Compressed Manifold Modes for Mesh Processing

T. Neumann\textsuperscript{1,2}, K. Varanasi\textsuperscript{3,4}, C. Theobalt\textsuperscript{3}, M. Magnor\textsuperscript{2} and M. Wacker\textsuperscript{1}

\textsuperscript{1}HTW Dresden, Germany \textsuperscript{2}TU Braunschweig, Germany \textsuperscript{3}Max-Planck-Institut für Informatik, Saarbrücken, Germany \textsuperscript{4}Technicolor Research, France

Abstract

This paper introduces compressed eigenfunctions of the Laplace-Beltrami operator on 3D manifold surfaces. They constitute a novel functional basis, called the compressed manifold basis, where each function has local support. We derive an algorithm, based on the alternating direction method of multipliers (ADMM), to compute this basis on a given triangulated mesh. We show that compressed manifold modes identify key shape features, yielding an intuitive understanding of the basis for a human observer, where a shape can be processed as a collection of parts. We evaluate compressed manifold modes for potential applications in shape matching and mesh abstraction. Our results show that this basis has distinct advantages over existing alternatives, indicating high potential for a wide range of use-cases in mesh processing.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Hierarchy and geometric transformations

1. Introduction

The eigenfunctions of the Laplace-Beltrami operator of a 3D surface define a basis, known as the manifold harmonic basis (MHB). This basis can be understood as a generalization of the Fourier spectrum for functions defined on a general manifold surface. The Laplace-Beltrami operator captures all the intrinsic properties of the shape and is invariant to extrinsic shape transformations such as isometric deformations. As such, its eigenfunctions constitute a compact and elegant basis for spectral shape processing that is independent of the actual shape representation and parameterization. In fact, the MHB is unique and characteristic of the geometric and topological properties of the shape. Several successful applications have been proposed that took advantage of these de-
They automatically group confined local regions like protrusions and ridges into separate basis functions, Fig. 1. Due to their unique spatial locality, they are robust to significant geometric and topological noise, such as what happens due to partial scans. Thus, the CMB can be considered as a tool for robust shape analysis and matching. At the same time, CMB is an orthogonal basis and can reconstruct any function defined on the shape, up to an arbitrary degree of precision. We qualitatively evaluate the CMB towards developing potential applications in shape matching, shape approximation, and feature detection. A reference implementation of the algorithm as well as all scripts to reproduce the results in this paper are available at https://github.com/tneumann/cmm

2. Background and Related Work

Before introducing the new compressed manifold basis (CMB), let us first give some background on the manifold harmonic basis (MHB) and on their applications in various areas in computer graphics. We will also review recent results on obtaining compactly supported eigenfunctions.

Manifold Harmonic Basis

The Laplace-Beltrami operator $\Delta$ on a 2D manifold surface embedded in 3D space induces a set of eigenfunctions $\phi_k$ which satisfy the classical equations

$$-\Delta \phi_k = \lambda_k \phi_k, \quad k \in \mathbb{N}, \quad \lambda_k \in \mathbb{R},$$

where $\lambda_k$ are the eigenvalues of the operator. The eigenfunctions form a basis that is called the manifold harmonic basis (MHB) [VL08]. In most practical applications, only a set of $K \in \mathbb{N}$ eigenfunctions corresponding to the smallest eigenvalues is required.

Applications

Due to their compactness, encoding efficiency, isometric invariance, and computational efficiency, manifold harmonics can be found in countless applications in computer graphics [LZ10]. Vallet and Lévy [VL08] present several mesh filtering applications and introduced an algorithm that is able to compute the MHB on very large meshes. The recently proposed functional map framework [OBCS+12] uses maps between functions on shapes, for example to transfer segmentations or to estimate correspondences between deformed shapes. Various point signatures based on Laplace-Beltrami eigenfunctions are successfully used in shape matching [Rus07,SOG09]. Other applications include mesh parameterization [MTAD08], shape segmentation [dGGV08] and compression of meshes [KG00]. The MHBs can even help in accurate facial tracking [BPW13].

