \Lambda^{l+1}$. Indeed, we have the following, which follows directly from the definitions and Equation (1):

$$(S^l)^T(\mathbb{W}^{l+1}S^l\Phi^l - \mathbb{M}^{l+1}S^l\Phi^l\Lambda^l) = 0, \tag{2}$$

since $\mathbb{W}^l = (S^l)^T\mathbb{W}^{l+1}S^l$ and similarly for $\mathbb{M}^{l+1}$.

Definition 1 The *prolonged* eigenvectors and eigenvalues at level $l+1$ from level l are given by:

$$\hat{\Phi}^{l+1} := S^l\Phi^l, \quad \hat{\Lambda}^{l+1} := \Lambda^l. \tag{3}$$

Using this definition and Equation (2) it is straightforward to show the following.

Lemma 1 The prolonged eigenvectors and eigenvalues $\hat{\Phi}^{l+1}, \hat{\Lambda}^{l+1}$ are *weak* eigenvectors and eigenvalues of $\mathbb{L}^{l+1}$ with respect to functions of level $l+1$ which are in the image of S^l. Explicitly, for any function $g^{l+1} \in Im(S^l)$ we have:

$$\langle g^{l+1}, \mathbb{L}^{l+1}\hat{\Phi}^{l+1} - \hat{\Phi}^{l+1}\hat{\Lambda}^{l+1} \rangle_{\mathbb{M}^{l+1}} = 0, \tag{4}$$

$$\langle \hat{\Phi}^{l+1}, \hat{\Phi}^{l+1} \rangle_{\mathbb{M}^{l+1}} = I. \tag{5}$$

All the proofs, though elementary, are provided in the appendix for completeness. The important point is that the inner products $\langle \cdot,\cdot \rangle_{\mathbb{M}}$ and $\langle \cdot, \mathbb{L}\cdot \rangle_{\mathbb{M}}$ are *invariant to the subdivision level* l when applied to subdivided functions (see Lemma 5).

Intuitively, Lemma 1 implies that the prolonged eigenvectors and eigenvalues provide a good approximation of the eigen decomposition of $\mathbb{L}^{l+1}$, when considering functions in the image of S^l.

Finally, we can bound the representation error of the projection on the prolonged eigenvectors, as follows.

Lemma 2 Let $g \in Im(S^l)$. Then

$$\|g - \hat{\Phi}\hat{\Phi}^\dagger g\|^2_{\mathbb{M}} \leq \frac{\|\nabla g\|^2_{\mathbb{M}}}{\lambda_{k+1}} \leq \frac{\lambda_{max}}{\lambda_{k+1}}\|g\|^2_{\mathbb{M}}, \tag{6}$$

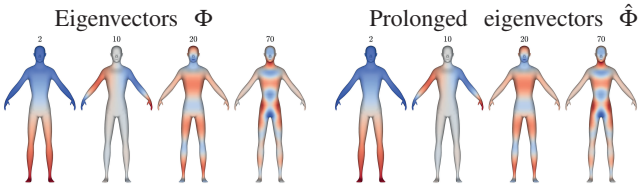

Figure 5: *A few low eigenvectors and the corresponding prolonged eigenvectors from a level below. Note the clear visual similarity.*

where all quantities are at level $l+1$, $\hat{\Phi}$ are the first k eigenvectors of $\mathbb{L}^l$ prolonged to $l+1$, λ_{k+1} is the $k+1^{\text{th}}$ eigenvalue of $\mathbb{L}^l$, λ_{max} is the largest eigenvalue of $\mathbb{L}^l$, and ∇g is a discrete gradient defined such that $\|\nabla g\|^2_{\mathbb{M}} = \langle g, \mathbb{L}g \rangle_{\mathbb{M}}$.

The proof uses a similar technique to the one that is used to show the bound for the eigenvectors of the Laplacian, see e.g., [CPK18]Eq.(17), for the first bound, and the Courant-Fisher Minimax Theorem [ANT09, ANT08] for the second bound. Note that on triangle meshes ∇g is the gradient field of the subdivision of g to the finest level f, because $\mathbb{W}^l = (\mathbb{S}^{fl})^T W^f \mathbb{S}^{fl}$. On general meshes ∇g can be defined through the L^2 norm of the 1-form $d_0 g$ subdivided to the finest level, with respect to the edge-based mass matrix.

Figure 4 demonstrates experimentally the result of Lemma 2. We show the average representation error, normalized by the functions' squared norm $\|g\|^2_{\mathbb{M}}$, as a function of k, the number of computed eigenvectors. We use Φ^{l+1} and $\hat{\Phi}^{l+1}$, for a set of 1000 random functions in the image of S^l. Note that while Φ^{l+1} leads to a better error after 100 eigenvectors, the difference is smaller than 1%.

Figure 5 visualizes the eigenvectors Φ^{l+1} and the prolonged eigenvectors $\hat{\Phi}^{l+1}$ for a few low eigenvectors. For visualization purposes, we chose eigenvectors which correspond to non-repeating eigenvalues. Note, that the sign of the eigenvectors is arbitrary, hence we show either the eigenvector or its negation, chosen so that the eigenvectors visually correspond between the sets. The figure demonstrates that in addition to having similar representation power, the eigenvectors themselves are visually very similar.

Figure 6 (left and center left) shows the eigenvalues of $\mathbb{L}^l$ at various levels. Note, that in addition to the eigenvectors, the eigenvalues are also quite similar, and the similarity breaks at higher eigenvalues for higher levels. This is demonstrated further in Figure 6 (center right and right), which shows the ratio Λ^{l+1}/Λ^l for the different subdivision levels.

The bound in Lemma 2 depends on the largest eigenvalue of $\mathbb{L}$. We leave further investigation of a general bound to future work, noting that similar results for B-Spline surfaces have been recently researched in Iso-Geometric Analysis [ESC18a].

4.3. HFM Basis

We construct our HFM basis by leveraging the representation power of prolonged eigenvectors.

Definition 2 Let $\tilde{k}^f$ be the total number of required basis vectors at the finest level f. Define $\{k^l \geq 0, l \in 0,..,f\}$, such that $\sum_{l=0}^f k^l = \tilde{k}^f$.

The *hierarchical basis* is given by:

$$\tilde{\Phi}^0 = \Phi^0, \quad \tilde{\Lambda}^0 = \Lambda^0, \tag{7}$$

$$\tilde{\Phi}^{l+1} = [S^l \tilde{\Phi}^l, \bar{\Phi}^{l+1}], \quad \tilde{\Lambda}^{l+1} = [\tilde{\Lambda}^l, \bar{\Lambda}^{l+1}], \tag{8}$$

where Φ^0, Λ^0 are the first k^0 eigenvectors and eigenvalues of $\mathbb{L}^0$, $\bar{\Phi}^{l+1}, \bar{\Lambda}^{l+1}$, are the k^{l+1} eigenvectors and eigenvalues of $\mathbb{L}^{l+1}$ in the band starting after the largest eigenvalue of $\tilde{\Lambda}^l$, and the operator $[\cdot,\cdot]$ denotes either column concatenation, or diagonal matrix concatenation according to the context.

Figure 7 illustrates the hierarchical construction of the basis. We provide further details on the splitting of the eigenvalue bands in the Implementation Section 6.

If we consider the basis at level l as an embedding of the polygonal mesh into $\mathbb{R}^{k^l}$, then the HFM basis is given by subdividing this embedding, and adding details using additional k^{l+1} dimensions.

4.4. Properties

Multi-scale. We have shown that prolonged eigenvectors have similar representation power to the exact eigenvectors, and the number of required eigenvectors depends on the norm of the gradient, thus the HFM basis is multi-scale.

Scalable. For high subdivision levels we only need to compute *bands* of eigenvectors. This tactic is a good fit with the common numerical computation of eigenvectors and eigenvalues, which is done per band [Ste02], leading to an efficient basis computation.

Complete. As we split the eigenspace into multiple bands, and compute each band separately, it is preferable that no eigenfunction is "lost", as that will reduce the basis' representation power. Completeness depends on the detailed strategy of band splitting, and we do not have a formal guarantee for its existence, since the splitting involves a heuristic for the behavior of the eigenvalues. In practice, as we discuss in the Implementation Section 6, we can validate that the bands have a non-trivial intersection, and discard the overlaps, by leveraging the approximate orthogonality property. If, however, repeating eigenvalues exist, the HFM basis may not be complete.

Approximately Blockwise Orthonormal. Our basis is computed by concatenating prolonged eigenvectors from multiple subdivision levels, and thus is not orthonormal by construction. We have the following, weaker, result.

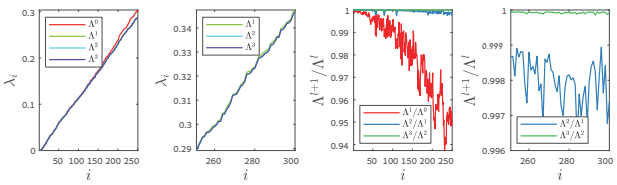

Figure 6: *(left and center left) The eigenvalues of $\mathbb{L}^l$ at different levels, note the similar trends. (center right and right) The ratio Λ^{l+1}/Λ^l for the different levels, note that the similarity breaks later for higher levels, and the ratio is very close to* 1.

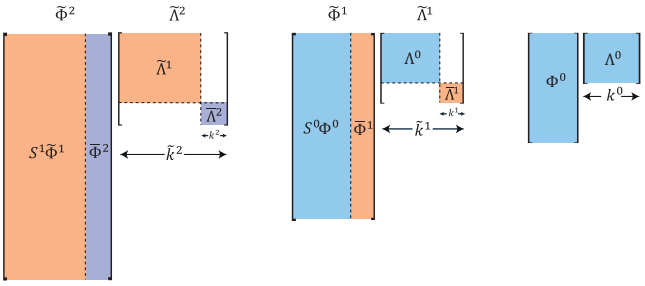

Figure 7: *An illustration of the hierarchical basis construction from Definition 2.*

Lemma 3 Assume $\tilde{\Lambda}^l$ and $\bar{\Lambda}^{l+1}$ are distinct, and both have no repeating eigenvalues. Let Φ^{l+1} be the first $\sum_{i=0}^{l} k^i = \tilde{k}^l$ eigenvectors of $\mathbb{L}^{l+1}$. Then the HFM basis $\tilde{\Phi}$ is *approximately blockwise orthonormal*:

$$\langle \tilde{\Phi}^{l+1}, \tilde{\Phi}^{l+1} \rangle_{\mathbb{M}^{l+1}} = \begin{bmatrix} \langle \tilde{\Phi}^l, \tilde{\Phi}^l \rangle_{\mathbb{M}^l} & E \\ E^T & I_{k^{l+1} \times k^{l+1}} \end{bmatrix}, \tag{9}$$

$$\langle \tilde{\Phi}^0, \tilde{\Phi}^0 \rangle_{\mathbb{M}^0} = I_{k^0 \times k^0}, \tag{10}$$

where the error matrix E is controlled by

$$\frac{1}{k^{l+1}} \sum_{ij} |E_{ij}| \leq \| S^l \tilde{\Phi}^l - \Phi^{l+1} \|^2_{\mathbb{M}^{l+1}}. \tag{11}$$

Intuitively, the lemma bounds the failure of the basis to be orthonormal by the failure of the lower eigenvectors of $\mathbb{L}^{l+1}$ to be prolongations of the basis at level l. As demonstrated experimentally in Figure 5, this error is indeed small, when no repeating eigenvalues exist. In practice, however, meshes often have repeating eigenvalues, thus in our computations, we do compute a pseudo-inverse matrix of the HFM basis, as we further discuss in the Limitations section 6.4.

4.5. Computation time

The hierarchical construction of the basis is much faster than the exact computation, where for each level l all of the $\sum_{i=0}^{l} k^i = \tilde{k}^l$ eigenvectors should be computed, instead of a band of eigenvectors. Table 1 shows a comparison of the basis computation times between HFM and the exact approach. It is clear that for very large shapes our method is much less expensive.

5. Inference

To formulate an optimization problem we first define a set of linear constraints per hierarchy level l, which are either *descriptor preservation* or *operator commutativity constraints*. We then iteratively solve a linear optimization problem at every level, which is bootstrapped by the solution at the previous level. Finally, we propose a novel improvement on the pointwise map extraction that leverages the subdivision structure and yields our output: a map between fine refinements of the subdivision surfaces.

