LAPLACE-BELTRAMI SPECTRA AS 'SHAPE-DNA' OF SURFACES USING THE CLOSEST POINT METHOD

by

Reynaldo Jose Arteaga

B.Math., Carleton University, 2012

a Thesis submitted in partial fulfillment
of the requirements for the degree of

Master of Science

in the
Department of Mathematics
Faculty of Science

SIMON FRASER UNIVERSITY
Summer 2014

APPROVAL

Name: Reynaldo Jose Arteaga

Degree: Master of Science

Title of Thesis: Laplace-Beltrami spectra as 'Shape-DNA' of surfaces using the closest point method

Examining Committee: Dr. Paul Tupper, Associate Professor, Mathematics
Chair

Dr. Steven Ruuth, Professor, Mathematics
Senior Supervisor

Dr. Nilima Nigam, Professor, Mathematics
Supervisor

Dr. Manfred Trummer, Professor, Mathematics
Internal/External Examiner

Date Approved: July 28, 2014

Partial Copyright Licence

Abstract

The need to compare two separate manifolds arises in a wide range of applications. In this thesis, 'Shape-DNA', i.e. the eigenvalues of the Laplace-Beltrami operator, are used to create a numerical signature representing an individual object. The corresponding spectrum is isometry invariant, which means it is independent of manifold representations such as parameterization or spatial positioning. Therefore, checking if two objects are isometry invariants does not require any alignment (registration/localization) of the objects but only comparing their spectra. We determine the Shape-DNA using the closest point method on the manifold. In 3D we illustrate the process for triangulated mesh and point cloud surfaces. Convergence studies demonstrate that the convergence rates correspond to those of the underlying finite difference methods. A 2D multidimensional scaling plot illustrates how identical objects are mapped to the same spot and similar objects form groups based on the Shape-DNA's.
Keywords: Laplace-Beltrami; closest point method; point cloud; spectra

To my loving family.

"But that's another story and shall be told another time."

— *The Neverending Story*, Michael Ende, *1979*

Acknowledgments

My deepest gratitude goes towards Dr. Steven Ruuth who was my mentor throughout the production of my thesis. His guidance and support was far beyond ideal and it has been a pleasure working along his side. I also like to thank my family for their support and always believing in their son/brother. To my friends, professors and colleagues, who have given me support in every part of this experience, I greatly appreciate your kindness, efforts and motivation. I am very thankful for my wonderful girlfriend Stephanie Di Perna who has helped me throughout every step of this journey. Lastly, thank you Lord Jesus Christ for holding me up when I was in need and being by my side in all my defeats and victories.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Determining how two objects are similar to or different from each other is a standard problem in both mathematics and computational science. Instances of such applications arise in biological systems, biometric technology, medical imaging and 3D scanning. For example, biological systems study how similar animals remain when they undergo a physical transformation such as inflation [2]. Biometric technology can be used to secure financial transactions [8] by scanning a humans fingerprints and checking the database for a match. As well, historical identification of artwork can be accomplished with 3D scanning (such as the digital Michelangelo project [18]) and find similarities to other artworks belonging to the same time period. With current technologies and the vast pool of applications, there is high demand for software that can systematically examine a database to determine similarly shaped objects (i.e. shape matching) efficiently.

In order to implement shape matching, we first require a method to characterize an object. "Visual recognition of objects requires an active process of mapping visual sensations onto stored mental representations", Parsons and Fox explain in [26]. We now discuss techniques for automating this innate human ability, which then can be used for shape matching.

A classical approach to identifying shapes is through the aid of watermarks [3], although there remain some disadvantages of this technique. Watermarks are information embedded into the representation or geometry of an object and may be either visible or invisible. Watermark data is commonly embedded into polygonal meshes by slightly modifying the vertex location, the connectivity of the mesh or the frequency domain involving mesh-spectral analysis. In most cases, anytime we manipulate the representation or reparametrize an object,

watermarks can be destroyed if they are not embedded in the geometry. However, embedding watermarks into the geometry changes the object, possibly rendering it inadequate for shape matching.

In the past fifty years, several methods have been proposed to solve the problem of shape matching (e.g. using moments, spherical harmonics or Reeb graphs—a survey can be found in Iyer et al. [15]). A majority of these approaches realign the geometric objects in a process which is commonly known as localization or registration [27]. These shape matching algorithms use a specific representation of the object (usually polygonal meshes). Additional techniques decompose the representation of the objects into smaller attributes [6], which are compared in a subsequent step. All these techniques can solve the shape matching problem, however there is a more desirable class of shape identifiers.

A possible alternative to localization and representation decomposition is to identify shapes by a *signature* or *fingerprint*. Fingerprints identify the shape of an object by geometric invariants. These fingerprints do not discriminate between different representations of a single object. Fingerprints leave the objects in their original forms, unlike the watermark approach, as previously mentioned. We adopt Reuter et al. [29] choice to have a fingerprint consist of a vector of numbers/shape invariants associated with the given object. Reuter et al. [29] propose that the ideal candidate will satisfy these properties:

1. [ISOMETRY]: Congruent solids (or isometric surfaces) should have the same fingerprint independently of the object's given representation and location. Therefore, the fingerprint should be an isometry invariant.

2. [SCALING]: For some applications, it is necessary that the fingerprint is independent of the object's size, therefore the fingerprint should be scaling invariant.

3. [SIMILARITY]: Similarly shaped solids should have similar fingerprints. The fingerprint should depend continuously on shape deformations.

4. [EFFICIENCY]: The effort needed to compute fingerprints should be reasonable.

5. [COMPLETENESS]: Fingerprints should give a complete characterization of the shape, thus representing the shape uniquely. Moreover, it would be desirable that the fingerprints could be used to reconstruct the original solid.

6. [COMPRESSION]: The fingerprint data should not be redundant, i.e. a part of it could not be computed from the rest of the data.

7. [PHYSICALITY]: Finally, it would be beneficial if an intuitive geometric or physical interpretation of the meaning of the fingerprints is available.

All together, these properties would make the fingerprint an excellent choice for shape matching as its target application.

An essential property that requires formalism is that of isometry.

Definition 1. *(Isometry)*
Two geometric objects are isometric if a homeomorphism[1] *from one to the other exists preserving (geodesic) distances, i.e. mapping curves to curves with equal arc length. This homeomorphism is then called an isometry.*

Surfaces remain isometric even after bending the original surface, despite not being congruent (assuming that there is no stretching or change to the metric) . When matching a given object A that is a transformation of another object B, the fingerprint will be similar for objects A and B. When comparing objects, the [ISOMETRY] property is arguably the most important feature.

Reuter et al. [29] have recognized that the eigenvalues of the Laplace-Beltrami operator adheres to most of these ideal properties (1. [ISOMETRY] - 7. [PHYSICALITY]). In order to understand the Laplace-Beltrami operator on general surfaces, we first consider the operator on a smooth and well studied manifold, that is, on a sphere. Afterwards, we will detail properties of the eigenvalues.

1.1 Laplace-Beltrami Operator

Let $\mathbf{e}_r, \mathbf{e}_\theta, \mathbf{e}_\phi$ be the standard unit vectors in spherical coordinates. The surface gradient in spherical coordinates is then defined as

$$\nabla_{\mathcal{S}} u(x) = \frac{\partial u}{\partial r}\mathbf{e}_r + \frac{1}{r\sin\theta}\frac{\partial u}{\partial \phi}\mathbf{e}_\phi + \frac{1}{r}\frac{\partial u}{\partial \theta}\mathbf{e}_\theta, \tag{1.1}$$

[1] A function $f : X \to Y$ between topological spaces is called a homeomorphism if it satisfies:

1. f is a bijection,
2. f is continuous,
3. the inverse function f^{-1} is continuous.

where θ is the angle with the z-axis, ϕ the rotation around the z-axis (commonly known as the zenith and azimuth angles respectively), $\mathcal{S}$ is a smooth closed surface, and $u(\cdot)$ is a $\mathbb{C}^1$ smooth scalar function, $u : \mathcal{S} \to \mathbb{R}$. The surface divergence in spherical coordinates is given by

$$\nabla_{\mathcal{S}} \cdot \mathbf{U}(x) = \frac{1}{r^2}\frac{\partial}{\partial r}(r^2 U_r) + \frac{1}{r\sin\theta}\frac{\partial}{\partial\theta}(\sin\theta U_\theta) + \frac{1}{r\sin\theta}\frac{\partial U_\phi}{\partial\phi}. \tag{1.2}$$

The Laplace-Beltrami operator, named after Pierre-Simon Laplace and Eugenio Beltrami, has the following definition:

$$\Delta_{\mathcal{S}} u(x) = \nabla_{\mathcal{S}} \cdot \nabla_{\mathcal{S}} u(x) \tag{1.3}$$

on a smooth closed surface $\mathcal{S}$. The Laplace-Beltrami operator is a generalization of the Laplacian since it can also operate on functions defined on surfaces in Euclidean space and, more generally, on Riemannian manifolds.

The problem that is central to this thesis is the following:

Problem 1. *(Laplace-Beltrami eigenvalue problem)*
Given a surface $\mathcal{S}$, determine the eigenfunctions $u : \mathcal{S} \to \mathbb{R}$ and eigenvalues λ^2 satisfying

$$\Delta_{\mathcal{S}} u(\boldsymbol{x}) = -\lambda^2 u(\boldsymbol{x}), \qquad \textit{for } \boldsymbol{x} \in \mathcal{S}. \tag{1.4}$$

Equation (1.4) does not have an analytic solution for a general manifold, hence the majority of our efforts will be concluded with numerical results. Before we investigate the general properties of the Laplace-Beltrami operator, we start off with an analytic result for the sphere to give some insight.

1.2 Eigenvalues of a Sphere

In order to determine the eigenvalues of the sphere, we can look at Helmholtz's equation in spherical coordinates. This approach follows common mathematical techniques and more importantly, avoids the difficulty of interpreting the surface Laplacian.

The Helmholtz equation was studied for various basic shapes in the 19th century. In 1829, Siméon Denis Poisson solved it for the rectangular membrane. In 1852 Gabriel Lamé solved it for the equilateral triangle. Alfred Clebsch solved it in 1862 for the circular membrane. The boundary-value problem for the Helmholtz equation is given by

$$\begin{cases} \Delta u(\mathbf{x}) + \lambda^2 u(\mathbf{x}) = 0, & \text{in } \Omega \\ \dfrac{\partial u}{\partial n} = 0, & \text{on } \partial\Omega. \end{cases} \tag{1.5}$$

where Ω is an open, bounded subset of $\mathbb{R}^n$, $u : \bar{\Omega} \to \mathbb{R}$ is the eigenfunction and $\lambda^2 \in \mathbb{R}$ is the corresponding eigenvalue. The case where $\lambda = 0$ becomes Laplace's equation. On $\mathcal{S}$, the Helmholtz equation can be derived by applying separation of variables to the wave equation (See §A.1 for details). Thus labeling λ, the wavenumber, which are spatial frequencies of a wave. This can give some intuition for the Helmholtz equation.

Considering the Helmholtz equation restricted to the unit sphere, we transform the coordinate system from the Euclidean space to spherical coordinates. Starting from a sphere centered at the origin, radius r, and $(x, y, z) \in \mathbb{R}^3$ we have:

$$\begin{aligned} x &= r\cos\theta\sin\phi, \\ y &= r\sin\theta\sin\phi, \\ z &= r\cos\phi, \end{aligned}$$

where $0 \leq \theta \leq 2\pi$ and $0 \leq \phi \leq \pi$. The Laplacian operator in spherical coordinates, Equation (1.3), is explicitly written as

$$\Delta = \frac{1}{r^2}\frac{\partial}{\partial r}\left(r^2\frac{\partial}{\partial r}\right) + \frac{1}{r^2\sin\theta}\frac{\partial}{\partial\theta}\left(\sin\theta\frac{\partial}{\partial\theta}\right) + \frac{1}{r^2\sin^2\theta}\frac{\partial^2}{\partial\phi^2}. \tag{1.6}$$

In order to solve Equation (1.5) we apply the classical approach of separation of variables. Let

$$u(\theta, \phi) = \Theta(\theta)\Phi(\phi) \tag{1.7}$$

with no dependence on r, since it is constant for all points on the sphere.

