Hierarchical Functional Maps between Subdivision Surfaces

Meged Shoham\textsuperscript{1} Amir Vaxman\textsuperscript{2} Mirela Ben-Chen\textsuperscript{1}

\textsuperscript{1}Technion - Israel Institute of Technology
\textsuperscript{2}Utrecht University

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
\begin{itemize}
\item Computing methodologies \rightarrow Mesh models;
\end{itemize}

1. Introduction

Subdivision surfaces are a popular shape representation for 3D modeling, used in the design pipeline of many artists [LVGL\textsuperscript{*}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\textsuperscript{*}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

\textsuperscript{1} 2019 The Author(s)
\textsuperscript{2} Computer Graphics Forum \textsuperscript{©} 2019 The Eurographics Association and John Wiley & Sons Ltd. Published by John Wiley & Sons Ltd.
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-Clark [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 Lité 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 [GBK18], 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. Extellas 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 [Zot06]. 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].
2.1. Notation

We work with a polygonal mesh \( M = (V, F, \mathcal{E}) \), given by its vertices, faces and edges, respectively. We denote \(|V| = n\) and \(|F| = m\), 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 on \( M \) and functions \( g \) and \( h \) from \( X \) to \( \mathbb{R}^d \), represented by their basis coefficients, to functions in the image of \( \Phi \), and is thus given by \( C_{12} = \Phi_1^T P_{12} \Phi_2 \).

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 \( T_{12} : V_1 \to V_2 \), from the vertices of \( M_1 \) to the vertices of \( M_2 \), there exists a corresponding functional map (FMap) \( P_{12} \) that maps PL functions on \( M_2 \) to PL functions on \( M_1 \). It is given by \( P_{12}(g_2) = g_2(T_{12}(v_1)) \) for all vertices \( v_1 \in V_1 \), and functions \( g_2 \in \mathbb{R}^{F_2} \). It is easy to check that \( P_{12} \) is a linear operator, and thus can be described by a matrix \( P_{12} \in \mathbb{R}^{n \times m} \).

The main strength of the functional map framework comes from working with a spectral basis for functions, usually taken to be \( \Phi \), 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 \( \Phi_2 \), represented by their basis coefficients, to functions in the image of \( \Phi_1 \), and is thus given by \( C_{12} = \Phi_1^T P_{12} \Phi_2 \).

2.1.3. Inference

To compute a functional map \( C_{12} \) between two meshes \( M_1, M_2 \), 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 \( F_1 \in \mathbb{R}^{k_1 \times d} \) and (2) commutativity constraints of the form \( C_{12} O_2 = O_1 C_{12} \), where \( O_i \in \mathbb{R}^{k_i \times k_i} \) is a linear operator on \( M_i \). Both the descriptors, \( F_i \), and the operators, \( O_i \), are given through their projection on the spectral bases \( \Phi_i \). 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 \( \rho \) is applied entry-wise to the diagonal of \( \Lambda \). Landmarks, on the other hand, require the knowledge of two corresponding vertices \( v_i \in V_i \) per landmark. Given a vertex \( v \in V \), 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 \( \Phi_i \) 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 $P_{12} \in \mathbb{R}^{n_2 \times n_1}$, from which a vertex-to-vertex map, $f_{12}$, 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 $\|C_{12} \Phi^* - \Phi^* P_{12}\|^2$ 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 $\mathcal{M}^i = (V^i, F^i, \mathcal{E}^i)$, with $i \in \{0, \ldots, f\}$, where $\mathcal{M}^0$ is the base mesh. Following de Goes et al. [dGDMD16], we define a subdivision matrix $S^i \in \mathbb{R}^{d+1 \times d}$ for the vertices at level $i$. Hence, the embedding of $V^{i+1}$ is given by $X^{i+1} = S^i X^i$, for $i > 0$, taking $X^0 = X^0$. We accumulate the subdivision of multiple levels by multiplying the corresponding subdivision matrices, namely $S^{i+1} = S^{i-1} S^{i-2} \ldots S^2 S^1$ for $0 \leq i < f$. 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 straightforward to show that we can compute the SEC unweighted Laplace-Beltrami operator by $W^l = (S^l)^T W^l S^l$. Similarly, the SEC mass matrix for the vertices is given by $M^l = (S^l)^T M^l S^l$. Note, that since the finest subdivision level $l$ 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.](image)