5.1. Descriptors Constraints

A spectral descriptor is given in terms of a 1-parameter family of functions $\rho_t : \mathbb{R}^+ \to \mathbb{R}^+$, where $t \in \mathbb{R}^+$, is a filter on the eigenvalues.

Figure	Shape	n^0	f	n^f	$\tilde{k}^l$	Exact	HFM
Fig. 13	woman	1.3k	3	85k	(100,50,50,50)	26.4	12.5
Fig. 13	man	1.3k	3	85k	(100,50,50,50)	26.6	12.9
Fig. 1	tiger	7.1k	3	454k	(100,100,50,50)	226.3	83.1
Fig. 1	cat	1.2k	4	317k	(100,50,50,50,50)	156.2	57.4
Fig. 16	zebra	9.8k	3	629k	(100,100,50,50)	322.5	127.7
Fig. 16	horse	1.7k	4	439k	(100,50,50,50,50)	227.5	80.8
Fig. 17	elephant	6.0k	3	385k	(100,100,50,50)	194.5	72.9
Fig. 17	mammoth	1.7k	4	442k	(100,50,50,50,50)	227.4	81.1
Fig. 18	troll TEX	4.2k	3	272k	(100,50,50,50)	104.5	47.8
Fig. 18	troll	2.2k	3	145k	(100,50,50,50)	49.0	23.7
Fig. 18	orc	2.2k	3	142k	(100,50,50,50)	48.7	23.5

Table 1: *Timing statistics (in seconds). From left to right: (n^0) number of vertices at the coarse level; (f) finest subdivision level; (n^f) number of vertices at the finest level; ($\tilde{k}^l$) The number of eigenvectors computed per level. Timing (in seconds) for: exact basis computation; HFM basis computation.*

For example, the heat kernel map [SOG09] of a vertex $v \in \mathcal{V}$ is given by $\Phi \rho_t(\Lambda) \Phi^T \delta_v$, with $\rho_t(\lambda) = \exp(-\lambda t)$. As we do not compute a full eigen-decomposition of the Laplacian, we would like to use the HFM basis in a similar way for computing spectral descriptors.

Definition 3 The *Hierarchical Spectral Descriptior Matrix* (HSD) is given by $\tilde{K}_{\rho,t} = \tilde{\Phi} \rho_t(\tilde{\Lambda}) \tilde{\Phi}^T$. We remove the level notation, as all the quantities are at the same subdivision level.

We additionally define a *hierarchical landmark descriptor* using $\tilde{K}_{\rho,t,v} = \tilde{K}_{\rho,t} \delta_v$ for a vertex $v \in \mathcal{V}$, and a *hierarchical signature descriptor* using $\tilde{K}_{\rho,t,\bullet} = diag(\tilde{K}_{\rho,t})$.

The heat kernel and other spectral descriptors have beneficial properties which we would like to preserve. Indeed, we have the following, which is a straightforward consequence of the definition:

Lemma 4 Let $\tilde{K}^{l+1}_{\rho,t}$ be the HSD at level $l+1$, then:

$$\tilde{K}^{l+1}_{\rho,t} = S^l \tilde{K}^l_{\rho,t} (S^l)^T + \bar{\Phi}^{l+1} \rho_t(\bar{\Lambda}^{l+1}) (\bar{\Phi}^{l+1})^T. \qquad (12)$$

Namely, the HSD at level $l+1$ is given by subdividing the HSD from level l, and adding the contribution of the new basis functions from level $l+1$. Thus, the difference between, e.g., the heat kernel and the hierarchical heat kernel again depends on how much the eigenvectors at level $l+1$ are similar to the subdivided eigenvectors from level l, and similarly for the eigenvalues. In fact, since for the heat kernel we have that the contribution of higher eigenvectors decays exponentially with t, the larger t is, the better the hierarchical heat kernel approximates the exact heat kernel.

Figure 8 shows the ratio of the hierarchical heat kernel signature and the exact heat kernel signature for a few vertices as a function of t (left), and for two t values for all vertices (right). Note that as t increases the error decreases, however even for small t the error is below 2 percent, and invisible to qualitative inspection.

In practice, we do not compute the full matrix $\tilde{K}_{\rho,t}$, but only the landmark descriptors $\tilde{K}_{\rho,t,v}$ for a given set of input landmarks, and the signature descriptor $\tilde{K}_{\rho,t,\bullet}$, which is given by the diagonal. The landmark descriptors are efficiently computed in the spectral

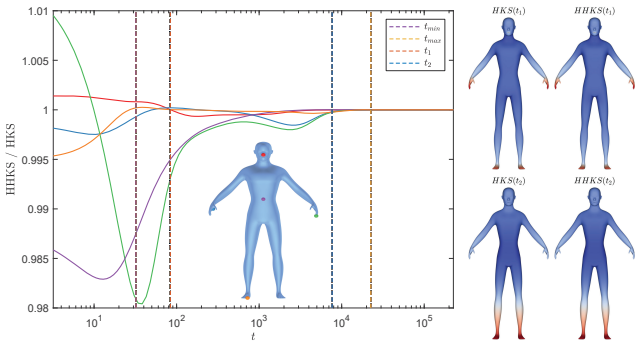

Figure 8: *Quantitative and qualitative evidence to the Hierarchical HKS approximation quality. (left) The ratio of the HHKS and the HKS for a few vertices as a function of t, (right) the exact and hierarchical HKS for two times t as a function on the mesh. Note that as t increases the error decreases, and that even for small t the error is less than 2 percent, and visually indistinguishable. t_{min} and t_{max} as recommended by [SOG09].*

domain, see Section 6.2.2, and the signatures are projected onto the HFM basis $\tilde{\Phi}^l_i$. This yields the descriptor matrices $F^l_i \in \mathbb{R}^{\tilde{k}^l \times d}$, where d is the number of descriptors, which is fixed for all levels.

5.2. Commuting Operators Constraints

We provide as building blocks hierarchical commuting operators for commutativity with the Laplacian and with a given intrinsic symmetry map, which are most commonly used, and additionally are *sparse* in the spectral basis. We discuss later how dense operators can be incorporated in our framework.

Laplacian. Isometric maps commute with the Laplacian operator. Therefore, as a regularization, a common constraint is given by $C_{12}\mathcal{O}_2 = \mathcal{O}_1 C_{12}$, where $\mathcal{O}_i = \Phi_i^\dagger L_i \Phi_i = \Phi_i^T M_i (\Phi_i \Lambda_i \Phi_i^T M_i) \Phi_i = \Lambda_i$. We therefore leverage the approximate orthogonality property of the HFM basis and set the Laplacian commutativity operator accordingly, taking: $\tilde{\mathcal{O}}^l_i = \tilde{\Lambda}^l_i$.

Symmetry. In many cases, especially if designed by artists, the input surfaces have intrinsic self-symmetry given as input self-maps, or permutations, $\mathcal{S}^0_i \in \mathbb{R}^{n^0_i \times n^0_i}$. Note that if the control polygon mesh is symmetric, then for symmetric schemes, such as Catmull-Clark, the subdivided meshes at all levels will be symmetric as well. Thus, given the symmetry at level l, we perform a nearest neighbor search between the vertices of the subdivided mesh at level $l+1$, namely $X^{l+1} = S^l X^l$ and the subdivided symmetric embedding at level $l+1$, given by $S^l \mathcal{S}^l X^l$, to find the symmetry map $\mathcal{S}^{l+1}$. Since the symmetry map is *combinatoric* this process is applicable to intrinsic as well as extrinsic symmetries. Finally, we use the given symmetry as a commutation constraint by projecting it on the HFM basis, namely we set: $\tilde{\mathcal{O}}^l_i = (\tilde{\Phi}^l_i)^\dagger \mathcal{S}^l_i \tilde{\Phi}^l_i$.

If the mesh is exactly symmetric, the symmetry operator commutes with the Laplacian, namely $\mathcal{S}\mathbb{L} = \mathbb{L}\mathcal{S}$, and its spectral representation is diagonal. Since the symmetry is often not exact, we restrict the operator to the main 3 diagonals of $\tilde{\mathcal{O}}_i$.

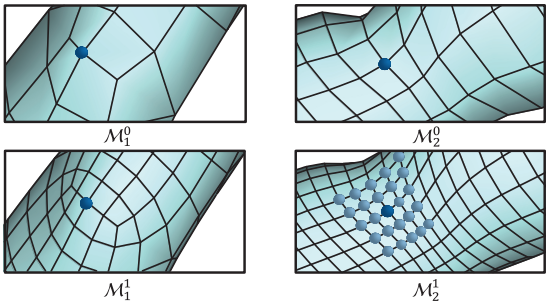

Figure 9: *Hierarchical nearest neighbor search. (top) Two matching vertices at level* 0*. (bottom) The only candidates considered for a match of a refined vertex of* $\mathcal{M}_1^1$ *(bottom left) are the non-zero entries in* S_2^0*, in the column that corresponds to the vertex from level* 0 *(bottom right).*

5.3. Optimization

Given the hierarchical construction of the basis $\tilde{\Phi}_i^l$, the descriptors $\tilde{F}_i^l$, and the commutativity operators $\tilde{\mathcal{O}}_i^l$, we proceed to optimize for the *Hierarchical Functional Map* $\tilde{C}_{12}^l$, as follows.

5.3.1. Coarse level 0

Solve for $\tilde{C}_{12}^0$ using the standard functional map optimization scheme, by minimizing:

$$\tilde{C}_{12}^0 = \arg\min_C \|CF_2^l - F_1^l\|_F^2 + \sum_{\mathcal{O}} \alpha(\mathcal{O}) \|C\mathcal{O}_2^0 - \mathcal{O}_1^0 C\|_F^2, \quad (13)$$

where the sum goes over all available commutativity operators, either Laplacian, or symmetry or both, α is the weight assigned to each operator, and the norm is the Frobenius norm. The parameter values are fixed for all experiments and provided in the Implementation section 6.

5.3.2. Level $l+1$

Given the solution $\tilde{C}_{12}^l$ from level l, we compute the functional map at the next level as follows. We define the solution as the matrix:

$$\tilde{C}^h = \begin{bmatrix} \tilde{C}^l & \bar{C}^{l\leftarrow h} \\ \bar{C}^{h\leftarrow l} & \bar{C}^h \end{bmatrix}, \quad (14)$$

where $h = l+1$ and for clarity we removed the subscript notation. The matrix block $\bar{C}^{l\leftarrow h} \in \mathbb{R}^{\tilde{k}^l \times k^h}$, for example, transfers high frequencies on $\mathcal{M}_2$ to low frequencies on $\mathcal{M}_1$.

Now we reformulate the descriptor and commutativity constraints to solve for the three unknown matrix blocks $\bar{C}$. To this end, we decompose the constraints as block matrices using:

$$\tilde{F}^{l+1} = \begin{bmatrix} \tilde{F}^l \\ \bar{F}^h \end{bmatrix}, \quad \tilde{\mathcal{O}}^{l+1} = \begin{bmatrix} \tilde{\mathcal{O}}^l & 0 \\ 0 & \bar{\mathcal{O}}^h \end{bmatrix}, \quad (15)$$

where, e.g., $\tilde{F}^l \in \mathbb{R}^{\tilde{k}^l \times d}$ contains the coefficients of the descriptors in the low eigenvectors of the basis, and similarly for the other matrices. The constraint matrices $\tilde{F}_i^{l+1}, \tilde{\mathcal{O}}_i^{l+1}$ are computed in terms of coefficients of $\tilde{\Phi}^{l+1}$ when possible, e.g. for landmark spectral constraints, and Laplacian commutativity constraints, or computed as full operators/functions and projected to the HFM basis.

With the constraints in hand, we solve for the partial matrix $\bar{C}^{l\leftarrow h}$, *independently* of the other two unknown blocks:

$$\bar{C}_{12}^{l\leftarrow h} = \arg\min_C \|C\bar{F}_2^h - \tilde{F}_1^l + \tilde{C}_{12}^l \tilde{F}_2^l\|_F^2 + \sum_{\mathcal{O}} \alpha(\mathcal{O}) \|C\bar{\mathcal{O}}_2^h - \tilde{\mathcal{O}}_1^l C\|_F^2. \quad (16)$$

The equations for $\bar{C}^{h\leftarrow l}$ and $\bar{C}^h$ are similarly formulated in terms of the block matrices of the linear constraints, leading to a linear least squares optimization, which is coupled in these two blocks.

