COMPUTING SHAPE DNA USING THE CLOSEST POINT METHOD

RACHEL HAN* AND CHINGYI TSOI†

Abstract. We demonstrate an application of the closest point method to numerically computing the truncated spectrum of the Laplace–Beltrami operator. This is known as the "Shape DNA" and it can be used to identify objects in various applications. We prove a result about the null-eigenvectors of the numerical discretization. We also investigate the effectiveness of the method with respect to invariants of the Shape DNA. Finally we experiment with clustering similar objects via a multi-dimensional scaling algorithm.

Key words. closest point method, Laplace–Beltrami operator, shape DNA, eigenvalue problem, multi-dimensional scaling, numerical analysis

1. Introduction. The Laplace–Beltrami operator is widely used in geometric modelling and computer graphics for applications such as smoothing, segmentation and registration of 2D or 3D shapes [9], [13]. Another novel application is characterizing shapes by extracting their "fingerprints" or "*Shape DNA*", as first introduced by Reuter, Wolter and Peinecke [10]. Storing and processing the Shape DNA enables fast retrieval and identification in a database of shapes and has applications in areas such as machine learning. The full spectrum of the Laplace–Beltrami operator on a surface is able to identify distinct shapes because it contains intrinsic information, such as volume and surface area [10]. Also, identification via Shape DNA is robust since it is isometry invariant (isometries include rotation, translation and reflection) and independent of parametrization, reducing the preprocessing of the shapes.

In this project, the robustness of Shape DNA is improved further by using the closest point method [12], a numerical technique which represents surfaces without using a parametrization. Building on previous work [7, 16, 1], we use the closest point method to discretize the Laplace–Beltrami eigenvalue problem and MATLAB to solve the resulting matrix eigenvalue problem on various surfaces. Figure 1 shows some eigenfunctions computed with this method. We measure the numerical errors and compare to expectations in the literature. Furthermore, we analyze the effectiveness of the default MATLAB "`eigs`" algorithm and compare its performance with our indirect approach. We improve the numerical analysis of the closest point method by proving that the numerical discretization is singular with a constant eigenvector in cases where the Laplace–Beltrami operator is itself singular (that is, closed surfaces and the Neumann problem), and that it does not have a constant eigenvector for the Dirichlet problem. Finally, we use multidimensional scaling plots to represent the similarities between different surfaces based on their Shape DNA.

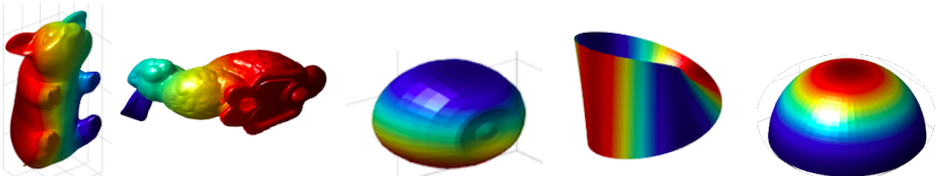

Fig. 1: Eigenfunctions on surfaces: pig, Stanford bunny, apple, Möbius strip and hemisphere.

*University of British Columbia (hanrach@alumni.ubc.ca)
†Hong Kong Baptist University (chingyitsoi@gmail.com)

2. The Laplace–Beltrami Eigenvalue Problem. The Laplace–Beltrami operator on curved surfaces is analogous to the standard Laplacian operator. For example, it can model in-surface diffusion. To compute the spectrum of the Laplace–Beltrami (LB) operator Δ_s on a surface S, we need to find eigenpairs (λ, u) which solve the functional problem

$$-\Delta_s u(x) = \lambda u(x) \quad \text{for } x \in S, \tag{1}$$

where the *eigenvalue* λ is a scalar number and the *eigenfunction* $u : S \to \mathbb{R}$ is a nonzero function defined on S. For example, this eigenvalue problem (1) appears when doing separation of variables on the wave equation. The eigenfunctions can then be interpreted as vibration patterns (standing waves) of the domain, and the eigenvalues are associated with the frequency of those waves.

The famous paper by Kac [5] poses the question whether one can identify the shape given the eigenvalues of the Laplacian operator. In fact, two different membranes can give an identical set of eigenvalues as shown through what are known as "isospectral domains", constructed by Gordon et al. [4]. Given that we can infer some information of the surfaces using the spectra, if we are given a frequency of a membrane, can we guess which shape of the drum it is most "similar" to? Also, in what ways are the surfaces "similar"?

In this paper, we attempt to classify surfaces using the LB spectra by automatically clustering them, where similarities are optimally determined through a clustering algorithm which uses the Euclidean dissimilarity metric, which will be discussed in Section 6. We measure the success of the clustering empirically, by observing how close the generated clusters are from the expected clustering based on intuitive characteristics of the surfaces. The characteristics include roundness of the surface and presence and treatment of boundaries. To do this, we use the following notion of Shape DNA defined by Reuter et al. [10].

DEFINITION 2.1. *For a given surface, the* Shape DNA *is the list of the first 50 smallest-in-magnitude eigenvalues of the Laplace–Beltrami operator, scaled by the first nonzero eigenvalue such that the first nonzero number in the Shape DNA is 1.*

The nonzero scaling factor makes the Shape DNA invariant to scaling. Appendix A demonstrates the derivation of the Shape DNA for a circle.

3. The Closest Point Method. Suppose we have a surface S and a function defined on it $u : S \to \mathbb{R}$. For each grid point in a narrow band near the surface, an extension operator maps the function value from the point on the surface that is closest to the grid point. That is, the closest point extension is a map from a function defined on S to a function defined on a narrow band of S, where the function defined on the narrow band is constant along the normals to the surface S. It can be used to express derivatives on the surface in terms of derivatives in the embedding space.

3.1. Mathematical formulation. We begin with some definitions before formulating eigenvalue problems in the embedding space.

DEFINITION 3.1 (Closest point function [12]). *Let S be a smooth surface in $\mathbb{R}^d$. Then $cp(x)$ refers to a point belonging to S which is closest to x.*

Knowing values of this function is how we represent the surface in our method. For example, the closest point representation of the circle is illustrated in Figure 2.

