SpectroMeter: Amortized Sublinear
Spectral Approximation of Distance on Graphs

Roee Litman
Tel-Aviv University
tau.ac.il/~roeelitm

Alex M. Bronstein
Technion, Israel Institute of Technology
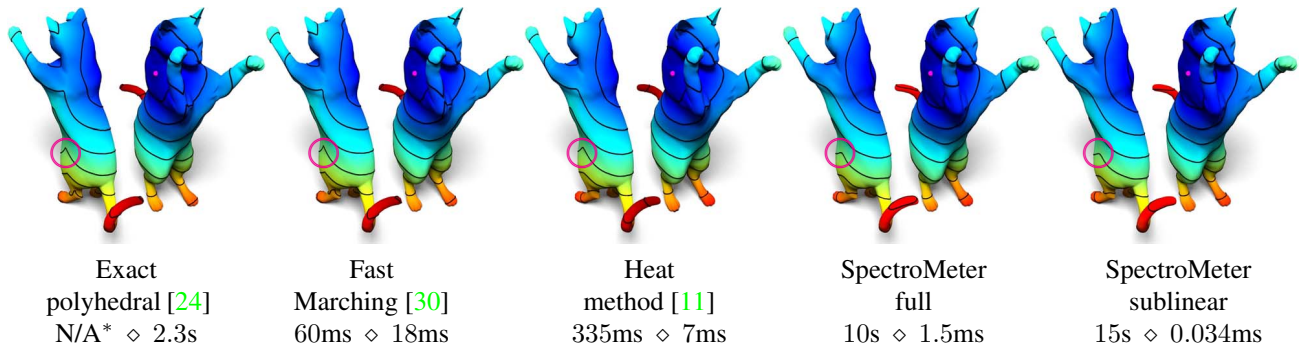cs.technion.ac.il/~bron

Exact	Fast	Heat	SpectroMeter	SpectroMeter
polyhedral [24]	Marching [30]	method [11]	full	sublinear
N/A* ◇ 2.3s	60ms ◇ 18ms	335ms ◇ 7ms	10s ◇ 1.5ms	15s ◇ 0.034ms

Figure 1: Comparison of five methods for distance map calculations on a triangular mesh from the TOSCA [8] dataset. Each result is presented from two viewpoints, underwhich the timiming result for pre-calculations (left) and run-time (right) are presented. Distances are measured from the magenta point on the cat shoulder, and are colormapped from blue (low) to red (high), and equidistant fronts are marked by black contours. The leftmost 'exact' map dictates the colormap values on the other four, as well as the locations of the black contours (set at steps of 10% the maximal distance). Both SpectroMeter maps (two rightmost) are very similar to the exact ones (two leftmost), but obtained with a fraction of the computational cost. Non-differentiable locations are harder to approximate, as in the smoothed cusp is highlighted by a magenta circle.

Abstract

We present a method to approximate pairwise distance on a graph, having an amortized sub-linear complexity in its size. The proposed method follows the so called heat method due to Crane et al. [11]. The only additional input are the values of the eigenfunctions of the graph Laplacian at a subset of the vertices. Using these values we estimate a random walk from the source points, and normalize the result into a unit gradient function. The eigenfunctions are then used to synthesize distance values abiding by these constraints at desired locations. We show that this method works in practice on different types of inputs ranging from triangular meshes to general graphs. We also demonstrate that the resulting approximate distance is accurate enough to be used as the input to a recent method for intrinsic shape correspondence computation.

1. Introduction

Distances on graphs are used on a daily basis, in many hardware and software applications, sometimes without us being fully aware of it. The best known example is map navigation, where one searches the best route to traverse a city, minimizing either distance or commute time. Another common application is packet routing through the internet, where two of the most commonly used algorithms (OSPF [26] and IS-IS [27]) allow multiple machines to collaboratively find the best route for a packet.

Two other common domains of application of distance on graphs are computer graphics and geometry processing, where images and shapes are often described as graphs (perhaps, with some additional structure as in the case of meshes). For example, being able to calculate distances between points on a shape allows to represent the intrinsic information in a way invariant to bending or other extrinsic deformations.

While existing methods for calculating distances on graphs are very efficient, they are still at least linear in the graph size. This may become prohibitively expensive when the graph is very big or when the number of distance calculations grows.

---

*Pre-calculation time could not be separated from the runtime in the implementation we used.

Our contribution. We introduce a spectral method to approximate distances on graphs, given only a few of the eigenfunctions of the graph Laplacian, possibly sampled at a subset of the vertex set. We see two main contributions of this paper: First, we show how to perform pairwise distance approximation in sub-linear time in the size of the graph (excluding some preprocessing). Second, unlike the previous method [11], we do not rely on a discrete *divergence* operator, which allows extension to types of graphs where such an operator is not trivial to define.

2. Distance maps on graphs

There exists a wealth of literature dedicated to distance computation on graphs. A useful dichotomy is to categorize such algorithms into *heap based methods* that gradually propagate distances over the graph, usually accessing its vertices in an unpredictable order, and *stack based methods* operating on the entire vertex set or on pre-fixed subsets thereof.

2.1. Heap based methods

Heap-based approaches for the computation of distances on graphs have been dominant until, probably, the last decade. One of the first such algorithms was suggested by Dijksrta [14], solving the shortest path problem for any directed graph with n vertices in $O(n \log n)$. Mitchell et al. [24] extended this algorithm to polyhedral meshes, providing an exact solution for the discrete geodesic distance computation in $O(n^2 \log n)$. It was later shown by Surazhsky et al. [33] that an approximation with rather low worst-case error bounds can be achieved in $O(n \log n)$.