Sparsity-inducing norms in graphics

The main mathematical tool used in this paper is sparsity-inducing regularization. This concept has been very popular in compressed sensing and machine learning as well as in image- and signal-processing. Recently, applications in computer graphics appeared. For example, Deng et al. [DBD+] show local modification of constrained architectural meshes using sparse regularization methods. Neumann et al. [NWW+] propose a variant of sparse PCA to automatically extract local deformation components from a dataset of meshes, e.g. mesh sequences captured from human facial performances. Pokrass et al. [PBB+] show that sparse regularization can also help to improve correspondence-estimation between deformed shapes. Rustamov [Rus11] proposes a multiresolution kernel which is centered locally around a specific point on the mesh, also using the sparsity-inducing $\ell_1$ norm. Due to a similar sparsity inducing objective, these multi resolution kernels look similar to some of the first compressed modes presented here. However, they look very different for higher number of compressed basis functions $K$. Another important difference is that the multi-resolution kernels [Rus11] are defined with respect to some given central point in the mesh and do not form a basis.

Compressed Modes

Ozoliņš, Lai, and Osher [OLCO13] propose to find compactly supported eigenfunctions of a general differential operator. To this end, they add a sparsity inducing $\ell_1$-norm into a variational formulation of a problem of
3. Computing Compressed Manifold Modes

As a preliminary, we begin by clarifying how we discretize both the Laplace-Beltrami operator $\Delta$ and the eigenvalue equation (1) , as well as the minimization problem (2) . With those building blocks in place, we then derive an algorithm that is able to compute compressed manifold modes by solving (2) .

3.1. Discretization

In this paper, we concentrate on triangle meshes. A very popular discretization of the Laplacian $\Delta$ for a triangle mesh with $N$ vertices may be realized as a sparse matrix $L \in \mathbb{R}^{N \times N}$ with the cotangent weights [MDSB02]. Along with the cotangent weights in $L$, which only respect angles between edges, we also use the lumped mass matrix $D$ containing the vertex areas along its diagonal. Notice that our discretization corresponds to the finite element formulation of the Laplace-Beltrami operator [LZ10], which leads to a generalized eigenvalue problem as explained in the following.

With the Laplace-Beltrami operator defined, we discretize the eigenfunctions. The manifold harmonic basis (MHB) is defined by [VL08]

$$-L\Phi_k = \lambda D\Phi_k ,$$

(3)

This can be solved using an off-the-shelf sparse iterative eigensolver or by using the efficient hand-by-hand computation method presented in [VL08]. Given a mesh with $N$ vertices, we assemble the first $K$ eigenvectors into a matrix $\Phi \in \mathbb{R}^{N \times K}$, where each column $\Phi_k$ is one eigenvector.

For computing the compressed manifold basis, we extend the variational formulation (2) from [OLCO13] to be applicable to triangle meshes. Here, we especially have to consider the area-matrix $D$. Without $D$, the eigenbasis is not independent of the mesh resolution, as demonstrated in Fig. 2. Our discretization of (2) thus reads

$$\min_{\Phi} \text{Tr}(\Phi^\top L\Phi) + \mu \|\Phi\|_1 , \quad \text{s.t. } \Phi^\top D\Phi = I .$$

(4)

The next section explains an efficient algorithm that solves this minimization problem.

3.2. Reformulation using ADMM

In this section, we reformulate the optimization problem (4) so that it can be solved efficiently using the alternating direction method of multipliers (ADMM). ADMM is a mathematical framework for convex optimization, interested readers are referred to the comprehensive article by Boyd et al. [BPC11]. ADMM can solve problems that involve two functions and constraints,

$$f(x) + g(z) , \quad \text{s.t. } Ax + Bz = c .$$

(5)

Our optimization problem (4) can be reformulated in the above form (5) . To this end, we replace the orthogonality constraint using an indicator function

$$\mathbf{1}(\Phi) = \begin{cases} 0 & \text{if } \Phi^\top D\Phi = I \\ \infty & \text{otherwise} \end{cases} .$$

(6)

This lets us transform the optimization objective (4) into a sum of three functions $\text{Tr}(\Phi^\top L\Phi) + \mu \|\Phi\|_1 + \mathbf{1}(\Phi)$. Since extensions of ADMM for more than two functions are still being heavily researched and not necessarily guaranteed to converge [CHYY13], we adopt a splitting strategy introduced recently [WHML13]. The idea is to use one of the three functions as the “main function” $f$ in (5) , and group the remaining functions into $g$, cf. (5) . For computing CMMs, we found that choosing $\Phi$ as the main function works best. We then arrive at the following reformulation of (4)

$$\min_{\Phi : S, E} \mathbf{1}(\Phi) + \text{Tr}(E^\top L E) + \mu \|S\|_1 , \quad \text{s.t. } \Phi = S , \Phi = E .$$