At the coarsest level we solve for $(k^0)^2$ variables, whereas at the level h we solve two systems, with $\tilde{k}^l k^h$ and $k^h \tilde{k}^h$ variables, respectively. Since the linear solve scales cubicly in the number of variables, these reductions are significant. Dense operators will break the block diagonal structure of $\tilde{\mathcal{O}}^l$, and then all the three unknown matrix blocks will be coupled. Nevertheless, solving for $\tilde{k}^l k^h + k^h \tilde{k}^h$ variables is still much more efficient than solving for $(\tilde{k}^h)^2$ variables. Some sample timings can be seen in Table 3, where we report the timing for all our results.

5.4. P2P recovery

While the functional map can be used as is for transporting functions, our final goal is to obtain fine pointwise maps. A standard way [OCB*17] to extract a permutation matrix P_{12} from the computed spectral functional map C_{12}, is to solve the optimization problem: $\arg\min_{P,C} \|C\Phi_2^T - \Phi_1^T P\|^2$ alternatingly for P and C, starting from the inferred functional map C_{12}. Optimizing for P while keeping C fixed is implemented using a nearest neighbor

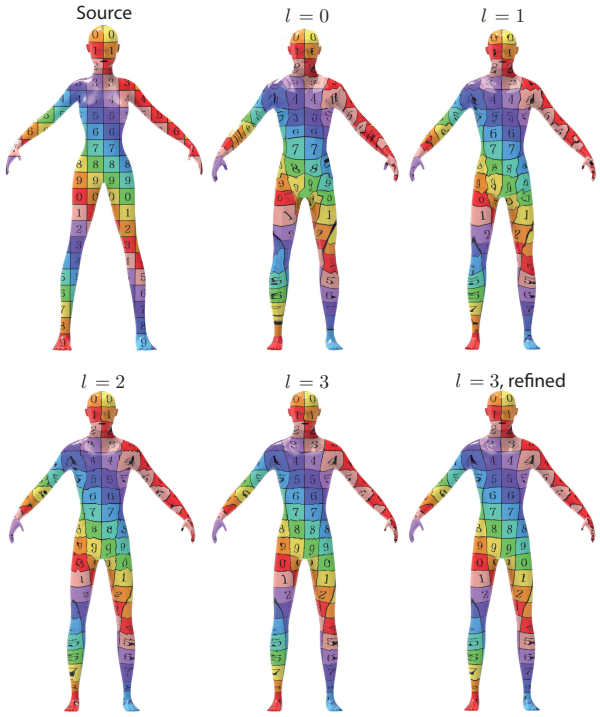

Figure 10: *Visualization of the map during the hierarchical process. Notice how the map improves as the resolution increases.*

search. Optimizing for an orthogonal C while keeping P fixed, is done using a linear solve and an SVD decomposition.

It is not feasible to use this approach directly at the finest subdivision level, as it may contain hundreds of thousands of polygons and hundreds of dimensions, making the nearest neighbors search computationally intractable. Thus, we leverage our hierarchical scheme to generalize this P2P recovery approach as follows.

Coarse level 0**.** Compute P_{12}^0 from C_{12}^0, using spectral ICP.

Level $l+1$**.** Our assumption is that corresponding points at level $l+1$ should be close to subdivided corresponding points from level l. Therefore, when optimizing for a permutation P_{12}^{l+1} using nearest neighbors search, we only consider matching candidates that correspond to the top r nearest neighbor matches from level l, see Figure 9. Namely, we consider candidates that have non-zero entries in the corresponding r columns of S_2^l. This considerably narrows down the search space, and reduces the computation time by orders of magnitude.

Refinement beyond the finest level. As the goal is to match between *subdivision surfaces*, we allow the matched points on $\mathcal{M}_2$ to be arbitrary points on the subdivision surface, and not necessarily vertices of the finest level $\mathcal{V}_2^f$. This is achieved by subdividing the spectral embedding of $\mathcal{M}_2$ and optimizing for a permutation P:

$$P_{12}^{f+1} = \arg\min_P \|C_{12}^f (S_2^f \Phi_2^f)^T - (\Phi_1^f)^T P\|^2, \tag{17}$$

where the dimensions of P_{12}^{f+1} are $n_1^f \times n_2^{f+1}$, again using only the r nearest neighbors from the level below as matching candidates. The final output map is then given by the matrix $P_{12}^f = P_{12}^{f+1} S_2^f$, whose dimensions are, as required, $n_1^f \times n_2^f$. Note that P_{12}^f is a general row stochastic matrix, that encodes a vertex-to-vertex map, from $\mathcal{V}_1^f$ to $\mathcal{V}_2^{f+1}$. The entries of the rows of P_{12}^f are the convex combination weights defined by the subdivision structure.

Figure 10 demonstrates this process, by visualizing the maps for levels $l = [0,..,f]$, as well as the final finest map after the refinement on the last level. Note the improvement of the map as the subdivision level increases, and the smooth map achieved after the last refinement step.

6. Implementation Details

6.1. Splitting the eigenspace

The computation of the HFM basis requires splitting the eigenspace into $f+1$ bands, given the required band widths $\{k^l \geq 0, l \in 0,..,f\}$, such that $\sum_{l=0}^f k^l = \tilde{k}^f$. For each band we employ Matlab's *eigs* solver [Mat18, LSY98, Ste02], that computes eigenvalues and eigenvectors near a given σ that is close to an eigenvalue.

We rely on Weyl's law, that predicts a linear growth of the eigenvalues as a function of their index [Ivr16] to estimate an eigenvalue in the "center" of the band of level $l+1$. Weyl's law applies to the asymptotic behavior for the continuous operator as $\lambda \to +\infty$, and our operator is discrete. However, a recent similar result exists for finite element discretization of elliptical PDEs [XZZ17]Thm 4.3, and indeed, the empirical behavior is close to linear, see Figure 6. Thus, we iteratively compute the bands as follows.

Level 0**.** Compute k^0 smallest generalized eigenpairs of $\mathbb{W}^0, \mathbb{M}^0$, yielding $\tilde{\Phi}^0, \tilde{\Lambda}^0$.

Level $l+1$**.**

- Estimate a linear function $\lambda_b(i)$ of the $\tilde{k}^l$ eigenvalues in $\tilde{\Lambda}^l$ as a function of their index.
- Set $\sigma = \lambda_b(\tilde{k}^l + \frac{1}{2}k^{l+1})$, and compute $k^{l+1}+h$ generalized eigenvalues and eigenvectors of $\mathbb{W}^{l+1}, \mathbb{M}^{l+1}$ near σ, yielding $\bar{\Phi}^{l+1}, \bar{\Lambda}^{l+1}$.
- Remove eigenvectors $\bar{\varphi}^{l+1}$ for which $\|\langle \tilde{\Phi}^l, \bar{\varphi}^{l+1} \rangle\|_\infty > \varepsilon$.

As the eigenvalues are not exactly linear, we allow some leeway in the computation of the bands, by computing h eigenpairs more than what is required. Then, we leverage the approximate orthogonality property from Lemma 3, to remove eigenvectors that are already well represented in the basis $\tilde{\Phi}^l$, where we filter eigenvectors with a maximal projection norm larger than ε. In all our experiments we used $h = 15$, and $\varepsilon = 0.4$.

Note that since repeating eigenvalues often do exist, we cannot guarantee that the HFM basis is complete. For example, if the band in level l is computed such that the last eigenvector is one of a pair of eigenvectors with similar eigenvalues, the computation at level $l+1$ may return the same pair rotated in eigenspace, in which case the projection on $\tilde{\Phi}$ may be above the threshold and the eigenvector will be discarded. In practice, we have not experienced problems due to this limitation.

6.2. Landmark Descriptors

6.2.1. Landmarks at fine levels

The input base meshes are often coarse, and therefore it is possible that semantic landmark points, e.g., an elbow, do not land on vertices, see e.g., Figure 11 (left). Thus, it is imperative to allow the user to place landmarks on *any* subdivision level l.

Due to the subdivision structure, the embedding of a vertex $v^f \in \mathcal{V}^f$ of the refined mesh is a convex combination of the embeddings of base mesh vertices. Specifically, the convex combination weights are the non-zero elements of the v^f-th row of $\mathbb{S}^{f0}$. Therefore, we compute the fine landmark descriptor as the corresponding convex combination of the coarse landmark descriptors, using $\tilde{K}_{\rho,t}^0 (\mathbb{S}^{f0})^T \delta_v^f$. The same applies for placing landmarks at any level $h \leq f$, and computing descriptors at any level $0 \leq l < h$, by taking $\tilde{K}_{\rho,t}^l$ and $\mathbb{S}^{hl}$. Figure 11 (right) shows the resulting coarse descriptors for the fine landmarks shown.

6.2.2. Efficient basis coefficients computation

A landmark descriptor of a vertex $v \in \mathcal{V}$ is given by Definition 3:

$$\tilde{K}_{\rho,t,v}^l = \tilde{\Phi}^l \rho_t(\tilde{\Lambda}^l) (\tilde{\Phi}^l)^T \delta_v.$$

For the functional map optimization we only need the basis coefficients of these descriptors, which are given by $(\tilde{\Phi}^l)^\dagger \tilde{K}_{\rho,t,v}^l = \rho_t(\tilde{\Lambda}^l) (\tilde{\Phi}^l)^T \delta_v$. Denoting the v-th row of $\tilde{\Phi}^l$ by $\tilde{\varphi}_v^l$ we therefore

have that the coefficients of the landmark descriptor are $\rho_t(\tilde{\Lambda}^l)\tilde{\varphi}^l_v$, which is a vector of size $\tilde{k}^l$. Thus, we can compute the coefficients directly, without computing the full descriptor first.

The same applies for descriptors of landmarks v^h at finer levels $h > l$, where the basis coefficients are $\rho_t(\tilde{\Lambda}^l)\,(\mathbb{S}^{hl}\tilde{\Phi}^l)^T\delta_{v^h}$.

6.3. Parameters

Hierarchy. The finest subdivision level f is set according to the required number of vertices in the finest level. In our examples we used meshes with up to $629K$ vertices at the finest subdivision level. For flexibility, we allow for different f levels for $\mathcal{M}_1$ and $\mathcal{M}_2$.

HFM Basis. In all the experiments, we set $k^0 = 100$, and $k^l = 50$ or 100 for the other levels. We additionally do not demand that $k^l_1 = k^l_2$, and allow for rectangular functional maps, see Table 1.

Linear Constraints. We choose between $7-21$ landmarks per shape for the landmark descriptors, depending the deviation from isometry of the expected map, with more landmarks required for less isometric shapes. We use WKS and WKM descriptors [ASC11], taking 100 energy levels distributed as recommended by the authors. We normalize each shape to unit area and normalize each descriptor to unit norm. Our models are extrinsically symmetric, thus we search for this symmetry explicitly, and use it as an operator commutativity constraint.

Inference. The α weights are set to 10^{-2} and 10^4 for the Laplacian commutation and symmetry commutation respectively. We use Matlab's direct solver to solve the linear system.

P2P Recovery. We use fixed parameters for the P2P reconstruction, using $s = 5$ alternating ICP iterations at the coarse level, $s = 3$ at the finer levels, and $r = 3$ for the hierarchical nearest neighbors search. At the last level of refinement we use only one ICP iteration.

6.4. Limitations

The HFM basis is not guaranteed to be complete in the presence of repeating eigenvalues. In practice we have not seen ill effects due to this, but a more principled approach for preventing missing eigenvectors is an interesting avenue for future work.

