Compressed Manifold Modes: Fast Calculation and Natural Ordering

Kevin Houston
School of Mathematics
University of Leeds
Leeds, LS2 9JT, U.K.
e-mail: k.houston@leeds.ac.uk
www.maths.leeds.ac.uk/∼khouston/

July 13, 2015

Abstract

Compressed manifold modes are locally supported analogues of eigenfunctions of the Laplace-Beltrami operator of a manifold. In this paper we describe an algorithm for the calculation of modes for discrete manifolds that, in experiments, requires on average 47% fewer iterations and 44% less time than the previous algorithm. We show how to naturally order the modes in an analogous way to eigenfunctions, that is we define a compressed eigenvalue. Furthermore, in contrast to the previous algorithm we permit unlumped mass matrices for the operator and we show, unlike the case of eigenfunctions, that modes can, in general, be oriented.

1 Introduction

The eigenvalues and eigenfunctions of the Laplace-Beltrami operator (LBO) on a discrete manifold have found many applications in geometry processing, for example, in shape matching, remeshing (such as quadrangulation), smoothing, and shape identification, see for example, [4, 8].

In analogy with a Fourier basis, the eigenfunctions of the LBO form a basis, called the **manifold harmonic basis** (see [14]), of the functions on the manifold. One major drawback of this basis is that it does not relate, at least in an intuitive way, to observable features of manifolds. In the pioneering work [11, 12] it is shown that one can produce on $\mathbb{R}^n$ a set of functions with localized support, i.e., compactly supported, which will function as a basis. They named these **compressed modes**. They proposed that these may be used as a natural basis for solving PDEs and suggested that they could be extended to general discrete manifolds. This generalization was made in [10] and were called **compressed manifold modes** (CMMs). These CMMs are locally supported and, crucially, they seem to be supported at important features of the manifold. For example in the top line of Figure 1 one can see that the collection of six CMMs is supported on the arms, legs, head and lower torso. That is, features that a human would identify are detected by the modes. The six modes pictured on the less symmetric Aquarius the Water Carrier, are also supported on regions which a human may identify as regions of interest. Twenty modes are given later in Figure 7 for the classic teapot.

Similarly the support for 15 modes on a L-shaped mesh is local and identifies significant features such as corners as shown in Figure 2.

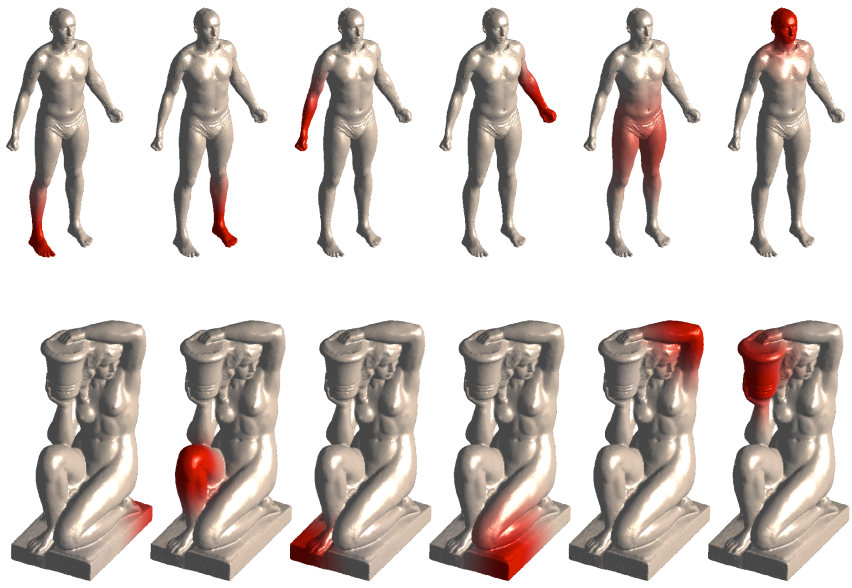


Figure 1: Local support and identification of natural features.

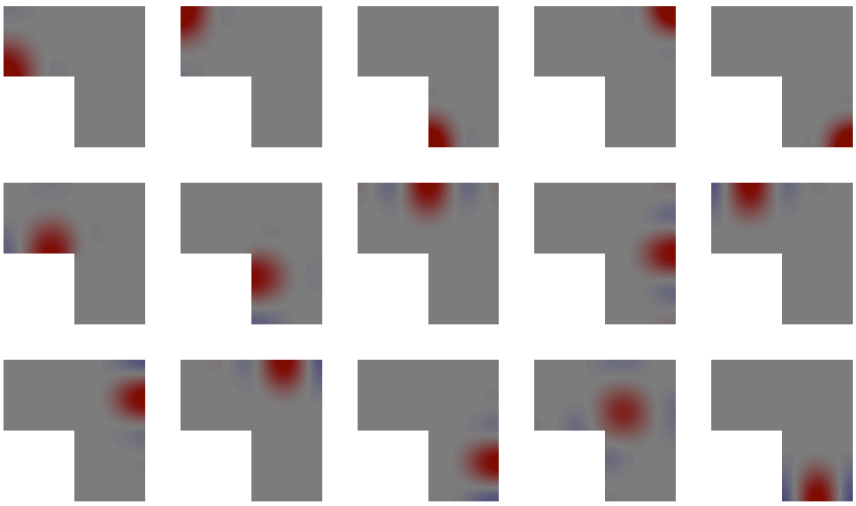


Figure 2: Support on a L-shaped mesh, red is positive and blue is negative.

In [10] the authors show that, in contrast to the eigenfunctions and eigenvalues of the LBO, the CMMs are robust with respect to noise and holes. Examples are given where shape matching is achieved in the presence of noise and partial sampling of meshes. Furthermore, the theory in [11, 12] is developed as a way of solving partial differential equations. Hence, CMMs are potentially important objects of study in a diverse range of subjects.