DEFINITION 3.2 (Closest point extension [12]). *Let S be a smooth surface in $\mathbb{R}^d$. The closest point extension of a function $u : S \to \mathbb{R}$ to a neighborhood $\Omega \subset \mathbb{R}^d$ of S*

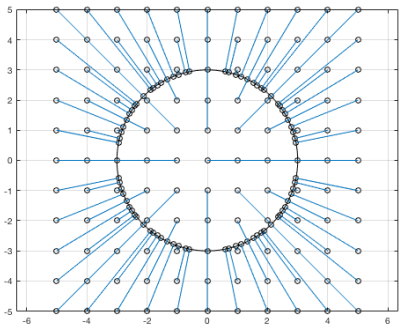

Fig. 2: Closest point function for a circle. In practice, we need this mapping only for a uniform grid, as shown. If the closest point is not unique (such as the centre of the circle), we choose any closest point.

is the function $v : \Omega \to \mathbb{R}$ *defined by* $v(x) = u(cp(x))$. *E is the operator which maps u to v:* $v(x) = u(cp(x)) \iff v = Eu$.

If $v = Eu$ then the intrinsic surface gradient $\nabla_s u(x)$ is same as $\nabla v(x)$ evaluated at $x \in S$, because the closest point extension of u is constant in the normal direction [12]. Similar results hold for other surface differential operators [12, 8], including the Laplace–Beltrami operator, which is the surface divergence of the surface gradient.

THEOREM 3.3 ([12]). *Let S be a smooth surface in $\mathbb{R}^d$ and $u : S \to \mathbb{R}$ be a smooth function. Assume the closest point function $cp(x)$ is uniquely defined in a narrow neighbourhood $\Omega \subset \mathbb{R}^d$ of S and that $v(x) = u(cp(x))$. Then*

$$\Delta_s u(x) = \Delta v(x) \qquad \text{for } x \in S.$$

Following the approach of [16], we start with (1) and extend both sides of the equation to obtain $-E\Delta_s u = E\lambda u$. Applying Theorem 3.3 gives $-E\Delta v = \lambda E u = \lambda v$, where $v = Eu$. This also means $v = Ev$ [16]. Based on this result, we consider an eigenvalue problem not on the surface, but rather on Ω, the narrow band of $\mathbb{R}^d$ surrounding the surface:

$$\begin{aligned} -E\Delta v &= \lambda v \quad \text{for } x \in \Omega, \\ \text{subject to} \quad v &= Ev \quad \text{for } x \in \Omega. \end{aligned} \tag{2}$$

To recover the eigenfunction on the surface we evaluate v on the surface, that is, restrict $u = v|_s$.

Notably, the eigenvalue problem (2) is not the same as solving the Laplacian eigenvalue problem $-\Delta v = \lambda v$; the constraint $v = Ev$ enforces the constant-along-normal constraint on the solution. Also, because of the E on the left-hand side, the embedded space eigenvalue problem (2) does not require imposing *additional* boundary conditions at the edges of Ω [12, 7].

To deal with the constraint, we use the method-of-lines penalty approach [16] by von Glehn et al. This is an equivalent formulation to (2), with a penalty parameter γ which controls the constraint. It gives the eigenvalue problem

$$-\big(E\Delta v - \gamma(v - Ev)\big) = \lambda v, \quad \text{for } x \in \Omega. \tag{3}$$

Combining the arguments of [7] and [16] we can show solutions of (1) solve (3). For $|\lambda| < |\gamma|$, solutions of (3) restricted to the surface solve (1).

3.2. Numerical discretization. To solve the Laplace–Beltrami eigenvalue problem, we use finite difference schemes to approximate the problem (3) on uniform Cartesian grids of the embedding space Ω. Let Δx be the grid size. Let $\mathbf{v}$ be the discretized vector consisting of samples of the function v at discrete grid points. We construct a sparse matrix L, the discretized Laplacian using standard Cartesian centered approximations of the second derivative with second-order accuracy [6].

In the implementation of the closest point method, we use a sparse matrix E_p to approximate the closest point extension operator E. The matrix E_p interpolates the function values of $\mathbf{v}$ onto the closest points of each grid point. This is done using Lagrange polynomial interpolation of degree p using the neighbouring grid points.

Combining these, we discretize the Laplace–Beltrami operator Δ_S using (3) to obtain

$$M = E_1 L - \gamma(I - E_3). \tag{4}$$

Here I is the identity matrix and E_1 and E_3 are the discretizations of the closest point extension operator using polynomial interpolation of degree 1 and 3. These choices are suggested in [3] for a second-order accurate method.

The parameter γ is chosen according to [16, 3] as $\gamma = \frac{2d}{\Delta x^2}$, where Δx is the grid-spacing and d is the dimension of the embedded space.

3.3. Open surfaces with boundaries. When the surface has a boundary ∂S, the eigenvalue problem (1) is different depending on what boundary conditions are imposed. Common boundary conditions include the *Dirichlet problem:*

$$\begin{aligned} -\Delta_s u(x) &= \lambda u(x) \quad \text{for } x \in S, \\ u(x) &= 0 \quad \text{for } x \in \partial S, \end{aligned} \tag{5}$$

and the *Neumann problem:*

$$\begin{aligned} -\Delta_s u(x) &= \lambda u(x) \quad \text{for } x \in S, \\ \frac{\partial u}{\partial n}(x) &= 0 \quad \text{for } x \in \partial S, \end{aligned} \tag{6}$$

where $\frac{\partial u}{\partial n}$ is the derivative in direction normal to ∂S and tangential to the surface.

If no changes are made to the closest point method as described above in Sections 3.1 and 3.2, the Neumann boundary condition will be imposed at any boundaries ∂S and we will solve (6). This is because the extension E extends the solution constant in the normal direction [12].

To instead solve the Dirichlet problem (5), the method can modified [12] as follows:

- consider the extension operator at each grid point x_i;
- if $cp(x_i)$ is not on the boundary ∂S, we extend the value $u(cp(x_i))$, as usual;
- if $cp(x_i)$ is on ∂S, we extend the negative value of $u(cp(x_i))$.

In practice this corresponds to "flipping the signs" of some rows of the matrices E_1 and E_3.

Imposing either Dirichlet or Neumann boundary conditions as just described will reduce the accuracy of the method to first-order. To recover second-order accuracy for surfaces with boundaries, we make minor modification to the extension operator following [7, §5]. This involves looking up a function value from a "mirror point" in the interior of S (instead of the closest point on the boundary); this gives a smoother extension [7].

3.4. On the Invertibility of the Closest Point Method. In this section, we present new arguments that relate the invertibility of the matrix M to that of the original functional problem.