Since graphs often constitute a discretization of a continuous domain, the question of consistency with the continuous geodesic distances naturally arises. While discrete geodesics are often inconsistent, "sampling theorems" have been derived guaranteeing the convergence of the discrete solution to the continuous one under certain conditions [5]. When a manifold is discretized as a graph satisfying these conditions, the discrete geodesic distances computed on it are guaranteed to converge to the continuous ones. As an alternative to solving a discrete geodesic problem, another line of approaches suggests to discretize directly the partial differential equation governing the distance map on the underlying continuous domain,

$$\|\nabla \mathbf{d}\| = 1 \quad s.t. \quad \mathbf{d}|_{\texttt{source}} = 0,$$

known as the eikonal equation. Sethian [30] and Tsitsiklis [36] have independently developed an efficient eikonal solver having $O(n \log n)$ runtime, which is essentially a continuous variant of Dijkstra's algorithm. This method, known as the *fast marching method* (FMM), was later extended from regular grids to other types of graphs like triangular meshes [19] and point clouds [22], just to mention

a few. A faster version was introduced by Yatziv et al.[42], solving at $O(n)$ at the cost of quantizing the distances.

Even though the mentioned methods differ in the types of data handled and the specific calculations performed, they are all dynamic programming algorithms and share the property of gradually propagating the distance calculation from the source(s) to the rest of the vertices. One potential advantage of this is that one can stop the calculation before covering the entire graph once the destination is achieved, saving time for 'close' distances. However, as a result of their inherent sequential structure and data-dependent access to memory, these methods are hard to parallelize and implement efficiently on modern computer architectures. Furthermore, they usually cannot reuse the calculations from previous runs.

2.2. Stack based methods

The other category of algorithms, in contrast to the former, operate on the entire vertex set or fixed subsets thereof.

One line of methods that provide a fast approximation of FMM are called *fast sweeping*, and can be dated back to Danielsson [13] who proposed to calculate a complete Euclidean distance map using only four 'sweeps' of a 2D plane in $O(n)$. The method was later expanded to other types of manifolds, as formally suggested by Zhao [43], after it was implicitly used in other works [44, 34, 35]. Even though these methods have $O(n)$ complexity, their main drawback is that for some manifolds it might require several repetitions to achieve sufficient accuracy, up to $O(2^n)$ for an exact solution, as shown by Hysing et al. [17]. A highly parallel variant called *parallel marching* was suggested by Weber et al. [40], who proposed to decompose the surface into regular grids, solving each in a manner similar to [13]. Even though parallel marching requires a low-distortion parameterization into quadrilateral patches, it is still one of the fastest implementations of surface distance computation to date.

The newest line of works in this category follows the heat method by Crane et al. [11], which will be covered in greater detail in the sequel.

Finally, we would like to mention a recent work by Aflalo et al. [3], where spectral-based distance approximations are also performed as the input of a subspace parameterized multidimensional scaling (MDS) problem. While this approach performs some approximation in a manner seemingly similar to ours, we will show in the sequel that it does not scale to general pairwise distance approximations.

3. Background

Prior to detailing how the heat method works, we briefly review some preliminaries, mainly regarding graph Laplacian.

Graph Laplacian. Let $\mathcal{G} = (V, E)$ be an undirected graph with $|V| = n$ vertices. Let the graphs edges be further equipped with nonnegative weights $\{w_{ij}\}$, defining the $n \times n$ *weighted adjacency matrix* of the graph as

$$[\mathbf{W}]_{ij} = \begin{cases} w_{ij}, & \text{if } (i,j) \in E \\ 0, & \text{otherwise.} \end{cases}$$

Additionally, the *vertex degree matrix* is an $n \times n$ diagonal matrix, whose elements are usually defined as

$$[\mathbf{A}]_{ii} = a_{ii} = \sum_j w_{ij}. \tag{1}$$

The *unnormalized* Laplacian of a graph is defined as the $n \times n$ matrix $\mathbf{L}_u = \mathbf{A} - \mathbf{W}$. As with all Laplacians, it is easy to observe that the sum of all rows and columns is zero. For an undirected graph, the Laplacian is a symmetric positive semidefinite matrix. For other properties of the Laplacian, the reader is referred to [25].

One of the ways to normalize the Laplacian adopted here is the so-called *random walk* Laplacian [21, 10], defined as

$$\mathbf{L}_{rw} = \mathbf{A}^{-1}\mathbf{L}_u = \mathbf{I} - \mathbf{A}^{-1}\mathbf{W}. \tag{2}$$

The name stems from its relation to the row-stochastic transition probability matrix $\mathbf{P} = \mathbf{I} - \mathbf{L}_{rw}$.

Mesh Laplacian. A triangular mesh is a specific type of graph, realizing a homogeneous simplicial complex of order 2. Since meshes typically represent continuous geometric objects (manifolds), specific constructions of mesh Laplacians exist that consistently discretize some properties of the Laplace-Beltrami operator (LBO) of the underlying continuous manifold. Mesh Laplacians have been extensively used in geometry processing; for a review, the reader is referred to [31].

We adopt the popular *cotangent weight scheme* first formalized in its current form by Meyer et al. [23]. Under this scheme, the $n \times n$ *stiffness matrix* $\mathbf{L}_c$ is constructed as an unnormalized Laplacian whose edge weights are

$$w_{ij} = \frac{1}{2}\cot(\alpha) + \cot(\beta),$$

with α and β being the two angles facing the edge (i, j), and the diagonal of degree matrix is the same as in (1). This scheme also defines the *mass matrix* $\mathbf{M}$ whose elements are $[\mathbf{M}]_{ii} = m_i$, where m_i is $\frac{1}{3}$ the area of all triangles incident on vertex i. Finally, the mesh Laplacian is defined as $\mathbf{L}_m = \mathbf{M}^{-1}\mathbf{L}_c$.
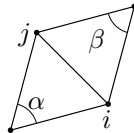
Graph geodesics. A *path* on the graph is a sequence of p vertices $P = \{v_1, \ldots, v_p\} \in V$, where each consecutive pair is one of the graph edges, i.e. $(v_i, v_{i+1}) \in E$. The length of a path P is the sum of all its edge weights,

$$L(P) = \sum_{i=1}^{p-1} w_{v_i, v_{i+1}}.$$

A shortest path (also known as minimum geodesic on a continuous manifold) between two vertices v and v' is a path starting at $v_1 = v$ and ending at $v_p = v'$ whose length is minimum over all such paths. Note that a minimizer might not be unique as there might be several such paths.