Two significant problems of compressed manifold modes that are identified in [10] are speed and ordering. They take longer to generate than eigenfunctions and no method for ordering was given. Further problems are that the algorithm in [10] is restricted to an LBO with diagonal mass matrix and it is noted that there is ambiguity of sign of the modes in the same way that unit eigenfunctions of a matrix are only defined up to sign.

This paper deals with all these problems. The main contributions are as follows:

(i). We adapt an algorithm from [5] to speed up the algorithm given in [10] for computation of the modes. This accelerated algorithm is described in Section 3 and experiments show an average 44% reduction in the time taken with a 47% decrease in the number of iterations required. The accuracy of the accelerated version is measured in Section 5.

(ii). We describe a natural ordering for the modes in Section 4 that is entirely analogous to the ordering of eigenvalues of a matrix.

(iii). The new algorithm allows non-diagonal mass matrices. The explanation of this is given in Section 7.

(iv). Section 6 shows how, in contrast to the case of eigenvectors, the modes can be oriented, i.e., can be given a sign. This can be seen by comparing the colours in Figure 5 with that of Figure 13 in [10]. The modes in Figure 5 have been 'flipped' where necessary.

The Matlab code of the implementation of the algorithms is freely available under a Creative Commons Attribution-NonCommercial-ShareAlike (CC BY-NC-SA) license. It can be found at `http://www1.maths.leeds.ac.uk/~khouston/`.

Acknowledgements: My thanks to Thomas Neumann for helpful discussions and comments. The meshes of humans come from the SCAPE dataset, [1] and the Aquarius mesh is from the EPFL Computer Graphics and Geometry Laboratory. The Algorithms Latex bundle by Rogério Brito was used in preparation of the manuscript.

2 Background, previous and related work

In recent years there has been much interest in the eigenvalues and eigenfunctions of the discrete Laplace-Beltrami operator. For a smooth manifold M the classical differential geometry Laplace-Beltrami operator, denoted Δ, has a set of eigenfunctions $\{\varphi_k\}$ and associated eigenvalues, $\{\lambda_k\}$, determined by

$$\Delta\varphi_k = \lambda_k\varphi_k$$

where $k \in \mathbb{N}$ and $\lambda_k \in \mathbb{R}$. See [2]. The self-adjointness of Δ implies that the eigenvalues are real and that the eigenfunctions are orthogonal with respect to the L_2-inner product: $\langle f, g\rangle = \int_M fg$.

In the discrete case we denote the LBO by L and decompose it as $L = A^{-1}W$ where A is a **mass matrix**, i.e., a matrix which relates to the area/volume around the vertices of the discrete manifold, and W is a weight matrix, such as the cotan matrix, (see [4, 8] for further references).

Both A and W are symmetric $N \times N$ matrices where N is the number of vertices. The matrix $L = A^{-1}W$ is in general not symmetric but is self-adjoint with respect to the L_2-inner product given by $\langle v, w \rangle = v^T A w$, where v^T denotes the transpose of the vector v. This implies that its eigenvalues are real and its eigenvectors are orthogonal with respect to the inner product. The eigenvalues and eigenvectors of L are solutions of the symmetric eigenvalue problem

$$W\varphi = \lambda A\varphi.$$

In this paper we write K eigenfunctions as the columns of the $N \times K$ matrix Φ.

In [11] **compressed modes** are introduced as a solution of a minimization problem involving the Hamiltonian operator $H = -(1/2)\Delta + V(x)$, where V is a potential function, with what is called an L_1-regularizing term, that is, $(1/\mu)\sum_j |\psi_j|_1$ for modes ψ_j where $\mu \in \mathbb{R}$ is a parameter controlling how localized the modes are.

We produce a set $\Psi_K = \{\psi_j\}_{j=1}^K$ arising from the variational problem

$$\min_{\Psi_K} \sum_{j=1}^{K} \left(\frac{1}{\mu} |\psi_j|_1 + \langle \psi_j, H\psi_j \rangle \right) \text{ such that } \Psi^T\Psi = \mathrm{Id}$$

where Ψ is the $N \times K$ matrix given by arranging the ψ_j in columns.

The generalization, called **compressed manifold modes**, described in [10] is as follows. For $K \in \mathbb{N}$ and $\mu \in \mathbb{R}$, the first K compressed manifold modes are the columns of the matrix Φ, where Φ is determined by the constrained optimization problem

$$\min_{\Phi} \mathrm{Tr}\left(\Phi^T W \Phi\right) + \mu ||\Phi||_1 \text{ such that } \Phi^T A \Phi = \mathrm{Id}\,.$$

Here $||\Phi||_1$ is the sum of the absolute values of entries of Φ. (Note that, rather confusingly, the compression parameter is $1/\mu$ in [11] and μ in [10]. We will follow the latter as our algorithm is a generalization of theirs.) A mode is an analogue of the Laplace-Beltrami eigenfunctions.

The parameter μ is a measurement of the compression of the support of the modes. A large μ means large compression of support, i.e, the support gets smaller, and a small μ has small compression, i.e., large support.

3 Accelerated ADMM Algorithm

This section contains the first contribution of the paper – an accelerated version of the algorithm presented in [10] to calculate the compressed manifold modes. In the tests this new algorithm required 47% fewer iterations to reach convergence and resulted in a 44% time improvement.

In [11], Ozoliņš et al propose a Splitting Orthogonality Constraint (SOC) algorithm (described in detail in [7]) for calculating compressed modes in the case that the manifold is $\mathbb{R}^n$. In [10], Neumann et al show that the SOC algorithm does not function well for more general manifolds and to counteract this they propose an Alternating Direction Method of Multipliers (ADMM) algorithm. The ADMM method is known to be particularly slow, see [3]. A method of acceleration based on the Nesterov method, [9], is described in [5] and we propose a modification of this to increase the speed of the algorithm from [10]. The algorithm is a variant of the gradient descent method with an over-relaxation step.