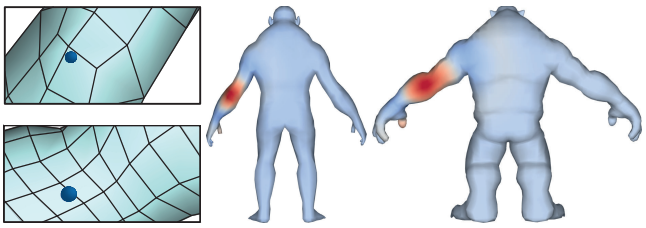

Figure 11: *(left) Corresponding landmark points do not land on vertices of the coarsest level $\mathcal{V}^0$, therefore landmark on fine vertices are required. (center, right) Coarse landmark descriptors on both meses for the fine vertices showed on the left.*

Figure	n^0_2, n^0_1	f_2, f_1	n^f_2, n^f_1	pts	T (m)
Fig. 1, tiger → cat	$7.1k, 1.2k$	3,4	$454k, 317k$	15	9.5
Fig. 16, zebra → horse	$9.8k, 1.7k$	3,4	$629k, 439k$	16	13.6
Fig. 17, eleph. → mam.	$1.7k, 6.0k$	3,4	$385k, 442k$	16	9.9
Fig. 18, troll tex → troll	$4.2k, 2.2k$	3,3	$272k, 145k$	21	4.3
Fig. 18, troll tex → orc	$4.2k, 2.2k$	3,3	$272k, 142k$	21	4.3
Fig. 13, woman → man	$1.3k, 1.3k$	3,3	$85k, 85k$	7	1.6
Fig. 19, troll → orc	$1.3k, 1.3k$	3,3	$145k, 142k$	21	2.9

Table 2: *Statistics and timing. From left to right: (n^0_2, n^0_1) number of vertices at the coarse level; (f_2, f_1) finest subdivision level of source and target models; (n^f_2, n^f_1) number of vertices at the finest levels; (pts) number of landmarks; (T) total time in minutes.*

We currently do not handle dense commutation operator constraints, such as [NO17]. Technically it is possible to incorporate them, however for high subdivision levels they slow down the process. Using an iterative solver with warm start, e.g. as has been done in [GBKS18], could improve our performance further.

Our approach inherits the existing problems of functional map based approaches that rely on WKS descriptors. In some cases, the map might have bad regions, e.g. the right tusk of the mammoth in Figure 17, and the nose of the troll in Figure 18. However, we do believe that our framework provides an excellent platform for improving the functional map machinery further.

We do not handle shapes with multiple components, which are common in models designed by artists, where there are often different components for the eyes, teeth, and others. Thus we pre-clean each shape from any small connected components and remove duplicate vertices if they exist.

7. Results

All the computations were performed on a machine with an i7 CPU and 64GB RAM. The code was written in Matlab except for the Catmull-Clark subdivision for quads which was written in C++ and used as a MEX file.

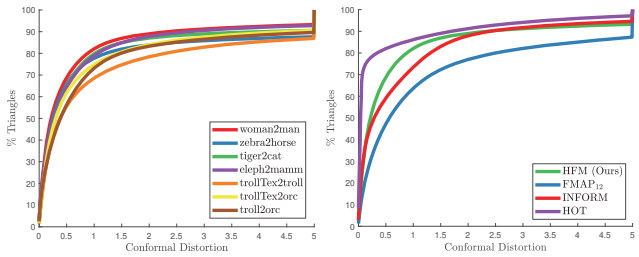

Figure 12: *(left) Conformal distortion of the maps for our experiments. (right) Quantitative comparison of the conformal distortion of the maps. We compare our method (HFM), the non heirarchical functional maps scheme FMAPS12 [OBCS*12], INFORM [NO17] and HOT [AL16]. HOT achieves the best conformal distortion, at a $25\times$ higher computational cost, our method is second.*

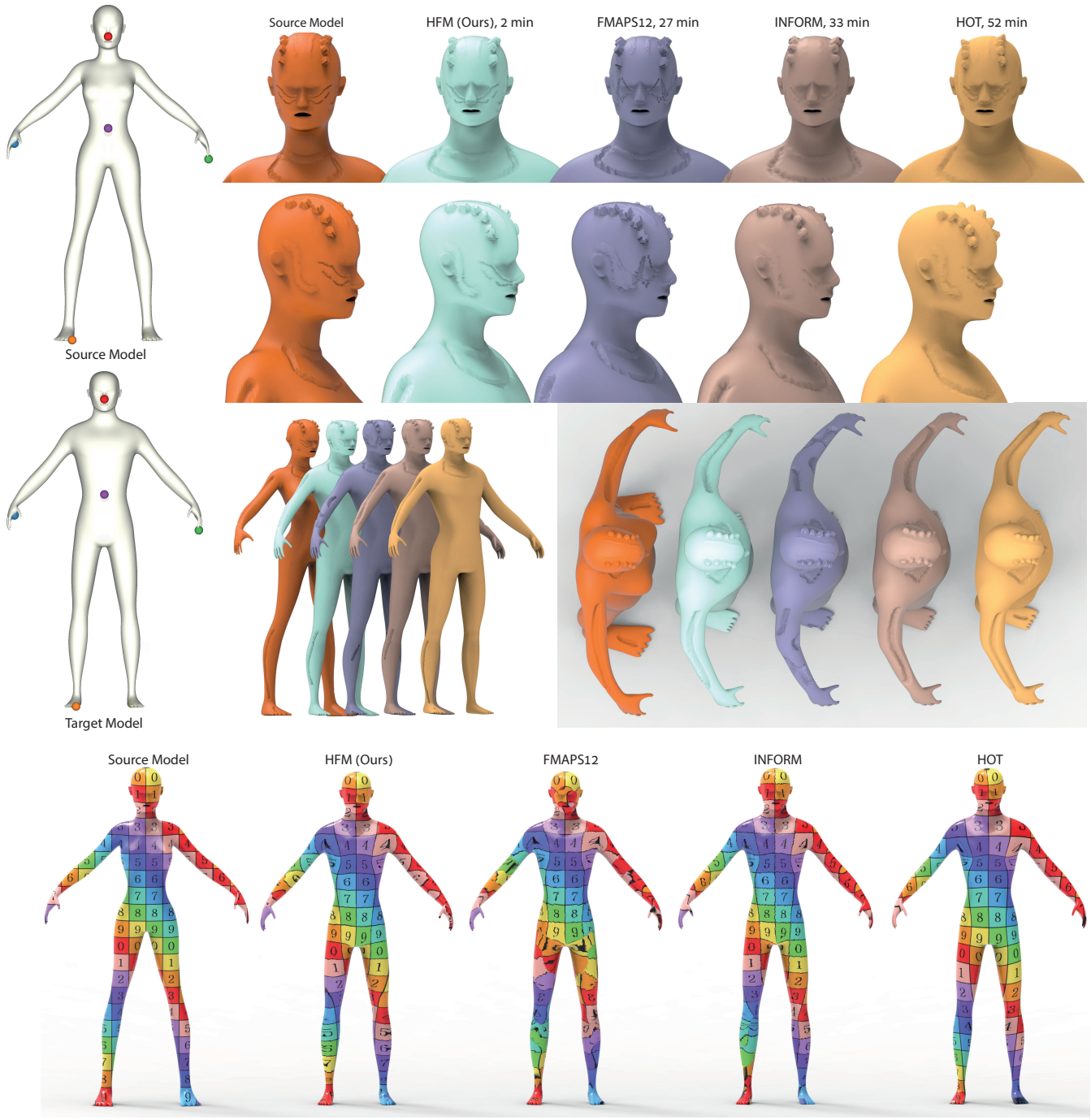

Figure 13: *Comparison of our method (HFM) with the non hierarchical functional maps scheme FMAPS12 [OBCS*12], INFORM [NO17] and HOT [AL16] for displacement transfer. Note that our map correctly transfers displacements at both low resolution (around neck) and high resolution (face and head). The original FMAPS approach does not handle well the fine details on the face, INFORM gives a result comparable to ours, and HOT produces a mostly good result, yet causes the head displacements to slide. All methods took an order of magnitude longer than ours to compute. The bottom row visualizes the map by transporting a checkerboard texture.*

Fig. 1, tiger → cat

	$l=0$	$l=1$	$l=2$	$l=3$	$l=4$	Total
Basis	2.8	13.9	32.6	66.6	33.4	149.3
Desc	0.2	0.4	1.4	5.6	2.2	9.8
Fmap	0.4	0.4	1.5	1.6	2.2	6.1
P2P	0.8	2.0	9.2	37.8	186.0	235.8
Refine					172.5	172.5
Total	4.2	16.7	44.7	111.6	396.3	9.5m

Fig. 16, zebra → horse

	$l=0$	$l=1$	$l=2$	$l=3$	$l=4$	Total
Basis	3.3	17.5	45.8	89.5	44.6	200.8
Desc	0.1	0.5	1.9	7.5	4.4	14.6
Fmap	0.3	0.3	1.4	1.5	2.1	5.7
P2P	1.4	2.4	11.8	50.1	263.9	329.6
Refine					265.0	265.0
Total	5.2	20.8	61.0	148.8	580.1	13.6m

Fig. 17, eleph. → mam.

	$l=0$	$l=1$	$l=2$	$l=3$	$l=4$	Total
Basis	2.7	12.8	31.8	68.5	47.0	163.6
Desc	0.1	0.4	1.4	5.2	4.7	11.9
Fmap	0.4	0.3	1.5	1.6	2.1	6.0
P2P	0.9	2.4	11.5	47.3	225.4	287.7
Refine					124.3	124.3
Total	4.1	16.0	46.3	122.7	404.2	9.9m

Fig. 18, troll tex → troll

	$l=0$	$l=1$	$l=2$	$l=3$	Total
Basis	2.4	6.5	22.5	44.3	75.9
Desc	0.1	0.3	1.4	4.2	5.7
Fmap	0.4	0.8	1.2	1.8	4.1
P2P	0.9	3.8	17.1	85.1	107.1
Refine				66.9	66.9
Total	3.7	11.5	42.0	202.4	4.3m

Fig. 18, troll tex → orc

	$l=0$	$l=1$	$l=2$	$l=3$	Total
Basis	1.8	6.4	22.6	44.3	75.3
Desc	0.1	0.3	1.1	4.2	5.6
Fmap	0.3	0.8	1.2	1.7	4.0
P2P	0.8	3.9	17.2	85.0	106.8
Refine				66.4	66.4
Total	3.1	11.3	42.0	201.5	4.3m

Fig. 13, woman → man

	$l=0$	$l=1$	$l=2$	$l=3$	Total
Basis	0.5	2.1	7.8	15.2	25.7
Desc	0.02	0.07	0.3	1.2	1.6
Fmap	0.3	0.5	0.9	1.3	3.0
P2P	0.2	2.3	8.9	34.7	46.0
Refine				21.6	21.6
Total	1.1	5.0	17.9	74.0	1.6m

Fig. 19, troll → orc

	$l=0$	$l=1$	$l=2$	$l=3$	Total
Basis	1.2	4.0	14.8	30.5	50.7
Desc	0.07	0.2	0.7	2.9	3.9
Fmap	0.4	0.8	1.2	1.7	3.9
P2P	0.5	3.5	14.5	32.7	81.1
Refine				37.8	37.8
Total	2.1	8.5	31.2	135.6	2.9m

Table 3: *Timing statistics (in seconds) for each step in each level of the hierarchy. Basis: subdividing and computing the HFM basis; Desc: computing the descriptors and symmetry operators; Fmap: computing the functional map; P2P: extracting a point to point map; Refine: the last refinement step on the finest level.*

7.1. Timing and map quality

Table 2 shows statistics and timings for all our experiments. The longest computation time is 13.6 minutes for the zebra and horse pair (Figure 16), where the models at the finest level have 629k and 439k vertices. Timings for each step per level are given in Table 3. The most time-consuming step in our approach is the P2P reconstruction step, in average close to 70% of the total computation time. Yet, this is a considerable speedup over the same computations in the non-hierarchical setup, where this step is the most time consuming one. We measure map quality by the conformal distortion induced by the map (Figure 12 (left)). The distortion of our maps are of the same order of magnitude as existing approaches.

7.2. Comparisons

We compare our method to HOT [AL16], INF [NO17] and BCICP [RPWO18] using code supplied by the authors. For all the methods, we triangulated the meshes, and used the same constraints as ours, when possible with the provided code. Specifically, for HOT we used only the landmark descriptors, and for INF and BCICP we used landmark and Laplacian commutativity constraints. We additionally compared to a functional maps setup without the hierarchy (FMAPS), where we used landmarks, Laplacian and symmetry constraints.

The timings in minutes were: FMAPS: 27, INF: 33 and HOT: 52. BCICP failed to complete the computation due to memory issues. Our timings are given in Table 2, where the total time was under 2 minutes, and thus an order of magnitude faster than the other approaches. More detailed timings, including each step in the pipeline for each level in the hierarchy, are given in Table 3.