As mentioned previously, sometimes graphs are used as a discretization of a continuous manifold. In these cases the above definition of a geodesic has to be relaxed in order to allow convergence of the discrete solution to the continues one [5]. Paths on a mesh, for example, will not be restricted to the edges and are allowed to traverse the faces.

Discretized gradient. We follow the definition of the intrinsic gradient for triangular meshes from [11], for the ease of comparison. Other realizations exist, e.g., by Pokrass et al. [28].

Let f be a piece-wise linear function on the mesh defined by the values f_i on the vertices. Let δ be a triangle formed w.l.o.g. by the vertices 1, 2 and 3 with the corresponding function values f_1, f_2 and f_3. The gradient of f on δ is constant and is defined by

$$\nabla\delta = \frac{1}{2 \cdot \text{area}(\delta)} \sum_{i=1}^{3} f_i \left(n_s \times e_i\right) \tag{3}$$

where n_s is the normal to δ, and e_i is the edge opposing vertex i oriented counterclockwise.

In a general (non-mesh) graph, we can no longer rely on faces, and define the gradient on edges. Specifically, an edge $e = (v_i, v_j)$ whose vertices have scalar values $\{f_i, f_j\}$ will have a gradient $\nabla e_{ij} = (f_i - f_j)/w_{ij}$.

3.1. The heat method

A recent seminal work by Crane et al. [11] proposed a new approach to distance calculation dubbed as the *heat method*. This method, which is extended in this paper, is summarized in Algorithm 1. While we focus on the specific realization of the heat method on triangular meshes, it is more general and can be applied to other types of data, as mentioned in [11].

The key idea behind the heat method stems from Varadhan's equation [37]: it can be shown that the solution u_t of the heat equation

$$(\mathbf{M} + t\mathbf{L}_c)\,\mathbf{u}_t = \mathbf{u}_0 \tag{4}$$

input : Laplacian $\mathbf{L}$, Gradient ∇, Divergence Δ,
time value t_0 and origin point(s)

output: Distance map $\mathbf{d}$

---

1. Solve the heat equation (4) from the source point(s) for time value t, obtaining $\mathbf{u}_t$.

2. Calculate the gradient of the result $\mathbf{u}_t$, and normalize to unit length according to (5), obtaining $\widehat{\mathbf{g}}$.

3. Solve Poisson equation (6) for the divergence of the normalized gradient $\widehat{\mathbf{g}}$, obtaining $\mathbf{d}$.

Algorithm 1: The Heat method [11]

describing the heat distribution at time t is a function whose gradient direction coincides with that of a distance map from the same source (yet, unlike the distance map, the gradient magnitude $\|\nabla\mathbf{u}_t\|$ is not necessarily unit). The initial heat placement $\mathbf{u}_0$ at $t = 0$ is a discretized Dirac distribution, having 1 in the source location(s) and 0 elsewhere.

Following this observation, we can normalize the gradient of $\mathbf{u}_t$ to unit length

$$\widehat{\mathbf{g}} = \frac{\nabla\mathbf{u}_t}{\|\nabla\mathbf{u}_t\|}, \tag{5}$$

and solve Poisson's equation for the distance map $\mathbf{d}$ from the same source such that its Laplacian coincides with the divergence of the normalized gradient $\widehat{\mathbf{g}}$

$$\mathbf{L}_c\mathbf{d} = \nabla \cdot \widehat{\mathbf{g}}. \tag{6}$$

We did not specify here the exact implementation of the divergence operator for triangular mesh, and refer the reader to Section 3.2 in [11] for details. Note however, that the discretization or even a consistent definition of this operator on more general graphs is a limitation of the heat method. In what follows, we describe an alternative technique for distance map calculation similar in spirit to the heat method, yet relying only on the gradient but not on the divergence.

4. Proposed method - SpectroMeter

The proposed method which we dub *SpectroMeter* can be thought of as a variant of the heat method [11] in the spectral (frequency) domain. The details of the method are given in the following section and are summarized in Algorithm 2. The proposed method relies on several spectral properties of the Laplacian that we briefly highlight next.

Laplacian Spectrum. Spectral graph theory is a vast field with numerous and ramified applications [25, 12]. For example, a recent tutorial by von Luxburg [39] covers applications of different types of Laplacians to clustering

input : Laplacian eigenfunctions $\mathbf{\Phi}$, their gradient $\nabla\mathbf{\Phi}$, $O(k)$ locations, origin point(s)

output: Approximate distance map

---

1. Calculate the heat kernel $\mathbf{h}_t$ from the source points at the given $O(k)$ locations on the mesh, according to (7).

2. Calculate $\nabla\mathbf{h}_t$ the gradient of $\mathbf{h}_t$ at these locations and normalize to unit length $\widehat{\mathbf{g}} = \nabla\mathbf{h}/\|\nabla\mathbf{h}\|$.

3. Find coefficients $\mathbf{a}$ that best approximate $\widehat{\mathbf{g}}$, i.e. $\widehat{\mathbf{g}} \approx \sum_{i=1}^{k} a_i \nabla\phi_i$, according to (8).

4. Set the first coefficient(s) a_i corresponding to constant ϕ_i's so that the result is non-negative (or has zero value at the source).

5. Synthesize approximate distance at desired location(s) x according to $d(x) = \phi(x)\mathbf{a}$.

Algorithm 2: SpectroMeter - the proposed sublinear distance map approximation algorithm. The full (non-sublinear) flavor of this algorithm simply uses *all* the graph points rather than the given $O(k)$ samples.

and graph-cuts. Being essentially an averaging operator, graph Laplacian excels at promoting smooth functions on the graph [31], and specifically, it has been shown that the Laplacian eigenbasis is optimal in the ℓ_2 sense for the approximation of smooth functions [2].