The Helmholtz equation in spherical coordinates using separation of variables gives:

$$\begin{aligned} \frac{1}{\sin\theta}\frac{\partial}{\partial\theta}\left(\sin\theta\Theta'\right)\Phi + \frac{1}{\sin^2\theta}\Phi''\Theta + \lambda^2\Phi\Theta &= 0, \\ \frac{\sin\theta}{\Theta}\frac{\partial}{\partial\theta}\left(\sin\theta\Theta'\right) + \lambda^2\sin^2\theta &= -\frac{\Phi''}{\Phi}. \end{aligned} \tag{1.8}$$

The left hand side of Equation (1.8) does not depend on the same variables as the right hand side. We therefore have the constant coefficient ordinary differential equation,

$$\frac{\Phi''}{\Phi} = -m^2, \tag{1.9}$$

for some integer m (in order that the solution is the same for ϕ and $\phi + 2\pi$). The solution to Equation (1.9) is of the form

$$\Phi(\phi) = e^{im\phi} = \sin(m\phi) + i\cos(m\phi). \tag{1.10}$$

Substituting Equation (1.9) into Equation (1.8) we have

$$\frac{\sin\theta}{\Theta}\frac{\partial}{\partial\theta}\left(\sin\theta\Theta'\right)+\lambda^2\sin^2\theta=m^2. \tag{1.11}$$

Simplifying the problem by azimuthal symmetry, where $m=0$, Equation (1.11) becomes

$$\frac{1}{\sin\theta\Theta}\frac{\partial}{\partial\theta}\left(\sin\theta\Theta'\right)=-\lambda^2. \tag{1.12}$$

Substituting $u=\cos\theta$, the associated Legendre equation is

$$\frac{d}{du}\left((1-u^2)\frac{d\Theta}{du}\right)+\lambda^2\Theta(u)=0. \tag{1.13}$$

To have a solution to Equation (1.13) it is necessary that u is finite and $\lambda^2=l(l+1), l\in\mathbb{N}^0$. The eigenvalues for the unit sphere are therefore $\lambda^2=l(l+1)=\{0,2,6,12,\ldots\}$, with multiplicities $2l+1$.

1.3 Properties of the Spectrum

After seeing the spectrum for the sphere, we are interested in the general properties of the Laplace-Beltrami operator. We will highlight important qualities of the spectrum demonstrating the connection to the ideal properties (1.[ISOMETRY] - 7. [PHYSICALITY]).

The spectrum is isometric invariant as it only depends on the gradient and divergence which in turn are defined to be dependent only on the Riemannian structure of the manifold [29]. In order to demonstrate this dependence, we first introduce some notation from Reuter et al. [29]. Let f and g be real-valued functions belonging to $\mathbb{C}^2$ defined on a Riemannian manifold $\mathcal{M}$. Given a local parametrization $\psi:\mathbb{R}^n\to\mathbb{R}^{n+k}$ of a submanifold $\mathcal{M}$ of $\mathbb{R}^{n+k}$ with

$$g_{ij}:=\partial_i\psi\cdot\partial_j\psi,\quad G:=(g_{ij}),\quad W:=\sqrt{\det G},\quad (g^{ij}):=G^{-1}$$

(where $i,j=1,\cdots,n$ and det denotes the determinant), we obtain

$$\begin{aligned}\nabla f\cdot\nabla g &= \sum_{i,j}g^{ij}\partial_i f\partial_j g \quad \text{and}\\ \Delta_{\mathcal{S}}f &= \frac{1}{W}\sum_{i,j}\partial_i(g^{ij}W\partial_j f).\end{aligned} \tag{1.14}$$

This underlying mathematical notion allows the [ISOMETRY] property to be fulfilled.

Moreover, if we scale a manifold by a factor a, then the eigenvalues become scaled by the factor $\frac{1}{a^2}$ (See §A.2 for details). This allows us to satisfy the [SCALING] property by multiplying by the appropriate constant.

We observe that the spectrum consists of a diverging sequence, $0 \leq \lambda_1 \leq \lambda_2 \leq \cdots$, with no accumulation (limit) points except at infinity. Eigenvalues can have multiplicity greater than 1. Reuter et al. [29] discuss the mutual independence of eigenvalues and the impossibility of finite characterization. Specifically they demonstrate that an arbitrary eigenvalue cannot be computed from a finite number of other eigenvalues for a Riemannian manifold.

Assertion 1. *(Mutual independence of eigenvalues)*
An arbitrary eigenvalue λ_k of a compact Riemannian manifold $(\mathcal{M}, g)$ cannot be computed from a finite number of other eigenvalues of $(\mathcal{M}, g)$ in general (i.e. if the manifold is unknown).

They also extend Assertion 1 to demonstrate that it is not only impossible to determine one arbitrary eigenvalue, but it is also impossible to determine the whole spectrum from a finite subsequence for an unknown manifold [29]. By "unknown", we mean that there is no prior knowledge on what the manifold is, since we can determine the entire spectrum for certain restricted classes of surfaces (i.e. if we know the object is a sphere, then from the first few eigenvalues we can determine the entire spectrum).

Corollary 1. *(Impossibility of finite characterization)*
No subsequence S of a spectrum, Spec$(\mathcal{M})$, of any unknown compact Riemannian manifold $\mathcal{M}$ determines the whole spectrum.

Assertion 1 and Corollary 1 are of interest in relation to the [COMPRESSION] property. The eigenvalue spectrum cannot be compressed into a finite subsequence without loss of information. Moreover, without any prior knowledge about the manifold, the first m-eigenvalues cannot be constructed from the first n-eigenvalues where $n < m$. The [COMPRESSION] property alongside Assertion 1 and Corollary 1 prevent any possible attempt at reconstructing the spectrum with only a subsequence and without knowing the manifold.

The remaining properties further support the choice of basing the fingerprints on the eigenvalues of the Laplace-Beltrami operator. Reuter et al. [29] demonstrate that a significant amount of information is stored in the fingerprint fulfilling [PHYSICALITY]. The

[EFFICIENCY] property holds for the finite element implementation taken by Reuter et al. [29] and the proposed method of this thesis (as we will see). Both involve solving an eigenvalue problem, which can be accomplished in a reasonable amount of computational time. Reuter et al. [29] detail that the [COMPLETENESS] property cannot be upheld due to non-isometric manifolds having the same spectrum. Lastly, [SIMILARITY] is satisfied due to the spectrum of the Laplace-Beltrami operator depending continuously on the manifold.

Together with all these properties, the fingerprint approach is an excellent candidate. The calculation of the fingerprint does not manipulate the structure of the manifold which allows it to be used in shape matching. As we investigate further in this thesis, we shall see the successful implementation for shape matching.

The thesis unfolds as follows. Chapter 2 gives a detailed explanation of the eigenvalues of the Laplace-Beltrami operator that we call 'Shape-DNA' and the shape matching algorithm. Chapter 3 illustrates the closest point method and exhibits the different components through a concrete example. Chapter 4 verifies the method with numerical results for triangulated surfaces, point clouds and shape matching application. The closest point method has never been used for shape-matching and point cloud surfaces until this study. Lastly, Chapter 5 summarizes the Shape-DNA algorithm and its findings.

Chapter 2

Shape-DNA

This chapter begins with a formal definition of Shape-DNA. Following this, we detail the overall shape matching process called the 'shape matching algorithm'. We explain each step in the shape matching algorithm, which is detailed in Reuter et al. [29].

2.1 Shape-DNA

Shape-DNA is extracted from the Laplace-Beltrami operator which encapsulates the geometry of the manifold since it is an isometry invariant.

Definition 2. *(Shape-DNA)*
Let $\mathcal{M}$ be a Riemannian manifold with a metric h. The cropped spectrum containing only the first n eigenvalues

$$cspec_n(\mathcal{M}, h) = \{\lambda_0 \leq \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_{n-1}\} \in \mathbb{R}^n_{\geq 0} \tag{2.1}$$

is called the Shape-DNA of $(\mathcal{M}, h)$.

Shape-DNA is a central concept in shape matching. See Reuter et al. [29] for a variety of other interesting applications.

2.2 Shape Matching Algorithm

Humans are able to categorize different objects by geometric similarities, and this is precisely what we try to automate. Shape-DNA quantifies this innate ability to categorize various

objects. In order to conduct shape comparison between several objects, we have the following major steps:

Definition 3. *(Shape Matching Algorithm)*

1. *Solve for the Shape-DNA of the Laplace-Beltrami for given manifolds.*
2. *Normalize the Shape-DNA of each manifold.*
3. *Conduct a MultiDimensional Scaling (MDS) plot with the Euclidean distances between the Shape-DNA from step 2.*

How one decides to solve the Laplace-Beltrami eigenvalue problem is entirely user-dependent. Reuter et al. [29] have chosen to implement the finite element method. The normalization step is strictly for the [SCALING] property, in order to scale objects of different magnitudes. Since each manifold is represented by its own Shape-DNA, multidimensional scaling is a classical approach to illustrate the distance between each and every Shape-DNA. We will now give a more in depth look at each step in the shape matching algorithm.

2.2.1 Finite element method

Firstly we must find (a finite number) of the eigenvalues of the Laplace-Beltrami operator, i.e. the Shape-DNA. We detail the finite element method for the Laplace-Beltrami eigenvalue problem, then proceed to the numerical discretization.

Variational problem

Recall that the Helmholtz equation is defined as

$$\Delta u(\mathbf{x}) + \lambda^2 u(\mathbf{x}) = 0, \tag{2.2}$$

where $u \in \mathbb{R}^n$ is the eigenfunction and $\lambda^2 \in \mathbb{R}$ corresponds to the eigenvalue. Rather than working with Equation (2.2), we wish to work with the variational problem or weak formulation.

We can reach the weak formulation of Helmholtz's equation restricted to some given smooth surface by applying Green's formula,

$$\int\int \phi \Delta u W d\mathcal{S} = -\int\int \nabla u \cdot \nabla \phi \, W d\mathcal{S}, \tag{2.3}$$

with test functions $\phi \in \mathbb{C}^2$, Neumann ($\frac{\partial u}{\partial n} = 0$) boundary condition and the Nabla operator under the notation given in §1.3

$$\nabla u \cdot \nabla \phi = \sum_{i,j} g^{ij} \partial_i u \, \partial_j \phi. \tag{2.4}$$

We multiply the Helmholtz equation with test functions. By integrating over the surface and using Green's formula, Equation (2.3) becomes

$$\begin{aligned} \int\int \phi \Delta u W d\mathcal{S} &= -\lambda^2 \int\int \phi u W d\mathcal{S}, \\ \iff \int\int \sum_{i,j} g^{ij} \partial_i u \partial_j \phi W d\mathcal{S} &= \lambda^2 \int\int \phi u W d\mathcal{S}. \end{aligned} \tag{2.5}$$

Every function u ($u \in \mathbb{C}^2$ on the boundary) that solves the weak formulation for all test functions ϕ, is a solution to the Helmholtz equation [7, pg. 35].

Discretization

We require a numerical discretization of the variational problem in order to obtain the eigenvalues (since an explicit solution to the problem posed on a general surface does not exist). Reuter et al. implement a Galerkin technique as follows (See e.g. Strang [34]):

- Choose n linearly independent form functions: $F_1(\mathbf{x}), \cdots, F_n(\mathbf{x})$ defined on the parameter space (i.e. the discretization/tessellation of the domain).
- Secondly, use a linear combination of these functions to form a basis of a vector space to approximate the solution:
  $u(\mathbf{x}) \approx F(\mathbf{x}) := U_1 F_1(\mathbf{x}) + \cdots + U_n F_n(\mathbf{x})$.
- Lastly, calculate the n unknown coefficients $U_i \in \mathbb{R}$ by substituting u in the variational equation and selecting n different test functions ϕ_i to obtain n equations. They are n-form functions chosen for the test functions to preserve symmetry in the linear system.

After a discretization/tessellation of the domain is constructed n-form functions are created. These form functions represent the basis of the solution space. This solution space is composed of piecewise polynomial functions on each finite element, which are called "form functions". Reuter et al. [29] have used linear, quadratic and cubic polynomials defined on triangular elements in the parameter space of the surface.

By looking at the discretized surface (i.e. finite dimensional problem) and with Equation (2.5), the two symmetric matrices are

$$\begin{aligned} A &= (a_{lm}) := \left(\int\int \left(\sum_{j,k} (\partial_j F_l)(\partial_k F_m) g^{jk} d\sigma \right) \right) \\ B &= (b_{lm}) := \left(\int\int F_l F_m d\sigma \right) \end{aligned} \tag{2.6}$$

and the variational problem then becomes a general eigenvalue problem:

$$AU = \lambda^2 BU. \tag{2.7}$$

We have that $U = (U_1 \cdots U_n)$ and A, B are sparse positive semi-definite symmetric matrices since all eigenvalues are greater than or equal to zero.