Let f and g be functions and that we wish to minimize $f(u) + g(v)$ subject to $Au + Bv = c$. Let ρ be a penalty parameter as in [3]. The accelerated algorithm is given in Algorithm 1. To compute the compressed manifold modes we proceed as

Algorithm 1 Accelerated Calculation of CMMs.

Require: $\alpha_1 = 1$, $\eta = 0.999$.

for $k = 1, 2, 3, \ldots$ **do**

$u_k = \arg\min f(u) + \frac{\rho}{2}||Au + B\widehat{v}_k + c - \widehat{\lambda}_k||^2$

$v_k = \arg\min g(v) + \frac{\rho}{2}||Au_k + Bv + c - \widehat{\lambda}_k||^2$

$\lambda_k = \widehat{\lambda}_k + Au_k + Bv_k + c$

$c_k = \rho\left(||\lambda_k - \widehat{\lambda}_k||^2 + ||B(v_k - \widehat{v}_k)||^2\right)$

if $c_k < \eta c_{k-1}$ **then**

$\alpha_k = 1$

end if

$\alpha_{k+1} = \dfrac{1 + \sqrt{1 + 4\alpha_k^2}}{2}$

$\widehat{v}_{k+1} = v_k + \dfrac{\alpha_k - 1}{\alpha_{k+1}}(v_k - v_{k-1})$

$\widehat{\lambda}_{k+1} = \lambda_k + \dfrac{\alpha_k - 1}{\alpha_{k+1}}(\lambda_k - \lambda_{k-1})$

end for

described in Section 3.2 of [10]. The optimization function is split into the sum of three (rather than two) functions:

$$\mathrm{Tr}(\Phi^T L \Phi) + \mu||\Phi||_1 + \iota(\Phi)$$

where the indicator function ι is defined by

$$\iota(\Phi) = \begin{cases} 0, & \text{if } \Phi^T A \Phi = \mathrm{Id}, \\ \infty, & \text{otherwise.} \end{cases}$$

They then reformulate the problem as

$$\min_{\Phi,S,E} \iota(\Phi) + \mathrm{Tr}(E^T L E) + \mu||S||_1 \text{ such that } \Phi = S,\ \Phi = E.$$

We proceed similarly to [10] and use the following in Algorithm 1 : $f = \iota$, $u = \Phi$, $v = \begin{bmatrix} E \\ S \end{bmatrix}$ and

$$g\left(\begin{bmatrix} E \\ S \end{bmatrix}\right) = \begin{bmatrix} \mathrm{Tr}(E^T L E) \\ \mu||S||_1 \end{bmatrix}.$$

We set

$$A = \begin{bmatrix} I \\ I \end{bmatrix}, \qquad B = \begin{bmatrix} -I & 0 \\ 0 & -I \end{bmatrix} \quad \text{and} \quad c = 0.$$

Note that there is no guarantee that this algorithm converges. That the algorithm in [5] converges for a weakly convex optimization problem is shown in [5]. The essential part of the proof is the monotonic decrease in a certain residual. The optimization problem we wish to solve is not weakly convex and the residual does not monotonically decrease (as can be seen in examples). Hence we make a modification to the restart rule to ensure that the algorithm does not become stuck on the same values. The acceleration process requires a parameter, denoted η, to initiate the restart. In all the experiments in this paper η was set to 0.999.

In the implementation used for the experiments in this paper (and in [10]) the penalty parameter is varied as in Section 3.4 of [3]. This greatly increases the speed of both algorithms.

Furthermore, convergence was determined by the use of the **primal and dual residuals**. The primal residual at iteration k is $r_k = Au_k + Bv_k - c$ and the dual residual is $s_k = \rho A^T B(v_k - v_{k-1})$. The precise condition for stopping the algorithm are described in Section 3.3.1 of [3]. The algorithm terminates when $||r_k||_2 \leq \epsilon^{\text{pri}}$ and $||s_k||_2 \leq \epsilon^{\text{dual}}$ for the two tolerances, ϵ^{pri} and ϵ^{dual}. These are determined from

$$\epsilon^{\text{pri}} = \sqrt{2n}\epsilon^{\text{abs}} + \epsilon^{\text{rel}} \max\left\{||\Phi||_2, \sqrt{||E||_2^2 + ||S||_2^2}\right\},$$
$$\epsilon^{\text{dual}} = \sqrt{n}\epsilon^{\text{abs}} + \epsilon^{\text{rel}}||\lambda||_2$$

where n is the number of vertices for the mesh, ϵ^{abs} is an absolute tolerance, ϵ^{rel} is a relative tolerance and λ is the dual variable in the algorithm. In all experiments, following [10], we took $\epsilon^{\text{abs}} = 10^{-8}$ and $\epsilon^{\text{rel}} = 10^{-6}$.

In the experiments the two algorithms were given the same random initialization of numbers between 0 and 1. This was repeated 30 times for each mesh. Table 1 compares the average number of iterations and average time taken for the accelerated algorithm described above and the one from [10]. The meshes Stand, Crouch and Bent Limbs are the those from first, second and third lines of Figure 5 respectively; L-shape is the L-shape from Figure 2; Aquarius is the water carrier statue mesh from Figure 1. The experiments were performed on an iMac with 3.4 GHz Intel core i7 and 8GB RAM.

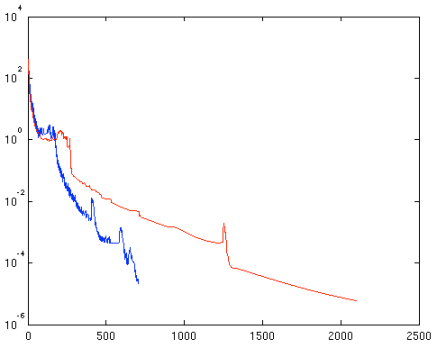
In the experiments the accelerated algorithm required on average 47% fewer iterations than the original algorithm with a range between 9% and 88%. The improvement in speed was nearly as good with an average 44% improvement. The range of improvements was 2% to 87%. Hence, we can conclude that the improvements are significant and the algorithm is worth implementing when compared to the original.