3. Hierarchical Functional Maps (HFM)

3.1. Requirements

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 $S^1$, normalized by the functions’ squared norm, as a function of the number of eigenvectors $k$. We compare using the prolonged eigenvectors $\Phi^{l+1}$, with the optimal representation error achieved by using $\Phi^{l+1}$. Note that while the exact eigenfunctions achieve a slightly better representation error for larger $k$ values, the graphs are almost indistinguishable.](image)
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 \( l \in \{0, \ldots, f \} \), given by \( L^l = (M^l)^{-1}W^l \), is positive semi-definite [dGDMD16], and thus we can compute its lowest \( k \) eigenvectors and eigenvalues, given by \( \Phi^l \in \mathbb{R}^{d \times k} \) and \( \Lambda^l \in \mathbb{R}^k \), respectively. By definition, these fulfill:

\[
(W^l \Phi^l - M^l \Phi^l) \Lambda^l = 0, \quad (\Phi^l, \Phi^l)_{M^l} = I. \tag{1}
\]

It is natural to consider the relation between the SEC Laplacian operators on multiple levels. First, note that for all levels \( l \in \{0, \ldots, f \} \), the operator \( L^l \) depends on the geometry of the finest level \( f \), given by \( M^f, W^f \), and on the multilevel subdivision operator \( S^l \), 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 \( \Phi^l, \Lambda^l \) and \( \Phi^{l+1}, \Lambda^{l+1} \). Indeed, this follows directly from the definitions and Equation (1):

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

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

**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 \( L^{l+1} \) with respect to functions of level \( l+1 \) which are in the image of \( S^l \). Explicitly, for any function \( g^{l+1} \in \text{Im}(S^l) \) we have:

\[
\langle g^{l+1} - \Phi^{l+1} \hat{\Phi}^{l+1}, M^{l+1} \rangle = 0, \tag{4}
\]

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

All the proofs, though elementary, are provided in the appendix for completeness. The important point is that the inner products \( \langle \cdot, \cdot \rangle_{M} \) and \( \langle \cdot, \cdot \rangle_{M^l} \) 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 \( L^{l+1} \), 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 \( g \in \text{Im}(S^l) \). Then:

\[
\|g - \Phi g\|^2_{M^l} \leq \frac{\|\nabla g\|^2_{M^l}}{\lambda_{k l + 1}} \leq \frac{\lambda_{\max}}{\lambda_{k l + 1}} \|g\|^2_{M^l}, \tag{6}
\]

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

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 \( \nabla g \) is the gradient field of the subdivision of \( g \) to the finest level \( f \), because \( W^f = (S^f)^T W^{f+1} S^f \). On general meshes \( \nabla g \) can be defined through the \( L^f \) norm of the 1-form \( d g \) 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 \( \|g\|^2_{M^l} \), as a function of \( k \), the number of computed eigenvectors. We use \( \Phi^{l+1} \) and \( \Phi^{l+1} \), for a set of 1000 random functions in the image of \( S^l \). Note that while \( \Phi^{l+1} \) leads to a better error after 100 eigenvectors, the difference is smaller than 1%.

Figure 5 visualizes the eigenvectors \( \Phi^{l+1} \) 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 \( L^l \) 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 \( \Lambda^{l+1} / \Lambda^l \) for the different subdivision levels.