2.2.2 Normalization

There are different possible choices for Step (2) of the Shape-DNA Algorithm introduced by Reuter et al. [29]. The Shape-DNA of a d-dimensional Riemannian manifold can be:

1. divided by the factor c of the fitting curve,

$$f(x) = cx^{2/d} \tag{2.8}$$

   fitting $f(n) := \lambda_n$ (cf. Weyl's law in Appendix A.3).

2. multiplied by $V^{2/d}$, where V is the real Riemannian volume of the manifold extracted by extrapolation from the spectrum (for $d = 2$ this is simply a multiplication by the surface area).

3. multiplied by $V^{2/d}$, where V is the real Riemannian volume that has been calculated externally via a pre-process.

We note that for the case $d = 2$, the slope of the fitting line gives a rough approximation to $4\pi/A$ where A is the area [29].

We will choose the first and most general normalization that does not require any knowledge of the manifold other than the eigenvalues we compute. Although Weyl's law is for large n, we obtain accurate results for $n \approx 50$ for shape matching in §4.2.4. Software automation is possible by selecting the first normalization factor. Although the second and

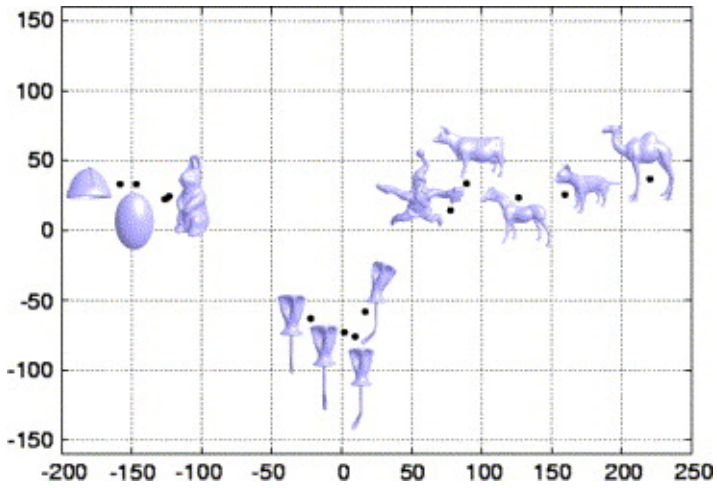

Figure 2.1: Example of 2D-MDS plot for shape matching on a set of objects. Reprinted with permission of the *Journal of Computer-Aided Design* [[29], Figure 19].

third options would be more accurate, we cannot always accurately estimate the area of a general manifold. MATLAB's [23] built in function, *nlinfit* which applies a least squares estimation is used to find the coefficient c.

2.2.3 Multidimensional scaling

In order to give a visual interpretation of a given set of objects describing the distances between each Shape-DNA, we will use multidimensional scaling. The 2D-MDS plot can be understood as an orthogonal projection of the n-dimensional vectors onto their best 2D-fitting plane. It illustrates how identical manifolds map to the same spot on the figure, and similar objects form groups (See Figure 2.1 and Figure 2.2 for some results from Reuter et al. [29, 28]).

We briefly mention the setup to achieve the 2D-MDS plot. Given N-objects, we compute the dissimilarity matrix where $\delta_{i,j}$:= distance between the i^{th} and j^{th} objects. Find N vectors, $x_1, \cdots x_N \in \mathbb{R}^d$ such that $\|x_i - x_j\|_2 \approx \delta_{i,j} \quad \forall i, j \in N$. T. Cox and M. Cox [11] describe this process by solving an equivalent eigenvalue problem in the following steps:

1. Obtain dissimilarities $\{\delta_{rs}\}$.
2. Find matrix $\mathbf{A} = \left[-\frac{1}{2}\delta_{rs}^2\right]$.

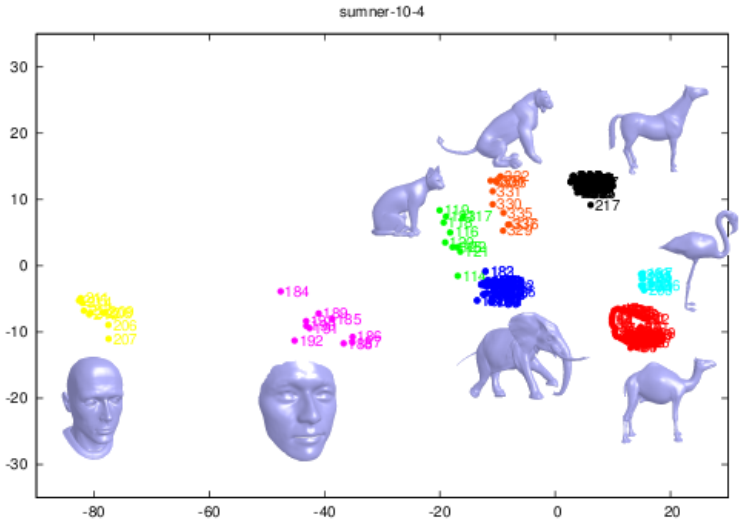

Figure 2.2: Example of 2D-MDS plot for shape matching on a set of objects. Reprinted with permission of *M. Reuter* [28].

3. Find matrix $\mathbf{B} = [a_{rs} - a_{r.} - a_{.s} + a_{..}]$,
   where $[\mathbf{A}]_{rs} = a_{rs}$, $a_{r.} = n^{-1}\sum_s a_{rs}$, $a_{.s} = n^{-1}\sum_r a_{rs}$, $a_{..} = n^{-2}\sum_r\sum_s a_{rs}$.

4. Find the eigenvalues $\lambda_1, \cdots, \lambda_{n-1}$ and associated eigenvectors $\mathbf{v}_1, \cdots, \mathbf{v}_{n-1}$, where the eigenvectors are normalized so that $\mathbf{v}_i^T\mathbf{v}_i = \lambda_i$.

5. Choose the number of dimensions p. This thesis considers $p = 2$ for all cases.

6. The coordinates of the n points in the p dimensional Euclidean space are given by $x_{ri} = v_{ir}$ $(r = 1, \cdots, n;\ \ i = 1, \cdots, p)$.

In this thesis, the vectors are represented by the Shape-DNA of N-objects. There are built-in functions for determining the solution to the multidimensional scaling problem given a dissimilarity matrix. We obtain accurate and reliable results by the function *cmdscale* in MATLAB [23].

The eigenvalues of the Laplace-Beltrami operator, i.e. Shape-DNA is the fingerprint that will identify objects. The shape matching algorithm has three components. Firstly, the finite element method was used to solve for the Shape-DNA of the objects. Then a normalization factor is taken into account based on Weyl's Asymptotic growth of eigenvalues. Finally, multidimensional scaling is how we visualize our shape matching results as a natural choice.

The shape matching algorithm has successful results using the finite element method. However this process relies on a triangulated surface. The closest point method can operate directly on both triangulated surfaces and point clouds. As we will see in the following chapter, by solving the Shape-DNA with the closest point method instead of the finite element method, this leads to improvements in several aspects.

Chapter 3

The Closest Point Method

A major consideration for determining the Shape-DNA of an object, is how to evaluate the surface differential operator. We briefly illustrate the reasoning behind selecting the closest point method (CPM) as the preferred candidate. We illustrate the CPM through a case study. Finally, we review the Implicit Closest Point Method (ICPM) for the Laplace-Beltrami operator.

3.1 Motivation

A standard approach for solving PDEs on surfaces is to create a parametrization on the surface, express the differential operators within this new set of coordinates and then perform a discretization. Distortions in either angles or other regions always occur from parametrizations, as discussed in [12]. Surface parametrization techniques are difficult to implement for general geometries and hence we consider other strategies.

Another class of methods embed the surface differential equations within differential equations posed on all $\mathbb{R}^3$. By restricting the corresponding embedding equations to only the surface, we obtain the original problem at hand. This approach accounts for general surface geometries while maintaining the simplicity of working in standard Cartesian grids.

One proposal takes a level-set representation of the underlying surface [5]. Although effective for a class of problems, embedding methods based on level-set methods have a list of restrictions. Notably, these methods do not naturally allow for open surfaces with boundaries although modifications involving additional level-set functions are possible [13]. A second substantial issue arises when restricting the problem from all space to a narrow

band width around the surface. This step requires the introduction of artificial boundary conditions, which are not well studied and in practice limit the order of accuracy to first order.

Other embedding methods may be considered. Notably, one may use a closest point representation of the surface instead of a level set representation. A closest point representation allows for open surfaces and does not introduce artificial boundary conditions, as we shall see in further discussion. We now detail the ideas underlying the method.

3.2 Closest Point Representation

The closest point method (CPM) is an embedding method. It replaces the surface PDE with a related PDE in the embedding space. This allows us to numerically solve the PDE using finite differences, finite element or other standard approaches. Applying the closest point method with finite differences is shown to be straightforward and easily implemented with uniform grids.

An important component of the CPM is the closest point function.

Definition 4. *(Closest point function)*
Given a surface $\mathcal{S}$ in $\mathbb{R}^d$, $\text{cp}(\boldsymbol{x})$ *refers to a (possibly non-unique) point belonging to $\mathcal{S}$ which is closest to $\boldsymbol{x} \in \mathbb{R}^d$.*

In practice, there are various techniques to determine the closest point function depending on the surface given. An analytic representation of the closest point function can be given for simple geometries such as a circle, sphere or torus. If given a parametrization of the object, standard numerical optimization techniques can be applied to compute the closest point function. For more complex geometries, we may be given the surface in the form of a triangulated surface. For triangulated surfaces, we can examine the nearest point for all grid points in the embedding space to their respective nearest triangle (i.e. the surface). Due to interpolation, the closest point function need only be computed once for a distinct surface. Hence, going through a list of triangles is feasible. For triangulated surfaces there are also fast methods for evaluating the closest point function using tree-based algorithms [33].

Now we will illustrate how to handle the surface Laplacian using the closest point function through two principles and a theorem [22].

Principle 1. *(Gradients)*
For points $\boldsymbol{x}$ on a smooth surface, $\nabla_{\mathcal{S}} u(\boldsymbol{x}) = \nabla u(\mathrm{cp}(\boldsymbol{x}))$ because the function $u(\mathrm{cp}(\boldsymbol{x}))$ is constant in the normal direction and therefore only varies along the surface. In other words, at points $\boldsymbol{x}$ on the surface, intrinsic surface gradients $\nabla_{\mathcal{S}} u(\boldsymbol{x})$ are the same as gradients of $u(\mathrm{cp}(\boldsymbol{x}))$.

Principle 2. *(Divergence)*
Let $\nabla_{\mathcal{S}} \cdot$ denote the divergence operator intrinsic to a smooth surface $\mathcal{S}$ and let $\boldsymbol{v}$ be any vector field on $\mathbb{R}^d$ that is tangent at $\mathcal{S}$ and also tangent at all surfaces displaced by a fixed distance from $\mathcal{S}$ (i.e., all surfaces defined as level sets of the distance function to $\mathcal{S}$). Then at points $\boldsymbol{x}$ on the surface $\nabla \cdot \boldsymbol{v}(\boldsymbol{x}) = \nabla_{\mathcal{S}} \cdot \boldsymbol{v}(\boldsymbol{x})$.

By combining Principle 1 and 2, we can investigate the Laplace-Beltrami operator on a surface with the closest point function.

Theorem 1. *Let $\mathcal{S}$ be a smooth closed surface in $\mathbb{R}^d$ and $u : \mathcal{S} \to \mathbb{R}$ be a smooth function. Assume the closest point function $\mathrm{cp}(\boldsymbol{x})$ is defined in a neighborhood $\Omega \subset \mathbb{R}^d$ of $\mathcal{S}$. Then*

$$\Delta_{\mathcal{S}} u(\boldsymbol{x}) = \Delta u(\mathrm{cp}(\boldsymbol{x})) \quad \text{for} \quad \boldsymbol{x} \in \mathcal{S}. \tag{3.1}$$

The closest point function in Theorem 1 gives a normal extension to the surface, which means that on the manifold the surface Laplacian and standard Euclidean Laplacian are equivalent. For time-dependent PDEs, both operators cannot agree for long times, however if we select a suitable extension the evolution of the embedding PDE will be accurate initially, which will be sufficient to update the solution in time [30].