Typical graphs comparing the sums of the prime and dual residuals for the two algorithms are shown in Figure 3. These were selected to give some insight into how the accelerated version behaves. We can see that its residual sum is a condensed and bumpier version of the unaccelerated version's. The combined residual for the accelerated algorithm looks like it has greater fluctuations and this is indeed the case. This is pictured in close up Figure 4 where one can see that these are Nesterov 'ripples' – a common feature of Nesterov methods. The algorithm restarts at the top of the bounce.

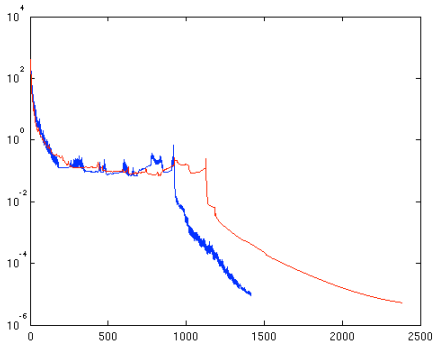
4 A natural order for compressed manifold modes

For the Laplace-Beltrami operator we can naturally order the eigenfunctions by their associated eigenvalues. We shall now describe the correct analogous natural ordering for compressed manifold modes. One could naively order the modes according to their Dirichlet energy. That is, ignore the additional L_1 term in the minimisation. This leads to poor results as one can see by comparing Figure 5 (our ordering) and Figure 6 (Dirichlet ordering). The former is better than the latter for these near-isometric surfaces.

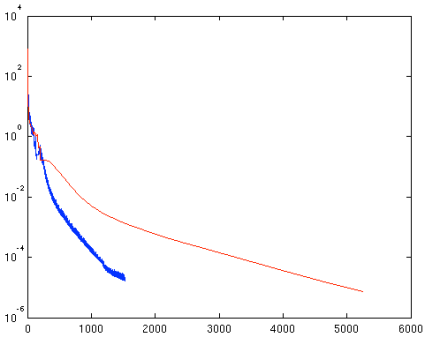
In Figure 5 the modes for a mesh fall naturally into pairs of consecutive modes with the exception of the third row where one could argue that positions 4 and 5 should be exchanged. One could also argue that the head- and torso-supported modes are not in the same sequence in the three meshes. However, they do only occur in positions 7 and 8. Contrast these small differences with the larger ones in Figure 6, the Dirichlet ordering. Here the head-supported mode appears in positions 3, 4, and 8. The torso supported mode appears in positions 5 and 6. Furthermore, no mesh consistently has what we could call natural pairs and in particular the bottom mesh has most of the modes not occurring in consecutive pairs.



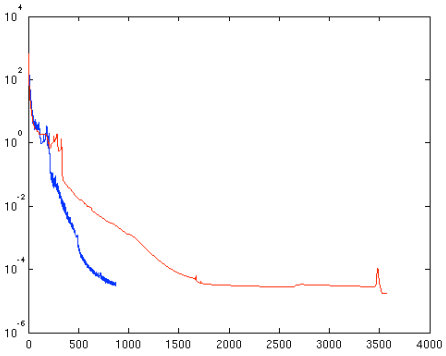
Stand, $\mu = 0.0075$, $K = 6$



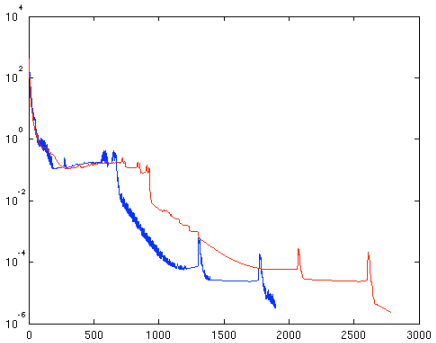
Stand, $\mu = 0.001$, $K = 6$



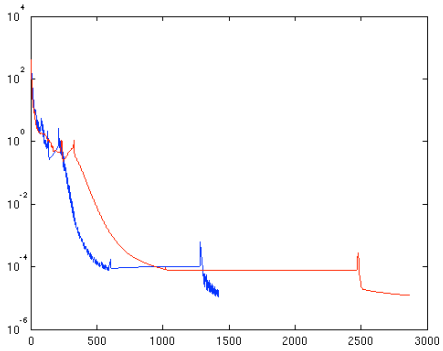
Aquarius, $\mu = 0.0075$, $K = 6$



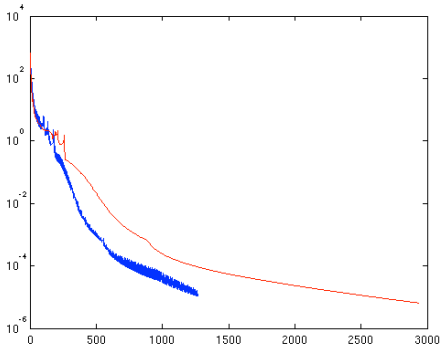
Stand, $\mu = 0.0075$, $K = 10$



Crouch, $\mu = 0.001$, $K = 6$



Crouch, $\mu = 0.0075$, $K = 6$



Crouch, $\mu = 0.0075$, $K = 10$

Figure 3: Comparison of typical sum of prime and dual residual for accelerated ADMM (blue) and ADMM (red).

Mesh	μ	K	Fast ADMM iter's	ADMM iter's	Percent reduction	Fast ADMM time (in sec)	ADMM time (in sec)	Percent reduction
Stand	0.008	10	1381	2889	48	43	86	45
Crouch	0.008	10	1140	3177	57	38	89	49
Bent limbs	0.008	10	1218	2373	49	38	71	47
Stand	0.001	6	1329	1980	33	37	53	30
Crouch	0.001	6	1916	2704	30	48	66	27
Bent limbs	0.001	6	1813	3640	46	50	95	43
Stand	0.0075	6	829	1498	44	23	40	42
Crouch	0.0075	6	902	1660	44	23	40	41
Bent limbs	0.0075	6	922	1645	43	25	44	41
L Shape	0.02	10	7867	18597	57	229	469	51
Aquarius	0.0075	6	1568	4349	60	243	661	59
Aquarius	0.0075	10	2220	5182	55	412	875	50
Average					47			44

Table 1: Comparison of the accelerated and original ADMM algorithms. For each mesh the average of 30 iterations is taken.