(7)
Notice how the two separate coupling constraints force the variable \( \Phi \) to be equal to \( S \) and \( E \). If those constraints are fulfilled exactly then we arrive again at (4). The equivalence to the standard formulation of ADMM (5) can be seen by first substituting \( x = \Phi \) and \( f = t \). Variable \( z \) and function \( g \) are then block-separable with

\[
g \left( \begin{bmatrix} E \\ S \end{bmatrix} \right) = \frac{\text{Tr}(E^\top L E)}{\mu \|S\|_1}.
\]

(8)

Finally, the constraint matrices from (5) may be written as

\[
A = \begin{bmatrix} I \\ I \end{bmatrix}, \quad B = \begin{bmatrix} -I & 0 \\ 0 & -I \end{bmatrix}, \quad c = 0.
\]

(9)

### 3.3. Numerical Algorithm

Rephrasing of our original minimization problem (4) into (7) enables us to apply the ADMM method [BPC11, Section 3.1.1]. To this end, we introduce the dual variable \( U \in \mathbb{R}^{2N \times K} \) which consists of two blocks \( U = [U_E; U_S] \), corresponding to two auxiliary variables \( E \) and \( S \). Given an initial guess for \( \Phi \), we set \( E = S = \Phi \) and \( U = 0 \). The algorithm then comes down to iterating the following steps:

\[
\Phi \leftarrow \text{arg min}_\Phi t(\Phi) + \frac{\rho}{2} \| \Phi - \frac{1}{\mu} \cdot E \|_F^2 \quad \text{(10)}
\]

\[
E \leftarrow \text{arg min}_E \text{Tr}(E^\top L E) + \frac{\rho}{2} \| \Phi - E + U_E \|_2^2 \quad \text{(11)}
\]

\[
S \leftarrow \text{arg min}_S \mu \| S \|_1 + \frac{\rho}{2} \| \Phi - S + U_S \|_2^2 \quad \text{(12)}
\]

\[
U \leftarrow U + \left[ \begin{bmatrix} \Phi \\ E \end{bmatrix} - \begin{bmatrix} S \\ E \end{bmatrix} \right].
\]

(13)

The variable \( \rho \) is called the penalty parameter. Usually, updates (11) and (12) are done in a single step involving \( g(z) \) in (5), but due to the independent blocks in (8) they can be separated.

We now explain how the individual update steps, which are small optimization problems themselves, can be performed efficiently. The objective of the minimization (10) can be transformed into

\[
t(\Phi) + \frac{\rho}{2} \| \Phi - \frac{1}{\mu} \cdot (S + E) \|_2^2 - 2 \text{Tr}(S + E) + \frac{1}{\mu} \| S + E \|_2^2.
\]

Ignoring terms that don’t change the minimum allows reduction of (10) to

\[
\Phi \leftarrow \text{arg min}_\Phi \| \Phi - Y \|_2^2 \quad \text{s.t.} \quad \Phi^\top D \Phi = I,
\]

(14)

with \( Y = \frac{1}{\mu} \cdot (S - U_S + E - U_E) \). Since \( D \) contains the vertex area weights, it is a positive diagonal matrix and trivial to invert. We can then substitute \( \Phi = D^{-1/2} \Psi \) to yield

\[
\Phi \leftarrow D^{-1/2} \left( \text{arg min}_\Psi \| D^{-1/2} \Psi - Y \|_2^2 \right). \quad \text{(15)}
\]

Based on the SVD factorization of \((D^{1/2} Y)^\top (D^{1/2} Y) = VWV^\top\), a closed-form solution is found as

\[
\Phi \leftarrow D^{-1/2} \left( Y V W^{-1/2} v^\top \right). \quad \text{(16)}
\]

Figure 3: Convergence of our method (green) vs the direct extension of [OLCO13] to 3D meshes (red). Some of the eigenfunctions computed by [OLCO13] show oscillations as visible in (a), which do not appear with our method (compare with our result in Fig. 1). (b) Plotting the primal residual across the iterations shows that [OLCO13] obtains infeasible solutions while our method converges.