Definition 5. *(Closest point extension)*
Let $\mathcal{S}$ be a smooth surface in $\mathbb{R}^d$. The closest point extension of a function $u : \mathcal{S} \to \mathbb{R}$ to a neighborhood Ω of $\mathcal{S}$ is the function $v : \Omega \to \mathbb{R}$ defined by

$$v(\boldsymbol{x}) \quad = \quad u(\mathrm{cp}(\boldsymbol{x})). \tag{3.2}$$

Let us note that the closest point extension is an interpolation step in the implementation. We require a high enough order of the interpolation to avoid having its errors dominate the solution. More concretely, with a q^{th}-order difference scheme and a PDE involving up

to r^{th}-order derivatives, the interpolation order should be $q + r$ or higher. Polynomial interpolation is used for problems that have smooth solutions, whereas techniques involving Essentially Non-Oscillatory (ENO) [32] or Weighted Essentially Non-Oscillatory (WENO) interpolation [16] are used for problems with non-smooth solutions.

3.3 Case Study: The Closest Point Method for the Heat Equation on a Circle

The CPM is best explained through an example, and the heat equation on a circle highlights all the different components. The heat equation is described by the PDE:

$$u_t(\mathbf{x}) \quad = \quad \Delta_{\mathcal{S}} u(\mathbf{x}). \tag{3.3}$$

The CPM is composed of these major steps [30]:

1. Determine a closest point representation of the surface.
2. Choose a computational domain normally in the form of a band around the surface.
3. Replace surface gradients with standard gradients in $\mathbb{R}^3$.
4. Extend the surface data on the computational domain using the closest point function.
5. Solve the embedding PDE using standard finite differences in the computational domain.

For time-dependent processes, the last two steps are repeated for every time iteration. If the PDE is time-independent, then we run through all steps once. We will detail the important features of the method.

3.3.1 Closest point representation

The closest point representation for the circle is given by an analytic formula. This allows the software to quickly assign all the grid points their respective closest points on the circle as seen in Figure 3.1. If there was not an analytic formula, then an optimization step would be required in order to determine the closest points on the surface. The closest point representation is applied to all the grid points in the band surrounding the circle.

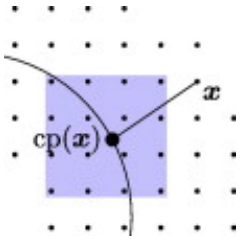

Figure 3.1: A discrete closest point extension being applied to extend $u(\text{cp}(\mathbf{x}))$ using degree $p = 3$ (bicubic) interpolation. The shaded region indicates the interpolation stencil for $\text{cp}(\mathbf{x})$. Reprinted with permission of the *Journal of Computational Physics* [[19], Figure 1].

3.3.2 Banding

Any efficient embedding method must solve the embedded PDE on a narrow band Ω_c, defined as

$$\Omega_c = \{x : \|x - \text{cp}(x)\|_2 \leq \text{bw}\},$$

where bw is the bandwidth. For the second-order centered difference Laplacian and gradient operators, the bandwidth value is given as

$$\text{bw} = \sqrt{(d-1)\left(\frac{p+1}{2}\right)^2 + \left(1 + \frac{p+1}{2}\right)^2}\,\delta x$$

where d is the dimension, p is the degree of the interpolating polynomials and δx is the stencil width. The derivation is detailed in Macdonald et al. [19]. Figure 3.2 illustrates the corresponding band of a circle with $\delta x = 0.1$.

The embedding PDE only agrees with the surface PDE when we have a constant normal extension of data off the surface. Although Step 4 of the CPM is required at every time iteration, it eliminates the need for artificial boundary conditions. This is a significant distinction between the closest point method and other embedding methods.

3.3.3 The extension matrix

The closest point extension is now represented using a matrix operator that we will refer to as the *extension matrix*, E. The matrix operator assigns a value of $u(\text{cp}(\mathbf{x}))$ to $u(\mathbf{x})$. This step takes the form of a matrix because barycentric Lagrange polynomial interpolation is a linear combination of values at neighbouring grid points in a hypercube in $\mathbb{R}^d$ (See A.4 for

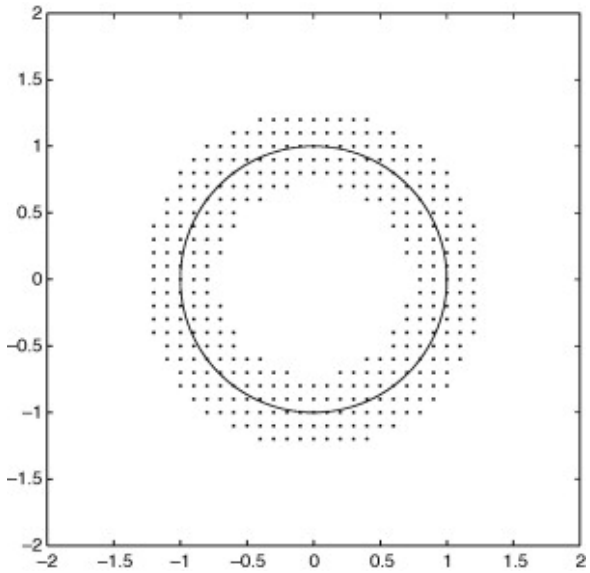

Figure 3.2: Computational grid for the closest point method for a circle with $\delta x = 0.1$. Five point differencing stencil and degree $p = 2$ interpolation are used. Reprinted with permission of the *Journal of Computational Physics* [[30], Figure 1].

details). These neighbouring grid points form the *interpolation stencil* of the scheme. We apply the interpolation of degree p, in a dimension-by-dimension fashion, so the hypercube has $p+1$ points in each coordinate direction. This stencil is illustrated by the shaded region in Figure 3.1 where a degree $p = 3$ interpolation is applied to the circle. We formalize the construction of the *extension matrix*.

Now we examine two ordered lists of points in the embedding space. Firstly, let $L = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m\}$ correspond to the list of all grid points in the interpolation stencil for a given point on the surface $\mathcal{S}$. The grid points in L compose the computational band on which the numerical solution is defined. Secondly we have $G = \{\mathbf{x}_{m+1}, \mathbf{x}_{m+2}, \ldots, \mathbf{x}_{m+m_g}\}$ corresponding to the "ghost points" along the edges of the computational band which are disjoint from the set L. The values of the points in set G do not propagate in time and are only used for the derivation.

Lastly, for the definition, we require defining two vectors over the list L and G. Let $u(\text{cp}(L)) \in \mathbb{R}^m$ denote the vector with components $u(\text{cp}(\mathbf{x}_i))$ for $\mathbf{x}_i \in L$ and let $u(\text{cp}(G)) \in \mathbb{R}^{m_g}$ denote the vector with components $u(\text{cp}(\mathbf{x}_{m+i}))$ for $\mathbf{x}_{m+i} \in G$. See Macdonald and Ruuth [22] for the complete derivation.

Definition 6. *(Extension matrix* E*)*
Given the vectors $\boldsymbol{u}, u(\mathrm{cp}(L)), u(\mathrm{cp}(G))$ *and an interpolation scheme as described above, the extension matrix is a matrix operator* E *such that*

$$\begin{pmatrix} u(\mathrm{cp}(L)) \\ u(\mathrm{cp}(G)) \end{pmatrix} \approx \mathrm{E}\boldsymbol{u}. \tag{3.4}$$

The nonzero entries in the i^{th} *row of* E *consist of the weights in the interpolation scheme for* $u(\mathrm{cp}(\boldsymbol{x}_i))$*. That is, the components of the matrix* $\mathrm{E} = [\gamma_{ij}]$ *are*

$$\gamma_{ij} = \begin{cases} w_j & \text{if } \boldsymbol{x}_j \text{ is in the interpolation stencil for } \mathrm{cp}(\boldsymbol{x}_i) \\ 0 & \text{otherwise.} \end{cases}$$

with w_j *denoting the weight associated with the grid point* x_j *in the interpolation scheme for point* $\mathrm{cp}(\boldsymbol{x}_i)$*.*

3.3.4 Time dependence

In order to step forward in time, we use some time stepping method such as the forward Euler time discretization:

$$u(t + \delta t) = u(t) + \delta t F(t, u(t)), \quad \forall u \in \Omega_c \tag{3.5}$$

where $F = \Delta u$. Macdonald et al. have successfully applied the closest point method to the problem of computing eigenvalues [19], reaction-diffusion processes on point clouds [20], segmentation on surfaces [35] as well as many other problems [30, 21, 22].

3.4 Implicit Closest Point Method

To solve the Laplace-Beltrami eigenvalue problem we need an implicit numerical method in space. The implicit closest point method introduces a matrix to carry out the interpolation step [22]. If we naively use an interpolation matrix that corresponds to the explicit approach (which was discussed in § 3.3.3), we find that some eigenvalues are real and positive. Figure 3.3 gives an example of the spectra for the 2D unit circle. Fortunately, there is a stabilized version of the implicit closest point method that eliminates these spurious eigenvalues. The spectra for the ICPM for the same test case is given in Figure 3.4 to illustrate the stability. We now give further details on the implicit closest point method.

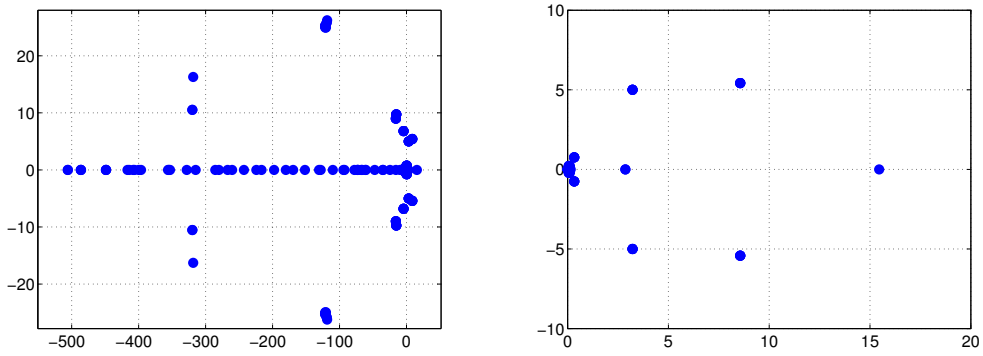

Figure 3.3: Spectra of $\widetilde{M}$ for a unit circle in 2D using biquartic interpolation ($p = 4$) and a mesh width of $\delta x = 0.1$. Some eigenvalues exist in the right half plane. Left: Full spectra. Right: Zoomed in.

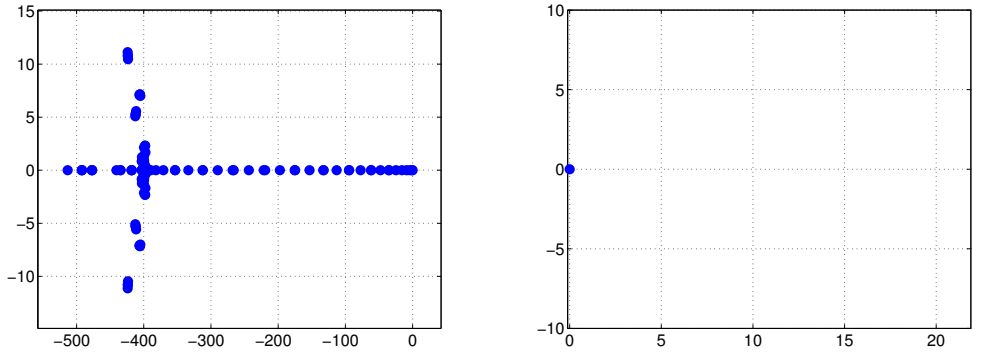

Figure 3.4: Spectra of M for a unit circle in 2D using biquartic interpolation ($p = 4$) and a mesh width of $\delta x = 0.1$. Eigenvalues do not exist in the right half plane. Left: Full spectra. Right: Zoomed in.

3.4.1 The discrete differential operator

The discretization of the differential operator gives us the following:

$$\Delta u(\text{cp}(\mathbf{x})) \approx \Delta_h \text{E}\mathbf{u} = \widetilde{M}\mathbf{u}, \tag{3.6}$$

where Δ_h is the finite difference scheme, and E is the extension matrix. Hence, this replaces the continuous operator in the original PDE. The choice is clear from taking the surface Laplacian of Definition 6. This naturally leads to the following problem:

Problem 2. *(Ill-posed embedded eigenvalue problem)*
Determine the eigenfunctions $v : \Omega \subset \mathbb{R}^d \to \mathbb{R}$ *and eigenvalues* λ *satisfying*

$$\Delta v(\text{cp}(\boldsymbol{x})) = -\lambda^2 v(\boldsymbol{x}), \tag{3.7}$$

in a neighbourhood $\Omega \subset \mathbb{R}^d$ *of* $\mathcal{S}$*.*

This immediate choice for the differential operator fails to correspond with the original Equation (1.4) for $\lambda = 0$ (null-eigenspace). We now detail this inconsistency.