We denote the eigendecomposition of a Laplacian $\mathbf{L}$ as $\mathbf{L}\phi_i = \lambda_i\phi_i$, with λ_i and ϕ_i being the eigenvalues and the corresponding eigenfunctions. We will assume ordering by increasing value of λ_i. It is easy to show that the smallest eigenvalue λ_0 has the value 0 with a constant corresponding eigenvector and has multiplicity equal to the number of connected components in the graph.

Preprocessing. As with the other distance methods [30, 11], some of the calculations have to be done for any input, and can be pre-computed and stored to reduce the amortized cost of distance computation.

The proposed technique, being a spectral method, relies on first performing eigendecomposition of the graph Laplacian. Specifically, we require the first smallest $k \ll n$ eigenvalues and the corresponding eigenfunctions. These low-frequency harmonics can be reused for many other tasks relying on the Laplacian spectrum. We henceforth denote the truncated eigenbasis collectively by the matrix $\mathbf{\Phi} = (\phi_1, \ldots, \phi_k)$. Next, we also calculate the (intrisic) gradients of all the eigenvectors (the constant one(s) have

vanishing gradients), denoting the results collectively as $\nabla \Phi = (\nabla \phi_1, \ldots, \nabla \phi_k)$.

Finally, we sample the graph at $O(k)$ locations, preferably using farthest point sampling [15] (FPS) based on an 'exact' method like FMM. This sampling is used for the sublinear version of the algorithm, as detailed in the sequel.

Heat in the spectrum. The major difference between [11] and the proposed method is that the solution is done in the spectral domain. This has two implications. First, the heat equation (4) can be solved using the eigenbasis. One way to do this is to apply the Woodbury matrix inversion identity [41] to (4), which will still cost at least $O(n^2)$ and might be inaccurate. We chose to approximate the solution directly through the spectrum (following, e.g., [32]) as the truncated series

$$\hat{h}_t(x_s, y) = \sum_{i=1}^{k} e^{-\lambda_i t} \phi_i(x_s) \phi_i(y). \qquad (7)$$

The function $\hat{h}_t(x_s, y)$ approximates as the *heat kernel*, which measures the amount of heat that flows from a source point x_s to point y after time t. For the case of multiple (or non-point) sources, the solution for a set of source points $\chi_s = \{x_i\}_{i=1}^s$ is the average of the heat kernel of all source points, $\hat{h}_t(\chi_s, y) = \frac{1}{s} \sum_i \hat{h}_t(x_i, y)$. We will henceforth denote the values of $\hat{h}_t(x_s, y)$ at all vertices y, and averaged over all sources χ_s in the graph as $\mathbf{h}_t$.

In essence (7) is the spectral approximation of the solution of the 'exact' heat diffusion (4), but is more efficient as it can be done in $O(nk)$ instead of $O(n^2)$ with $k \ll n$. The approximation is accurate as long as the time constant t is sufficiently big and the residual of the truncated series (7) is small. This formulation allows further acceleration down to $O(k^2)$ as described in the sequel.

The second implication is that we no longer need to solve the full $n \times n$ Poisson equation (6), but rather find a set of k Fourier coefficients $\mathbf{a}$ defining a low-rank approximation of the function $\hat{g} = \mathbf{\Phi a}$, such that its gradient $\nabla \hat{d} = \nabla \mathbf{\Phi a}$ is the closest in the ℓ_2 sense to the normalized gradient $\hat{g} = \nabla \mathbf{h}_t / \|\nabla \mathbf{h}_t\|$ of the approximate heat kernel, resulting in the following closed-form expression

$$\mathbf{a}^* = \underset{\mathbf{a}}{\operatorname{argmin}} \|\widehat{\mathbf{g}} - \nabla \mathbf{\Phi a}\|_2 = (\nabla \mathbf{\Phi})^\dagger \widehat{\mathbf{g}}, \qquad (8)$$

where † marks pseudo inverse. Note that this formulation does not rely on the divergence operator. Since the coefficient(s) of the constant eigenfunction(s) cannot be determined from (8) (as the corresponding gradients vanish), we set them to make the entire map non-negative or zero at the source.

Random walk. The only difference for general (non-mesh) graphs is that the heat equation is replaced with random walk process, where the notion of 'heat' is replaced with probability of arrival at a location y after t steps. We follow the definition from [39] for this probability

$$\widehat{h}_t(x_s, y) = \sum_{i=1}^{k} (1 - \lambda_i)^t \phi_i(x_s) \phi_i(y), \qquad (9)$$

where here $\{\lambda_i, \phi_i\}$ are the eigenvalues and the corresponding eigenfunctions of the random-walk laplacian $\mathbf{L}_{rw}$.

Sublinear approximation. Both the approximation of the heat kernel (7) and the solution of (8) are $O(nk)$. However, we observe that both (7) and (8) are highly overdetermined, in the case of $k \ll n$, and still can be solved unambiguously if only a subset of $O(k)$ graph vertices is considered. This reduces the overall complexity of the method to $O(k^2)$, which is sublinear in n. This excludes, of course, the pre-computation mentioned above which costs no less than $o(nk)$ for eigen decomposition, then $O(nk \log n)$ for FPS, and finally $O(k^3)$ for the matrix inversion in (8).

It is important to emphasize that this sublinear complexity is for a single pairwise distance, and increasing the number of sources or destinations will increase the overall complexity. For example, approximating the distance from one point to the entire vertex set will still cost $O(nk)$. That said, empirical results show that our method is still considerably faster than all mentioned methods, even in the latter case.

In practice we sample the graph using FPS based on one of the 'exact' method mentioned previously. The minimum amount of samples is the number of non-constant eigenfunctions. We use an amount of distinct samples slightly higher than what makes $\nabla \mathbf{\Phi}$ in (8) numerically full rank.

A similar sampling is performed by Aflalo et al. [3], but is not limited by the basis size $O(k)$, nor can it be generalized to full distance computation, as shown below.