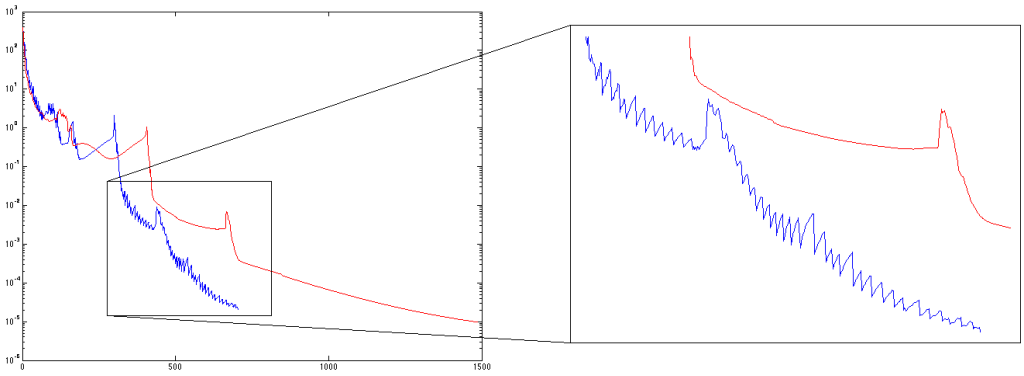


Figure 4: A close up (with scale) of the Nesterov ripples. Mesh was Stand with $\mu = 0.008$ and $K = 6$.

Having shown the superiority of the new ordering, let us now describe it in detail. Let $L = A^{-1}W$ denote the Laplace-Beltrami operator where A is the mass matrix and W the weight matrix. Suppose we wish to find K modes. Then the modes can be represented as the K columns of Φ where Φ is determined by the constrained optimization problem

$$\min_{\Phi} \operatorname{Tr}\left(\Phi^T W \Phi\right) + \mu||\Phi||_1 \text{ such that } \Phi^T A \Phi = \mathrm{Id}\,.$$

Since A is symmetric the constraint $\Phi^T A\Phi = \mathrm{Id}$ determines $K(K-1)/2$ distinct equations. If we apply the Lagrange multiplier method we require $K(K-1)/2$ Lagrange multipliers λ_{ij}, with $1 \le i \le k$ and $i \le j \le k$.

Lemma 4.1 *The Lagrangian function, $\mathcal{L}$, for this constrained optimization problem is*

$$\mathcal{L}(\Phi) = \operatorname{Tr}\left(\Phi^T W \Phi\right) + \mu||\Phi||_1 - \operatorname{Tr}\left(\left(\Phi^T A\Phi - \mathrm{Id}\right)\Lambda\right)$$

where Λ is the real symmetric $K \times K$ matrix with entries $[\Lambda]_{ij} = \lambda_{ij}$ for $1 \le i \le k$ and $i \le j \le k$.

Proof. We can define the Hadamard product of matrices X and Y, denoted $X \circ Y$, to be their elementwise product. That is,

$$[X \circ Y]_{ij} = [X]_{ij} \cdot [Y]_{ij},$$

where $[X]_{ij}$ is the (i,j) entry of the matrix X. Now, the Lagrangian function is

$$\mathcal{L}(\Phi) = \operatorname{Tr}\left(\Phi^T W \Phi\right) + \mu||\Phi||_1 - \sum_{i,j}\left[\left(\Phi^T A\Phi - \mathrm{Id}\right) \circ \Lambda\right]_{ij}.$$

From Lemma 5.1.5 of [6] we have that $\sum_{i,j}[X \circ Y]_{ij} = \operatorname{Tr}(XY^T)$. From this the result follows. □

The method of Lagrangian multipliers requires the solution of $\dfrac{\partial \mathcal{L}}{\partial \Phi} = 0$. Using the facts that for matrices X, Y, Z and function f,

$$\frac{\partial}{\partial X}\operatorname{Tr}(XYX^TZ) = ZXY + Z^TXY^T, \quad \text{([13] equation 118)},$$
$$\frac{\partial}{\partial X^T} f(X) = \left(\frac{\partial}{\partial X} f(X)\right)^T,$$

we see that the equation $\dfrac{\partial \mathcal{L}}{\partial \Phi} = 0$ gives

$$\begin{aligned} 2W\Phi + \mu\,\mathrm{sign}(\Phi) - 2A\Phi\Lambda &= 0 \\ \Phi^T W\Phi + \frac{\mu}{2}\Phi^T\,\mathrm{sign}(\Phi) &= \Phi^T A\Phi\Lambda \\ \Phi^T W\Phi + \frac{\mu}{2}\Phi^T\,\mathrm{sign}(\Phi) &= \Lambda. \end{aligned}$$

By considering the diagonal of both sides, and noting that for a vector φ we have $\varphi^T\,\mathrm{sign}(\varphi) = ||\varphi||_1$, we can associate a number to each column of Φ, i.e., to a compressed manifold mode.