THEOREM 3.4. *Let S be a smooth closed surface in $\mathbb{R}^d$. Then, the Laplace–Beltrami operator has a constant eigenfunction and the matrix discretization M from (4) has a constant eigenvector. Consequently, both the Laplace–Beltrami operator and M are singular.*

Proof. Assume $u : S \to \mathbb{R}$ is a nonzero constant function, $u(x) = c$ where $c \neq 0$. We want to show that $(u, 0)$ is an eigenpair of Δ_s such that $\Delta_s u = 0$. Applying Theorem 3.3, we have $\Delta_s u = \Delta E u$ for $x \in S$. But $Eu = c$, since the extension of a constant c is c itself. Therefore,

$$\Delta E u = \sum_{j=1}^{d} \frac{\partial^2 c}{\partial x_j^2} = 0.$$

We have shown that the constant function u is an eigenfunction of Δ_s corresponding to the zero eigenvalue, $\lambda = 0$. Therefore, the operator Δ_s is singular.

For the matrix problem, let $\mathbf{v} = \langle c, c, \ldots, c \rangle$, a vector of constants in $\mathbb{R}^d$. Consider

$$M\mathbf{v} = (E_1 L - \gamma(I - E_3))\mathbf{v} = E_1 L\mathbf{v} - \gamma\mathbf{v} + \gamma E_3 \mathbf{v}.$$

Here L is the discrete Laplacian, and $\mathbf{v}$ is a constant vector, so $L\mathbf{v} = \mathbf{0}$. Also, $E_3\mathbf{v} = \mathbf{v}$ because each row of E_3 corresponds to a cubic polynomial interpolation which is exact for constant data (from the uniqueness of the polynomial interpolant [14]). Therefore,

$$M\mathbf{v} = E_1\mathbf{0} - \gamma\mathbf{v} + \gamma\mathbf{v} = \mathbf{0}.$$

We have shown that $(\mathbf{v}, 0)$ is an eigenpair of M such that $M\mathbf{v} = \mathbf{0}$. Therefore, M is a singular matrix. ∎

The presence of boundary conditions modifies the matrix M and the extension operators. The case of Neumann boundary conditions is similar to Theorem 3.4 and the proof is unchanged. However for Dirichlet boundary conditions, the results are quite different as the next theorem shows.

THEOREM 3.5. *Let S be a smooth open surface in $\mathbb{R}^d$. Then, the Laplace–Beltrami Dirichlet problem (5) on S does not have a constant eigenfunction. Furthermore, the constant vector is not an eigenvector of the matrix discretization M from (4).*

Proof. For the first part, we note that $u(x) = c$, a constant, satisfies the condition $-\Delta_s u = 0$, similarly to the proof in Theorem 3.4. However, the boundary condition $u = 0$ (on ∂S) implies that $c = 0$ whereas eigenfunctions must be nonzero.

By way of contradiction, assume the vector of constants $\mathbf{v} = \langle c, c, \ldots, c \rangle$ is an eigenvector of M. As in the proof of Theorem 3.4, $L\mathbf{v} = \mathbf{0}$. Consider a grid point x_i for which $cp(x_i) \in \partial S$. As noted in Section 3.3, this ith row of matrix E_3 will now correspond to the negative interpolant. Thus $E_3(i, :)\mathbf{v} = -c$; in fact all entries of $E_3\mathbf{v}$ are either c or $-c$ (with at least one $-c$). We calculate

$$M\mathbf{v} = E_1 L\mathbf{v} - \gamma\mathbf{v} + \gamma E_3\mathbf{v} = E_1\mathbf{0} - \gamma\mathbf{v} + \gamma\langle \pm c, \pm c, \ldots, \pm c \rangle = \mathbf{w},$$

where the vector $\mathbf{w}$ has jth component

$$w_j = \begin{cases} 0 & \text{if } cp(\mathbf{x}_j) \notin \partial S, \\ -2\gamma c & \text{if } cp(\mathbf{x}_j) \in \partial S. \end{cases}$$

since $\gamma \neq 0$, $M\mathbf{v} \neq \mathbf{0}$. ∎

Figure 3 illustrates the difference between Theorem 3.4 and 3.5.

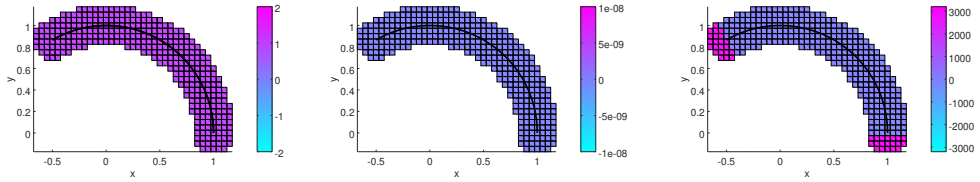

Fig. 3: Narrow bands of grid points surrounding the open black curve S. Left: The figure shows a vector $\mathbf{v}$ of ones at each grid point. Middle: with Neumann boundary conditions, the figure shows $-M\mathbf{v} = \mathbf{0} = 0\mathbf{v}$ at each grid point. It is a multiple of $\mathbf{v}$ and thus $\mathbf{v}$ is an eigenvector. Right: with Dirichlet boundary conditions (right), the figure shows $-M\mathbf{v} = -\mathbf{w}$, evidently not a multiple of $\mathbf{v}$, and hence $\mathbf{v}$ is not an eigenvector. The computations are done with negative sign $-M\mathbf{v}$, to relate back to the original eigenvalue problem (1).

Note Theorem 3.5 does not itself imply that M is nonsingular in the case of an open surface with Dirichlet boundary conditions. Nevertheless, we hypothesize that M is nonsingular in such cases based on numerical observations such as a reasonable condition number of the matrix. In the next section, we will see that the explicit inverse of M is not required in our implementation, further avoiding any issues that may occur if M is close to being singular.

4. Implementation. Once the matrix M in (4) is computed, we can use the MATLAB command "`eigs`" to compute the eigenvalues and the corresponding eigenvectors (u in vector form). Note that the matrix M is a sparse, nonsymmetric matrix. The `eigs` command calls the `ARPack` software which uses Arnoldi iteration to compute the eigenvectors. The idea is similar to power iteration, where we start with a random vector $\mathbf{v}$, and iteratively compute $M\mathbf{v}, M^2\mathbf{v}, M^3\mathbf{v}, ..., M^{n-1}\mathbf{v}$. These form a Krylov matrix

$$\begin{bmatrix} \mathbf{v} & M\mathbf{v} & M^2\mathbf{v} & \cdots & M^{n-1}\mathbf{v} \end{bmatrix}.$$

Arnoldi iteration orthogonalizes this column space through Gram–Schmidt orthogonalization [15]. This sequence converges to the largest eigenvector $\mathbf{v}_{\max}$ corresponding to the largest eigenvalue. However, we want the 50 smallest-in-magnitude eigenvalues, which forces us to compute the largest eigenvectors for the inverse of M. This requires us to solve the subproblem $M\mathbf{v}_n = \mathbf{v}_{n-1}$ every step of the Arnoldi iteration. In the implementation, the shifted inverse iteration method is used and solves

$$(M - \sigma I)\mathbf{v}_n = \mathbf{v}_{n-1} \tag{7}$$

every iteration. The shift σ alleviates the complication associated with singular M.