The quantitative and visual results are summarized in Figures 12(right) and 13. We show the source and target meshes, with the corresponding landmarks, the deformed source mesh, and the resulting deformed target mesh after displacement transfer using the computed map. We also show a visualization of the map by transferring a checkerboard texture. Note that our result provides the best visual map for transporting the deformation. The quantitative results show that our approach is better than state-of-the-art methods for all methods except of HOT, which is 25× slower than our approach.

7.3. Application: Transferring textures

Given a pointwise map P_{12}^f between the finest subdivision levels we can transfer texture images.

Assuming both models have texture coordinates U_i^f that are subdivided to the finest level, and given a texture image for $\mathcal{M}_2$, we construct a new texture image for $\mathcal{M}_1$. This is done by first computing the *deformed* texture coordinates of $\mathcal{M}_1$, given by $\tilde{U}_1 = (P_1^{u\leftarrow x})P_{12}(P^{x\leftarrow u})_2 U_2$, where all the quantities are at the finest level. Here, $P^{x\leftarrow u}$ maps texture vertices to model vertices, and vice versa for $P^{u\leftarrow x}$. Next, the model is saved and rendered, with coordinate locations given by U_1 and texture coordinates given by $\tilde{U}_1$. The resulting image is the new texture image for $\mathcal{M}_1$.

A technical issue remains—the texture seams of $\mathcal{M}_2$ do not necessarily correspond to texture seams of $\mathcal{M}_1$, leading to visible artifacts in the new texture. To remedy this, we identify quads of $\mathcal{M}_1$ that are mapped to vertices in the 1-ring neighborhood of the texture seams of $\mathcal{M}_2$, and remove them from the rendering, leading to missing texture regions. Finally, we use an off-the-shelf image inpainting tool [Inp18] for recovering the missing regions, where we use the removed quads as the inpainting mask. This process does not require user intervention, and is demonstrated in Figure 14.

Combined with our high quality maps at the finest levels, this approach is highly effective for texturing base objects using a detailed high resolution texture of a different model. Figures 1, 15, 16, 17, 18 demonstrate our results, with statistics and timings given in Table 2 and Table 3. Note that the transported texture closely follows the semantic correspondence between the shapes. To the best of our knowledge, such detailed transfer was difficult to achieve before.

7.4. Application: Transferring displacement maps

Another common workflow with subdivision surfaces is *sculpting* on a refined mesh, and then baking the resulting displacements

Figure 14: *The texture transfer process. (left) the input texture image, (center) the transported image with the missing data along the texture seams of the source model, (right) the final inpainted results. The textured model appears in Figure 1.*

into a displacement map. Transferring displacement maps created this way is in fact simpler than transferring texture images, since the displacement map can be reproduced with linear interpolation of vertex values at the finest level (as opposed to texture images, where the pixel data is denser than the vertex data). Therefore, we render the new displacement image using linear vertex colors instead of a texture image. Since texture vertices have the same displacement value even if they are on a texture seam, no texture discontinuities are introduced, and thus this application does not require inpainting. If a low level polygonal model is not required, we can skip the baking step, and simply transport the displacement function directly, as a function on the surface.

Figure 19 demonstrates this approach. We deformed one of the troll models from Figure 18 using Blender's multi-resolution sculpting [vG09]pp. 101, and computed the resulting displacement map as a function on the surface. Then, we computed a map to a different troll model, transported the displacement function with the map and applied the displacement. Note the similar semantic locations of the ornaments on the two trolls. Figure 13 used the computed map in the same way, to transport displacement functions. We show our results, and the results of other map computation approaches on the same model, leading to inferior or similar results with an order of magnitude longer computation times. Note that our map was accurate enough to transport the details in a semantic way to the correct locations on the face and head of the target model.

8. Conclusions and Future Work

We presented a method for computing correspondences between subdivision surfaces, which to the best of our knowledge was not possible before. We investigated the spectral structure of the SEC Laplace Beltrami operator at different subdivision levels, and leveraged the results to construct a hierarchical spectral basis. Using this basis, we designed a hierarchical functional map inference scheme that given input landmarks generates very detailed maps, an order of magnitude faster than existing approaches for triangle meshes. Finally, we showed how our maps can be used for texturing and detailing subdivision models, by transferring highly detailed texture images and displacement maps.

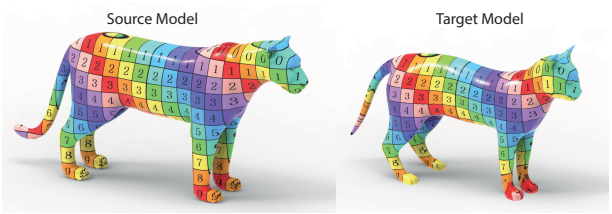

Figure 15: *Map visualization with checkerboard texture transfer for the tiger and cat pair from Figure 1.*

Our approach has many avenues for future work. We chose the SEC Laplace Beltrami operator as a truncated basis to represent functions and operators, because it allows us to derive theoretical representation bounds. Alternatively, a wavelet hierarchical basis [Ber04] could be more appropriate for subdivision surfaces. To that end, the computational consequences of using such a basis, e.g. in terms of the sparsity of the resulting constraints, would need to be evaluated. On the other hand, a local wavelet basis might be more suitable for partial matching, or for transporting discontinuous textures, e.g. for models with semi-sharp creases. We additionally note that although the hierarchical scheme seems independent from the subdivision structure, our theoretical claims rely on differential operators that use the metric of the fine mesh through the subdivision structure. Another technical route to investigate is the number of eigenpairs taken at each subdivision level. In general, the number of eigenfunctions we need depend on the frequency of the functions that we want to represent, and not on the total number of vertices, hence a data driven approach might be appropriate.

From the application standpoint, our computed maps scan potentially be incorporated into 3D modeling environments, e.g., Blender, for simultaneous sculpting on two shapes, like symmetry is used for sculpting today. Finally, generalizing our approach to collections of subdivision surfaces would enable tasks such as joint shape analysis on the abundant datasets of open 3D movies.

Acknowledgements

M. Ben-Chen acknowledges funding from the European Research Council (ERC starting grant no. 714776 OPREP), and the Israel Science Foundation (grant no. 504/16).

References

[ACBCO17] AZENCOT O., CORMAN E., BEN-CHEN M., OVSJANIKOV M.: Consistent functional cross field design for mesh quadrangulation. *ACM Transactions on Graphics (TOG) 36*, 4 (2017), 92. 3

[AL16] AIGERMAN N., LIPMAN Y.: Hyperbolic orbifold tutte embeddings. *ACM Trans. Graph. 35*, 6 (2016), 217–1. 2, 11, 12, 13

[ANT08] AVRON H., NG E., TOLEDO S.: *A Generalized Courant-Fischer Minimax Theorem*. Tech. rep., Lawrence Berkeley National Lab.(LBNL), Berkeley, CA (United States), 2008. 6

[ANT09] AVRON H., NG E., TOLEDO S.: Using perturbed qr factorizations to solve linear least-squares problems. *SIAM Journal on Matrix Analysis and Applications 31*, 2 (2009), 674–693. 6, 19

[ASC11] AUBRY M., SCHLICKEWEI U., CREMERS D.: The wave kernel signature: A quantum mechanical approach to shape analysis. In *Computer Vision Workshops (ICCV Workshops), 2011 IEEE Int. Conf. on* (2011), IEEE, pp. 1626–1633. 4, 11

[AW11] ALEXA M., WARDETZKY M.: Discrete laplacians on general polygonal meshes. In *ACM Transactions on Graphics (TOG)* (2011), vol. 30, p. 102. 4

[Ber04] BERTRAM M.: Biorthogonal loop-subdivision wavelets. *Computing 72*, 1 (Apr 2004), 29–39. 14

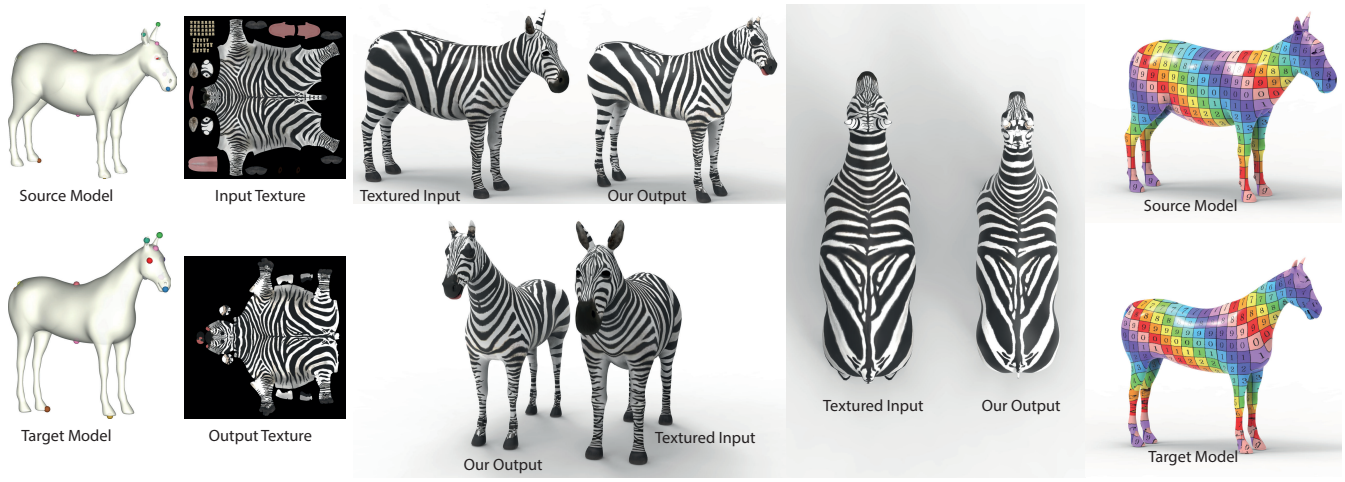

Figure 16: *Texture transfer of a zebra to a horse. (left to right): input source and target models with corresponding landmarks, input and output texture image, pairs of textured inputs and textured outputs from multiple views, map visualization with checkerboard texture transfer.*

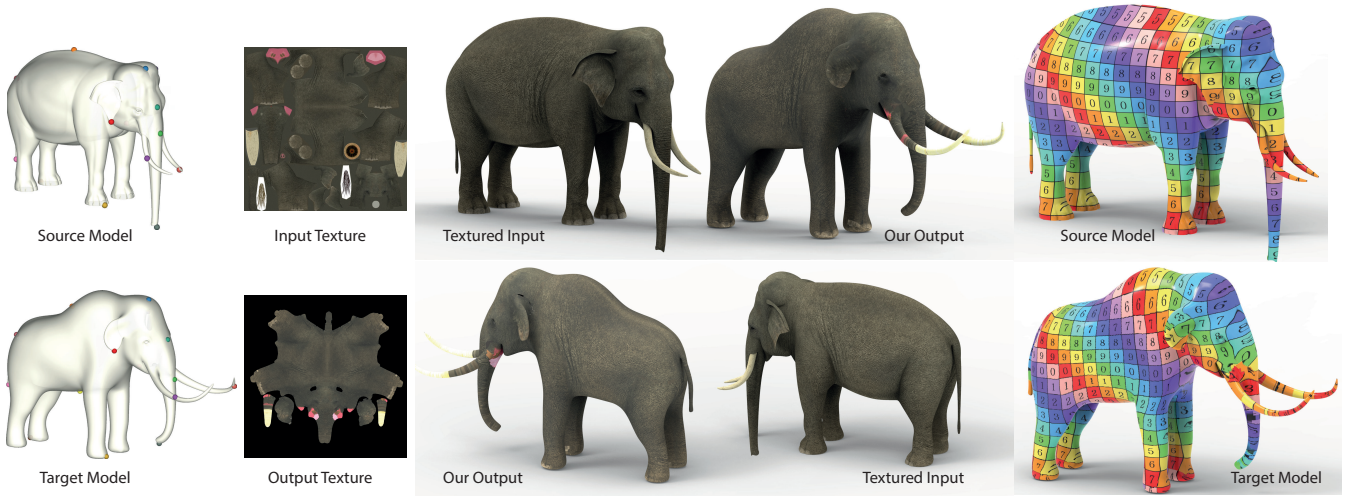

Figure 17: *Texture transfer of an elephant to a mammoth. (left to right): input source and target models with corresponding landmarks, input and output texture image, pairs of textured inputs and textured outputs from multiple views, map visualization with checkerboard texture.*