A proof of the last step appears in various sources, e.g. in [LO14]. Thus, step (10) requires an SVD of a \( K \times K \) matrix - the number of vertices \( N \) is irrelevant (usually \( K \ll N \)).

To solve the minimization problem (11), we set its derivative to zero, which gives a closed form solution

\[
\left( \rho I - L - L^\top \right) \Phi = E - \rho (\Phi + U_E), \quad \text{(17)}
\]

so we have to solve \( K \) very sparse linear systems in each iteration. By prefactorizing \((\rho I - L - L^\top)\), e.g. using Cholesky factorization, this step can be significantly accelerated. Updating the factorization is only necessary when \( \rho \) changes.

Finally, the third step of updating \( S \) in (12) is separable for each entry \( S_{ij} \). It has a simple closed-form solution that may be expressed concisely using the proximal operator (denoted \( \text{prox} \)) of the \( \ell_1 \) norm [BJMO12],

\[
S_{ij} \leftarrow \text{prox}_{\mu} \| v \| = \text{sgn}(v) \max \left( \| v \| - \frac{\mu}{\rho}, 0 \right), \quad \text{(18)}
\]

where we substitute \( v = \Phi_{ij} + (U_S)_{ij} \) for brevity.

### 3.4. Convergence

ADMM is guaranteed to converge for a convex \( f \) and \( g \). However, the minimization task in (4) contains the nonconvex constraint of orthogonality of the eigenvectors, so convergence to a global minimum cannot be guaranteed. Nevertheless, the proposed method finds local minima which are suitable for practical applications.

To monitor convergence we use the primal residual \( \| \epsilon \|_2 \) and the dual residual \( \| d \|_2 \), cf. [BPC11]. When those values fall below a numerical threshold, our method stops. To automatically set the parameter \( \rho \), our method adopts the adjustment strategy discussed in [BPC11, Section 3.4.1]. This means that the only parameters for our method are \( \mu \) which
set the sparsity and controls the locality, and \( K \) which is the number of CMMs to compute. We also compare the convergence of our method with our reimplementation of the algorithm in [OLCO13] for 3D meshes. We set \( D = I \), because [OLCO13] cannot handle arbitrary \( D \), and run the algorithm on the mesh from Fig. 1. In this example only our algorithm is able to find a feasible solution, as can be seen in Fig. 3. We believe that this behavior is caused mainly by our splitting strategy that groups the two convex functions together and selects the non-convex \( t \) as “main function” \( f \).

4. Experiments and Results

Comparison to Varimax Varimax (or Orthomax) finds a unitary transformation of the eigenspace that aims to localize the basis, through maximizing the second order moments. An example application in computer graphics can be found in [SBCBG11]. Applying the Varimax method on the MHB indeed localizes the functions, but global oscillations remain on the mesh: The eigenfunctions are often not exactly zero and are thus not sparse (compare Fig. 5a to Fig. 5b). Also, for small \( K \) the locality of Varimax diminishes completely (Fig. 5c) while the CMMs are always local (Fig. 5d).

Influence of Varying Sparsity The locality of the CMMs can be controlled by varying the parameter \( \mu \), which is not possible with Varimax. This is demonstrated in Fig. 4: large \( \mu \) will result in smaller local support. To additionally make the parameter \( \mu \) invariant to the number of vertices \( N \) we usually multiply it by \( N \).

Initialization Our method can be initialized with a random \( \Phi \). We typically observe that the algorithm converges to the same set of basis functions, although their ordering might be different. For example, we always found CMMs for the five fingers and the palm if we compute six CMMs on a hand. Similarly, the set of eigenfunctions in Fig. 1 were always the same, but their ordering was different because some CMMs have eigenvalues that are close to each other. When \( K \) is very small, e.g. \( K = 1 \), we observe that different initializations lead to different local minima, see Fig. 6. Instead of giving random initialization, we can also give a user-drawn scribble as initialization, which might be useful for certain applications. Other initialization strategies are also possible, for example in Fig. 8 we used Varimax, because we wanted to have reproducible results even though \( K \) is quite low.