The default method used by `eigs` for solving the linear system (7) is MATLAB's `backslash` operator. Since M is a nonsymmetric matrix, the LU solver is used. This

direct method does not take advantage of the sparsity of M, and exhausts available memory for a modest $\Delta x = 0.0125$ for solving the eigenvalue problem on a sphere of radius 1. To resolve finer grids for accurate computation using smaller memory space, we implemented an iterative solver using GMRES to solve the subproblem (7). Incomplete LU (ILU) factorization was performed for preconditioning, which was essential for the solution to converge in a reasonable amount of time. Note that the algorithm was applied to $-M$ since we have $-\Delta_s$ from (1). The difference in runtime of computing the Shape DNA of a sphere with ILU preconditioner is tabulated in Table 1. The modified iterative solver was tested on a Linux machine with a 4-core Intel Core Processor (Haswell) with 16 GB of RAM.

Table 1: Runtime of computing Shape DNA of a sphere. ILU includes the total of pre-computing the preconditioner as well as solving the system.

Δx	No preconditioner	GMRES + Incomplete LU
0.05	1047.5 s	132.7 s
0.1	139.8 s	21.6 s
0.2	29.8 s	7.5 s

Table 2: RAM usage percentage when computing sphere Shape DNA.

Δx	Direct solver (%)	Iterative solver (%)
0.0111	N/A	88
0.0125	96.4	71.9
0.025	40.9	17.6
0.05	9	7.4
0.1	6.7	5.9
0.2	6.6	7.2

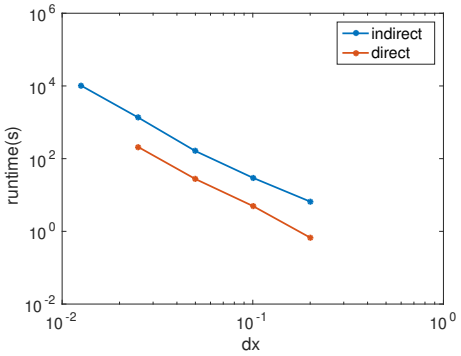

Fig. 4: Log-log scale plot of runtime comparison between indirect and direct eigenvalue solvers. The direct solver outperforms the indirect solver in terms of speed. However, for $\Delta x = 0.0125$, the direct solver runs out of memory and fails to complete the computation.

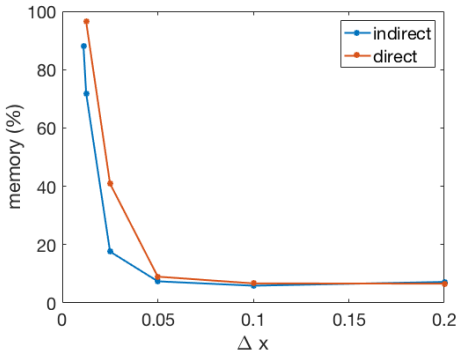

Fig. 5: Memory usage comparison between indirect and direct eigenvalue solvers. The indirect solver outperforms the direct solver for $\Delta x \leq 0.05$. However in both cases, as Δx becomes finer, the memory usage increases rapidly.

We emphasize the trade-off between runtime and memory usage for the smaller Δx. As plotted in Figure 4, the iterative approach is slower than the default direct approach. In Figure 5, we compare the percentage of RAM usage of each solver. The direct solver uses almost all the available memory when $\Delta x = 0.2/16$, whereas the indirect solver exhibits a slower increase for finer values of Δx.

5. Numerical Experiments.

5.1. Error convergence study. We hypothesize that the error in computing the nth eigenvalue λ_n of the LB operator using the closest point method behaves like $C(n)\Delta x^2$. Here Δx is the grid size used in the finite difference scheme, and $C(n)$ is some constant associated with nth eigenvalue. The constant $C(n)$ grows with n because higher eigenvalues are associated with more oscillatory eigenfunctions, so finer grid is required to resolve the increasing oscillations. Therefore, for a fixed Δx, we expect the error to increase as n increases. On the other hand, for a fixed truncated spectrum of $\lambda_1, \ldots, \lambda_{50}$ (the Shape DNA), we expect second-order convergence of errors as Δx decreases. This is because the closest point method is based on a second-order accurate discrete Laplacian and on the choices described in [12, 3].

For our first experiment, we computed the Shape DNA of a unit sphere numerically, and compared with the analytic values, which are known to be $n(n+1)$ with multiplicities $2n+1$ for $n \in \mathbb{N}$ [2]. Figure 6 shows second-order convergence when the error on the vector of Shape DNA is computed in the max norm $\|\cdot\|_\infty$ in $\mathbb{R}^{50}$. Similar results are observed in other norms or for the errors in individual eigenvalues (e.g., [7]).

5.2. Rotational Invariance. We also test the rotational invariance, which is one of the isometry properties of the spectrum. However, the closest point method uses a uniform Cartesian grid and it is possible the numerical solutions are influenced by that grid. As far as we know, this question has not been carefully studied in the literature. We experiment with rotating both open and closed surfaces (which are represented via the closest point method) by $\frac{\pi}{5}, \frac{\pi}{4}, \frac{\pi}{3}$ and $\frac{\pi}{2}$. We compute the Euclidean difference (in $\mathbb{R}^{50}$) between the truncated spectrum of the rotated and non-rotated case. We expect that this error converges at second-order (the design

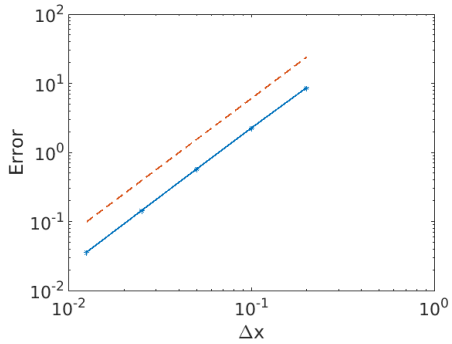

Fig. 6: Log-log scale plot of $\|\cdot\|_\infty$ error in computing the Shape DNA of a sphere. The errors align with the red dotted line of slope 2, indicating second-order convergence.

order of the method) as $\Delta x \to 0$

Figures 7 and 8 show the results for an ellipsoid and hemisphere. These confirm the expected second-order accuracy. In almost all cases, no significant effect on the rotation angle is observed. The one slightly anomolous case is $\frac{\pi}{2}$ of the ellipsoid; where the errors are very similar to no rotation. We suspect that this is due to the fact that the grid rotated by $\frac{\pi}{2}$ aligns well with the grid with no rotation.