The bound in Lemma 2 depends on the largest eigenvalue of \( L \). 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 \( \hat{k} \) be the total number of required basis vectors at the finest level \( f \). Define \( \{k_l^l \geq 0, l \in \{0, \ldots, f \} \} \), such that \( \sum_{l=0}^{f} k_l^l = \hat{k} \).

\[\text{Figure 5: A few low eigenvectors and the corresponding prolonged eigenvectors from a level below. Note the clear visual similarity.}\]
The hierarchical basis is given by:
\[
\Phi^0 = \Phi_0, \quad \Lambda^0 = \Lambda^0,
\]
\[
\Phi^{l+1} = \left[\tilde{S}^l \Phi^l, \Phi^{l+1}\right], \quad \Lambda^{l+1} = [\Lambda^l, \Lambda^{l+1}],
\]
where \(\Phi^0, \Lambda^0\) are the first \(k^0\) eigenvectors and eigenvalues of \(\mathbb{L}^0\), \(\Phi^{l+1}, \Lambda^{l+1}\) are the \(k^{l+1}\) eigenvectors and eigenvalues of \(\mathbb{L}^{l+1}\) in the band starting after the largest eigenvalue of \(\Lambda^l\), 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 \(\mathbb{R}^k\), 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 eigenvalues 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.

\[
\Lambda^{l+1} \approx \Lambda^l + \frac{1}{k^{l+1}} \sum_{ij} |E_{ij}| \leq \|S^l \tilde{\Phi} - \Phi^{l+1}\|_{2^l}.
\]

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 \(\sum_{k^l} k^l\) 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 \(\rho_t : \mathbb{R}^+ \rightarrow \mathbb{R}^+\), where \(t \in \mathbb{R}^+\), is a filter on the eigenvalues.
For example, the heat kernel map [SOG09] of a vertex $v \in V$ is given by $\Phi v = \Phi t \Phi^T$, with $\Phi t = \exp(-\lambda t)$. 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 Descriptor Matrix (HSD) is given by $K_p = \Phi \Lambda \Phi^T$. We remove the level notation, as all the quantities are at the same subdivision level.

We additionally define a hierarchical landmark descriptor using $K_{p,i,s} = \delta_i \Phi$, for a vertex $v \in V$, and a hierarchical signature descriptor using $K_{p,i} = \diag(K_{p,i})$.

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 $K_{p,i}^{l+1}$ be the HSD at level $l+1$, then:

$$ K_{p,i}^{l+1} = S^l K_{p,i} (S^l)^T + \Phi_i \Lambda_i^{l+1} (\Phi_i^{l+1})^T. \tag{12} $$

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 $K_{p,i}$, but only the landmark descriptors $K_{p,i,s}$ for a given set of input landmarks, and the signature descriptor $K_{p,i}^s$, which is given by the diagonal. The landmark descriptors are efficiently computed in the spectral domain, see Section 6.2.2, and the signatures are projected onto the HFM basis $\Phi_i$. This yields the descriptor matrices $F_i^s \in \mathbb{R}^{k \times d}$, 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 $C_{12} \Psi = \Psi C_{12}$, where $\Psi = \Phi_{i} L \Phi_{i} = \Phi_{i} M_{i} \Phi_{i} \Lambda_{i} M_{i} \Phi_{i} = \Lambda_{i}$. We therefore leverage the approximate orthogonality property of the HFM basis and set the Laplacian commutativity operator accordingly, taking: $\tilde{\Omega}_i = \tilde{\Lambda}_i$.

**Symmetry**: In many cases, especially if designed by artists, the input surfaces have intrinsic self-symmetry given as input self-maps, or permutations, $\Psi_0 \in \mathbb{R}^{k \times n}$. 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 $l$, given by $S^l \tilde{S}^l X^l$, to find the symmetry map $S^{l+1}$. 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: $\tilde{\Omega}_i = (\Phi_i ')^T S^l \Phi_i '$.

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

Given the hierarchical construction of the basis $\Phi_l$, the descriptors $F_l$, and the commutativity operators $O_l$, we proceed to optimize for the Hierarchical Functional Map $C_{l,12}$, as follows.

5.3.1. Coarse level

Solve for $C_{l,12}^0$ using the standard functional map optimization scheme, by minimizing:

$$C_{l,12}^0 = \arg \min_C \|CF_l^2 - F_l^1\|_F^2 + \sum_O \alpha(O) \|CO_0^2 - O_1^0C\|_F^2,$$

where the sum goes over all available commutativity operators, either Laplacian, or symmetry or both, $\alpha$ 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 $C_{l,12}^{l+1}$ from level $l$, we compute the functional map at the next level as follows. We define the solution as the matrix:

$$C^{l+1} = \begin{bmatrix} C^{l+1}_l & \tilde{C}^{l+1}_{l+1} \end{bmatrix},$$

where $h = l+1$ and for clarity we removed the subscript notation. The matrix block $\tilde{C}^{l+1} \in \mathbb{R}^{k^h \times k^h}$, for example, transfers high frequencies on $M_2$ to low frequencies on $M_1$.