Finding optimal value of t. In [11], the time value t in the heat equation was set to be the squared average edge length, based on discretization arguments. This selection seem to work in practice for most shapes even though there is no worst-case guarantee, and still failing in some isolated cases (e.g. in Figure 1 we had to use ten times the latter value).

As shown in [8], the spectral heat kernel in (7) does not depend on the discretization, but does not scale trivially with the global scaling of the shape. A scale agnostic version of (7) would then be

$$\hat{h}_t(x_s, y) = \operatorname{tr}(\mathbf{M}) \sum_{i=1}^{k} e^{-\lambda_i t \operatorname{tr}(\mathbf{M})} \phi_i(x_s) \phi_i(y), \qquad (10)$$
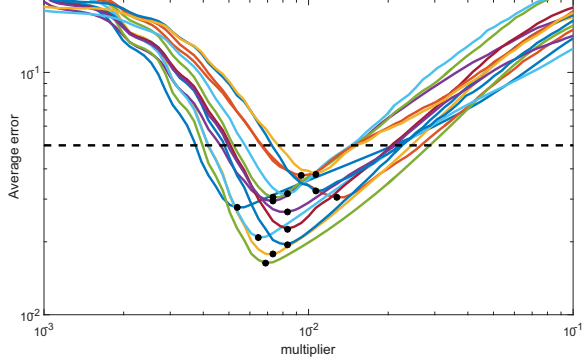
Figure 2: Average error (measured relative to FMM and normalized by the shape diameter) as a function of the multiplier m in the selection of the time parameter $t = m \operatorname{tr}(\mathbf{M})$. Each curve corresponds to a shape set from Table I in [11] or from the TOSCA dataset [8]. Notice that in most examples a uniform setting $m = 8 \times 10^{-3}$ is close to the shape-specific optimal parameter value (black dots) and yields mean error below 5% (dashed line). This figure was inspired by its counterpart in [11].

basis size	25	50	100	250	350	500
relative	4.28	2.51	1.79	0.77	0.60	0.43
ℓ_2	1.36	0.87	0.64	0.29	0.24	0.17
ℓ_∞	4.90	3.18	2.39	1.39	1.15	0.96

Table 1: Average error of projecting FMM distance on the Laplacian basis of the cat shape (26×10^3 vertices) from TOSCA [8] shown in Figure 1. Shown for different number of harmonics are the ℓ_2 and ℓ_∞ errors in % of intrinsic diameter as well as the relative error in %. The ℓ_∞ error does not decay fast due to the Gibbs phenomenon [16]. The errors are averaged over several runs with difference source points.

where $\operatorname{tr}(\mathbf{M})$ is the shape area. In a manner similar to [11], we also choose define the time value t by empirically setting a multiplier m

$$t = m \operatorname{tr}(\mathbf{M}). \qquad (11)$$

Testing on several shapes from various sources, we set $m = 8 \times 10^{-3}$, as can be drived from Figure 2. Note, that while in [11] the time value is discretization dependent, our time value depends only on the total area of the shape.

Comparison to spectral MDS. As mentioned previously, one part of the Spectral MDS (SMDS) method [3] includes distance calculations seemingly similar the SpectroMeter. This part of the SMDS algorithm includes interpolation of distance maps sampled on a sparse set of points, based on a projection on the Laplacian eigenbasis. While

FMM [19]   SMDS [3]

[3] report very low reconstruction errors, these are true only for maps whose origin is close to one of the former samples. If one would use the same method with origin far from these samples, the result will look similar to the result of (7), i.e. non-unit magnitude gradients with the correct direction (as can be seen in the inset).

5. Experimental evaluation

In what follows, we assess the actual performance of the discussed approximation method compared to the 'exact' methods, and highlight some of its limitations. Two fla-

vors of SpectroMeter will be tested: *full*, which uses the entire vertex set, and *sublinear*, which uses only the predefined $O(k)$ samples. For comparison, we used a Matlab mex implementation of FMM from the authors of [19], and a Matlab implementation of the Heat method accelerated by the CHOLMOD algorithm [9], used originally in [11]. We chose not to compare to the exact geodesic distance on polyhedra [24] in the large scale experiments, as its results are very similar to FMM result (less than 0.5% error), while the computational time is prohibitively slow.

Errors are measured using the ℓ_2 and ℓ_∞ norms, normalized by the graph diameter. Another criterion we measure is relative error, defined as $\frac{|d - d_{ref}|}{d_{ref} + \epsilon}$ for a result d versus a reference value of d_{ref}. We emphasize that latter error is somewhat biased *against* our method: errors of spectral representation usually concentrate in points where the function is non-differentiable, e.g., at the sources, where the relative error is most sensitive (small denominator). Nevertheless, we include these statistics here as it was used in previous studies [3, 11, 33] to which we compare.

5.1. Spectral representation accuracy

First, we test the accuracy of projecting (essentially, compressing) distance maps on the Laplacian eigenbasis. Table 1 shows reconstruction error of FMM distance maps, averaged over the shapes from Figure 2, using different numbers of harmonics. As suggested by [2], the ℓ_2 error is decaying very fast, and is quite negligible even for $k = 100$ harmonics. We attribute this to the fast decay of coefficients of distance maps, which can be shown to be quadratic for Euclidean domains [1]. The ℓ_∞ error does not decay as fast, which can be related to the *Gibbs phenomenon* [16].