Shape Approximation Since both MHs and CMMs form a basis, it is possible to use them for encoding the actual mesh coordinates. This process was described for MHs in [VL08]. Specifically, we have the transformation to “fre-
frequency space \( \Phi \) by \( \hat{x} = \mathbf{D} \Phi^\top x \), and its inverse transformation by \( x = \Phi \hat{x} \). Here, \( x \) are the \( x \), \( y \), and \( z \) coordinates of the vertices. For \( K \ll N \), applying the transform followed by its inverse gives an approximation of the mesh. Comparing those approximations can help better understanding the properties of the CMB compared to the MHB. MHs show better compression guarantees and retain the rough shape of the whole mesh (Fig. 7b). When \( K \) is increased, the MH approximation will move all the vertices towards the input mesh. In contrast, the CMM approximation is more abstract and almost looks like a skeleton of the mesh (Fig. 7c). Each time \( K \) is increased, a local change is done to the approximated mesh by changing only the vertices of a local feature, for example by placing the finger tip. Another instructive example is given in Fig. 8. Here, two things can be clearly seen: First, the CMMs automatically form at high-curvature regions and topological protrusions of the mesh. To comply with the sparsity regularization, the first \( K \) CMMs ignore flat regions or regions connecting the geodesic extremities.

**Time and Space Complexity for Reconstruction** In Table 1 we quantitatively evaluate the CMB in comparison to the MHB for different number of components \( K \) and for different meshes. The reconstruction error of the vertex coordinates (\( \ell_2 \) norm of the difference between reconstruction and ground truth) approaches that of the MHB when the number of basis functions \( K \) goes up. It is also interesting to compare the required memory size for the MHB, stored as a dense array (64bit double), and to the CMB \( \Phi \) stored as a sparse matrix in CSR format. The CMB is smaller since \( \Phi \) contains many zeros. The runtimes in Table 1 were measured on a Linux system with Intel i7 3.4 GHz CPU and 16 GB RAM. We compare our unoptimized Python/NumPy implementation for computing the CMB versus the highly optimized ARPACK eigensolver to compute the MHB.

<table>
<thead>
<tr>
<th>mesh</th>
<th>basis</th>
<th>( K )</th>
<th>( \mu )</th>
<th>error</th>
<th>size</th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td>hand (868)</td>
<td>MHB</td>
<td>10</td>
<td>5</td>
<td>9.34</td>
<td>68 KB</td>
<td>0.03 s</td>
</tr>
<tr>
<td></td>
<td>CMB</td>
<td>10</td>
<td>-</td>
<td>14.39</td>
<td>43 KB</td>
<td>3 s</td>
</tr>
<tr>
<td></td>
<td>MHB</td>
<td>30</td>
<td>5</td>
<td>4.00</td>
<td>203 KB</td>
<td>0.08 s</td>
</tr>
<tr>
<td></td>
<td>CMB</td>
<td>30</td>
<td>-</td>
<td>3.97</td>
<td>156 KB</td>
<td>29 s</td>
</tr>
<tr>
<td>fertility (4994)</td>
<td>MHB</td>
<td>10</td>
<td>5</td>
<td>45.02</td>
<td>390 KB</td>
<td>0.18 s</td>
</tr>
<tr>
<td></td>
<td>CMB</td>
<td>10</td>
<td>-</td>
<td>17.54</td>
<td>216 KB</td>
<td>44 s</td>
</tr>
<tr>
<td></td>
<td>MHB</td>
<td>40</td>
<td>5</td>
<td>7.92</td>
<td>1.5 MB</td>
<td>0.39 s</td>
</tr>
<tr>
<td></td>
<td>CMB</td>
<td>40</td>
<td>-</td>
<td>6.93</td>
<td>933 KB</td>
<td>560 s</td>
</tr>
<tr>
<td>bunny (34834)</td>
<td>MHB</td>
<td>10</td>
<td>1.25</td>
<td>36.89</td>
<td>2.7 MB</td>
<td>1.4 s</td>
</tr>
<tr>
<td></td>
<td>CMB</td>
<td>10</td>
<td>-</td>
<td>121.28</td>
<td>1.7 MB</td>
<td>419 s</td>
</tr>
<tr>
<td></td>
<td>MHB</td>
<td>40</td>
<td>1.25</td>
<td>20.49</td>
<td>10.6 MB</td>
<td>3.5 s</td>
</tr>
<tr>
<td></td>
<td>CMB</td>
<td>40</td>
<td>-</td>
<td>20.68</td>
<td>8.2 MB</td>
<td>1357 s</td>
</tr>
</tbody>
</table>

(a) Input Mesh (b) MH reconstr. (c) CMM reconstr.