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

$$\tilde{F}^{l+1} = \begin{bmatrix} F_l^1 \\ F_l^h \end{bmatrix}, \quad \tilde{O}^{l+1} = \begin{bmatrix} O^l & 0 \\ 0 & O^h \end{bmatrix},$$

where, e.g., $F_l^1 \in \mathbb{R}^{k^l \times d}$ contains the coefficients of the descriptors in the low eigenvectors of $l$, and similarly for the other matrices. The constraint matrices $\tilde{F}^{l+1}$, $\tilde{O}^{l+1}$ are computed in terms of coefficients of $\tilde{O}^{l+1}$ 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 $\tilde{C}^{l+1}$ independently of the other two unknown blocks:

$$\tilde{C}^{l+1}_{l,12} = \arg \min_C \|CF_l^2 - F_l^1 + \tilde{C}^{l+1}_{l,12}||_F^2 + \sum_O \alpha(O) \|CO_0^2 - O_1^0C\|_F^2.$$

The equations for $C^{h-l}$ and $C^h$ 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 $(k^h)^2$ variables, whereas at the level $h$ we solve two systems, with $k^h$ and $k^h k^h$ variables, respectively. Since the linear solve scales cubically in the number of variables, these reductions are significant. Dense operators will break the block diagonal structure of $\tilde{O}$, and then all the three unknown matrix blocks will be coupled. Nevertheless, solving for $k^h k^h + k^h k^h$ variables is still much more efficient than solving for $(k^h)^2$ 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 $P_l$ from the computed spectral functional map $C_{l,12}$, is to solve the optimization problem: $\arg \min_{PC} \|CO_2^2 - O_2^1P\|^2$ alternatingly for $P$ and $C$, starting from the inferred functional map $C_{l,12}$. Optimizing for $P$ while keeping $C$ fixed is implemented using a nearest neighbor
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 $P_0^{f}$ from $C_0^{f}$, 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 $P_{12}^{f+1}$ 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 $S_f^{\dagger}$. 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 $M_2$ to be arbitrary points on the subdivision surface, and not necessarily vertices of the finest level $V_2^{f}$. This is achieved by subdividing the spectral embedding of $M_2$ and optimizing for a permutation $P$:

$$P_{12}^{f+1} = \arg \min_p \|C_1^{f+1} (S_{12}^{f+1} \Phi_f^{f+1})^T \Phi_f^{f+1} P \|^2,$$

(17)

where the dimensions of $P_{12}^{f+1}$ are $n_f^{f+1} \times n_f^{f+1}$, 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, $n_f^{f} \times n_f^{f}$. Note that $P_{12}^{f}$ is a general row stochastic matrix, that encodes a vertex-to-vertex map, from $V_f^{f}$ to $V_f^{f+1}$. 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 $\sum_{k^l=0}^{k^l} k^l = k$. For each band we employ Matlab’s eigs solver [Mat18, LSY98, Ste02], that computes eigenvalues and eigenvectors near a given $\sigma$ 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 [Ivu16] 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 $\Phi^0, \bar{\Lambda}^0$.

**Level** $l+1$.

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

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 $\Phi^l$, where we filter eigenvectors with a maximal projection norm larger than $\epsilon$. In all our experiments we used $h = 15$, and $\epsilon = 0.4$.

