Laplace-based Spectral Method for Point Cloud Processing

Dissertation

Presented in Partial Fulfillment of the Requirements for the Degree
Doctor of Philosophy in the Graduate School of The Ohio State
University

By

Chuanjiang Luo, M.S.

Graduate Program in Computer Science and Engineering

The Ohio State University

2013

Dissertation Committee:

Yusu Wang, Advisor
Mikhail Belkin
Tamal Dey
Patrick Woodward
Abstract

Laplace-Beltrami operator is a fundamental geometric object. It encodes a lot of intrinsic geometric information of an input Riemannian manifold. Over the past fifteen years, a great number of spectral methods for mesh processing applications have been proposed in the literature, relying on the eigen-analysis of the Laplace-Beltrami operator to carry out desired tasks. However, in practice, input data is often given as a point cloud and connectivity information such as a mesh structure is not always available. A point cloud based discrete Laplace operator becomes a natural choice for such tasks. In my thesis work, I study a spectral point cloud processing framework based on the Laplace-Beltrami operator. There are two aspects involved. First, given a point cloud data, how can we faithfully and efficiently compute the Laplace operator and/or its related spectral structure? Second, how can we use this spectral structure for data analysis? My thesis work make contributions in both aspects, providing theoretical understanding as well as developing algorithms for practical geometry processing applications.

Previously, there have been some discrete Laplace operators for point cloud data. Two most important ones are the Gaussian weighted graph Laplace operator and the so-called PCD-Laplace operator. However, the former only converges to the manifold Laplacian under the random uniform sampling condition. While the latter
converges under a more general sampling condition, it is not symmetric and has complex eigenvalue and eigenfunctions. We propose a new point cloud Laplace operator called Voronoi-Laplacian which is symmetric, convergent and works better under the so-called \((\epsilon, \delta)\) sampling condition (the price is that the sampling condition is slightly more stringent than that of the PCD-Laplace operator). It is a modification of the PCD-Laplace operator where we change the way the area weights are computed. This comes from the findings that weighted sum of a function with our modified Voronoi scheme converges to the true integral of the function over the manifold under \((\epsilon, \delta)\) sampling condition.

When data set is large, computation of its eigen-structure is expensive, limiting its applications. A natural strategy is to sub-sample the data and compute the structure on the decimated data. Unfortunately, this may smooth out important features in the original data. Here we propose a new heuristic method to simplify the data while preserve its spectral property. In particular, we aim to preserve the so-called heat kernel signature, which is based on the eigen-structure and almost as informative as the eigen-structure itself.

Once we have the spectral structure, the next question is how to retrieve useful geometric information from it. We present two such applications: one that deals with calculating gradient of functions and one that uniformizes the distribution of the point cloud or achieves a target distribution.

Specifically, in the first application, we initiate the investigation of computing gradients under a different metric on the manifold other than the original natural metric induced from the ambient space. It is easier to approximate gradients in the eigenspace for discrete inputs and the new approach is robust to noises in the input
function and in the underlying manifold. More importantly, we can easily smooth
the gradient field at different scales within this eigenspace framework. In the second
application, given a set of points approximating an unknown surface and a target
function, the goal is to compute a set of points that matches the target distribution.
We propose a simple and robust framework for this problem that is effective at both
local refinement and global adjustment of point distribution. Our approach uses the
information encoded in the graph Laplacian that is orthogonal to the mean-curvature
flow.
To my parents
Acknowledgments

My first and sincere appreciation goes to my advisor, Prof. Yusu Wang. This thesis would not be possible without her tremendous help throughout my years in this department. It is she who introduced me to the field of geometric computing and continuously guided and supported me in all stages of this thesis. My gratitude to her extends to every aspect of doing research, from topic selection to problem solving, to effective presentation. I have been extremely lucky to have an advisor who cared so much about my work, responded my questions so promptly and helped me to improve my technical skills so patiently. Her encouragement and insights not only permeate this thesis, but also my future career. For everything you’ve done for me, Dr. Wang, I thank you.

I would like to thank my committee members: Prof. Mikhail Belkin, Prof. Tamal Dey and Prof. Patrick Woodward. Their suggestions and comments helped a lot in making this thesis better. I would like to express my deep gratitude to Prof. Tamal Dey and Prof. Rephael Wenger, who are the other two faculty members in this research group, for all the help during my study in the department.

In addition, I would like to thank Prof. Jian Sun for his help in the two research projects we worked together. His research attitude and critical thinking skills serves as a role model to me.
My former and current cubicle-mates also deserve my sincere thanks: Joshua Levine, Kuiyu Li, Issam Safa, William Harvey, Oleksiy Busaryev, Andrew Slatton, Pawas Ranjan, Lei Wang, Xiaoyin Ge, and Suyi Wang. Their friendship and assistance has meant more to me than I could ever express.