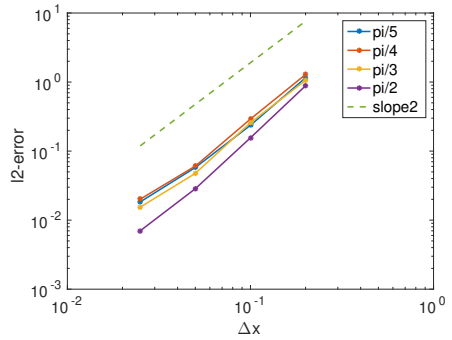

Fig. 7: Ellipsoid spectrum exhibits a second-order convergence for various angles, demonstrating rotational invariance.

5.3. Scaling Invariance. As mentioned in Definition 2.1, we scale the eigenvalue spectra by the first nonzero vector to ensure the Shape DNA is invariant to different scales of shapes. For example, as $\Delta x \to 0$, a sphere of radius 1 should have the same Shape DNA as the sphere of radius 2.

We test this in practice: the error convergence plot over Δx was generated using a torus of minor radius 0.5 and major radius 1, and a torus of minor radius 1 and major radius 2. The Euclidean norm of the difference between the two Shape DNA vectors was calculated over varying grid sizes Δx. In Figure 9, we observe second-order convergence.

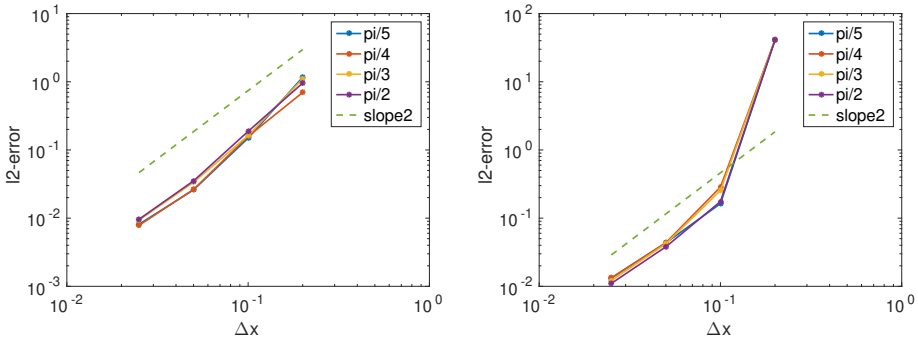

Fig. 8: Rotational invariance test for a hemisphere with Neumann (left) and Dirichlet (right) boundary conditions. The results exhibit the expected second-order convergence of error.

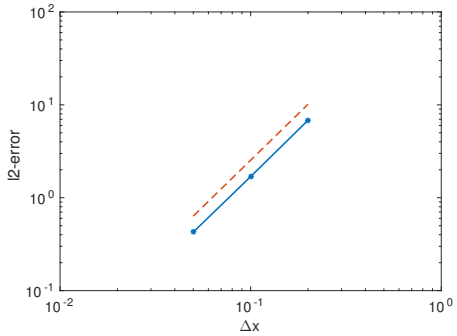

Fig. 9: Computation illustrating scale invariance on torus. The error between the Shape DNA of two Tori of different scales shows second-order accuracy. The red dotted line indicates slope 2.

6. Multidimensional Scaling Plots. It is not clear how to visualize the similarities between surfaces if we are given vectors of Shape DNA of different surfaces. Therefore, we examine the similarities through multidimensional scaling. Multidimensional scaling is a non-supervised learning algorithm that takes in objects with many features and clusters them through nonlinear dimension reduction, as discussed in the paper by Roweis and Saul [11]. The algorithm achieves this by minimizing the cost function

$$f(\hat{Z}) = \sum_{i=1}^{n}\sum_{j=1}^{n} d_3(d_2(\hat{z}_i - \hat{z}_j) - d_1(\hat{x}_i - \hat{x}_j)). \tag{8}$$

$\hat{Z}$ denotes a $m \times n$ matrix composed of n vectors $\hat{z}_i$ in $\mathbb{R}^m$. This algorithm directly minimizes the distance between objects in $\mathbb{R}^m$ ($\hat{z}$-space which is the space we want to visualize, typically $\mathbb{R}^2$ or $\mathbb{R}^3$) and the objects in the original space $\hat{x}$ ($\mathbb{R}^{50}$ since Shape DNA vectors in $\mathbb{R}^{50}$). We can use different metrics when measuring distances in the $\hat{x}$-space and $\hat{z}$-space. This is reasonable because often the two spaces are in

different dimensions ($\hat{x}$-space will usually be in a higher dimension). In (8), d_1 is the high dimensional distance function that measures the pairwise distance of objects in the original space ($\mathbb{R}^{50}$). The distance function d_2 measures in the visualization space (either $\mathbb{R}^2$ or $\mathbb{R}^3$). Another distance function d_3 compares the objects in the original space and the visualization space.

First we compute the spectra of different surfaces. Then we can compute the dissimilarity matrix D which contains pairwise distances between every spectrum. In other words, D contains the distances between objects in the original space $\hat{x}$. For the results below, the standard Euclidean distances were used for all three metrics d_1, d_2 and d_3. Then, we use the MATLAB command `mdscale` to run the multidimensional scaling algorithm and obtain the objects' locations on the $2D$ or $3D$ plot such that the vector distances in $\mathbb{R}^{50}$ between the Shape DNA's are reflected on the plot.

6.1. Torus Experiment. We test a simple notation of similarity: the thickness of tori. We expect tori of similar thickness to be clustered together in the MDS plot. The tori were labelled form 1 to 6, thickest to thinnest. Major radius was fixed to 1 and minor radii were 0.4, 0.3, 0.2, 0.15, 0.1 and 0.05. The shapes were computed using the grid size $\Delta x = 0.05$ such that the thinnest torus can be properly resolved by the closest point method.