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 $\Phi$ 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 $v^f \in V_f$ 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 $S_f^{\dagger}$. Therefore, we compute the fine landmark descriptor as the corresponding convex combination of the coarse landmark descriptors, using $\Phi_{f, \delta} = \Phi_{f, \delta}^{\dagger} S_f^{\dagger} \delta$. The same applies for placing landmarks at any level $h \leq f$, and computing descriptors at any level $0 \leq l < h$, by taking $\Phi_{f, \delta}^{l} \delta^{h}$ and $S_f^{h}$. 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 $v \in V$ is given by Definition 3:

$$\hat{K}^l_{f, \delta} = \phi^l_{f, \delta} (\bar{\Lambda}^l)^{-\frac{1}{2}} \delta.$$ 

For the functional optimization map we only need the basis coefficients of these descriptors, which are given by $\langle \Phi^l \rangle^T \hat{K}^l_{f, \delta} = \phi^l_{f, \delta} (\bar{\Lambda}^l)^{-\frac{1}{2}} \delta$. Denoting the $i$-th row of $\Phi^l$ by $\phi_{f, \delta}^i$, we therefore
have that the coefficients of the landmark descriptor are $p_i(\tilde{A}^t\Phi^t)$, which is a vector of size $k^l_1$. Thus, we can compute the coefficients directly, without computing the full descriptor first.

The same applies for descriptors of landmarks $\phi^h$ at finer levels $h > l$, where the basis coefficients are $p_i(\tilde{A}^t(\tilde{S}_h^l\Phi^h)^t)\delta_{ik}$.

### 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 $M_1$ and $M_2$.

**HFM Basis.** In all the experiments, we set $k^0 = 100$, and $k^l_1 = k^l_2 = 50$ 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 and 21 landmarks per shape for the landmark descriptors, depending on the deviation from isometry of the expected map, with more landmarks required for less isometric shapes. We use WKS and WK 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 $\alpha$ 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](https://example.com/figure11.png)

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

---

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](https://example.com/figure12.png)

**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-hierarchical functional maps scheme FMAPS12 [OBCS12], 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 [OBCS12], 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.
We measure map quality by the conformal distortion step, in average close to 70% of the total computation. The most time-consuming step in our approach is the P2P reconstruction step, in average close to 70% of the total computation. We compare our method to HOT [AL16], INF [NO17] and BCICP [RPW018] 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 25x slower than our approach.

7.4. Application: Transferring textures

Given a pointwise map $P f_{n}$ between the finest subdivision levels we can transfer texture images.

Assuming both models have texture coordinates $U f_{m}$ that are subdivided to the finest level, and given a texture image for $M_{2}$, we construct a new texture image for $M_{1}$. This is done by first computing the deformed texture coordinates of $M_{1}$, given by $\hat{U}_{I} = (P f_{n})^T P f_{m}(P f_{m})U_{f_{m}}$, where all the quantities are at the finest level. Here, $P f_{m}$ maps texture vertices to model vertices, and vice versa for $P f_{n}$. Next, the model is saved and rendered, with coordinate locations given by $U_{I}$ and texture coordinates given by $\hat{U}_{I}$. The resulting image is the new texture image for $M_{1}$.

A technical issue remains—the texture seams of $M_{2}$ do not necessarily correspond to texture seams of $M_{1}$, leading to visible artifacts in the new texture. To remedy this, we identify quads of $M_{1}$ that are mapped to vertices in the 1-ring neighborhood of the texture seams of $M_{2}$, and remove them from the rendering, leading to missing texture regions. Finally, we use an off-the-shelf image inpainting tool [bap18] 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.

7.7. Application: Sculpting textures

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.

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

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 paired 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


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 transfer.


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.


The eigenvalues and eigenvectors of $L_{k+1}$ are weak eigenvectors and eigenvalues of $L_{k+1}$ with respect to functions of level $l+1$ which are in the image of $S'$. Explicitly, for any function $g^{k+1} \in \text{Im}(S')$ we have:

$$
\langle (g^{k+1}, L_{k+1} g^{k+1})_{S'_{k+1}} = 0,
$$

(4)

and

$$
\langle (\Phi^{k+1}, \Phi^{k+1})_{S'_{k+1}} = I,
$$

(5)

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

$$
\langle (g^{k+1}, L_{k+1} g^{k+1})_{S'_{k+1}} = (g', L_{k} \Phi)_{S'_{k+1}} - \Phi^{k+1} L_{k+1} \Phi^{k+1} = 0
$$

and

$$
\langle (\Phi^{k+1}, \Phi^{k+1})_{S'_{k+1}} = I
$$

(6)

which completes the proof. Note that the same reasoning holds for inner eigenvectors and eigenvalues of $L_{k+1}$. And similarly,

$$
\langle (g^{k+1} g^{k+1})_{S'_{k+1}} = 0.
$$

(7)