Finally, I must mention my parents who have provided their unconditional support all through my life. This work is dedicated to them.
Vita

2004 .............................................. B.S. Electrical Engineering,
           USTC, China
2007 .............................................. M.S. Computer Science,
           Chinese Academy of Science, China
2007-2008 ................................. Ph.D. Student,
              Electrical and Computer Engineering,
                  The Ohio State University, OH
2011 .............................................. Summer Intern,
              Palo Alto Research Center, CA
2009-present ............................... Ph.D. Candidate,
              Computer Science and Engineering,
                  The Ohio State University, OH

Publications

Research Publications

Chuanjiang Luo, Jian Sun, Yusu Wang. Integral estimation from point cloud in d-
dimensional space: a geometric view. In J. Hershberger and E. Fogel, editors, Sym-

Chuanjiang Luo, Issam Safa, Yusu Wang. Approximating gradients for meshes and

Tamal Dey, Kuiyu Li, Chuanjiang Luo, Pawas Ranjan, Issam Safa, Yusu Wang. Per-

Chuanjiang Luo, Issam Safa, Yusu Wang. Feature-aware streamline generation of
planar vector fields via topological methods. In Comput. Graph., 36(6):754–766,
Fields of Study

Major Field: Computer Science and Engineering
# Table of Contents

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Abstract</strong></td>
<td>ii</td>
</tr>
<tr>
<td></td>
<td><strong>Dedication</strong></td>
<td>v</td>
</tr>
<tr>
<td></td>
<td><strong>Acknowledgments</strong></td>
<td>vi</td>
</tr>
<tr>
<td></td>
<td><strong>Vita</strong></td>
<td>viii</td>
</tr>
<tr>
<td></td>
<td><strong>List of Tables</strong></td>
<td>xii</td>
</tr>
<tr>
<td></td>
<td><strong>List of Figures</strong></td>
<td>xiii</td>
</tr>
<tr>
<td></td>
<td><strong>Chapters:</strong></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td><strong>Introduction</strong></td>
<td>1</td>
</tr>
<tr>
<td>1.1</td>
<td>Laplace-Beltrami Operator and Its Properties</td>
<td>3</td>
</tr>
<tr>
<td>1.2</td>
<td>Discrete Laplace Operator</td>
<td>5</td>
</tr>
<tr>
<td>1.2.1</td>
<td>Properties</td>
<td>6</td>
</tr>
<tr>
<td>1.2.2</td>
<td>Existing Discrete Laplace Operators</td>
<td>7</td>
</tr>
<tr>
<td>1.3</td>
<td>Spectral Methods</td>
<td>13</td>
</tr>
<tr>
<td>1.4</td>
<td>Contribution</td>
<td>14</td>
</tr>
<tr>
<td>1.4.1</td>
<td>Integral Estimation and Voronoi Laplacian</td>
<td>15</td>
</tr>
<tr>
<td>1.4.2</td>
<td>HKS-preserving Data Simplification</td>
<td>15</td>
</tr>
<tr>
<td>1.4.3</td>
<td>Eigen Gradient for Point Cloud</td>
<td>16</td>
</tr>
<tr>
<td>1.4.4</td>
<td>Point Cloud Uniformization and Density Adaptation</td>
<td>16</td>
</tr>
<tr>
<td>2</td>
<td>Integral Estimation and Voronoi Laplacian</td>
<td>18</td>
</tr>
<tr>
<td>2.1</td>
<td>Preliminaries</td>
<td>22</td>
</tr>
<tr>
<td>2.2</td>
<td>Voronoi Scheme</td>
<td>24</td>
</tr>
</tbody>
</table>
List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1 Relative error of integral estimation for 2500/5000/10000 points from a flat-2 torus $T^2$ embedded in $\mathbb{R}^4$. The last three columns are results based on the Monte Carlo integration, by using density estimation methods provided by the KDE toolbox, and by equal weight, respectively. All numbers are scaled by $10^{-4}$.</td>
<td>42</td>
</tr>
<tr>
<td>2.2 Relative error of integral estimation for 2500/5000/10000 skew-sampled points from flat-2 torus $T^2$ embedded in $\mathbb{R}^4$. All numbers are scaled by $10^{-4}$.</td>
<td>42</td>
</tr>
<tr>
<td>2.3 Relative error of integral estimated by various methods for molecular data. All numbers are scaled by $10^{-4}$.</td>
<td>43</td>
</tr>
<tr>
<td>2.4 Timing in seconds of various methods for 2500/5000/10000 points from flat-2 torus $T^2$ embedded in $\mathbb{R}^4$.</td>
<td>43</td>
</tr>
<tr>
<td>2.5 Timing in seconds for 2500/5000/10000 points from flat-2 torus $T^2$ embedded in $\mathbb{R}^4$, $\mathbb{R}^{20}$, $\mathbb{R}^{40}$ and $\mathbb{R}^{60}$.</td>
<td>44</td>
</tr>
<tr>
<td>2.6 Normalized $L_2/L_\infty$ error for different sampling on $S^2$, all numbers are scaled in percentage.</td>
<td>45</td>
</tr>
<tr>
<td>3.1 Each entry: the averaged $L_1$-error / Statistical correlation between the approximated HKS and ground truth. All original meshes are of size 30K, and they are simplified to 3K by either our method or QSlim.</td>
<td>56</td>
</tr>
<tr>
<td>5.1 Correlation/$L_\alpha$ error comparisons. The “Lloyd’s Input” is already close to target distribution, while input to our algorithm is not.</td>
<td>94</td>
</tr>
</tbody>
</table>
List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1 Angles involved in the calculation of cotangent Laplace operator</td>
<td>10</td>
</tr>
<tr>
<td>2.1 Illustrations for the proofs of Lemma 2 in (a) and of Lemma 3 in (b)</td>
<td>27</td>
</tr>
<tr>
<td>2.2 The molecular surface of the protein (PDB id: 1BRS, chain A) with sample points</td>
<td>39</td>
</tr>
<tr>
<td>2.3 (a) Estimated Integral for function $f = \sin x + 1$ under heat diffusion. (b) Estimated Integral for function $f = \sin^2 x + 1$ under heat diffusion.</td>
<td>44</td>
</tr>
<tr>
<td>3.1 Given a query Armadillo model that is pose-altered, incomplete, and partially scanned, we can first computes the heat kernel signature function at a certain scale, and then extract a set of HKS maxima (red dots) using persistent homology. Feature vectors computed at these maxima are then used to search for the most similar models, be it complete, partial, or incomplete, in a shape database. A few top matches are shown. The black curves are the boundary curves of either partial or incomplete models. Correspondence between segmentations of different models is shown with consistent coloring.</td>
<td>54</td>
</tr>
<tr>
<td>4.1 The gradient vector at a point on $\Omega = \Phi(M)$ is the projection of $V$ onto the tangent space at this point.</td>
<td>64</td>
</tr>
<tr>
<td>4.2 from (a) to (e) the gradient field reconstructed from 1500, 50, 20, 10, and 5 eigenvectors respectively. in (f) and (g) the magnitude and angular error plots with their standard deviation.</td>
<td>71</td>
</tr>
</tbody>
</table>
4.3 Top row from left to right, the ground truth gradient field for \( f(x, y, z) = (\sin(5x) + 5)(y^3 + 5)e^z \), gradient field by eigen method for 1% surface noise and 2% function noise, and gradient field by quadratic fit for the 1% surface noise and 2% function noise. The table shows the angular and magnitude errors for different noise configurations.