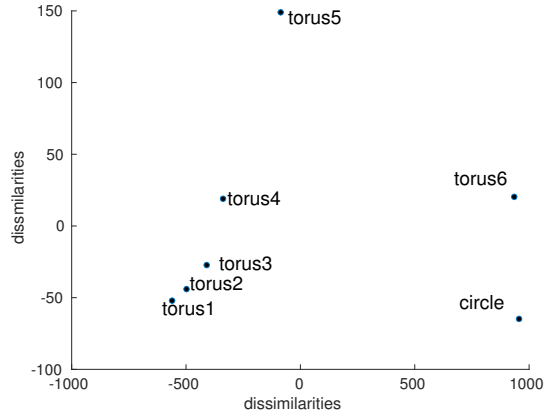

Fig. 10: Torus MDS Plot. We observe the thickness of the torus varying on the horizontal axis – from the thickest torus on the left-most corner to the circle on the far right.

The MDS plot of this experiment is shown in Figure 10. Between 'torus 1', 'torus 2', 'torus 3', and 'torus 4', the minor radius decrements by 0.1 where as for the rest, it decrements by 0.05. However, we observe a more distinct clustering for the thicker tori. One hypothesis is that the thickness of 'torus 5' and 'torus 6' is still poorly resolved by the given grid size, $\Delta x = 0.05$. Also note that the vertical dissimilarity distance scale is smaller than the horizontal scale by magnitudes. While it may seem that 4 and 5 are oddly far apart, their horizontal distance is reasonable. The thinnest 'torus 6' is closest to the circle, which is an expected clustering. This experiment demonstrates that the MDS plots are able to capture thickness of objects.

6.2. Holes on Sphere Experiment. We test various sizes of holes on a unit sphere. We call these shapes "sphere rings"; examples are shown in Figure 11 (left).

Sphere ring 1 and 2 are spheres with small punctures with radii 0.05 and 0.1. Sphere ring 4 and 5 have big holes with radii 0.90 and 0.95, close to that of a unit hemisphere. Sphere ring 3 has a medium sized hole with radius 0.5. The surface labelled 'ring' is created by removing top and bottom caps from a sphere, with radii 0.3 and 0.5 respectively. Neumann boundary conditions were imposed on the open surfaces.[1] Our hypothesis is that Shape DNA of the sphere rings with Neumann boundary conditions will be closer to that of the closed sphere. Thus we expect that 1 and 2 will cluster with the sphere, 4 and 5 with the hemisphere, 3 in the middle and the ring with the Möbius strip.

In Figure 11, MDS better recognizes global similarities than particularities. For example, the general shape of a sphere, hemisphere and ring is distinguished whereas the differences in the openness and the orientation of objects are more subtle.

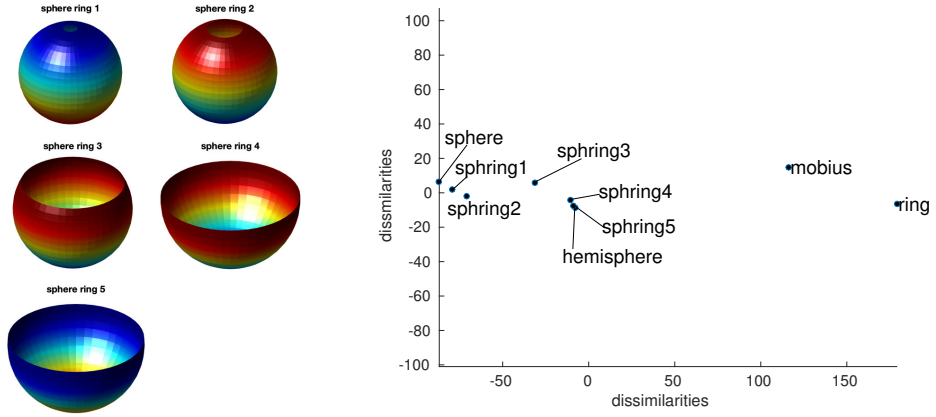

Fig. 11: Sphere Ring experiment. Left: example surfaces. Right: MDS plot. Sphere rings with small holes are clustered with the sphere. Sphere rings with big holes are clustered with the hemisphere. Möbius strip and 'ring' are clustered together.

6.3. 2D versus 3D multidimensional scaling. We demonstrate the potential effectiveness of a 3D plot. With higher dimensional plots, the optimized location of each surface in the corresponding space contains more information about Shape DNA, although it may be harder to visualize. Our hypothesis is that adding a third axis to the plots will capture additional characteristics of the surfaces as they can vary along more axes, with each axis representing a characteristic. In the 2D MDS plot of Figure 12 (left), the horizontal axis scale is much more significant. We observe that the closed surfaces (apple and sphere) are at the left and the open surfaces scatter to the right. However, the boundary conditions are not distinguished and it is unclear how the surfaces vary along the vertical axis. Distinguishing boundary conditions is crudely achieved by the 3D plot in Figure 12 (right) as each axis shows a unique type of characteristic that the surfaces vary upon. Note that in the 3D plot, "apple" and

[1] Open surfaces with Neumann boundary conditions and closed surfaces both have zero as their first eigenvalue in the Shape DNA. This is not true of open surfaces with fixed Dirichlet boundary conditions. The different boundary condition can be distinguished in MDS plots. How this affects clustering will be explored in the next section.

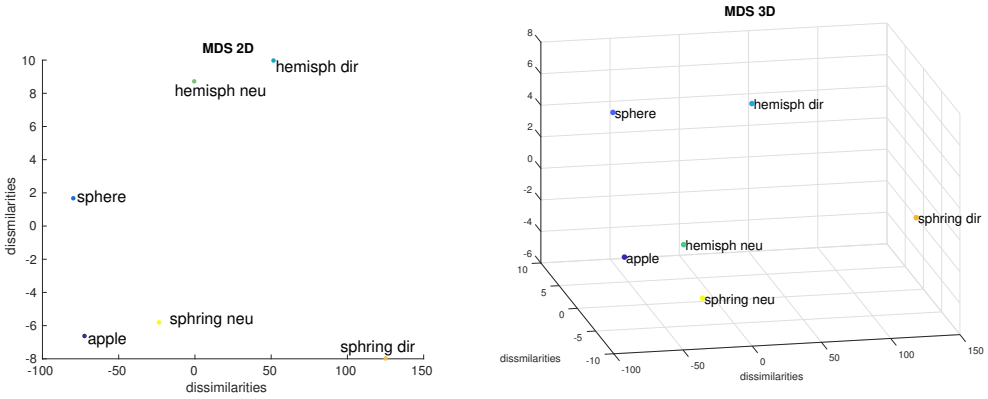

Fig. 12: 2D MDS plot (left) and 3D MDS plot (right). In the 2D plot, closed surfaces are on the left and open surfaces scatter to the right. The 2D MDS plot captures the openness of the surfaces along the horizontal axis. In the 3D plpt, on the z-axis, boundary conditions of open surfaces are roughly distinguished.