3.4.2 The null-eigenspace

The constant eigenfunction $u(\mathbf{x}) = c$ and $\lambda = 0$ is a solution to Equation (1.4). If we evaluate a function on Ω which agrees with u on the surface (but differs off the surface)

$$v : \Omega \to \mathbb{R}, \qquad \text{such that } v(\mathbf{x}) = c \text{ for } \mathbf{x} \in \mathcal{S}, \tag{3.8}$$

and observe that $v(\mathbf{x})$ is a null-eigenfunction of Equation (3.7). The set of null-eigenfunctions for Equation (3.7) is infinite-dimensional since $v(\mathbf{x})$ is arbitrary for $\mathbf{x} \in \Omega \backslash \mathcal{S}$. Moreover, any linearly independent change off the surface gives a new linearly independent eigenfunction, as Macdonald et al. [19] discuss.

3.4.3 A revised discrete differential operator

As previously mentioned, a study of the spectra of $\widetilde{M}$ divulges that some eigenvalues have positive real parts [22] (see Figure 3.3). A slight modification to the initial implicit closest point method can eliminate the spurious eigenvalues and accurately capture the null-eigenspace.

The adjustment is made within the finite difference stencil represented by the matrix Δ_h. The diagonal elements will involve u_i instead of $u(\text{cp}(\mathbf{x}_i))$ and all other neighbouring points of $\mathbf{x}_j$ of the stencil map back to their closest points $\text{cp}(\mathbf{x}_j)$.

$$M = stab(\Delta_h, \text{E}) := diag\Delta_h + (\Delta_h - diag\Delta_h)\text{E}. \tag{3.9}$$

The Laplace-Beltrami operator when discretized then becomes:

$$\Delta u(\text{cp}(\mathbf{x})) \approx Mu = -\lambda^2 u. \tag{3.10}$$

Now that we have the ICPM, which gives accurate results for the surface Laplacian, we can give a formal description of the Laplace-Beltrami eigenvalue problem in combination with the closest point method.

3.5 Closest Point Method Approach

The closest point method will be applied with standard finite difference schemes and interpolation to obtain subgrid resolution at the points representing the surface. We will apply the ICPM as discussed in this chapter.

3.5.1 The embedded eigenfunction problem

As mentioned in §3.4.1, Equation (3.6) leads to an ill-posed embedded eigenvalue problem, Equation (3.7). The proof can be found in Macdonald et al. [19]. Instead, we define our embedded formulation of the surface Laplacian as follows.

Definition 7. *(Operator $\Delta_\epsilon^\#$)*
Given $\Omega \subset \mathbb{R}^d$ containing a surface $\mathcal{S}$ and a function $v : \Omega \to \mathbb{R}$, the operator $\Delta_\epsilon^\#$ is defined as

$$\Delta_\epsilon^\# : \quad = \quad \Delta(v(\text{cp}(\boldsymbol{x}))) - \frac{2d}{\epsilon^2}\left[v(\boldsymbol{x}) - v(\text{cp}(\boldsymbol{x}))\right] \tag{3.11}$$

where $0 < \epsilon \ll 1$.

Macdonald et al. [19] have the factor $2d$ for notational convenience and a penalty term for large change in the normal direction. We investigate the newly posed eigenvalue problem which is consistent with Equation (1.4):

Problem 3. *Regularized embedded eigenvalue problem.*
Determine all eigenfunctions $v : \Omega \subset \mathbb{R}^d \to \mathbb{R}$ *and eigenvalues* λ *satisfying*

$$\Delta_\epsilon^{\#} v(x) \quad = \quad -\lambda^2 v(\boldsymbol{x}), \tag{3.12}$$

in an embedding space $\Omega \subset \mathbb{R}^d$ *containing the surface* $\mathcal{S}$*.*

Before illustrating the strengths of the CPM by means of numerical results, we first motivate the CPM by qualitative reasoning for common scenarios.

3.6 Finite Element Approach

Reuter et al. [29] have solved the Laplace-Beltrami operator by the finite element method. In order to obtain a more accurate solution to the PDE, their method requires a better triangulation on the surface. The major concern with triangulations is that, in real world situations, we require human intervention to get a good triangulation, and surface scans do not normally come as triangulated data–they come as points in a point cloud. The finite element method may also be complicated on surfaces, when high order and curved elements are needed. Moreover, creating a triangulation is a costly step and the refinement can vary depending on what triangulation is applied[1]. Refinement on the surface for the CPM is uniform and easily implemented simply by changing one parameter: the stencil of the CPM. In this thesis, we propose to use the CPM since it can be implemented on either a triangulation or a point cloud.

Now that the groundwork has been laid down, we can commence some examples with the ICPM determining the eigenvalues of the Laplace-Beltrami operator. The focus being on the accuracy of the eigenvalues and shape matching.

[1]Delaunay triangulation is one common implementation [25].

Chapter 4

Numerical Results

This chapter begins by computing the eigenvalues of several manifolds. In practice, surface representations of objects may be given in different forms. Two classical forms for the surface of an object are as triangulations and point clouds. We consider both of these fundamental data sets in this chapter, where applying the ICPM to a point cloud for the Laplace-Beltrami is unprecedented. We validate results by performing convergence studies and performing shape matching for clusters of similar objects.

4.1 Two-Dimensional Results

In two dimensions, the surface (or boundary) of the shapes correspond to a one-dimensional curve. Numerical convergence studies are carried out for surfaces on which the exact solution is known.

The eigenvalues of the unit circle are given by $\{0, 1, 4, 9, 16, \ldots\}$, with multiplicities 2 for eigenvalues greater than 0. Refining the stencil size of the closest point method we expect convergence corresponding to the orders of the interpolation and finite difference schemes. Taking a degree three interpolation and standard second-order finite differences, we achieve second-order convergence (See top of Figure 4.1). Similarly for polynomial interpolation of degree five and a finite difference stencil of fourth-order, we obtain fourth-order convergence (See bottom of Figure 4.1). Changing the geometry to an ellipse also gives second-order accuracy similar to that of the circle, see Figure 4.2.

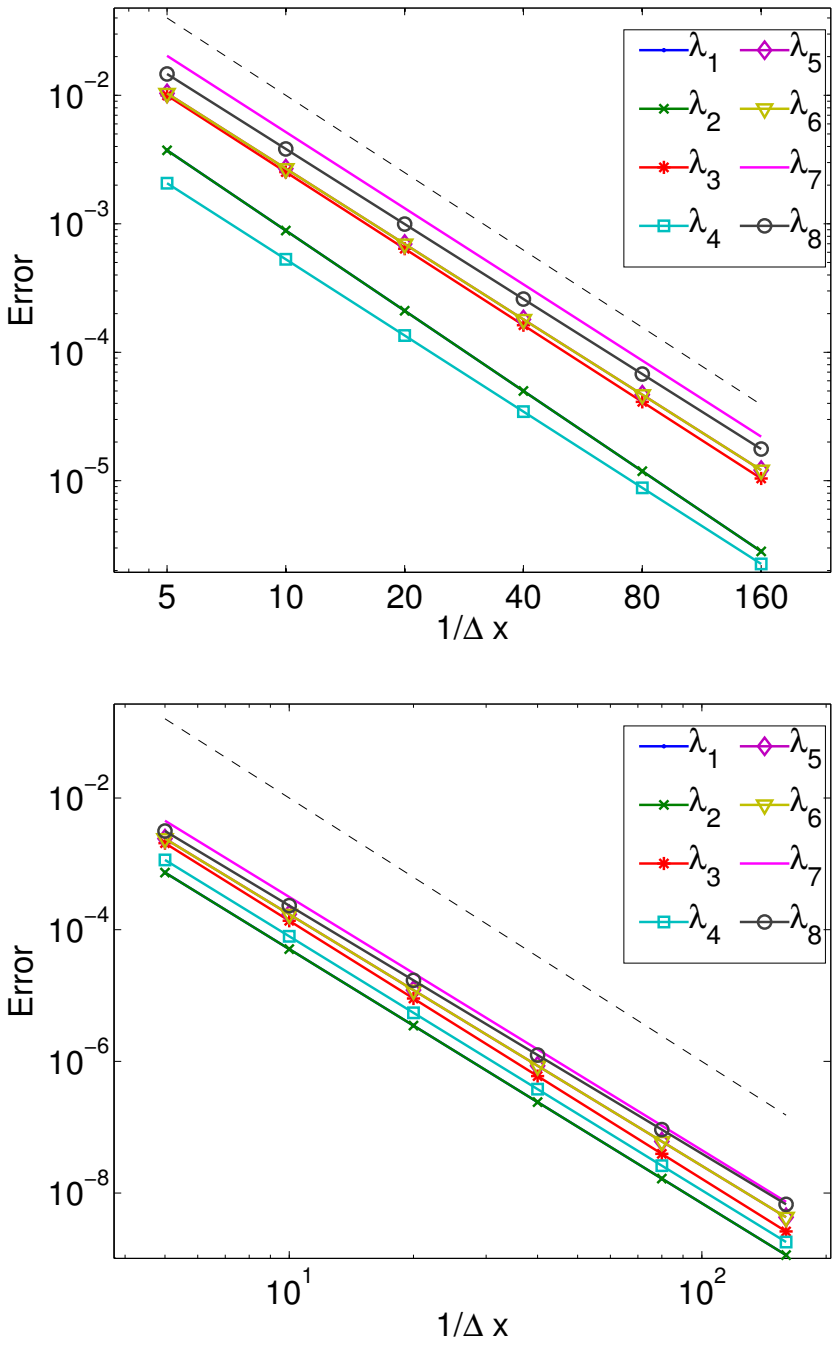

Figure 4.1: Numerical convergence study for the first few eigenvalues of the unit circle. Top: Degree $p = 3$ interpolation and second-order centered finite differences. The dashed line has slope 2. Bottom: interpolation $p = 5$, and fourth-order centered finite differences. The dashed line has slope 4.

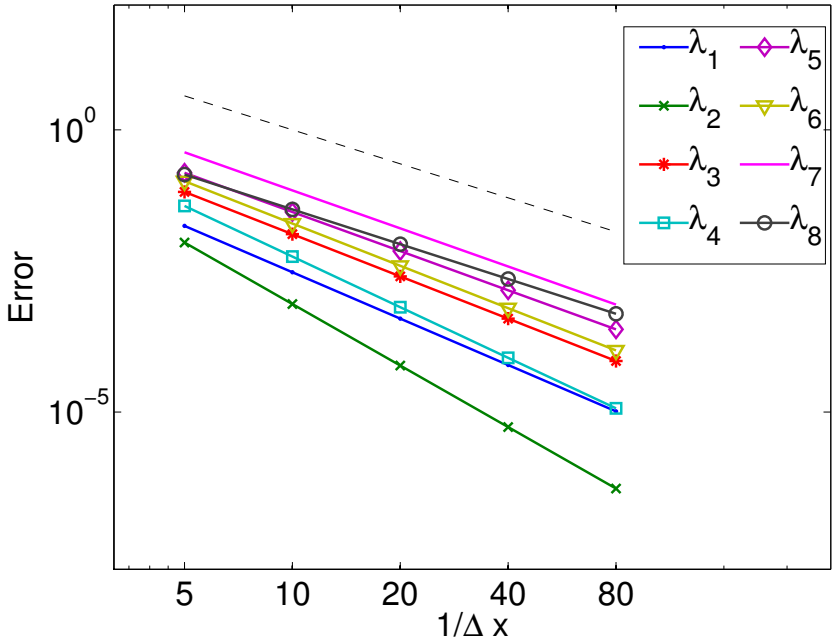

Figure 4.2: Numerical convergence study for the first few eigenvalues for the ellipse with major axis 1.5 and minor axis 0.75. Second-order finite differences and degree $p = 3$ interpolation are used. The dashed line has slope 2.

The eigenvalues are computed using MATLAB's built in *eigs* function which is based on ARPACK [17]. We remark that the error in the eigenvalues increases for larger eigenvalues but we still observe the expected convergence rates. The *eigs* function uses the Arnoldi iteration to determine the eigenvalues of the matrix representing the Laplace-Beltrami operator.