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

$$
\langle (g^{k+1}, L_{k+1} g^{k+1})_{S'_{k+1}} = (g', L_{k} h')_{S'_{k+1}},
$$

(8)

and

$$
\langle (\Phi^{k+1}, \Phi^{k+1})_{S'_{k+1}} = (g', L_{k} h')_{S'_{k+1}}.
$$

(9)

Proof.

$$
\langle (g^{k+1}, L_{k+1} g^{k+1})_{S'_{k+1}} = (g', L_{k} h')_{S'_{k+1}}
$$

(10)

and

$$
\langle (\Phi^{k+1}, \Phi^{k+1})_{S'_{k+1}} = (g', L_{k} h')_{S'_{k+1}}.
$$

(11)

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

### Lemma 1

The (weak) eigenvectors and eigenvalues $\Phi^{k+1}, \Lambda^{k+1}$ are orthogonal to the remaining $n - k$ eigenvectors and eigenvalues of $L_{k+1}$.

Proof.

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

$$
\langle (g^{k+1}, L_{k+1} g^{k+1})_{S'_{k+1}} = (g', L_{k} \Phi)_{S'_{k+1}} - \Phi^{k+1} L_{k+1} \Phi^{k+1} = 0
$$

and

$$
\langle (\Phi^{k+1}, \Phi^{k+1})_{S'_{k+1}} = I
$$

(6)

which completes the proof. Note that the same reasoning holds for inner eigenvectors and eigenvalues of $L_{k+1}$. And similarly,

$$
\langle (g^{k+1} g^{k+1})_{S'_{k+1}} = 0.
$$

(7)

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

$$
\langle (g^{k+1}, L_{k+1} g^{k+1})_{S'_{k+1}} = (g', L_{k} h')_{S'_{k+1}},
$$

(8)

and

$$
\langle (\Phi^{k+1}, \Phi^{k+1})_{S'_{k+1}} = (g', L_{k} h')_{S'_{k+1}}.
$$

(9)

Proof.

$$
\langle (g^{k+1}, L_{k+1} g^{k+1})_{S'_{k+1}} = (g', L_{k} h')_{S'_{k+1}}
$$

(10)

and

$$
\langle (\Phi^{k+1}, \Phi^{k+1})_{S'_{k+1}} = (g', L_{k} h')_{S'_{k+1}}.
$$

(11)

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

### Lemma 1

The (weak) eigenvectors and eigenvalues $\Phi^{k+1}, \Lambda^{k+1}$ are orthogonal to the remaining $n - k$ eigenvectors and eigenvalues of $L_{k+1}$.

Proof.

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

$$
\langle (g^{k+1}, L_{k+1} g^{k+1})_{S'_{k+1}} = (g', L_{k} \Phi)_{S'_{k+1}} - \Phi^{k+1} L_{k+1} \Phi^{k+1} = 0
$$

and

$$
\langle (\Phi^{k+1}, \Phi^{k+1})_{S'_{k+1}} = I
$$

(6)

which completes the proof. Note that the same reasoning holds for inner eigenvectors and eigenvalues of $L_{k+1}$. And similarly,

$$
\langle (g^{k+1} g^{k+1})_{S'_{k+1}} = 0.
$$

(7)
3.4 we further have that $\langle \vec{g}', \vec{g}' \rangle_{L^2} \leq \lambda_{\max} \| \vec{g}' \|^2_{L^2}$, where $\lambda_{\max}$ is the largest eigenvalue of $L^1$, completing the proof of the second bound.