Next, for comparison, we repeat the experiment only this time using SpectroMeter sublienar (full has very similar stats). The result, summarized in Table 2, suggest that our method achieves maximum performance around $k = 250$ harmonics after which diminishing returns are observed. This as well, may be due to the fast decay of coefficients [1]. Since this result was consistent across many shapes we

basis size	25	50	100	250	350	500
relative	55.86	32.41	15.80	9.93	9.74	9.81
ℓ_2	19.33	13.63	6.35	2.84	2.88	3.10
ℓ_∞	46.17	37.05	22.14	7.46	6.88	7.65

Table 2: Average error of our method compared to FMM, in the same setting as Table 1. It can be seen that our method achieves its best performance at $k = 250$, and does not improve with more harmonics, which can be attributed to the fast decay of coefficients [1] in the presence of noise. The relative error is high since spectral reconstruction has high error around non-differentiable areas like the source.
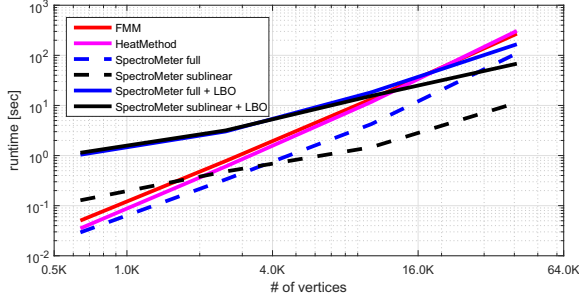

Figure 3: Runtime of different methods, on different graph sizes. Presented is the time for calculating pairwise distance from 15% of the vertices to all the rest on a sphere mesh. Shown are: FMM, Heat method and SpectroMeter, both full and sublinear. Even though this task has quadratic complexity, our method scales better than the exact ones, even when counting the complexity of basis construction (marked 'LBO' in the plot).

tested, in all the following experiments we set $k = 250$.

5.2. Complexity analysis

In order to measure how all methods scale with the graph size, we ran them on an implicit shape discretized as a mesh, with gradually increasing vertex count. For each mesh, we measured the distance from 5% of the vertices to all the rest. Note that due to the size of the result, this task has quadratic complexity in the size of graph, even for our sublinear method. As can be seen in Figure 3, our approach outperforms the other ones, and the incline of the sub-linear flavor of our method is the lowest. If we include the eigendecomposition into the timing, both SpectroMeter implementations are still faster for large graphs.

Table 3 shows error of the latter experiment compared to FMM. While the errors of SpectroMeter are roughly twice as high as the heat method, it scales more gracefully for larger graphs.

5.3. Distance on a nearest neighbor graph

One of the contributions of this work is that we do not need do define a divergence operator. This allows us to

method	# vertices err type	0.6K	2.5K	10K	20K
heat	relative	3.24	2.19	1.52	1.11
method	ℓ_2	1.35	0.92	0.66	0.51
[11]	ℓ_∞	1.95	1.42	1.08	0.86
	relative	6.25	6.53	6.32	6.09
full	ℓ_2	2.37	2.39	2.27	2.16
	ℓ_∞	2.92	3.41	3.69	3.82
	relative	6.23	6.53	7.17	6.77
sublinear	ℓ_2	2.36	2.39	2.26	2.13
	ℓ_∞	2.92	3.40	3.67	3.79

Table 3: Errors of distance methods w.r.t. to an FMM result on the same corpus, measured in % of the intrinsic diameter of a sphere mesh with different vertex count.

apply the method on non-manifold graphs, for example a graph generated by running k nearest neighbors (k-NN) on data from a Euclidean domain. We generated a graph from 7-NN of 3×10^3 points in $\mathbb{R}^5$, which resulted in a graph Laplacian with similar sparsity to the meshes in other experiments.

Figure 4 depicts distance maps from three source points, visualized in two dimensions using t-SNE [20]. The resulting maps are very similar to the exact ones generated using Dijkstra's method [14], but require a fraction of the computational cost.

In order to give a qualitative measure for the results, we sorted the vertices according to the resulting distance in each map separately, and measured Kendel's τ [18] as the permutation distance

$$\tau = \frac{\#\,\text{discordant pairs}}{n\,(n-1)\,/\,2}, \qquad (12)$$

compared to Dijkstra's result. On average, projection onto the Laplacian eigenbasis gave 12% error, while both types of SpectroMeter gave 14% error. While this may seem high, note that random guess results in 50% error, and the probability of randomly achieving error below 30% vanishes quickly even for permutations with $n = 50$.

5.4. Shape correspondence

We now show our methods' result on a recent method for shape correspondence by Vestner et al. [38]. Due to lack of space, this section is not self-contained, as we only cover the parts relevant for this work. We refer the reader to the aforementioned manuscript for details. In a nutshell (skipping all theoretic justification), the latter method finds bijective correspondence by solving a linear assignment problem (LAP) over all the vertices

$$\mathbf{\Pi}^* = \min_{\mathbf{\Pi} \in \mathcal{P}} \text{tr}\left(\exp(-\mathbf{D}_2/\sigma^2)\mathbf{\Pi}_0\mathbf{D}_1\mathbf{\Pi}\right), \qquad (13)$$

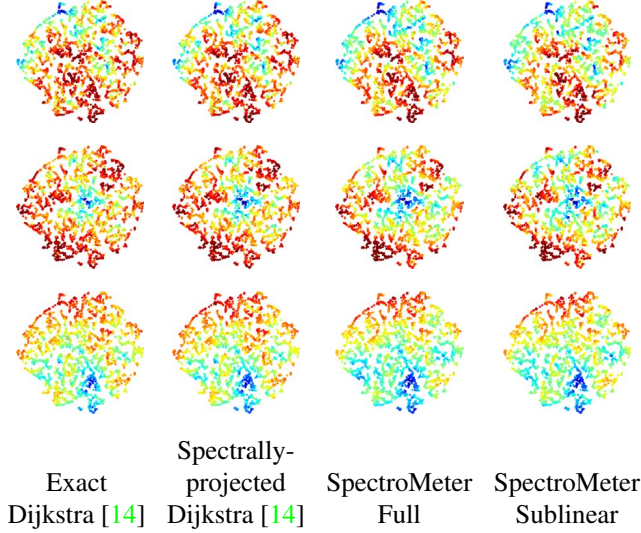| Exact Dijkstra [14] | Spectrally-projected Dijkstra [14] | SpectroMeter Full | SpectroMeter Sublinear |