4.2 Three-Dimensional Results

4.2.1 The sphere

For some common surfaces we have an explicit formula for the closest point representation. We first test our method on the sphere. Figure 4.3 demonstrates the expected second-order

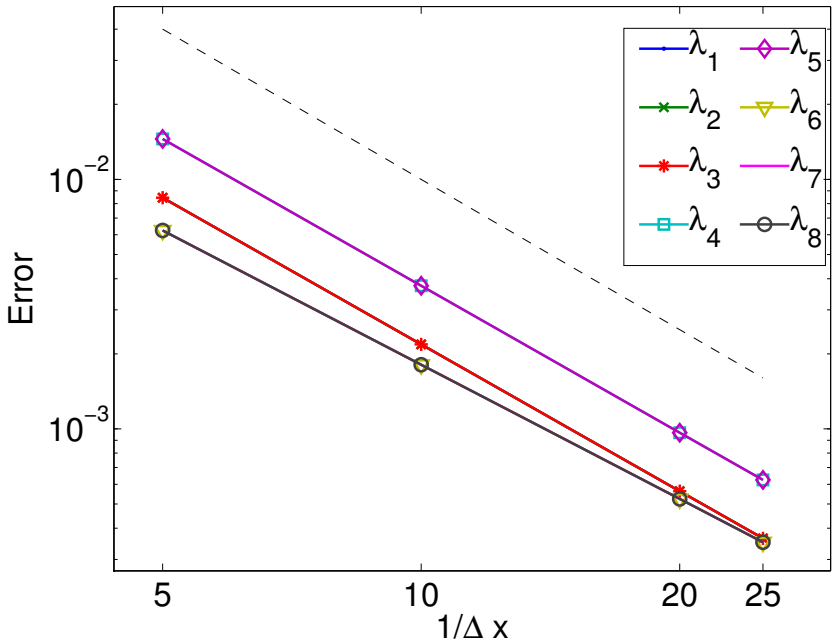

Figure 4.3: Numerical convergence study for the first few eigenvalues of the unit sphere. Degree $p = 3$ interpolation and second-order centered finite differences are used. The dashed line has slope 2.

accuracy.

4.2.2 Orthogonality

The processing of the surface structure is *orthogonal* to the CPM solver for the eigenvalues of the Laplace-Beltrami operator. The processing refers to determining the closest points of the embedding space to the surface. Orthogonality refers to the computer science term of decoupled or independent functions within a program. After computing the closest points of the embedding space, the resulting data is given to the CPM solver. If these two steps were intertwined then the code [1] would require drastic changes for every new surface we are interested in. This provides great flexibility with different data inputs such as analytic representations, triangulated surfaces or point clouds (as we shall see in this chapter).

4.2.3 Closest point representation of the surface

If we are not given a closest point representation we must determine the closest points on the surface to the computational grid, along with their distances. The naive approach would be to directly determine the triangle (or point) closest to each grid node in the embedding space. However this approach is computationally expensive and inefficient. This would lead to a cost of $\mathcal{O}(N_{\text{triangles}} \times N_{\text{band grid points}})$. The procedure is instead as follows:
For each triangle:

- Determine the radius and center of a sphere surrounding the triangle. Place a uniform grid surrounding the sphere (and triangle in turn).
- For each grid point in the cube, determine the closest point on the surface of the triangle.
- If this distance is smaller than any pre-existing distance for this grid point, then replace the value. Otherwise, do nothing.

Figure 4.4 demonstrates the combined result of creating the cube for the closest point representation for one triangle (of the triangulated surface). The number of grid points surrounding each triangle is constant, and this process leads to $\mathcal{O}(N_{\text{triangles}})$ operations. In regards to the computational band for the closest point method, this is the collection of the all the cubes surrounding each triangle. Note that one could also compute the closest point for every point in the embedding space (i.e. without employing banding), however this is only beneficial when working in Fourier Space [20].

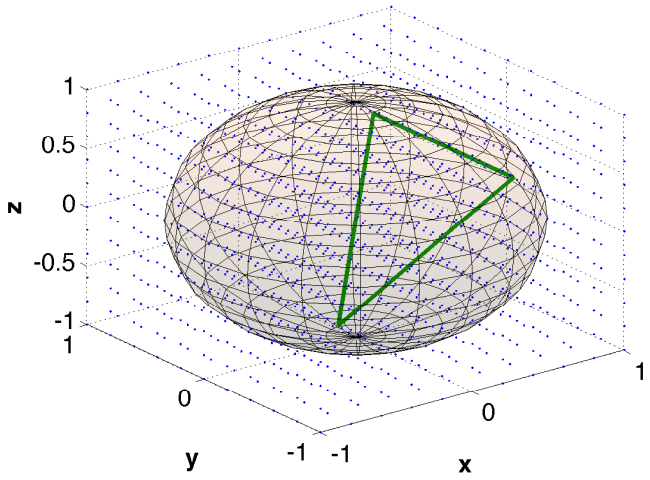

Figure 4.4: Creating the computational grid surrounding the triangle for the closest point method.

4.2.4 Triangulated surfaces

Certain simple surfaces and curves are excellent for validating our methods and software, since we have analytic formulas at our disposal. Much more complex shapes are of interest, but we do not have the same analytic formulas. A triangulation is a common form that describes the surface of an object. The closest point representation changes from an analytic form to an approximate form. The approximate form will be constructed from the underlying triangulation that represents the object. Clearly, the accuracy of the solution depends on the accuracy of the triangulation to the exact object.

Convergence

To illustrate the dependence on a good triangulation, we see that for a unit sphere with 2,562 vertices, and 5,120 faces we do not obtain second-order accuracy (Figure 4.5), whereas with 2,621,422 vertices, and 5,242,880 faces (Figure 4.6) we do indeed observe the correct order.

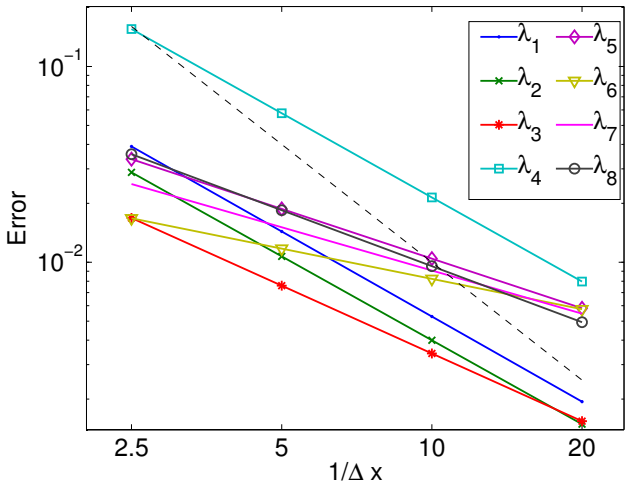

Figure 4.5: Numerical convergence study for the first few eigenvalues of a coarse triangulation for a sphere. Degree p=3 interpolation and second-order centered finite differences. The dashed line corresponds to second-order accuracy.

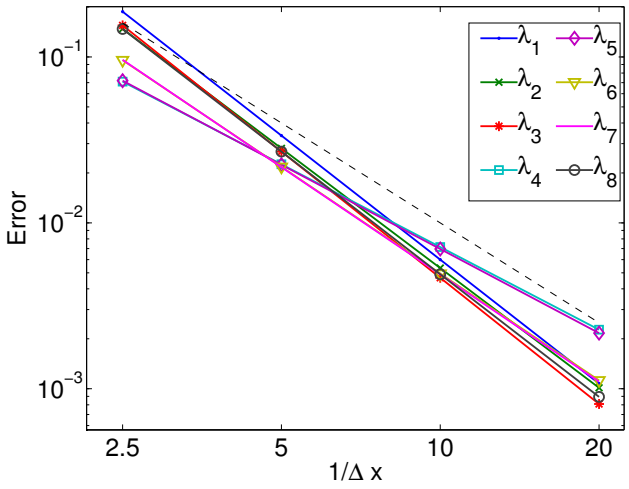

Figure 4.6: Numerical convergence study for the first few eigenvalues of a fine triangulation for a sphere. Degree p=3 interpolation and second-order centered finite differences. The dashed line has slope 2.

Size of triangulation

We now quantify the size of the triangulation and how fine it must be to give reliable results. Our approach uses the radius of the largest triangle where the radius of the triangle is defined as the radius of an inscribed circle (See Figure 4.7 and Equation 4.1). Now that we defined a measurement for the triangles that depicts the surface, we can now investigate how small the triangulation need be in order to obtain the expected convergence. Table 4.1 has listed the order of convergence for the first few eigenvalues for different maximum radius values. In Table 4.1, 4.2 and 4.3, each λ_i is the next distinct eigenvalue for the exact values of the unit sphere, i.e. $\lambda_1 = 2$, $\lambda_2 = 6$, $\lambda_3 = 12$ and so forth. Both tables have computed the first 160 eigenvalues using *eigs* in MATLAB. We expect the order of convergence to match the order of the finite difference and interpolation schemes used. However, as we calculate more eigenvalues in one call of the *eigs* function, the accuracy of the approximated eigenvalues decreases. In Table 4.3, we compute only 30 eigenvalues and see that the approximation for the first few distinct eigenvalues have improved results compared to Table 4.1. The size of the triangulation depends on how many eigenvalues are required by the user. We recommend that a triangulation with maximum radius, $R_{\max}$, to be $R_{\max} \leq 0.01$ for taking few eigenvalues, and $R_{\max} \leq 0.004$ for taking more than 150 eigenvalues. Let us note that further refinement is possible with a system that contains more storage and RAM (Mac OS X was used with 2.5 GHz Intel Core i5 and 8 GB 1600 MHz DDR3).

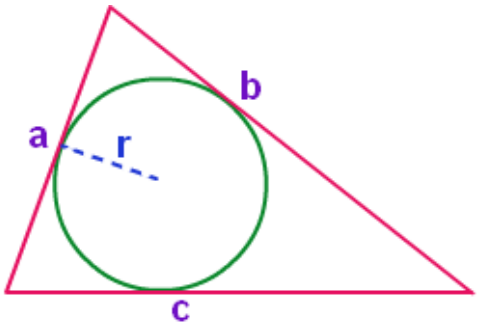

Figure 4.7: A circle inscribed in a triangle.

$$Radius = \frac{TriangularArea}{k} = \frac{\sqrt{k(k-a)(k-b)(k-c)}}{k}, \tag{4.1}$$

$$\text{where } k = \frac{1}{2}(a+b+c). \tag{4.2}$$

$R_{\max}$ \ λ_i	λ_1	λ_2	λ_3	λ_4	λ_5	λ_6	λ_7	λ_8	λ_9
0.03343480	1.1632	0.7610	1.3477	1.7114	2.0296	2.2770	2.2329	2.3215	2.3391
0.02371572	0.9938	0.9332	1.4924	1.7058	2.0263	2.3057	2.3059	2.2724	2.3593
0.01670541	0.8740	0.8837	1.5548	1.8529	2.0642	2.2209	2.3477	2.2983	2.3328
0.01178180	1.3749	1.0900	1.5796	1.8928	2.1735	2.3040	2.3606	2.3648	2.3740
0.0083981	1.4775	1.0688	1.5914	1.9673	2.1452	2.3578	2.3720	2.3594	2.3441
0.00386704	1.6724	1.0345	1.6699	2.0067	2.2588	2.3678	2.3954	2.3459	2.3930

Table 4.1: Order of convergence for the first few distinct eigenvalues of the unit sphere for different size triangulations. ICPM was used with second-order centered finite differences and a degree three interpolation. MATLAB's *eigs* function was used to compute the first 160 eigenvalues.

$R_{\max}$ \ λ_i	λ_1	λ_2	λ_3	λ_4	λ_5	λ_6	λ_7	λ_8	λ_9
0.03343480	1.5302	1.3688	1.0460	2.0337	2.2561	3.1527	3.4391	3.8256	3.9598
0.02371572	1.4389	1.2866	0.6907	2.0685	2.1889	3.4355	3.6169	3.7949	4.2088
0.01670541	1.5391	1.3019	0.7214	2.2069	2.1897	3.4771	3.5245	3.9022	4.0072
0.01178180	1.6009	1.3604	0.6584	2.2637	2.1687	3.3331	3.7529	4.0001	4.2356
0.0083981	1.6647	1.4707	0.8711	2.1357	2.3497	3.5401	3.6635	4.0780	4.0577
0.00386704	1.6897	1.4285	0.8632	2.5833	2.2824	3.5867	3.7874	4.0964	4.2553

Table 4.2: Order of convergence for the first few distinct eigenvalues of the unit sphere for different size triangulations. ICPM was used with fourth-order centered finite differences and a degree five interpolation. MATLAB's *eigs* function was used to compute the first 160 eigenvalues.