Definition 4.2 *Let φ be a compressed manifold mode of the operator $A^{-1}W$ with compression factor μ. The* ***compressed eigenvalue****, λ, associated to φ is the number*

$$\lambda = \varphi^T W\varphi + \frac{\mu}{2}||\varphi||_1.$$

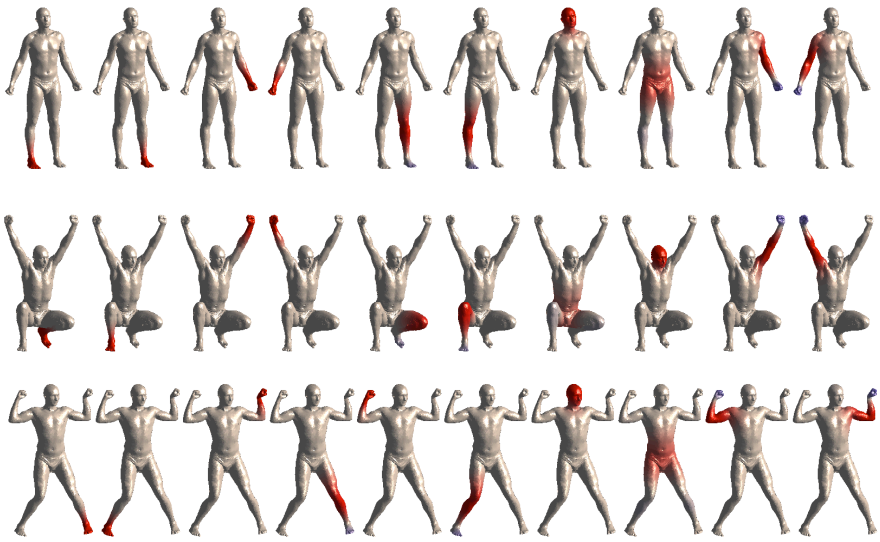


Figure 5: Ordering for three meshes, cf. Figure 13 of [10].

	Stand	Crouch	Bent limbs	Teapot $10^{-4}\times$
1	19.6155	20.3833	20.5455	34.1979
2	19.8722	20.5321	20.6391	57.9528
3	21.8682	21.4825	24.6300	60.1676
4	23.7901	23.0605	26.7960	60.1705
5	27.3304	25.6135	27.9656	60.9936
6	28.1828	27.9401	29.0192	62.9510
7	28.7432	30.4192	29.0294	63.0666
8	34.6949	33.1840	33.6681	63.2446
9	39.4302	36.5946	38.3945	63.3030
10	39.5416	36.8892	38.4758	63.3360

Table 2: Compressed eigenvalues for some meshes.

Note that when $\mu = 0$ and K is the the number of rows of W we get the standard eigenvalues of the operator $A^{-1}W$.

Figure 13 of [10] demonstrated that their algorithm consistently found similar modes for non-isometrically deformed meshes, in particular for a standing, crouching and limb bending man from the SCAPE dataset. This experiment was rerun with the accelerated algorithm (with $\mu = 0.008$) and the modes were ordered and flipped (see Section 6 for an explanation of the latter). This is pictured in Figure 5 and the resulting compressed eigenvalues are given in Table 2.

The compressed eigenvalues can be closely grouped. For example, Aquarius, the Water Carrier, pictured in Figure 1 has compressed eigenvalues 2.2223, 2.4737, 2.5792, 2.6721, 2.8335, and 3.4063 for $K = 6$ and $\mu = 0.001$. (The accelerated algorithm took 3308 iterations from a random initialization.)

For the human meshes we can see in Table 2 that they range from 19.6155 to 39.5416. To put this into perspective, the first ten eigenvalues of the LBO range from 0 to 31.7886. One can see clearly in Figure 5 that eight of the modes occur in obvious pairs: pair of feet, pair of legs, pair of hands, and pair of arms. This

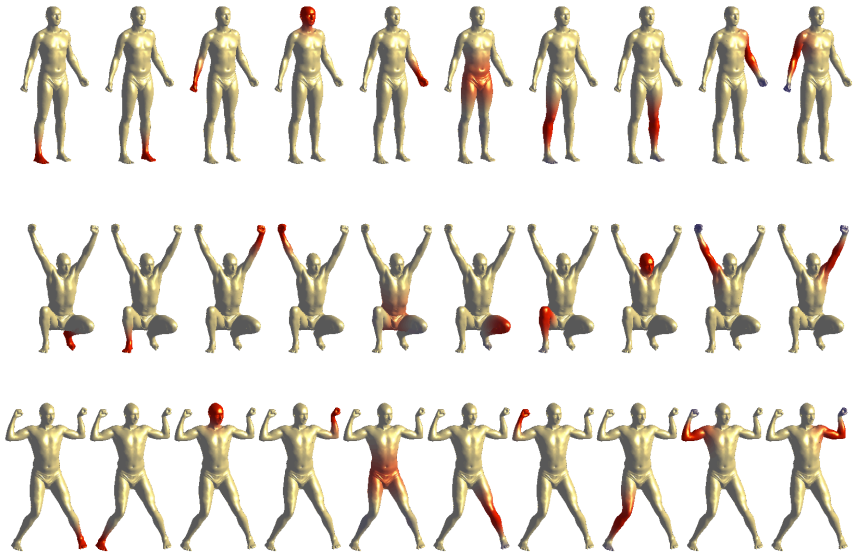


Figure 6: Ordering for three meshes by only the Dirichlet energy, note that the ordering is not as good as Figure 5.

pairing is reflected in the compressed eigenvalues, for example for the Stand mesh the compressed modes supported on the feet correspond to compressed eigenvalues 19.6155 and 19.8722. This means that it is highly likely that, just as in the eigenvalue case of symmetric objects, numerical errors can cause compressed eigenvalues to be out of order. This is possibly what is happening with the human poses where the first compressed eigenvalue is sometimes the left foot, sometimes the right.

A set of twenty modes is shown in Figure 7 for the classic teapot. Here $\mu = 0.0008$ and the initialisation was not random but the first twenty eigenfunctions of the discrete Laplace-Beltrami operator. Note again that the supports of the modes coincide with what a human might identify: the teapot spout, handle, lids, and knob of lid. The modes divide the spout in three. It would interesting to investigate how varying the parameter μ affects symmetry in the support of the modes.

Note again, that similar modes arising from symmetries have closely grouped compressed eigenvalues. This can be seen in Table 2 where the 3rd to 5th compressed eigenvalues correspond to the three regions on the teapot lid.