[BHS*17] BERKITEN S., HALBER M., SOLOMON J., MA C., LI H., RUSINKIEWICZ S.: Learning detail transfer based on geometric features. In *Comp. Graph. Forum* (2017), vol. 36, pp. 361–373. 2

[BK04] BOTSCH M., KOBBELT L.: A remeshing approach to multiresolution modeling. In *Proc. 2004 Eurographics/ACM SIGGRAPH symp. on Geometry processing* (2004), pp. 185–192. 3

[BM92] BESL P. J., MCKAY N. D.: Method for registration of 3-d shapes. In *Sensor Fusion IV: Control Paradigms and Data Structures* (1992), vol. 1611, International Society for Optics and Photonics, pp. 586–607. 5

[BMBZ02] BIERMANN H., MARTIN I., BERNARDINI F., ZORIN D.: Cut-and-paste editing of multiresolution surfaces. In *ACM Transactions on Graphics (TOG)* (2002), vol. 21, pp. 312–321. 3

[CC78] CATMULL E., CLARK J.: Recursively generated b-spline surfaces on arbitrary topological meshes. *Computer-aided design 10*, 6 (1978), 350–355. 2

[CPK18] CHOUKROUN Y., PAI G., KIMMEL R.: Sparse approximation of 3d meshes using the spectral geometry of the hamiltonian operator. *J. Math. Imaging Vis. 60*, 6 (2018), 941–952. 6

[dGDMD16] DE GOES F., DESBRUN M., MEYER M., DEROSE T.: Subdivision exterior calculus for geometry processing. *ACM Transactions on Graphics (TOG) 35*, 4 (2016), 133. 2, 3, 5, 6

[DKT98] DEROSE T., KASS M., TRUONG T.: Subdivision surfaces in character animation. In *Proc. 25th ann. conf. on Comp. graph. and interactive techniques* (1998), pp. 85–94. 2

[EBC17] EZUZ D., BEN-CHEN M.: Deblurring and denoising of maps between shapes. In *Comp. Graph. Forum* (2017), vol. 36, pp. 165–174. 3, 5

[EGKT08] EPPSTEIN D., GOODRICH M. T., KIM E., TAMSTORF R.: Approximate topological matching of quadrilateral meshes. In *Shape Modeling and Applications, 2008. SMI 2008. IEEE Int. Conf. on* (2008), IEEE, pp. 83–92. 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. 3

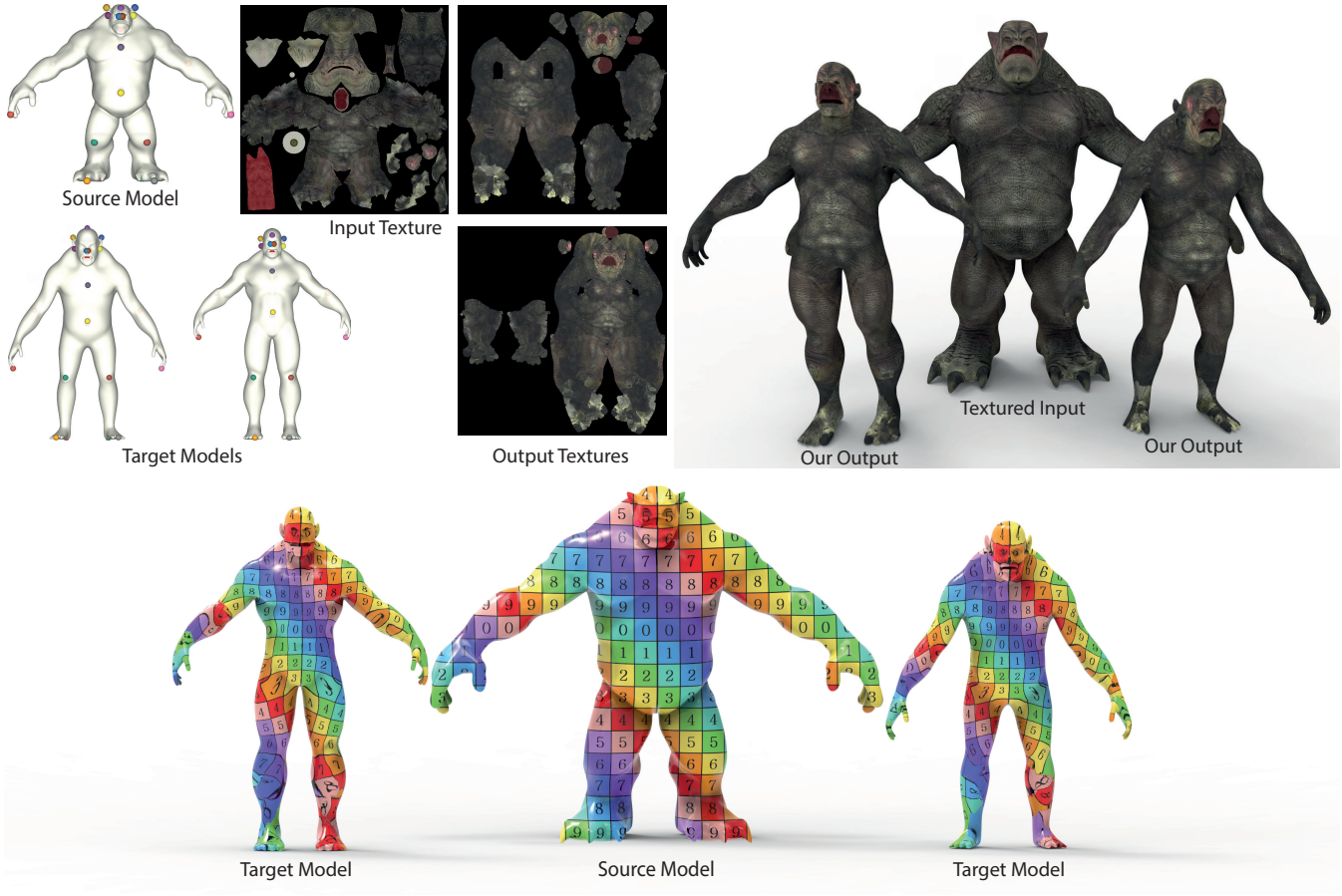

Figure 18: *Troll family. Texture transfer from a troll to two other troll models. (left to right) input source and target models with corresponding landmarks, input and output texture images, the textured input and our textured output results, map visualization with checkerboard texture.*

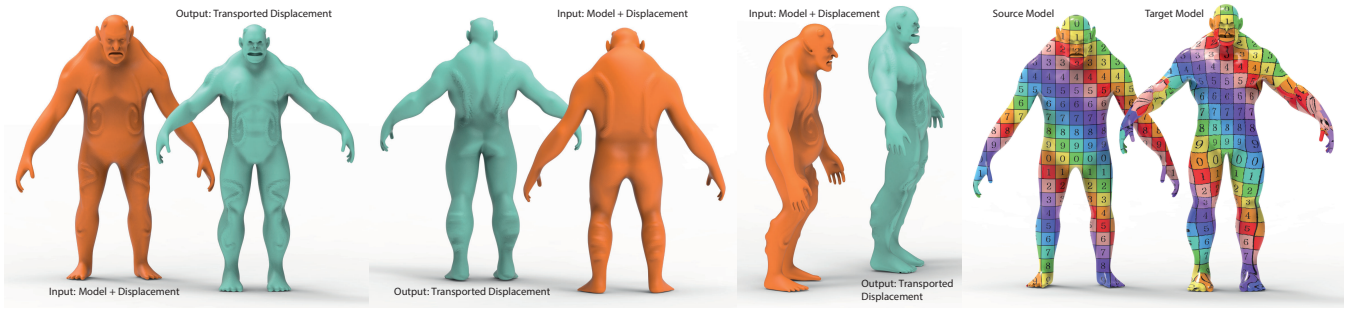

Figure 19: *Displacement transfer from a troll to an orc model. We show the input and output models with the displaced geometry, from multiple views, and map visualization with checkerboard texture.*

[ESBC19] EZUZ D., SOLOMON J., BEN-CHEN M.: Reversible harmonic maps between discrete surfaces. *ACM Transactions on Graphics (TOG) 38*, 2 (2019), 15:1–15:12. 3

[ESC18a] EKSTRÖM S.-E., SERRA-CAPIZZANO S.: *Eigenvalue isogeometric approximations based on B-splines : Tools and results*. Tech. Rep. 2018-012, 2018. 6

[ESC18b] ESTELLERS V., SCHMIDT F., CREMERS D.: Robust fitting of subdivision surfaces for smooth shape analysis. In *2018 Int. Conf. on 3D Vision (3DV)* (2018), IEEE, pp. 277–285. 2, 3

[GBKS18] GEHRE A., BRONSTEIN M., KOBBELT L., SOLOMON J.: Interactive curve constrained functional maps. In *Comp. Graph. Forum* (2018), vol. 37, pp. 1–12. 3, 11

[GKS02] GRINSPUN E., KRYSL P., SCHRÖDER P.: Charms: a simple framework for adaptive simulation. In *ACM transactions on graphics (TOG)* (2002), vol. 21, pp. 281–290. 2

[GSS99] GUSKOV I., SWELDENS W., SCHRÖDER P.: Multiresolution signal processing for meshes. In *Proc. 26th ann. conf. on Comp. graph. and interactive techniques* (1999), pp. 325–334. 3

[HCO18] HUANG R., CHAZAL F., OVSJANIKOV M.: On the stability of functional maps and shape difference operators. In *Comp. Graph. Forum* (2018), vol. 37, pp. 145–158. 3

[Inp18] INPAINT: Inpaint photo restoration software. https://www.theinpaint.com/, 2018. 13

[Ivr16] IVRII V.: 100 years of weyl's law. *Bulletin of Mathematical Sciences 6*, 3 (2016), 379–452. 10

[Lag18] LAGA H.: A survey on nonrigid 3d shape analysis. In *Academic Press Library in Signal Processing, Volume 6*. Elsevier, 2018, pp. 261–304. 2

[LI15] LI X., IYENGAR S.: On computing mapping of 3d objects: A survey. *ACM Computing Surveys (CSUR) 47*, 2 (2015), 34. 2

[LJG14] LIU S., JACOBSON A., GINGOLD Y.: Skinning cubic bézier splines and catmull-clark subdivision surfaces. *ACM Transactions on Graphics (TOG) 33*, 6 (2014), 190. 2

[LLS01] LITKE N., LEVIN A., SCHRÖDER P.: Fitting subdivision surfaces. In *Proc. of the conf. on Visualization'01* (2001), IEEE Computer Society, pp. 319–324. 2

[Loo87] LOOP C.: Smooth subdivision surfaces based on triangles. *Master's thesis, University of Utah, Department of Mathematics* (1987). 2

[LPW*06] LIU Y., POTTMANN H., WALLNER J., YANG Y.-L., WANG W.: Geometric modeling with conical meshes and developable surfaces. In *ACM transactions on graphics (TOG)* (2006), vol. 25, pp. 681–689. 2

[LSY98] LEHOUCQ R. B., SORENSEN D. C., YANG C.: *ARPACK users' guide: solution of large-scale eigenvalue problems with implicitly restarted Arnoldi methods*, vol. 6. Siam, 1998. 10

[LV18] LOUKAS A., VANDERGHEYNST P.: Spectrally approximating large graphs with smaller graphs. In *Proceedings of the 35th International Conference on Machine Learning* (2018), pp. 3237–3246. 3

[LVGL*13] LOOP C., VAN GELDER D., LITKE N., EL GUERRAB R., ELMIEH B., KRAEMER M.: Opensubdiv from research to industry adoption. In *ACM SIGGRAPH 2013 Courses* (2013), SIGGRAPH '13, pp. 16:1–16:1. 1

[Mat18] MATHWORKS: eigs, subset of eigenvalues and eigenvectors. https://www.mathworks.com/help/matlab/ref/eigs.html, 2018. 10

[MK05] MARINOV M., KOBBELT L.: Optimization methods for scattered data approximation with subdivision surfaces. *Graphical Models 67*, 5 (2005), 452–473. 2