$R_{\max}$ \ λ_i	λ_1	λ_2	λ_3	λ_4	λ_5
0.03343480	1.4913	1.6799	1.5070	1.9218	2.0683
0.02371572	1.3192	1.8452	1.6631	1.9271	2.0652
0.01670541	1.1882	1.7686	1.7193	2.0770	2.1024
0.01178180	1.7306	1.9431	1.7334	2.1256	2.2117
0.0083981	1.8059	1.9741	1.7470	2.1781	2.1838
0.00386704	2.0491	1.9233	1.8163	2.2484	2.2967

Table 4.3: Order of convergence for the first few distinct eigenvalues of the unit sphere for different size triangulations. ICPM was used with second-order centered finite differences and a degree three interpolation. MATLAB's *eigs* function was used to compute the first 30 eigenvalues.

Shape matching

Now by applying the method to a class of objects, through the aid of multidimensional scaling (MDS), we can visualize clustering of similar objects (See Figure 4.8). The objects consist of a horse, cat, cow, camel, 3 bunnies with varying mesh refinement, hemisphere, ellipsoid, set of pliers with slight deformations. As expected, clusters begin to form where similarly shape objects are closer in Shape-DNA. The four legged animals remain within the same region. The bunnies have a base structure similar to the hemisphere and ellipsoid. The pliers do not resemble any other object closely and thus form their own cluster. A second example in Figure 4.9 has two human models, 3 bunnies and 3 pigs, each of which have a deformation from the original (i.e. scale, rotation, smoothing). We see three clusters formed from this given data set. Lastly, Figure 4.10 has two human models, a dolphin and a shark. The two marine mammals distances are close, as are the two human models. As expected the distance between the two different species is larger.

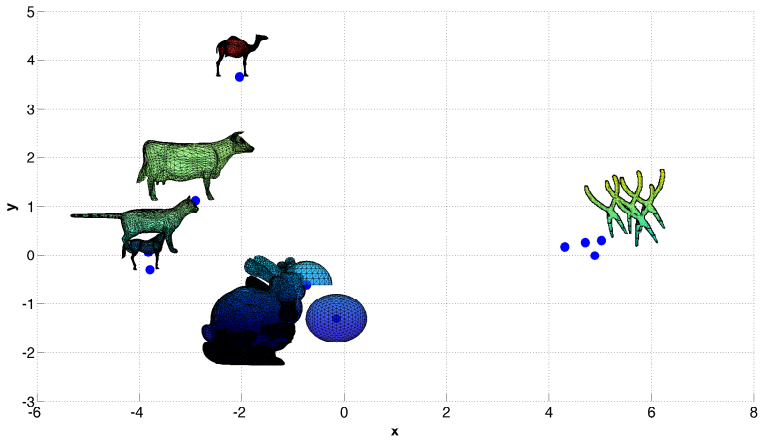

Figure 4.8: MDS plot illustrating the 3D objects. The ICPM stencil has the value 0.05.

4.2.5 Point clouds

Implementation of the point cloud, although is completely new, follows from the discussion in 4.2.3. The only exception is that we no longer have pieces of the surface, only points. Hence, we require an adequate resolution in order to obtain accurate results since the closest point on the surface only consists of the points in the cloud, and no points in between. In

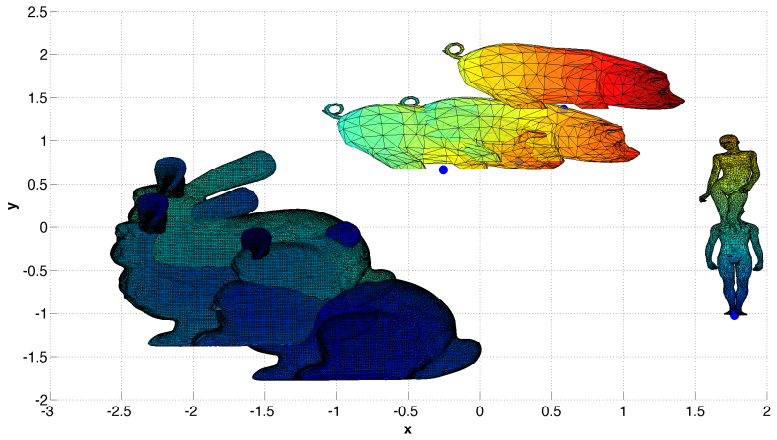

Figure 4.9: MDS plot illustrating the 3D objects. The ICPM stencil has the value 0.05.

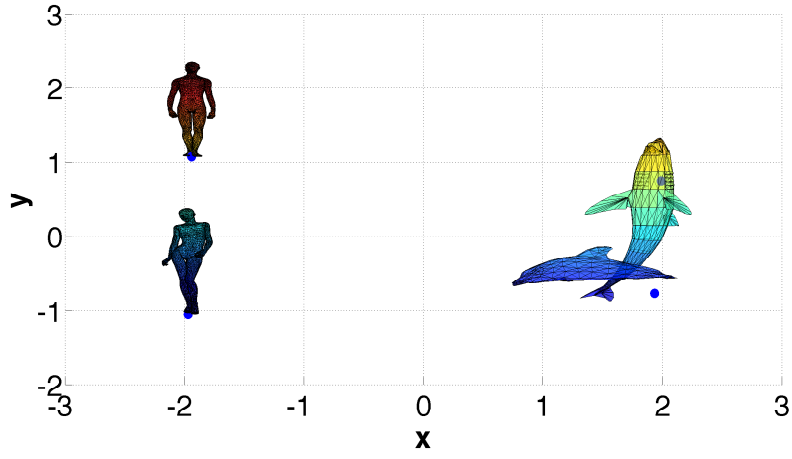

Figure 4.10: MDS plot illustrating the 3D objects. The ICPM stencil has the value 0.05.

many applications, laser scanners extract millions of data points representing the surface of the manifold, which justifies the high degree of density for this class of problem we are investigating. The analytic representation analog simply selects closest points on the surface as well, except the refinement is exact. Intuitively, if the refinement of the point cloud had a similar density, then we could obtain a result just as accurate as Figure 4.3. For the sphere, we investigate the convergence dependent on the point cloud approximation to the exact surface.

Convergence

In order to confirm that the implementation for the point cloud does give the correct result, we perform a convergence test for the sphere. The density of the point cloud determines the accuracy of the ICPM. A convergence study which fixes the surface representation, and varies the ICPM stencil, does not result in the expected order of accuracy (See Figure 4.11). This is due to the competing errors of the ICPM and point cloud representation. In order to determine the order of accuracy for the point cloud, we perform a different convergence study. We fix the stencil of the ICPM, and refine the point cloud (See Figure 4.12). As we see in Figure 4.12, the error is first-order accurate, and this explains the results in Figure 4.11. The error of the point cloud representation dominates the error of the ICPM. Thus, even if we refine the mesh of the ICPM applied to a fine mesh representation, the result does not coincide with the accuracy of the ICPM. Regardless of this first-order accuracy, we still have the solution converging to the exact solution as we refine the point cloud which is what we require.

We have successfully applied the ICPM to solve for the Shape-DNA of the Laplace-Beltrami operator. The verification comes from the convergence study in both two and three-dimensions. A required accuracy is needed for both the triangulation and point cloud representation. Using multidimensional scaling we were able to depict clustering of similarly shaped objects from a given data set based on their Shape-DNA.

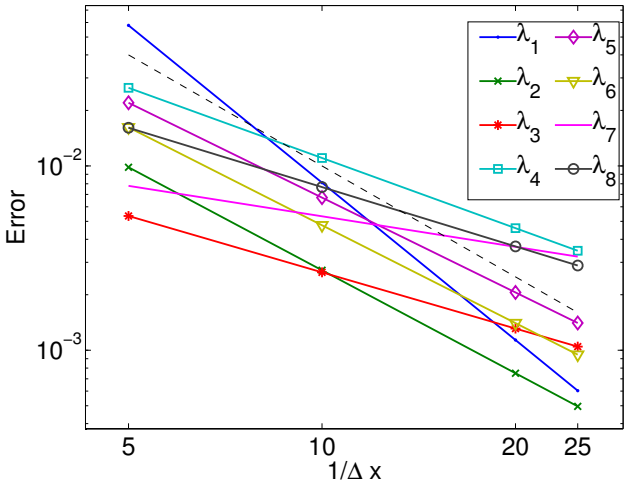

Figure 4.11: Convergence study of point cloud on a log-log plot. ICPM stencil varies and the cloud has $2,621,442$ points. The black dashed reference line has slope 2.

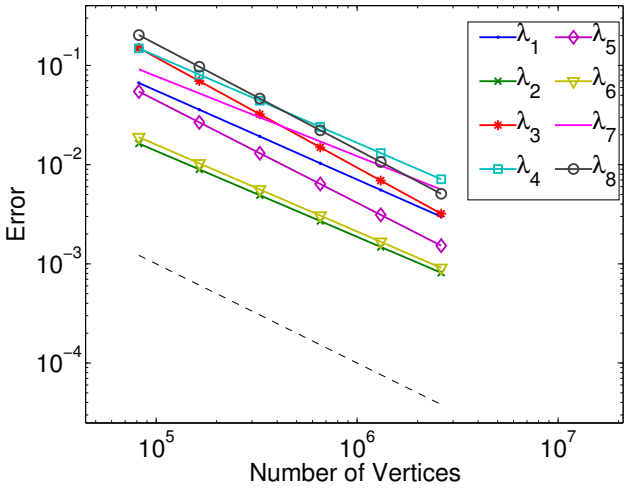

Figure 4.12: Convergence study of point cloud on a log-log plot. ICPM stencil is $\delta x = 0.04$ and the point cloud varies by the number of vertices in the representation. The black dashed reference line at the bottom has slope 1.

Chapter 5

Conclusion

In this thesis, we have successfully applied the ICPM for determining the eigenvalues of the Laplace-Beltrami operator on surfaces for shape matching. We have numerical simulations for manifolds in two dimensions, three dimensions, triangulations, and dense point clouds.

The initial motivation came from applying the ICPM for shape matching. This was accomplished by solving for the eigenvalues of the Laplace-Beltrami operator by the ICPM (based on the code [1]). By determining the spectra, we were able to conduct shape matching without affecting the representation of the manifolds, which is common for a watermark implementation. The shape matching was based on the shape matching algorithm comprised of determining the Shape-DNA, normalizing the Shape-DNA, and plotting the results which demonstrate similar objects being clustered together.

In the simulations, we first had a convergence study verifying the accuracy of the Shape-DNA coincided with the stencil of the ICPM. This convergence test was ran for surfaces given by an explicit formula, triangulated surfaces, and point cloud representations. The triangulated surface and point cloud representation were shown to require a degree of refinement. Since the Shape-DNA was proven to be accurate, a visualization of similar objects (based on their eigenvalues of the Laplace-Beltrami operator) was seen for different test cases using multidimensional scaling. This is the first time the ICPM has been used for shape matching. Moreover, this is the first time the ICPM has been applied to a point cloud.

Further investigation remains for applying the ICPM on a point cloud which is not as dense as the ones illustrated in this thesis. This would add to a much wider range of objects. Moreover, there is potential for applying this method for automated structure detection for

the nonlocal Cahn-Hilliard equation (See Choksi et al. [10]). They require identification of triply files which consist of spheres, double gyroids, lamellae, cylinders, perforated lamellae, etc (See Figure 5.1).

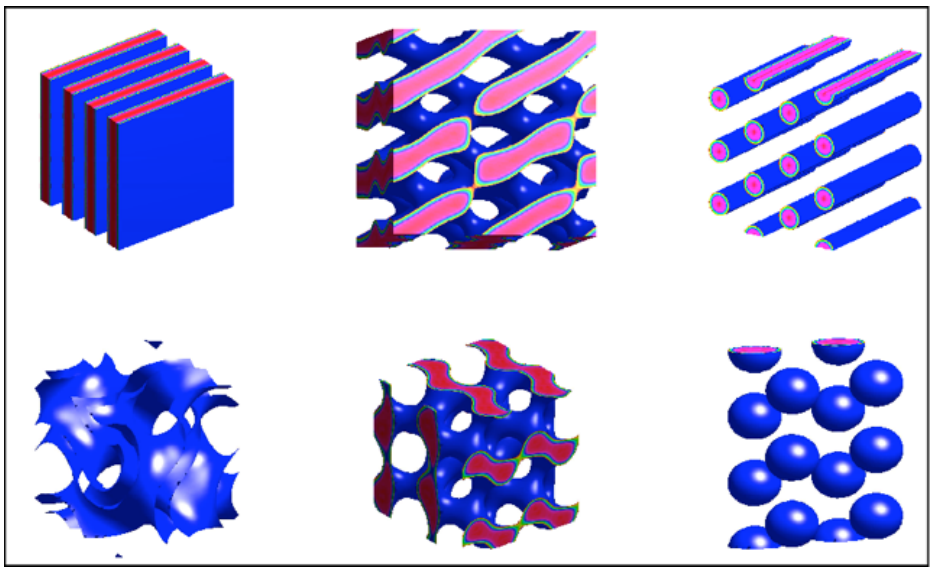

Figure 5.1: Shapes which are of interest to the nonlocal Cahn-Hilliard equation. Reprinted with permission of the *SIAM Journal on Applied Mathematics* [[10], Figure 5].