(a) \( K = 3 \) (b) \( K = 7 \) (c) \( K = 3 \) (d) \( K = 7 \)

Figure 7: Reconstruction of vertex positions from a small number of \( K = 6 \) (b) manifold harmonics and (c) our proposed compressed manifold modes. See text for details.

Figure 8: Soft segmentation using CMMs for two meshes and different number of CMMs \( K \). Protrusions and geodesic extremities are segmented, grey areas are not covered by any of the first CMMs.

Figure 9: Reconstruction error of a constant function on the mesh for varying \( K \) on three different meshes. (a) shows that with increasing \( K \) the function can be better reproduced and (b) measures \( \ell_\infty \) norm of that error, a value below 1 means that all vertices are covered by at least one CMM.

© 2014 The Author(s)
Computer Graphics Forum © 2014 The Eurographics Association and John Wiley & Sons Ltd
Insensitivity to noise Since the MH basis is based on the intrinsic Laplace operator, it is robust to various kinds of noise. We now empirically show that the CMM basis functions share this property with the MH basis. In Fig. 10, we added gaussian noise with \( \sigma = 50\% \) of the average edge length - the CMMs align up to sign flip. In another experiment, we randomly removed vertices as well as their neighboring triangles between the original and the noisy mesh. Up to sign flip, the CMM basis functions align well between the original and the noisy mesh.

5. Applications in Shape Matching

Typically, shape matching first involves robust feature detection and matching, and then a geometry aware regularization that extends this correspondence to the entire shape. For practical shape correspondence, both these stages have to be robust to geometric and topological surface noise that occurs inevitably in real world scanning systems. Occlusions and partial scans make this problem even more challenging.

Ovsjanikov et al. [OBCS12] use the manifold harmonic basis to propose the elegant framework of functional maps. Often, the manifold harmonic basis between two shapes do not exactly correspond to each other. The functional map still gives a full correspondence of the two shapes even in this case, but only if the sparse input correspondences (point or region features) that are required to construct the map are correct. On the other hand, if the shapes are related by an isometry, the manifold harmonics align well. Then, the functional map yields a diagonal correspondence matrix (or at least diagonal after permutation). Pokrass et al. [PBB+13] use this idea for regularizing the functional map to be sparse and close to a diagonal. However, this is limited to situations where the manifold harmonics can be aligned between the shapes. One important case where this assumption fails is with large holes and severe topological noise due to partial scans. We show an example in Fig. 12 where the mesh of a hand is badly corrupted by holes. “Dense” harmonics of the MHB are severely damaged by these holes and no longer align with the original mesh. However, up to a sign flip and ordering, CMMs align robustly with those of the original mesh. At the same time, the CMMs align very well in the case of purely isometric deformations, which is demonstrated in Fig. 13. Notice that in both examples, the same set of CMMs is recovered even though completely different random initializations were used for the different meshes.

Because of their spatial locality and robustness, compressed manifold modes can be used as shape features in the first stage of shape matching. But since the CMB provides an orthogonal basis that is invariant to isometric deformations, they can also be used in the second stage of shape-aware regularization. In this paper, we tested this second aspect by using the CMB to replace the MHB in the functional map framework. In the discrete setting the functional correspon-
dence between two shapes, $S_1$, resp. $S_2$, can be given as a simple $K \times K$ matrix $C$. If both shapes are equipped with the basis then we have matrices $\Phi_{S_1}$ and correspondingly $\Phi_{S_2}$). Those can be seen as $K$-dimensional point clouds (in eigenspace), and the matrix $C$ aligns those point clouds by $\Phi_{S_2} = \Phi_{S_1} C^T$. Thus, $\Phi_{S_1} C^T$ is a rotated basis such that the basis vectors (columns) overlap with those in $\Phi_{S_2}$.