"sphere" are not applicable when distinguishing different boundary conditions along the z-axis.

The results in Figure 13 show how the MDS plot roughly captures three characteristics. The tested shapes are two sphere rings with Dirichlet and Neumann conditions, two hemispheres with Dirichlet and Neumann conditions, a sphere and an apple. These surfaces differ by their openness, boundary conditions and the general geometry.

6.4. Collection of Shapes. Finally, we compute the spectra of seemingly unrelated objects and try to uncover similarities using MDS. Figure 14 shows outlines of shapes of some triangulated animals. Figure 15 shows the 2D and 3D MDS plots for a collection of various shapes including some of the animals. The hypothesis given this collection of shapes was that the objects with boundaries (Möbius strip and sphere ring) would cluster as would round objects (torus, apple). This is indeed the case in Figure 15. The triangulated 'pigloop2' (a coarsely-trangulated pig that has been smoothed twice by the Loop triangle subdivision procedure) stands somewhat alone, in between the two clusters.

To investigate further, we add more triangulated objects (the original coarse pig– 'annie', 'pigloop1' smoothed by one Loop subdivision, and the reasonably-smooth Stanford bunny). We also add a sphere which we expect to cluster with the closed, round objects. The result in Figure 16 is as expected, except that the shape 'annies' is far from the other animal shapes. One speculation is that since this particular coarse triangulation has many sharp edges and corners, it cannot be properly resolved by the closest point method. Hence, it lies away from the rest of the other pig shapes, and further away from the round objects. The other reasonably-smooth triangulated animals lie in between.

7. Conclusions. In this work, we used the closest point method to classify shapes using their Laplace–Beltrami spectra. We reviewed how the closest point method is used to discretize the LB operator to obtain a matrix. We showed that the singularity of that matrix is related to the singularity of the LB operator for closed

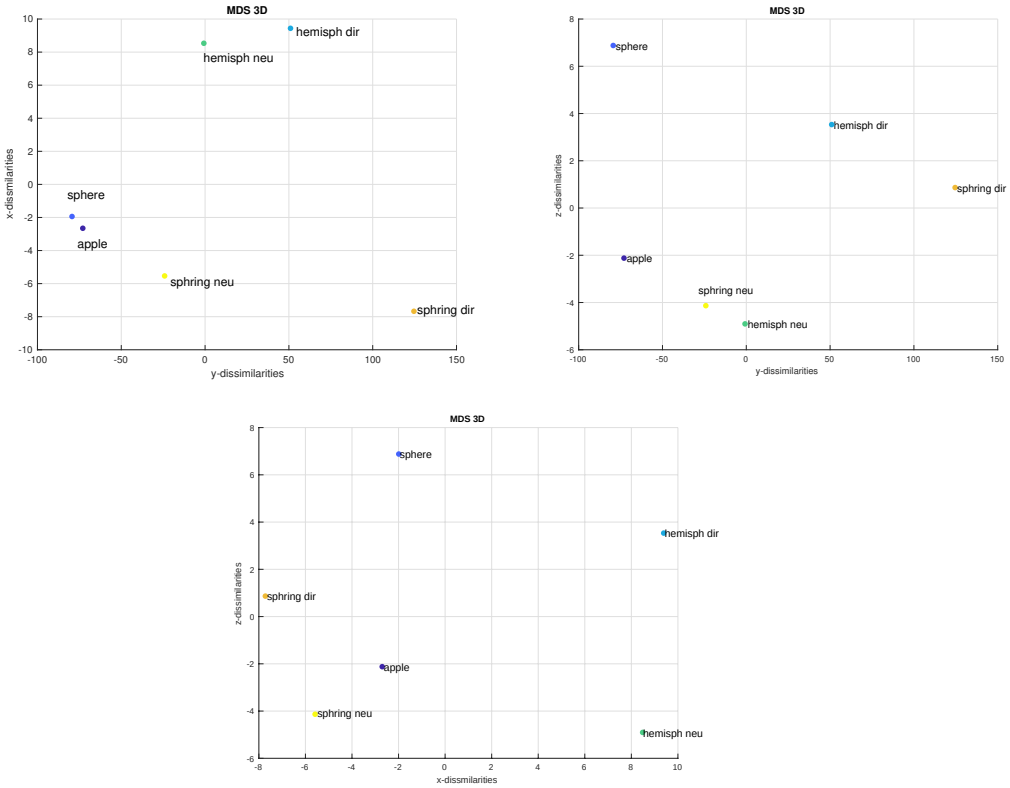

Fig. 13: In the top-left plot of y-x projection of the 3D plot, the x-axis (vertical) distinguishes the shapes. In the second y-z projection plot, the z-axis (vertical) distinguishes the Neumann or Dirichlet boundary conditions. In the third x-z projection plot, the y-axis (horizontal) distinguishes the open and closed surfaces.

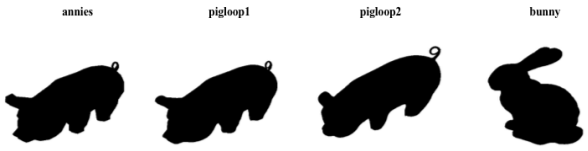

Fig. 14: Triangulated animal shapes

and open surfaces. In practice, we solve the LB eigenvalue problem using MATLAB `eigs`. We implemented an iterative solver within `eigs` to handle fine grid sizes in our computations, without running out of memory.

We performed various numerical experiments to demonstrate the accuracy of computing the Shape DNA using the closest point method. Then, we used a non-supervised clustering algorithm known as multi-dimensional scaling (MDS) to cluster shapes based on their Shape DNA. We observed that the resulting clusters are qualitatively reasonable. Using 3D MDS plots, we were able to extract three distinct

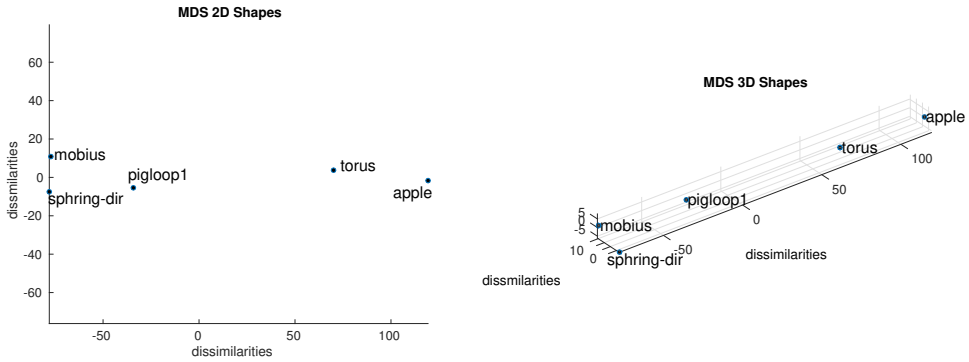

Fig. 15: Left: open surfaces with boundaries (Möbius, sphere ring) are in the left-most cluster and closed surfaces (torus, apple) are in the right-most cluster. The triangulated pig (closed surface) lies in the middle, closer to the left-most cluster. Right: we observe clusterings as in the 2D plot. The rounder surfaces are slightly lower on the z-axis.