Figure 4: Distance maps produced by four methods on a graph created using 7-NN in $\mathbb{R}^5$. The graph vertices are visualized in 2D using t-SNE [20]. Shown in each row is a results for a different source, color mapped from blue (small distance) to red (large distance).

where $\mathbf{D}_1$ and $\mathbf{D}_2$ are pairwise distance matrices, $\exp$ is element-wise exponent, $\mathbf{\Pi}_0$ is an initial correspondence guess (possibly non bijective) and $\mathcal{P}$ is the set of all possible permutations of size n. LAP has an $O(n^3)$ worse-case complexity, but can be solved in $O(n^2 \log n)$ operation on average using the auction method [6] (taking, e.g., a few seconds for a problem of size $n = 10^3$). In practice, the smooth geometry of the problem allows further multi-scale acceleration, which makes the LAP cost matrix sparse (zero entries indicate forbidden matches). In the particular case of triangular meshes, truncating the pairwise distance matrices at 5% of the diameter we can achieve about 99% sparsity, allowing solution of a large LAP problem of size $n = 10^4$ in about one minute. The main bottleneck that remains is the calculation of the full pairwise distance matrices.

We tested the correspondence method on the FAUST dataset [7] containing 3D scans of human shapes with 7×10^3 vertices. We solve LAP using the sparse auction solver publicly provided by the authors of [4]. The initial guess input $\mathbf{\Pi}_0$ was a obtained by using the current state-of-the-art correspondence method by Rodolà et al. [29] on a coarse version of the shape with 2×10^3 vertices, and interpolating the result to full resolution using nearest neighbors. Note that the initial guess method [29] also uses the Laplacian eigendecomposition, making Spectrometer's complexity even more attractive in this case: Average run times in seconds (excluding LAP) is 260 for FMM [19], 120 for the heat method [11], compared to 42 and 24 respectively for the full and sublinear flavors of SpectroMeter.

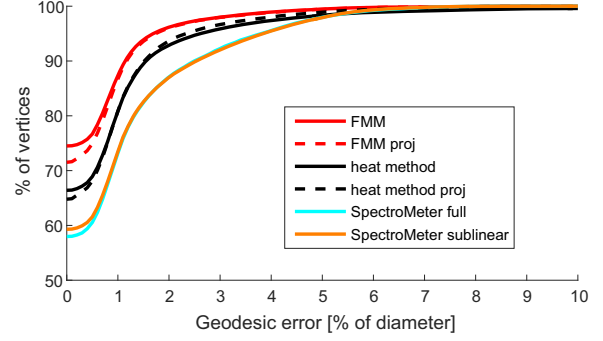Figure 5 shows results of the method [38] using several

Figure 5: Performance of the shape correspondence method from [38] on the FAUST dataset [7], based on different distance methods as input. Dashed lines indicate that the distance was projected onto the LBO basis (i.e. compressed). SperctroMeter yields slightly lower results due to the fact it is an approximation, but can further scale up to very big shapes.

distance methods as input. We also tested the effect of projecting the distance matrix onto the Laplacian eigenbasis. While the performance is lower for Spectrometer compared to the exact distance, the overall method of [38] becomes more scalable as it can be run for shapes of very high vertex count.

6. Conclusion

We presented an algorithm allowing to approximate pairwise distances on sparsely-connected graphs (such as, for example, triangular meshes) in sub-linear time given the low-frequency eigenfunctions of the Laplacian. Empirical evidence exhibits very good ℓ_2 approximation errors and reasonable ℓ_∞ errors.

Similarly to both the heat method [11] and FMM [30], the obtained distance map might violate the triangle inequality or distance symmetry. For similar reasons, it is possible that these distance maps cannot be used to compute the geodesics themselves via back-tracking, due to possible existence of local extrema along the path.

The influence of boundary conditions that has been ignored here deserves attention. Additionally, we would like to devise a criterion more robust than ℓ_2 for the solution of (8). Lastly, but most importantly, rigorous theoretical bounds on the average and worst-case approximation errors shall be developed.

References

[1] The fourier transform of absolute value of *t*. http://www.thefouriertransform.com/pairs/absT.php. Accessed: 2016-07-11. 6, 7

[2] Y. Aflalo, H. Brezis, and R. Kimmel. On the optimality of shape and data representation in the spectral domain. *SIAM Journal on Imaging Sciences*, 8(2):1141–1160, 2015. 4, 6

[3] Y. Aflalo and R. Kimmel. Spectral multidimensional scaling. *Proceedings of the National Academy of Sciences*, 110(45):18052–18057, 2013. 2, 5, 6

[4] F. Bernard, N. Vlassis, P. Gemmar, A. Husch, J. Thunberg, J. Goncalves, and F. Hertel. Fast correspondences for statistical shape models of brain structures. In *SPIE Medical Imaging*, pages 97840R–97840R. International Society for Optics and Photonics, 2016. 8

[5] M. Bernstein, V. De Silva, J. C. Langford, and J. B. Tenenbaum. Graph approximations to geodesics on embedded manifolds. Technical report, Technical report, Department of Psychology, Stanford University, 2000. 2, 3

[6] D. P. Bertsekas. *Network optimization: continuous and discrete models*. Citeseer, 1998. 8

[7] F. Bogo, J. Romero, M. Loper, and M. J. Black. FAUST: Dataset and evaluation for 3D mesh registration. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, Piscataway, NJ, USA, June 2014. IEEE. 8

[8] A. M. Bronstein, M. M. Bronstein, and R. Kimmel. *Numerical geometry of non-rigid shapes*. Springer Science & Business Media, 2008. 1, 5, 6

[9] Y. Chen, T. A. Davis, W. W. Hager, and S. Rajamanickam. Algorithm 887: Cholmod, supernodal sparse cholesky factorization and update/downdate. *ACM Transactions on Mathematical Software (TOMS)*, 35(3):22, 2008. 6

[10] F. R. Chung. *Spectral graph theory*, volume 92. American Mathematical Soc., 1997. 3