[MRCB18] MELZI S., RODOLÀ E., CASTELLANI U., BRONSTEIN M. M.: Localized manifold harmonics for spectral shape analysis. In *Comp. Graph. Forum* (2018), vol. 37, pp. 20–34. 3

[NBH18] NASIKUN A., BRANDT C., HILDEBRANDT K.: Fast approximation of laplace-beltrami eigenproblems. In *Comp. Graph. Forum* (2018), vol. 37, pp. 121–134. 3

[NKF*16] NIESSNER M., KEINERT B., FISHER M., STAMMINGER M., LOOP C., SCHAFER H.: Real-time rendering techniques with hardware tessellation. *Comp. Graph. Forum 35*, 1 (2016), 113–137. 1

[NMR*18] NOGNENG D., MELZI S., RODOLÀ E., CASTELLANI U., BRONSTEIN M., OVSJANIKOV M.: Improved functional mappings via product preservation. In *Comp. Graph. Forum* (2018), vol. 37, pp. 179–190. 3

[NO17] NOGNENG D., OVSJANIKOV M.: Informative descriptor preservation via commutativity for shape matching. In *Comp. Graph. Forum* (2017), vol. 36, pp. 259–267. 3, 4, 11, 12, 13

[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* (2012). 2, 3, 11, 12

[OCB*17] OVSJANIKOV M., CORMAN E., BRONSTEIN M., RODOLA E., BEN-CHEN M., GUIBAS L., CHAZAL F., BRONSTEIN A.: Computing and processing correspondences with functional maps. In *ACM SIGGRAPH 2017 Courses* (2017), pp. 1–62. 2, 3, 4, 9

[RMC15] RODOLÁ E., MOELLER M., CREMERS D.: Point-wise Map Recovery and Refinement from Functional Correspondence. In *Vision, Modeling and Visualization* (2015), Bommes D., Ritschel T., Schultz T., (Eds.). 3, 5

[RPWO18] REN J., POULENARD A., WONKA P., OVSJANIKOV M.: Continuous and orientation-preserving correspondences via functional maps. In *SIGGRAPH Asia 2018 Technical Papers* (2018), SIGGRAPH Asia '18, pp. 248:1–248:16. 3, 13

[Rus11] RUSTAMOV R. M.: Multiscale biharmonic kernels. In *Comp. Graph. Forum* (2011), vol. 30, pp. 1521–1531. 3

[SBSCO06] SHARF A., BLUMENKRANTS M., SHAMIR A., COHEN-OR D.: Snappaste: an interactive technique for easy mesh composition. *The Visual Computer 22*, 9-11 (2006), 835–844. 3

[SOG09] SUN J., OVSJANIKOV M., GUIBAS L.: A concise and provably informative multi-scale signature based on heat diffusion. In *Comp. Graph. Forum* (2009), vol. 28, pp. 1383–1392. 2, 4, 8

[SP04] SUMNER R. W., POPOVIĆ J.: Deformation transfer for triangle meshes. In *ACM Transactions on Graphics (TOG)* (2004), vol. 23, pp. 399–405. 2

[SS10a] SCHMIDT R., SINGH K.: *Drag, drop, and clone: An interactive interface for surface composition*. Tech. rep., Tech. Rep. CSRG-611, Department of Computer Science, University of Toronto, 2010. 3

[SS10b] SCHMIDT R., SINGH K.: Meshmixer: An interface for rapid mesh composition. In *ACM SIGGRAPH 2010 Talks* (2010), SIGGRAPH '10, pp. 6:1–6:1. 3

[Sta03] STAM J.: Flows on surfaces of arbitrary topology. *ACM Transactions On Graphics (TOG) 22*, 3 (2003), 724–731. 2

[Ste02] STEWART G. W.: A krylov–schur algorithm for large eigenproblems. *SIAM Journal on Matrix Analysis and Applications 23*, 3 (2002), 601–614. 7, 10

[TCL*13] TAM G. K., CHENG Z.-Q., LAI Y.-K., LANGBEIN F. C., LIU Y., MARSHALL D., MARTIN R. R., SUN X.-F., ROSIN P. L.: Registration of 3d point clouds and meshes: a survey from rigid to nonrigid. *IEEE trans. on vis. and comp. graphics 19*, 7 (2013), 1199–1217. 2

[TSS*11] TAKAYAMA K., SCHMIDT R., SINGH K., IGARASHI T., BOUBEKEUR T., SORKINE O.: Geobrush: Interactive mesh geometry cloning. In *Comp. Graph. Forum* (2011), vol. 30, pp. 613–622. 3

[Tur19] TURBOSQUID: 3d models for professionals, 2019. URL: http://www.turbosquid.com. 1

[TWS06] THOMASZEWSKI B., WACKER M., STRASSER W.: A consistent bending model for cloth simulation with corotational subdivision finite elements. In *Proc. 2006 ACM SIGGRAPH/Eurographics symp. on Comp. animation* (2006), pp. 107–116. 2

[VBCG10] VAXMAN A., BEN-CHEN M., GOTSMAN C.: A multi-resolution approach to heat kernels on discrete surfaces. In *ACM Transactions on Graphics (TOG)* (2010), vol. 29, p. 121. 3

[vG09] VAN GUMSTER J.: *Blender For Dummies*. For Dummies, 2009. 1, 14

[VKZHCO11] VAN KAICK O., ZHANG H., HAMARNEH G., COHEN-OR D.: A survey on shape correspondence. In *Comp. Graph. Forum* (2011), vol. 30, pp. 1681–1707. 2

[VL08] VALLET B., LÉVY B.: Spectral geometry processing with manifold harmonics. In *Comp. Graph. Forum* (2008), vol. 27, pp. 251–260. 2

[VLB*17] VESTNER M., LÄHNER Z., BOYARSKI A., LITANY O., SLOSSBERG R., REMEZ T., RODOLA E., BRONSTEIN A., BRONSTEIN M., KIMMEL R., ET AL.: Efficient deformable shape correspondence via kernel matching. In *2017 International Conference on 3D Vision (3DV)* (2017), IEEE, pp. 517–526. 3

[WW01] WARREN J., WEIMER H.: *Subdivision methods for geometric design: A constructive approach*. Elsevier, 2001. 2

[XZZ17] XU J., ZHANG H., ZIKATANOV L.: On the weyl's law for discretized elliptic operators. *arXiv preprint arXiv:1705.07803* (2017). 10

[Zor06] ZORIN D.: Modeling with multiresolution subdivision surfaces. In *ACM SIGGRAPH 2006 Courses* (2006), pp. 30–50. 3

[ZS00] ZORIN D., SCHRÖDER P.: *SIGGRAPH Courses*. 2000. 2

Appendix - Proofs.

Lemma 5 Let $g^{l+1}=S^l g^l$ and $h^{l+1}=S^l h^l$. Then we have that

$$\begin{aligned}\langle g^{l+1},\mathbb{L}^{l+1}h^{l+1}\rangle_{\mathbb{M}^{l+1}}&=\langle g^l,\mathbb{L}^l h^l\rangle_{\mathbb{M}^l},\\ \langle g^{l+1},h^{l+1}\rangle_{\mathbb{M}^{l+1}}&=\langle g^l,h^l\rangle_{\mathbb{M}^l}.\end{aligned}\tag{18}$$

Proof.

$$\begin{aligned}&\langle g^{l+1},\mathbb{L}^{l+1}h^{l+1}\rangle_{\mathbb{M}^{l+1}}=\\ &(g^{l+1})^T\mathbb{W}^{l+1}h^{l+1}= \quad (since\ \mathbb{L}=\mathbb{M}^{-1}\mathbb{W})\\ &(S^l g^l)^T\mathbb{W}^{l+1}S^l h^l= \quad (since\ g^{l+1}=S^l g^l\ and\ same\ for\ h)\\ &(g^l)^T\mathbb{W}^l h^l= \quad (since\ \mathbb{W}^l=(S^l)^T\mathbb{W}^{l+1}S^l)\\ &=\langle g^l,\mathbb{L}^l h^l\rangle_{\mathbb{M}^l}. \quad (since\ \mathbb{L}=\mathbb{M}^{-1}\mathbb{W})\end{aligned}\tag{19}$$

And similarly,

$$\begin{aligned}&\langle g^{l+1},h^{l+1}\rangle_{\mathbb{M}^{l+1}}=\\ &(S^l g^l)^T\mathbb{M}^{l+1}S^l h^l= \quad (since\ g^{l+1}=S^l g^l\ and\ same\ for\ h)\\ &(g^l)^T\mathbb{M}^l h^l= \quad (since\ \mathbb{M}^l=(S^l)^T\mathbb{M}^{l+1}S^l)\\ &=\langle g^l,h^l\rangle_{\mathbb{M}^l},\end{aligned}\tag{20}$$

which completes the proof. Note that the same reasoning holds for inner product of matrices.

Lemma 1 The prolonged eigenvectors and eigenvalues $\hat{\Phi}^{l+1},\hat{\Lambda}^{l+1}$ are *weak* eigenvectors and eigenvalues of $\mathbb{L}^{l+1}$ with respect to functions of level $l+1$ which are in the image of S^l. Explicitly, for any function $g^{l+1}\in Im(S^l)$ we have:

$$\langle g^{l+1},\mathbb{L}^{l+1}\hat{\Phi}^{l+1}-\hat{\Phi}^{l+1}\hat{\Lambda}^{l+1}\rangle_{\mathbb{M}^{l+1}}=0,\tag{4}$$

$$\langle\hat{\Phi}^{l+1},\hat{\Phi}^{l+1}\rangle_{\mathbb{M}^{l+1}}=I.\tag{5}$$

Proof. Since $g^{l+1}\in Im(S^l)$ there exists a function $g^l\in\mathbb{R}^{n^l}$ such that $g^{l+1}=S^l g^l$. Then, we have:

$$\begin{aligned}&\langle g^{l+1},\mathbb{L}^{l+1}\hat{\Phi}^{l+1}-\hat{\Phi}^{l+1}\hat{\Lambda}^{l+1}\rangle_{\mathbb{M}^{l+1}}=\\ &\quad\langle g^l,\mathbb{L}^l\Phi^l\rangle_{\mathbb{M}^l}-\langle g^l,\Phi^l\Lambda^l\rangle_{\mathbb{M}^l}= \quad (from\ Lemma\ 5,\ Def.\ 1)\\ &\quad(g^l)^T(\mathbb{W}^l\Phi^l-\mathbb{M}^l\Phi^l\Lambda^l)=0, \quad (from\ Eq.\ 1)\end{aligned}\tag{21}$$

which completes the proof of the first part. For the second part, note that

$$\langle\hat{\Phi}^{l+1},\hat{\Phi}^{l+1}\rangle_{\mathbb{M}^{l+1}}=\langle S^l\Phi^l,S^l\Phi^l\rangle_{\mathbb{M}^{l+1}}=\langle\Phi^l,\Phi^l\rangle_{\mathbb{M}^l}=I,\tag{22}$$

where we again used Definition 1, Lemma 5 and Equation 1, in this order.

Lemma 2 Let $g\in Im(S^l)$. Then

$$\|g-\hat{\Phi}\hat{\Phi}^\dagger g\|^2_{\mathbb{M}}\le\frac{\|\nabla g\|^2_{\mathbb{M}}}{\lambda_{k+1}}\le\frac{\lambda_{max}}{\lambda_{k+1}}\|g\|^2_{\mathbb{M}},\tag{6}$$

where all quantities are at level $l+1$, $\hat{\Phi}$ are the first k eigenvectors of $\mathbb{L}^l$ prolonged to $l+1$, λ_{k+1} is the $k+1^{\text{th}}$ eigenvalue of $\mathbb{L}^l$, λ_{max} is the largest eigenvalue of $\mathbb{L}^l$, and ∇g is a discrete gradient defined such that $\|\nabla g\|^2_{\mathbb{M}}=\langle g,\mathbb{L}g\rangle_{\mathbb{M}}$.