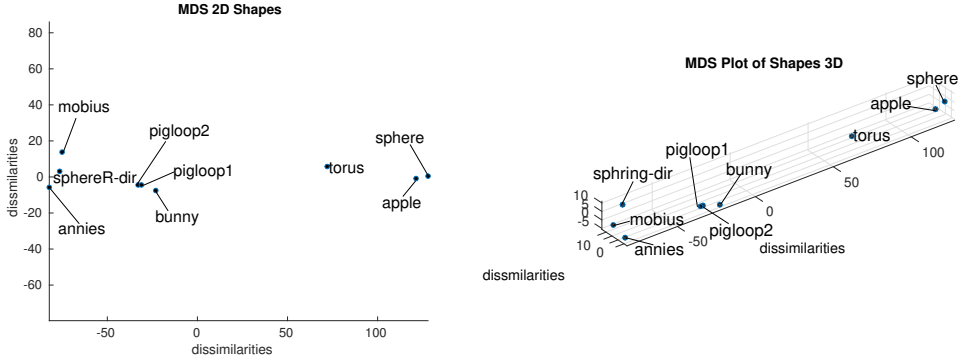

Fig. 16: Clustering a collection of shapes with additional triangulated animal shapes. The triangulated animals mostly lie in between two main clusters.

qualities of shapes that were chosen artificially in a set of input shapes.

Acknowledgements. This work was done through Undergraduate Summer Research Award from NSERC, under the supervision of Dr. Colin Macdonald and Dr. Steve Ruuth. We would like to thank Colin for his invaluable support and guidance on this project, and Dr. Steve Ruuth for motivating this work and teaching the closest point method during the PIMS-SFU Summer School. Also, many thanks to the UBC Math IT Department for the computational support.

Appendix A. Shape DNA of a unit circle. We show a simple analytic calculation of the Laplace–Beltrami spectrum on a circle. The Laplacian operator of a function v of two variables in polar coordinates is

$$v_{rr} + \frac{1}{r} v_r + \frac{1}{r^2} v_{\theta\theta}$$

If consider v as the extension of $u : S \to \mathbb{R}$ from the unit circle then v is constant in the r direction. On the unit circle, $r = 1$ and u is dependent on θ only. So we find the Laplace–Beltrami operator for the unit circle is $u_{\theta\theta}$. The eigenvalue problem is thus

$$u'' + \lambda u = 0.$$

For a circle, we impose a periodic boundary condition $u(0) = u(2\pi)$. Then,

$$u(\theta) = A\cos(\sqrt{\lambda_n}\theta),\ B\sin(\sqrt{\lambda_n}\theta), \quad \lambda_n = n^2,\ n = 0, 1, 2, ... \tag{9}$$

The Laplace–Beltrami spectrum of a circle is has multiplicity two: $0, 1, 1, 4, 4, 16, 16, ...$ These are also the first few entries of the Shape DNA. They are consistent with our numerical results.

REFERENCES

[1] R. J. Arteaga and S. J. Ruuth, *Laplace–Beltrami spectra for shape comparison of surfaces in 3D using the closest point method*, 2015 IEEE International Conference on Image Processing (ICIP), (2015), pp. 4511–4515, https://doi.org/10.1109/ICIP.2015.7351660.

[2] Y. Canzani, *Analysis on manifolds via the Laplacian*, Lecture Notes available at: http://www.math.harvard.edu/canzani/docs/Laplacian.pdf, (2013).

[3] Y. Chen and C. Macdonald, *The closest point method and multigrid solvers for elliptic equations on surfaces*, SIAM J. Sci. Comput., 37 (2015), pp. A134–A155, https://doi.org/10.1137/130929497.

[4] C. Gordon, D. Webb, and S. Wolpert, *Isospectral plane domains and surfaces via Riemannian orbifolds.*, Inventiones Mathematicae, 110 (1992), p. 1, https://doi.org/10.1007/BF01231320.

[5] M. Kac, *Can one hear the shape of a drum?*, American Mathematical Monthly, 73 (1996).

[6] R. J. LeVeque, *Finite Difference Methods for Ordinary and Partial Differential Equations*, SIAM, Philadelphia, PA, 2007.

[7] C. B. Macdonald, J. Brandman, and S. J. Ruuth, *Solving eigenvalue problems on curved surfaces using the closest point method*, J. Comput. Phys., 230 (2011).

[8] T. März and C. B. Macdonald, *Calculus on surfaces with general closest point functions*, SIAM J. Numer. Anal., 50 (2012), pp. 3303–3328, https://doi.org/10.1137/120865537, https://arxiv.org/abs/1202.3001.

[9] M. Reuter, S. Biasotti, G. P. D. Giorgi, and M. Spagnuol, *Discrete Laplace–Beltrami operators for shape analysis and segmentation*, Computers & Graphics, 33 (2009).

[10] M. Reuter, F.-E. Wolter, and N. Peinecke, *Laplace–Beltrami spectra as "shape-DNA" of surfaces and solids*, Computer-Aided Design, 38 (2006).

[11] S. Roweis and L. Saul, *Nonlinear Dimensionality Reduction by Locally Linear Embedding*, Science, 290 (2000), pp. 2323–2326.

[12] S. J. Ruuth and B. Merriman, *A simple embedding method for solving partial differential equations on surfaces*, J. Comput. Phys., 227 (2008).

[13] S. Seo, M. Chung, and H. Vorperian, *Heat kernel smoothing using Laplace–Beltrami eigenfunctions*, Medical Image Computing and Computer-Assisted Intervention, 230 (2010).

[14] E. Süli and D. Mayers, *An introduction to numerical analysis*, Cambridge, UK, (2003).

[15] L. N. Trefethen and D. Bau III, *Numerical linear algebra*, vol. 50, Siam, 1997.

[16] I. von Glehn, T. März, and C. B. Macdonald, *An embedded method-of-lines approach to solving partial differential equations on surfaces*, ArXiv e-prints, (2013), https://arxiv.org/abs/1307.5657.