[11] K. Crane, C. Weischedel, and M. Wardetzky. Geodesics in Heat: A New Approach to Computing Distance Based on Heat Flow. *ACM Trans. Graph.*, 32, 2013. 1, 2, 3, 4, 5, 6, 7, 8

[12] D. M. Cvetković, M. Doob, and H. Sachs. *Spectra of graphs: theory and application*, volume 87. Academic Pr, 1980. 4

[13] P.-E. Danielsson. Euclidean distance mapping. *Computer Graphics and image processing*, 14(3):227–248, 1980. 2

[14] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959. 2, 7, 8

[15] Y. Eldar, M. Lindenbaum, M. Porat, and Y. Y. Zeevi. The farthest point strategy for progressive image sampling. *IEEE Transactions on Image Processing*, 6(9):1305–1315, 1997. 5

[16] E. Hewitt and R. E. Hewitt. The gibbs-wilbraham phenomenon: an episode in fourier analysis. *Archive for history of Exact Sciences*, 21(2):129–160, 1979. 6

[17] S.-R. Hysing and S. Turek. The eikonal equation: numerical efficiency vs. algorithmic complexity on quadrilateral grids. In *Proceedings of ALGORITMY*, volume 2005, pages 22–31, 2005. 2

[18] M. G. Kendall. A new measure of rank correlation. *Biometrika*, 30(1/2):81–93, 1938. 7

[19] R. Kimmel and J. A. Sethian. Fast marching methods on triangulated domains. In *Proceedings of the National Academy of Science*, volume 95, pages 8341–8435, 1998. 2, 6, 8

[20] L. v. d. Maaten and G. Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008. 7, 8

[21] M. Meila and J. Shi. A random walks view of spectral segmentation. 2001. 3

[22] F. Mémoli and G. Sapiro. Distance functions and geodesics on submanifolds of \r^d and point clouds. *SIAM Journal on Applied Mathematics*, 65(4):1227–1260, 2005. 2

[23] M. Meyer, M. Desbrun, P. Schröder, and A. H. Barr. Discrete differential-geometry operators for triangulated 2-manifolds. In *Visualization and mathematics III*, pages 35–57. Springer, 2003. 3

[24] J. S. Mitchell, D. M. Mount, and C. H. Papadimitriou. The discrete geodesic problem. *SIAM Journal on Computing*, 16(4):647–668, 1987. 1, 2, 6

[25] B. Mohar. The laplacian spectrum of graphs. In *Graph Theory, Combinatorics, and Applications*, 1991. 3, 4

[26] J. Moy. Ospf version 2. 1997. 1

[27] D. Oran. Osi is-is intra-domain routing protocol. 1990. 1

[28] J. Pokrass, A. M. Bronstein, and M. M. Bronstein. A correspondence-less approach to matching of deformable shapes. In *International Conference on Scale Space and Variational Methods in Computer Vision*, pages 592–603. Springer, 2011. 3

[29] E. Rodola, M. Moeller, and D. Cremers. Point-wise map recovery and refinement from functional correspondence. In *International Symposium on Vision, Modeling and Visualization (VMV)*, 2015. 8

[30] J. A. Sethian. A fast marching level set method for monotonically advancing fronts. *Proceedings of the National Academy of Sciences*, 93(4):1591–1595, 1996. 1, 2, 4, 8

[31] O. Sorkine. Differential representations for mesh processing. In *Computer Graphics Forum*, volume 25, pages 789–807. Wiley Online Library, 2006. 3, 4

[32] J. Sun, M. Ovsjanikov, and L. Guibas. A concise and provably informative multi-scale signature based on heat diffusion. In *Computer graphics forum*, volume 28, pages 1383–1392. Wiley Online Library, 2009. 5

[33] V. Surazhsky, T. Surazhsky, D. Kirsanov, S. J. Gortler, and H. Hoppe. Fast exact and approximate geodesics on meshes. In *ACM transactions on graphics (TOG)*, volume 24, pages 553–560. ACM, 2005. 2, 6

[34] Y.-h. R. Tsai. Rapid and accurate computation of the distance function using grids. *Journal of Computational Physics*, 178(1):175–195, 2002. 2

[35] Y.-H. R. Tsai, L.-T. Cheng, S. Osher, and H.-K. Zhao. Fast sweeping algorithms for a class of hamilton–jacobi equations. *SIAM journal on numerical analysis*, 41(2):673–694, 2003. 2

[36] J. N. Tsitsiklis. Efficient algorithms for globally optimal trajectories. *IEEE Transactions on Automatic Control*, 40(9):1528–1538, 1995. 2

[37] S. R. S. Varadhan. On the behavior of the fundamental solution of the heat equation with variable coefficients. *Commu-*

nications on Pure and Applied Mathematics, 20(2):431–455, 1967. 3

[38] M. Vestner, R. Litman, A. Bronstein, E. Rodola, and D. Cremers. Bayesian inference of bijective non-rigid shape correspondence. *arXiv preprint arXiv:1607.03425*, 2016. 7, 8

[39] U. Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4):395–416, 2007. 4, 5

[40] O. Weber, Y. S. Devir, A. M. Bronstein, M. M. Bronstein, and R. Kimmel. Parallel algorithms for approximation of distance maps on parametric surfaces. *ACM Transactions on Graphics (TOG)*, 27(4):104, 2008. 2

[41] M. A. Woodbury. Inverting modified matrices. *Memorandum report*, 42:106, 1950. 5

[42] L. Yatziv, A. Bartesaghi, and G. Sapiro. O (n) implementation of the fast marching algorithm. *Journal of computational physics*, 212(2):393–399, 2006. 2

[43] H. Zhao. A fast sweeping method for eikonal equations. *Mathematics of computation*, 74(250):603–627, 2005. 2

[44] H.-K. Zhao, S. Osher, B. Merriman, and M. Kang. Implicit and nonparametric shape reconstruction from unorganized data using a variational level set method. *Computer Vision and Image Understanding*, 80(3):295–314, 2000. 2