A necessary condition for $C$ to be an area-preserving vertex-to-vertex map is that it must be orthogonal [OBCS’12, Theorem 5.1]. To illustrate CMMs in a shape matching scenario, we take two shapes with ground-truth correspondence differing by isometric deformations. Those given correspondences are then used to find the optimal orthogonal $C$ for different number of basis functions $K$. We then measure how much error (in geodesic distance) the resulting map between the basis functions introduces, depending on the number of basis functions $K$. This is exactly the same experiment as performed in [OBCS’12, Fig. 3], which we apply to the new CMB. Here, we use the unweighted Laplacian for both. We find that, despite the CMB not being as information-dense as the MHB, it achieves similar geodesic error with a fairly low number of basis functions $K$. Around 35 compressed modes are good enough to meet the accuracy of dense harmonics, Fig. 14. More interesting is the phenomenon that the correspondence matrix $C$ is much sparser for the CMB, compare Fig. 14b to Fig. 14c. Notice that at no point during the estimation of $C$ we enforced this sparsity. A direct implication of this is that the matrix $C$ for the CMB is closer to a permutation. For visualization, we choose the permutation that yields a diagonalized correspondence matrix, similar to Pokrass et al [PBB’13]. This is a trivial step and we show the final diagonalized correspondence matrix. For matching partial shapes, this effect is even more apparent, see Fig. 14, middle and bottom row. The respective correspondence matrix $C$ with CMB is again much closer to a permutation matrix, whereas that of the MHB shows much confusion and is much denser. We think this is a strong indicator that CMMs can be extremely useful for shape matching that is robust against large holes or even for partial shape matching.

6. Discussion and Future Work

It is possible to use CMMs as skinning weights for mesh editing applications. But it is desirable to produce skinning weights that are positive and that sum up to one for all the vertices on the shape [JBK’12]. In contrast to bounded biharmonic weights [JBPS11], CMMs cannot meet those requirements and do not fulfill exact interpolation constraints. But while bounded biharmonic weights also achieve some level of sparsity, they require the user to provide a set of point constraints or a skeleton rig. This is not needed for CMMs. The CMB automatically provides meaningful “areas” that can be used to restrain certain edits. CMMs can regularize sparse input positional constraints for applications such as posing a shape and key-framing an animation, or tracking a deformable object in video. Additionally, the CMB can be used for projecting certain deformation ener-
gies, thereby reducing their dimensionality and thus computationally simplifying and regularizing the deformation process. These ideas as well as the connection with bounded biharmonic weights [JBPS11] for skinning deformation would be worth exploring in the future.

The Laplace-Beltrami eigenfunctions are related to the heat kernel operator for heat diffusion on the mesh surface. This heat kernel has several applications, particularly heat diffusion on mesh surfaces in the future. Local support was proven for compressed dependence on the initialization. [SOG09]. Investigating the effect of sparsity and $\ell_1$ minimization in the heat diffusion framework was partly done in [Rus11]. CMMs can help in further investigations in the future. Local support was proven for compressed modes in the plane [BCO14], we showed empirically that this also seems to be the case on manifolds. Proving theoretically that the obtained functions are connected on the manifold is a related question open for future research.

A final point we would like to discuss is the non-convexity of our objective function in (2) which leads to a certain dependence on the initialization. [LLO14] recently proposed a convex relaxation of (2). However, this approach involves optimizing over an $N \times N$ matrix, which means that it is infeasible for meshes with large number of vertices $N$. So this is still an avenue for future work.

7. Conclusion

In this paper, we introduced compressed manifold modes (CMMs) on mesh surfaces, a novel orthogonal basis for general manifold surfaces in 3D. The compressed basis functions have local support around key shape features that are automatically detected. Our paper presented a complete mathematical derivation of a numerical algorithm that extracts this basis from a given 3D mesh. We empirically demonstrated desirable properties of the CMMs, such as full support and robustness to noise and partial scans. We qualitatively evaluated the CMMs for potential applications such as shape matching and shape approximation. Our results indicate high potential for a wider range of applications in the future.

References


Figure 14: Given two meshes and ground truth vertex-to-vertex correspondence, column (a), we compute the functional representation of this map between the two shapes. Column (b) shows the functional map between the MHB. In column (c), the functional map between the CMBs clearly shows much more sparsity. In (d) we plot the mapping error of the maps between MHB and CMB. With increasing number of basis functions $K$ the error of the CMB approaches that of the MHB. Notice how the effect of sparsity in the correspondence matrix is retained even in cases of partial correspondences between meshes with huge holes, middle and bottom row.