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

$$
(\vec{\Phi}'^{l+1}, \vec{\Phi}'^{l+1})_{L^2} = \left[ \begin{array}{c} \vec{\Phi}'_M \\ E^T \\ I_{2^{l+1}x2^{l+1}} \end{array} \right],
$$

$$
(\vec{\Phi}'^l, \vec{\Phi}'^l)_M = I_{2^l x 2^l},
$$

where the error matrix $E$ is controlled by

$$
\frac{1}{2^{l+1}} \sum_{ij} |E_{ij}| \leq \| \vec{\Phi}'^{l+1} - \vec{\Phi}'^{l+1} \|_{L^2}.
$$

**Proof.** According to Definition 2 we have that $\vec{\Phi}'^{l+1} = [S' \vec{\phi}', \bar{\lambda}'^{l+1}]$, therefore we get that $\langle \vec{\Phi}'^{l+1}, \vec{\Phi}'^{l+1} \rangle_{L^2} = 0$, since their corresponding eigenvalues are distinct. Hence we have:

$$
E = \langle \vec{\Phi}'^{l+1}, \vec{\Phi}'^{l+1} \rangle_{L^2} = \langle S' \vec{\phi}' - \vec{\Phi}'^{l+1}, \vec{\Phi}'^{l+1} \rangle_{L^2},
$$

Denote $A = S' \vec{\phi}' - \vec{\Phi}'^{l+1}$, $B = \vec{\Phi}'^{l+1}, M = 2^{l+1}$, thus $E = \langle A, B \rangle_M$, and let $A_i$ be the $i$-th column of $A$. We have:

$$
\sum_{ij} |E_{ij}| = \sum_{ij} |(A_i, B_j)_M| \leq \sum_{ij} \|A_i\|_M \|B_j\|_M,
$$

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 $\bar{a}(i) = \|A_i\|_M^2$, and similarly set $\bar{b}$ to the diagonal of the $B, B_M$. In addition, let $1$ be a column vector of ones:

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

since $\bar{b}^T 1 = \text{tr}(B^T MB) = \|B\|_M^2$, and similarly for $A$. Finally, note that $\|B\|_M^2 = \text{tr}(\vec{\Phi}'^{l+1}, \vec{\Phi}'^{l+1})_{M+1}) = \text{tr}(I_{2^{l+1}})$, and therefore $\|B\|_M^2 = 2^{l+1}$. Combining all the results, and plugging back the definition of $A$ we have:

$$
\sum_{ij} |E_{ij}| \leq \|A\|_M \|B\|_M = \|S' \vec{\phi}' - \vec{\Phi}'^{l+1} \|_{L^2}.
$$

If the first $\vec{\lambda}'$ eigenvalues at level $l$ and at level $l+1$ include no repeating eigenvalues, then the basis vectors in $\vec{\Phi}'^{l+1}$ 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 $K_{p,l}^{l+1}$ be the HSD at level $l+1$, then:

$$
K_{p,l}^{l+1} = S' K_{p,l}^{l} (S')^T + \vec{\Phi}'^{l+1} p_i(\vec{\lambda}'^{l+1}) (\vec{\Phi}'^{l+1})^T.
$$

**Proof.** By definition, we have that $K_{p,l}^{l+1} = \vec{\Phi}'^{l+1} p_i(\vec{\lambda}'^{l+1}) (\vec{\Phi}'^{l+1})^T$.

Plugging in the definition of the hierarchical basis $\vec{\Phi}'^{l+1} = [S' \vec{\phi}', \bar{\lambda}'^{l+1}]$, $\bar{\lambda}'^{l+1} = [\vec{\lambda}', \bar{\lambda}'^{l+1}]$, we get:

$$
K_{p,l}^{l+1} = [S' \vec{\phi}', \bar{\lambda}'^{l+1}] \begin{bmatrix} p_i(\vec{\lambda}') & 0 \\ 0 & p_i(\bar{\lambda}'^{l+1}) \end{bmatrix} \begin{bmatrix} (S' \vec{\phi}')^T \\ (\bar{\lambda}'^{l+1})^T \end{bmatrix} = S' \vec{\phi}' p_i(\vec{\lambda}') (S')^T + \bar{\lambda}'^{l+1} p_i(\bar{\lambda}'^{l+1}) (\bar{\lambda}'^{l+1})^T = S' K_{p,l}^{l} (S')^T + \bar{\lambda}'^{l+1} p_i(\bar{\lambda}'^{l+1}) (\bar{\lambda}'^{l+1})^T.
$$