Proof. Set Φ^l,Λ^l to be the first k eigenvectors and eigenvalues of $\mathbb{L}^l$, and $\bar{\Phi}^l,\bar{\Lambda}^l$ the remaining n^l-k eigenvectors and eigenvalues. Thus, we have:

$$\mathbb{L}^l=\left(\Phi^l\Lambda^l(\Phi^l)^T+\bar{\Phi}^l\bar{\Lambda}^l(\bar{\Phi}^l)^T\right)\mathbb{M}^l.\tag{23}$$

Since $g^{l+1}\in Im(S^l)$ there exists a function $g^l\in\mathbb{R}^{n^l}$ such that $g^{l+1}=S^l g^l$. Now, using Lemma 5 we get:

$$\langle g^{l+1},\mathbb{L}^{l+1}g^{l+1}\rangle_{\mathbb{M}^{l+1}}=\langle g^l,\mathbb{L}^l g^l\rangle_{\mathbb{M}^l}.\tag{24}$$

Plugging in Equation 23 leads to:

$$\begin{aligned}\langle g^l,\mathbb{L}^l g^l\rangle_{\mathbb{M}^l}&=(g^l)^T\mathbb{M}^l\left(\Phi^l\Lambda^l(\Phi^l)^T+\bar{\Phi}^l\bar{\Lambda}^l(\bar{\Phi}^l)^T\right)\mathbb{M}^l g^l\\ &\ge(g^l)^T\mathbb{M}^l\bar{\Phi}^l\bar{\Lambda}^l(\bar{\Phi}^l)^T\mathbb{M}^l g^l\\ &\ge\lambda_{k+1}(g^l)^T\mathbb{M}^l\bar{\Phi}^l(\bar{\Phi}^l)^T\mathbb{M}^l g^l,\end{aligned}\tag{25}$$

where in the last step we used $\bar{\Lambda}^l\ge\lambda_{k+1}I$ entrywise, since λ_{k+1} is the smallest eigenvalue in $\bar{\Lambda}$. Now we have:

$$\begin{aligned}&\|g^l-\Phi^l(\Phi^l)^\dagger g^l\|^2_{\mathbb{M}^l}=\\ &=\|\bar{\Phi}^l(\bar{\Phi}^l)^\dagger g^l\|^2_{\mathbb{M}^l} \quad (since\ [\Phi,\bar{\Phi}]\ is\ a\ full\ basis)\\ &=\|\bar{\Phi}^l(\bar{\Phi}^l)^T\mathbb{M}^l g^l\|^2_{\mathbb{M}^l} \quad (since\ \bar{\Phi}^l is\ ortho\ wrt\ \mathbb{M}^l)\\ &=(g^l)^T\mathbb{M}^l\bar{\Phi}^l(\bar{\Phi}^l)^T\mathbb{M}^l\bar{\Phi}^l(\bar{\Phi}^l)^T\mathbb{M}^l g^l\\ &=(g^l)^T\mathbb{M}^l\bar{\Phi}^l(\bar{\Phi}^l)^T\mathbb{M}^l g^l \quad (since\ \bar{\Phi}^l is\ ortho\ wrt\ \mathbb{M}^l).\end{aligned}\tag{26}$$

On the other hand, we have:

$$\begin{aligned}&\|g^l-\Phi^l(\Phi^l)^\dagger g^l\|^2_{\mathbb{M}^l}=\\ &=\|S^l(g^l-\Phi^l(\Phi^l)^\dagger g^l)\|^2_{\mathbb{M}^{l+1}} \quad (from\ Lemma\ 5)\\ &=\|g^{l+1}-\hat{\Phi}^{l+1}(\Phi^l)^T\mathbb{M}^l g^l\|^2_{\mathbb{M}^{l+1}}=\\ &=\|g^{l+1}-\hat{\Phi}^{l+1}(\hat{\Phi}^{l+1})^T\mathbb{M}^{l+1}g^{l+1}\|^2_{\mathbb{M}^{l+1}}=\\ &=\|g^{l+1}-\hat{\Phi}^{l+1}(\hat{\Phi}^{l+1})^\dagger g^{l+1}\|^2_{\mathbb{M}^{l+1}},\end{aligned}\tag{27}$$

where the last step is due to Lemma 1 which implies that $\hat{\Phi}^{l+1}$ is orthonormal with respect to $\mathbb{M}^{l+1}$. Combining all the results we get:

$$\begin{aligned}&\|g^{l+1}-\hat{\Phi}^{l+1}(\hat{\Phi}^{l+1})^\dagger g^{l+1}\|^2_{\mathbb{M}^{l+1}}=\|g^l-\Phi^l(\Phi^l)^\dagger g^l\|^2_{\mathbb{M}^l}=\\ &=(g^l)^T\mathbb{M}^l\bar{\Phi}^l(\bar{\Phi}^l)^T\mathbb{M}^l g^l\le\tfrac{1}{\lambda_{k+1}}\langle g^l,\mathbb{L}^l g^l\rangle_{\mathbb{M}^l}=\\ &=\tfrac{1}{\lambda_{k+1}}\langle g^{l+1},\mathbb{L}^{l+1}g^{l+1}\rangle_{\mathbb{M}^{l+1}},\end{aligned}\tag{28}$$

which completes the proof of the first bound.

Using the generalized Courant-Fisher Minimax Theorem [ANT09]Thm

3.4 we further have that $\langle g^l, \mathbb{L}^l g^l \rangle^l_{\mathbb{M}} \leq \lambda_{max} \|g^l\|^2_{\mathbb{M}^l}$, where λ_{max} is the largest eigenvalue of $\mathbb{L}^l$, completing the proof of the second bound.

Lemma 3 Assume $\tilde{\Lambda}^l$ and $\bar{\Lambda}^{l+1}$ are distinct, and both have no repeating eigenvalues. Let Φ^{l+1} be the first $\sum_{i=0}^{l} k^i = \tilde{k}^l$ eigenvectors of $\mathbb{L}^{l+1}$. Then the HFM basis $\tilde{\Phi}$ is *approximately blockwise orthonormal*:

$$\langle \tilde{\Phi}^{l+1}, \tilde{\Phi}^{l+1} \rangle_{\mathbb{M}^{l+1}} = \begin{bmatrix} \langle \tilde{\Phi}^l, \tilde{\Phi}^l \rangle_{\mathbb{M}^l} & E \\ E^T & I_{k^{l+1} \times k^{l+1}} \end{bmatrix}, \tag{9}$$

$$\langle \tilde{\Phi}^0, \tilde{\Phi}^0 \rangle_{\mathbb{M}^0} = I_{k^0 \times k^0}, \tag{10}$$

where the error matrix E is controlled by

$$\frac{1}{k^{l+1}} \sum_{ij} |E_{ij}| \leq \|S^l \tilde{\Phi}^l - \Phi^{l+1}\|^2_{\mathbb{M}^{l+1}}. \tag{11}$$

Proof. According to Definition 2 we have that $\tilde{\Phi}^{l+1} = [S^l \tilde{\Phi}^l, \bar{\Phi}^{l+1}]$, therefore we get that $\langle \tilde{\Phi}^{l+1}, \tilde{\Phi}^{l+1} \rangle_{\mathbb{M}^{l+1}}$ is a blockwise matrix of inner products $\begin{bmatrix} A & E \\ E^T & B \end{bmatrix}$. Where,

$$\begin{aligned} A &= \langle S^l \tilde{\Phi}^l, S^l \tilde{\Phi}^l \rangle_{\mathbb{M}^{l+1}} = \langle \tilde{\Phi}^l, \tilde{\Phi}^l \rangle_{\mathbb{M}^l}, \\ B &= \langle \bar{\Phi}^{l+1}, \bar{\Phi}^{l+1} \rangle_{\mathbb{M}^{l+1}} = I \\ E &= \langle S^l \tilde{\Phi}^l, \bar{\Phi}^{l+1} \rangle_{\mathbb{M}^{l+1}}. \end{aligned} \tag{29}$$

To obtain the bound on E, note that $\langle \Phi^{l+1}, \bar{\Phi}^{l+1} \rangle_{\mathbb{M}^{l+1}} = 0$, since their corresponding eigenvalues are distinct. Hence we have:

$$E = \langle S^l \tilde{\Phi}^l, \bar{\Phi}^{l+1} \rangle_{\mathbb{M}^{l+1}} = \langle S^l \tilde{\Phi}^l - \Phi^{l+1}, \bar{\Phi}^{l+1} \rangle_{\mathbb{M}^{l+1}}. \tag{30}$$

Denote $A = S^l \tilde{\Phi}^l - \Phi^{l+1}, B = \bar{\Phi}^{l+1}, M = \mathbb{M}^{l+1}$, thus $E = \langle A, B \rangle_M$, and let A_i be the i-th column of A. We have:

$$\sum_{ij} |E_{ij}| = \sum_{ij} |\langle A_i, B_j \rangle_M| \leq \sum_{ij} \|A_i\|^2_M \|B_j\|^2_M, \tag{31}$$

where for the last step we used the Cauchy-Schwarz inequality. Set $\bar{a}$ to be the diagonal of the matrix $\langle A, A \rangle_M$, namely $\bar{a}(i) = \|A_i\|^2_M$, and similarly set $\bar{b}$ to the diagonal of $\langle B, B \rangle_M$. In addition, let $\mathbf{1}$ be a column vector of ones:

$$\sum_{ij} \|A_i\|^2_M \|B_j\|^2_M = \mathbf{1}^T (\bar{a}\bar{b}^T) \mathbf{1} = \|A\|^2_M \|B\|^2_M, \tag{32}$$

since $\bar{b}^T \mathbf{1} = \mathrm{tr}(B^T M B) = \|B\|^2_M$, and similarly for A. Finally, note that $\|B\|^2_M = \mathrm{tr}(\langle \bar{\Phi}^{l+1}, \bar{\Phi}^{l+1} \rangle_{\mathbb{M}^{l+1}}) = \mathrm{tr}(I_{k^{l+1}})$, and therefore $\|B\|^2_M = k^{l+1}$. Combining all the results, and plugging back the definition of A we have:

$$\sum_{ij} |E_{ij}| \leq \|A\|^2_M \|B\|^2_M = \|S^l \tilde{\Phi}^l - \Phi^{l+1}\|^2_M k^{l+1}. \tag{33}$$

If the first $\tilde{k}^l$ eigenvalues at level l and at level $l+1$ include no repeating eigenvalues, then the basis vectors in Φ^{l+1} are expected to correspond (up to sign) to the prolonged eigenvectors (see Figures 5, 6), reducing the error on the right hand side.

Lemma 4 Let $\tilde{K}^{l+1}_{\rho,t}$ be the HSD at level $l+1$, then:

$$\tilde{K}^{l+1}_{\rho,t} = S^l \tilde{K}^l_{\rho,t} (S^l)^T + \bar{\Phi}^{l+1} \rho_t(\bar{\Lambda}^{l+1}) (\bar{\Phi}^{l+1})^T. \tag{12}$$

Proof. By definition, we have that

$$\tilde{K}^{l+1}_{\rho,t} = \tilde{\Phi}^{l+1} \rho_t(\tilde{\Lambda}^{l+1}) (\tilde{\Phi}^{l+1})^T.$$

Plugging in the definition of the hierarchical basis

$$\tilde{\Phi}^{l+1} = [\mathbb{S}^l \tilde{\Phi}^l, \bar{\Phi}^{l+1}], \quad \tilde{\Lambda}^{l+1} = [\tilde{\Lambda}^l, \bar{\Lambda}^{l+1}],$$

we get:

$$\begin{aligned} \tilde{K}^{l+1}_{\rho,t} &= \begin{bmatrix} \mathbb{S}^l \tilde{\Phi}^l & \bar{\Phi}^{l+1} \end{bmatrix} \begin{bmatrix} \rho_t(\tilde{\Lambda}^l) & 0 \\ 0 & \rho_t(\bar{\Lambda}^{l+1}) \end{bmatrix} \begin{bmatrix} (\mathbb{S}^l \tilde{\Phi}^l)^T \\ (\bar{\Phi}^{l+1})^T \end{bmatrix} = \\ &= \mathbb{S}^l \tilde{\Phi}^l \rho_t(\tilde{\Lambda}^l) (\tilde{\Phi}^l)^T (\mathbb{S}^l)^T + \bar{\Phi}^{l+1} \rho_t(\bar{\Lambda}^{l+1}) (\bar{\Phi}^{l+1})^T = \\ &= \mathbb{S}^l \tilde{K}^l_{\rho,t} (\tilde{\Phi}^l)^T (\mathbb{S}^l)^T + \bar{\Phi}^{l+1} \rho_t(\bar{\Lambda}^{l+1}) (\bar{\Phi}^{l+1})^T. \end{aligned} \tag{34}$$