Bibliography

[1] *Closest point method software*. 2014. Available at https://github.com/cbm755/cp_matrices.

[2] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(4):509–522, Apr 2002.

[3] O. Benedens. Geometry-based watermarking of 3d models. *Computer Graphics and Applications, IEEE*, 19(1):46–55, Jan 1999.

[4] Jean-Paul Berrut and Lloyd N. Trefethen. Barycentric lagrange interpolation. *SIAM Rev*, 46:501–517, 2004.

[5] Marcelo Bertalmo, Li-Tien Cheng, Stanley Osher, and Guillermo Sapiro. Variational problems and partial differential equations on implicit surfaces. *Journal of Computational Physics*, 174(2):759 – 780, 2001.

[6] Dmitriy Bespalov, Ali Shokoufandeh, William C. Regli, and Wei Sun. Scale-space representation of 3d models and topological matching. In *Proceedings of the Eighth ACM Symposium on Solid Modeling and Applications*, SM '03, pages 208–215, New York, NY, USA, 2003. ACM.

[7] D. Braess. *Finite Elemente*. Berlin: Springer, 1997.

[8] Alexander M. Bronstein, Michael M. Bronstein, and Ron Kimmel. Three-dimensional face recognition. *International Journal of Computer Vision*, 64(1):5–30, 2005.

[9] I. Chavel. *Eigenvalues in Riemannian Geometry*. Pure and Applied Mathematics. Elsevier Science, 1984.

[10] Rustum Choksi, Mark A. Peletier, and J. F. Williams. On the phase diagram for microphase separation of diblock copolymers: An approach via a nonlocal cahn–hilliard functional. *SIAM Journal of Applied Mathematics*, 69(6):1712–1738, 2009.

[11] T.F. Cox and A.A. Cox. *Multidimensional Scaling, Second Edition*. Chapman & Hall/CRC Monographs on Statistics & Applied Probability. Taylor & Francis, 2000.

[12] Michael S. Floater and Kai Hormann. *Surface parameterization: a tutorial and survey*. Springer, 2005.

[13] John B. Greer. An improvement of a recent eulerian method for solving pdes on general geometries. *J. Sci. Comput.*, 29(3):321–352, December 2006.

[14] A. Hunt and D. Thomas. *The Pragmatic Programmer: From Journeyman to Master*. Pearson Education, 1999.

[15] Natraj Iyer, Subramaniam Jayanti, Kuiyang Lou, Yagnanarayanan Kalyanaraman, and Karthik Ramani. Three-dimensional shape searching: state-of-the-art review and future trends. *Computer-Aided Design*, 37(5):509 – 530, 2005. Geometric Modeling and Processing 2004.

[16] Guang-Shan Jiang and Chi-Wang Shu. Efficient implementation of weighted {ENO} schemes. *Journal of Computational Physics*, 126(1):202 – 228, 1996.

[17] R. B. Lehoucq, D. C. Sorensen, and C. Yang. *ARPACK Users Guide: Solution of Large Scale Eigenvalue Problems by Implicitly Restarted Arnoldi Methods, SIAM.* 1998.

[18] Marc Levoy, Kari Pulli, Brian Curless, Szymon Rusinkiewicz, David Koller, Lucas Pereira, Matt Ginzton, Sean Anderson, James Davis, Jeremy Ginsberg, Jonathan Shade, and Duane Fulk. The digital michelangelo project: 3d scanning of large statues. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '00, pages 131–144, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[19] Colin B. Macdonald, Jeremy Brandman, and Steven J. Ruuth. Solving eigenvalue problems on curved surfaces using the closest point method. *Journal of Computational Physics*, 230(22):7944 – 7956, 2011.

[20] Colin B. Macdonald, Barry Merriman, and Steven J. Ruuth. Simple computation of reaction-diffusion processes on point clouds. *Proceedings of the National Academy of Sciences*, 110(23):9209–9214, 2013.

[21] Colin B Macdonald and Steven J Ruuth. Level set equations on surfaces via the closest point method. *Journal of Scientific Computing*, 35(2-3):219–240, 2008.

[22] Colin B. Macdonald and Steven J. Ruuth. The implicit Closest Point Method for the numerical solution of partial differential equations on surfaces. *SIAM J. Sci. Comput.*, 31(6):4330–4350, 2009. doi:10.1137/080740003.

[23] MATLAB. *version 7.10.0 (R2010a)*. The MathWorks Inc., Natick, Massachusetts, 2010.

[24] H. P. McKean, Jr. and I. M. Singer. Curvature and the eigenvalues of the laplacian. *Journal of Differential Geometry*, 1(1-2):43–69, 1967.

[25] Atsuyuki Okabe, Barry Boots, and Kokichi Sugihara. *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. John Wiley & Sons, Inc., New York, NY, USA, 1992.

[26] Lawrence M. Parsons and Peter T. Fox. *The neural basis of implicit movements used in recognising hand shape.* Cognitive Neuropsychology, Vol 15(6-8), pg. 583-615. 1998.

[27] J. Prokop and Anthony P. Reeves. A survey of moment-based techniques for unoccluded object representation and recognition. In *CVGIP: Graphical Models and Image Processing*, pages 438–460, 1992.

[28] M. Reuter. Mds plots. http://reuter.mit.edu/research/shapedb1/mdsplots/.

[29] Martin Reuter, Franz-Erich Wolter, and Niklas Peinecke. Laplacebeltrami spectra as shape-dna of surfaces and solids. *Computer-Aided Design*, 38(4):342 – 366, 2006.

[30] Steven J. Ruuth and Barry Merriman. A simple embedding method for solving partial differential equations on surfaces. *J. Comput. Phys.*, 227(3):1943–1961, January 2008.

[31] Minakshisundaram S. Eigenfunctions on riemanian manifolds. (17):43–69, 1953.

[32] Chi-Wang Shu and Stanley Osher. Efficient implementation of essentially non-oscillatory shock-capturing schemes, ii. *J. Comput. Phys.*, 83(1):32–78, July 1989.

[33] John Strain. Fast tree-based redistancing for level set computations. *Journal of Computational Physics*, 152(2):664 – 686, 1999.

[34] G. Strang. *Introduction to Applied Mathematics*. Wellesley-Cambridge Press, 1986.

[35] Li Tian, C.B. Macdonald, and S.J. Ruuth. Segmentation on surfaces with the closest point method. In *Image Processing (ICIP), 2009 16th IEEE International Conference on*, pages 3009–3012, Nov 2009.

Appendix A

Mathematical Results

A.1 Deriving Helmholtz Equation

Helmholtz equation can be derived via the wave equation. First recall the wave equation:

$$u_{tt}(\mathbf{r}, t) = c^2 \Delta u(\mathbf{r}, t) \tag{A.1}$$

Using the technique of separation of variables, we assume the solution has the form, $u(r, t) = R(r)T(t)$. Substituting this form into the PDE gives:

$$\begin{aligned} \frac{d^2}{dt^2} R(r)T(t) &= c^2 \Delta (R(r)T(t)), \\ R(r)\frac{d^2T}{dt^2} &= c^2 T(t) \Delta R(r), \\ \frac{1}{c^2 T^2}\frac{d^2T}{dt^2} &= \frac{\Delta R(r)}{R(r)}, \end{aligned}$$

The left hand side depends only on t and the right hand side depends only on r. Hence, they are independent from one another, and so we can set the equations equal to some arbitrary constant, $-\lambda^2$. This gives the following set of equations:

$$\begin{aligned} \frac{d^2T}{dt^2} &= -\lambda^2 c^2 T^2, \quad \text{second-order ODE,} \\ \Delta R(r) &= -\lambda^2 R(r), \quad \text{Helmholtz equation.} \end{aligned}$$

A.2 Scaling a n-Dimensional Manifold

We will prove that the eigenvalues of a scaled manifold can be computed from the original surface simply by multiplying by an appropriate factor. In order to illustrate this, we first introduce notation from Reuter et al. [29] in order to simplify the computations. Given a local parametrization $\psi : \mathbb{R}^n \to \mathbb{R}^{n+k}$ of a submanifold $\mathcal{M}$ of $\mathbb{R}^{n+k}$ with

$$g_{ij} : \quad = \quad \partial_i\psi \cdot \partial_j\psi, \qquad G := (g_{ij}), \qquad W := \sqrt{\det G}, \qquad (g^{ij}) := G^{-1}. \tag{A.2}$$

(where $i, j = 1, \cdots, n$ and det denotes the determinant).

Let $\mathcal{M}$ be a compact n-dimensional Riemannian manifold in C^∞ with the local parametrization $h : \mathbb{R}^n \to \mathbb{R}^{n+k}$. Let $\bar{h} := ah$ for the scaled manifold, along with other scaled values:

$$\begin{aligned} \partial_k\bar{h} &= a\partial_k h \quad (k = 1 \cdots n) \quad \text{implying } \bar{g}^{ij} = \frac{1}{a^2} g^{ij} \text{and} \\ \bar{W} &= a^2 W. \end{aligned}$$

Let u be a solution to

$$\Delta_h u = \frac{1}{W} \sum_{i,j} \partial_i (g^{ij} W \partial_j u) = -\lambda^2 u.$$

Then we have that

$$\Delta_{\bar{h}} u = \bar{W}^{-1} \sum_{i,j} \partial_i (\bar{g}^{ij} \bar{W} \partial_j u) = \frac{1}{a^2 W} \sum_{i,j} \partial_i (g^{ij} W \partial_j u) = -\frac{1}{a^2} \lambda^2 u. \qquad \blacksquare$$

A.3 Weyl's Asymptotic Growth of Eigenvalues

The normalization factor for Shape-DNA is an approximation to $4\pi/A$. This term derives from the asymptotics of the following theorem.

Theorem 2. *(Weyl-Asymptotic growth of eigenvalues)*
If D is a bounded region of $\mathbb{R}^d$ with piecewise smooth boundary B and if $0 < \lambda_1 \leq \lambda_2 \cdots$ is the spectrum and $N(\lambda) \leq \lambda$ where $N(\lambda)$ is the number of eigenvalues, counted with multiplicity, then

$$N(\lambda) \sim \frac{\omega_d vol(D) \lambda^{d/2}}{(2\pi)^d} \tag{A.3}$$

as $\lambda \to +\infty$. The volume of D is denoted by vol(D) and

$$\omega_d := \frac{\pi^{d/2}}{\Gamma(\frac{d}{2} + 1)} \tag{A.4}$$

is the volume of the unit disk in $\mathbb{R}^d$*. In particular,*

$$\lambda_n \sim 4\pi^2 \left(\frac{n}{\omega_d vol(D)} \right)^{2/d} \quad as\ n \to \infty. \tag{A.5}$$

A.4 Barycentric Interpolation

Barycentric interpolation [4] derives from the Lagrange interpolating polynomials with a slight modification leading to a faster computation. Define the barycentric weights for equispaced grid points by

$$w_j = (-1)^j \binom{n}{j}. \tag{A.6}$$

The 'second form of the barycentric formula' is defined as:

$$p(x) = \frac{\sum_{j=0}^{n} \frac{w_j}{x - x_j} u_j}{\sum_{j=0}^{n} \frac{w_j}{x - x_j}} \tag{A.7}$$

The weights are simplified from their general form due to equispaced grid points. Also note that the weights are independent of the values u_i which makes it efficient for changing u_i and interpolation points, x, fixed. This is the case for the closest point method, and allows us to write it in the form of a matrix when computing.

Appendix B

Lists

B.1 Database

The objects used in this thesis came from multiple sources listed here:

- Aim at Shapes Repository: http://shapes.aim-at-shape.net/
- The Stanford 3D Scanning Repository: https://graphics.stanford.edu/data/3Dscanrep/
- M. Reuter's Shape Database I: http://reuter.mit.edu/research/shapedb1/