5 Accuracy and Consistency

A measure of the accuracy of a numerical approximation of an eigenvector v of a linear map L can be calculated by $Lv - \lambda v$. The theory behind the calculation of ordering gives us an analogous measure of the accuracy of the algorithm in calculating modes. A mode φ with compressed eigenvalue should satisfy the condition $W\varphi + (\mu/2)\,\mathrm{sign}(\varphi) = \lambda A\varphi$ and this can be used to check accuracy.

In both the original and accelerated versions of the algorithm the average error of entries of the vector $W\varphi + (\mu/2)\,\mathrm{sign}(\varphi) - \lambda A\varphi$ was of the order 10^{-3} to 10^{-4}. Hence, although we cannot have much confidence in the accuracy of the methods we can say that accuracy is not lost by using the faster algorithm. The accuracy aspect of CMMs requires further investigation. In particular, what is the trade-off between speed and accuracy and what is the relationship between the algorithm

Figure 7: Modes on the teapot.

stopping tolerances ϵ^{abs} and ϵ^{rel} and accuracy.

One method to improve the accuracy was to perform 200-400 iterations of the accelerated algorithm, set all entries in Φ below a preset tolerance to zero and then restart the algorithm with this new Φ. This did not conclusively improve accuracy but had the advantage of occasionally increasing the speed significantly. The results were inconsistent and further study is required. One could also try a similar approach by setting to zero the entries of Φ that correspond to large absolute entries of $W\varphi + (\mu/2)\,\text{sign}(\varphi) - \lambda A\varphi$ for the different φ.

We now turn to consistency. Given a random initialization it is clearly plausible that resulting modes are inconsistent, that is distinct runs of the algorithm with fixed μ and K, etc, may produce a different collection of modes. However, experiments show that for certain values of μ the calculation of modes by the accelerated method is very consistent. For the SCAPE dataset models (Stand, Crouch and Bent Limbs in this paper) repeated experiments on random initializations involving numbers from 0 to 1 produced less than 1% inconsistent results for μ around 0.008. For other models, for example Aquarius, it was harder to locate a good value of μ to use.

The value of μ is important for applications as μ determines the 'size' of the support of modes, that is the amount the support covers the model. If μ is large, then the supported area is small. If the supported area is small, then it is likely that supported areas do not interact during the iterations and this can in theory lead to inconsistency. Hence, the value of μ is important for consistency but is poorly understood.

One way of avoiding inconsistency is to use the eigenfunctions of the Laplace-Beltrami operator as the initialization of Φ. This need not give the same K modes as the random initialization and it would be interesting to investigate further the use of this initialization as it worked quite well in some experiments. For example, the teapot in Figure 7 was calculated using the first twenty eigenfunctions and this was very consistent for small changes of μ in the sense we had the same modes in the same order. Nonetheless, what are the quantitative changes to the modes when μ is varied is still an open question.

When one changes the number K of modes to be calculated, then one gets inconsistent results. For example, compare Figure 8 with Figure 5 where K is

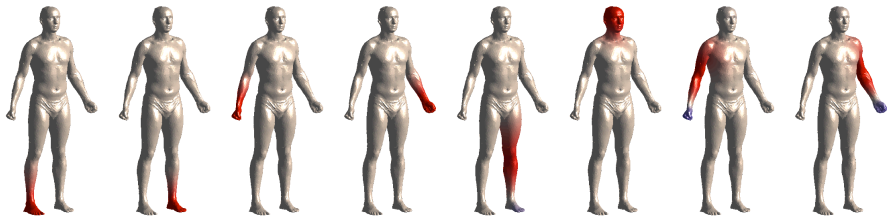


Figure 8: Effect of the choice of K.

8 and 10 respectively. The modes calculated for $K = 8$ are not the first 8 modes calculated for $K = 10$. However, Figure 5 gives one confidence that good consistency is achievable even between near isometric shapes.

6 Orientation and flipping of modes

A disadvantage of the Laplacian unit eigenfunctions is that they are defined only up to sign and there is no obvious way to 'flip' them so that eigenfunctions between different manifolds can be directly compared. For these eigenfunctions one expects an 'equal amount' of positive and negative values. That is, the integral of the positive values equals the integral of the negative values. (This is because the eigenfunctions are orthogonal to the constant vector, the eigenfunction for the zero eigenvalue.) However, for general compressed modes there is not this balancing of the positive and negative. Hence we can flip modes so that the ambiguity of parity is removed..

In practice, the values of a mode are dominated by numbers of either positive or negative values. For a mode φ the number $\operatorname{sign}(\max(\varphi) + \min(\varphi))$ will, in practice, be ± 1 depending on the sign of the dominant values. One can then use this to change the sign of the columns of Φ. In this way, the modes chosen to be dominated by positive values. A problem only occurs when $\max(\varphi) = -\min(\varphi)$ but this is unlikely in practice.

An alternative and perhaps more theoretically sound way to flip the mode is to integrate it over the manifold and define its sign to be the sign of the resulting value.

That the positive values dominate after flipping can be seen in the figures where red is positive.

7 Lumped and unlumped mass matrices

The mass matrix A in $L = A^{-1}W$ contains information about the area around the vertices. Many variations are possible, for example one-third area, Voronoi area or uniform area, see [8]. Given such a matrix one can 'lump' the matrix by summing all the elements of a row and putting the result in the diagonal. A mass matrix with non-zero entries only on the diagonal is called a **lumped matrix**.

The algorithm in [10] is given for lumped matrices. In that paper D is a diagonal matrix and it is easy to calculate $D^{1/2}$ and its inverse $D^{-1/2}$, where $D^{1/2}$ is formed from the square roots of the diagonal entries. For a positive definite matrix B such as the mass matrix there exists a square root matrix $B^{1/2}$ such that $B^{1/2}B^{1/2} = B$. However, the calculation of this can be computationally expensive and hence though

it is theoretically possible with this method to include unlumped mass matrices in the algorithm of [10] this may seriously impact the speed of computation.

Nonetheless, unlumped matrices can be used. In [10] the first part of the algorithm involves letting $Y = D^{1/2}(S - U_S + E + U_E)$, finding an SVD factorization of $Y^TY = VWV^T$ and then the closed form of the minimization problem is

$$\Phi = D^{-1/2}YVW^{1/2}V^T.$$

If instead we take $\widetilde{Y} = S - U_S + E + U_E$, and so $Y = D^{1/2}\widetilde{Y}$ then $Y^TY = VWV^T$ simply becomes, $\widetilde{Y}^TD\widetilde{Y} = VWV^T$ and the closed form of the minimization problem is then

$$\Phi = D^{-1/2}\left(D^{1/2}\widetilde{Y}\right)VW^{1/2}V^T = \widetilde{Y}VW^{1/2}V^T.$$

Thus with this modification to the algorithm we can avoid the expensive computation of the square root of the mass matrix and allow use of unlumped matrices.

Numerous experiments with different meshes, values of K and μ, failed to show that either the lumped or unlumped version was superior in terms of speed or accuracy. It seems unlikely that lumping does not at least have some effect and so while the experiments were inconclusive it may be the case that there do exist identifiable situations in which one method is superior to the other.

To ensure consistency and comparability the experiments described in this paper were done with lumped matrices. It should be noted that lumped matrices are used for the LBO much more commonly than unlumped. The aim here is to show that we can use unlumped if we wish.

8 Conclusions and future work

Compressed manifold modes were demonstrated in [10] to be robust with respect to noise and holes and have the potential for use in geometry processing applications. Indeed as they generalize to surfaces the compressed modes introduced in [11] they also have potential for application in solving PDEs on surfaces. In this paper it has been shown that the algorithm of [10] for the computation of compressed manifold modes can be considerably improved - by almost a factor of 2. The correct method for ordering modes has been demonstrated (with proof) and the modes have been oriented.

The study of the numerical calculation of eigenvalues has had the advantage of decades of work and the development of many optimization techniques and so unsurprisingly the calculation of compressed manifold modes is not competitive in terms of speed. This paper helps reduce this uncompetitiveness but the development of further improvements would be welcome.

Compressed manifold modes are of interest because they are functions with local support. A crucial insight is that if one is looking for a collection of functions to form a basis of functions on the manifold, then speed and accuracy are not that important, it is the orthonormality of the collection that is important. In the case of the CMM algorithms the locality of support and orthonormality can, and in the experiments usually do, appear rapidly, say within 400-600 iterations of the algorithm. Therefore it would be good to analyse how quickly these properties are achieved.

Further work of interest includes the following:

(i). Investigate and define for numerical calculations what it means for the modes to be locally supported and orthonormal. For the latter we need to check that $\Phi^TA\Phi$ is close to the identity. How close for practical applications?

(ii). The random initialization can lead to a wide variation in computation time. For example, in the 30 calculations used in Table 1, the Bent Limbs mesh with $K = 10$, $\mu = 0.008$ took between 736 and 4323 iterations, the latter is a factor of nearly 6 greater than the former. Can we do better or at least be more consistent?

(iii). Can inconsistency of the collection of modes calculated be reduced by restricting the band of random numbers used as an initialization? See Section 5.

(iv). Can accuracy of the algorithms be improved by some method, for example truncating the support as in Section 5.

(v). How do changes in the compression factor μ affect the consistency and accuracy of the algorithms?

References

[1] Dragomir Anguelov, Praveen Srinivasan, Daphne Koller, Sebastian Thrun, Jim Rodgers, and James Davis. Scape: Shape completion and animation of people. *ACM Trans. Graph.*, 24(3):408–416, July 2005.

[2] M. Berger. *A Panoramic View of Riemannian Geometry*. Springer Berlin Heidelberg, 2003.

[3] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Found. Trends Mach. Learn.*, 3(1):1–122, January 2011.

[4] Keenan Crane, Fernando de Goes, Mathieu Desbrun, and Peter Schröder. Digital geometry processing with discrete exterior calculus. In *ACM SIGGRAPH 2013 courses*, SIGGRAPH '13, New York, NY, USA, 2013. ACM.

[5] Tom Goldstein, Brendan O'Donoghue, and Simon Setzer. Fast alternating direction optimization methods. *CAM report*, pages 12–35, 2012.

[6] Roger A. Horn and Charles R. Johnson. *Topics in Matrix Analysis*. Cambridge University Press, 1991. Cambridge Books Online.

[7] Rongjie Lai and Stanley Osher. A splitting method for orthogonality constrained problems. *Journal of Scientific Computing*, 58(2):431–449, 2014.

[8] Bruno Lévy and Hao (Richard) Zhang. Spectral mesh processing. In *ACM SIGGRAPH 2010 Courses*, SIGGRAPH '10, pages 8:1–8:312, New York, NY, USA, 2010. ACM.

[9] Yurii Nesterov. A method of solving a convex programming problem with convergence rate $O(1/k^2)$. *Soviet Mathematics Doklady*, 27(2):372–376, 1983.

[10] Thomas Neumann, Kiran Varanasi, Christian Theobalt, Marcus Magnor, and Markus Wacker. Compressed manifold modes for mesh processing. *Computer Graphics Forum (Proc. of Symposium on Geometry Processing SGP)*, 33(5):1–10, July 2014.

[11] Vidvuds Ozoliņš, Rongjie Lai, Russel Caflisch, and Stanley Osher. Compressed modes for variational problems in mathematics and physics. *Proceedings of the National Academy of Sciences*, 110(46):18368–18373, 2013.

[12] Vidvuds Ozoliņš, Rongjie Lai, Russel Caflisch, and Stanley Osher. Compressed plane waves yield a compactly supported multiresolution basis for the laplace operator. *Proceedings of the National Academy of Sciences*, 111(5):1691–1696, 2014.

[13] K. B. Petersen and M. S. Pedersen. The matrix cookbook, Nov 2012. Version 20121115.

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