4.4 (a) Gradient field computed from 1000 eigenvectors (left) and 50 eigenvectors (right). (Note the critical point on the leftmost protrusion on the left image disappearing on the right image). (b) Gradient field computed from quadratic fitting (left), and the same field recomputed after being smoothed for 30 iterations (right).

4.5 Critical points computed from the mesh structure (left) and from the point cloud with 250 eigenvectors (right).

4.6 Left: critical points of the Connolly function computed from the mesh structure. Right: those computed from point clouds by the eigen-approach using 200 eigenvectors.

5.1 (a): 5K input points \( Q \) sampled from a random distribution far from the target distribution. The hidden surface is approximated by 50K points (not shown). (b): our output adapted to a target density distribution as shown in (c). The density map of our output is shown in (d). This is a hard case for relaxation-based methods due to the many tunnels. (e) shows the output density obtained by a PCD variant of the Lloyd relaxation method but with an input \( Q' \) that is already very close to the target distribution (see Section 5.3 for explanation).

5.2 Left: input 170K raw scan data; Middle: 10K downsampled and then uniformized; Right: zoom-in details of boundaries.

5.3 (a): Input are 5K points \( Q \) over 50K dense points \( P \): uniform noise of magnitude 1.5% of the diameter is added at each point. (b): Output points adapted to the target density function shown in (c). Note that these points are also smoothed by our algorithm. (d): Density map of our output points. The correlation of output density and target density is 0.99 and \( L_2 \)-error is 0.015.
5.4 (a) plots the $L_2$-error of the density of intermediate point set $Q$ as the parameter $t$ decreases in Algorithm 1. The points obtained when $t = t_0$ and $t = 0.5t_0$ are shown in (b) and (c): note that at $t = t_0$, while the distribution is already close to uniform (small $L_2$-error), points are yet well-separated. 88

5.5 (a) Input points and (c) output points on a human face. Small holes in (b) around nose are filled after uniformization in (d). 89

5.6 (a) Target density and (b) our output density function as achieved by the 10K output points shown in (c). 91

5.7 Experiments with sparse $P$. (a) Input 5K points $Q$. (b) Our output when $P = Q$. The output of [Huang et al. 2009] (c) when $P$ has 10K or (d) when $P$ has 40K points. (The output of [Huang et al.] when $P = Q$ is almost identical to input points.) 93

5.8 (a) (smoothed) mean curvature as target density, (b) our output density map, and (c) the output samples. The correlation between the target and our output distribution is 0.9875, and $L_2$-error of our output distribution is 0.0271. 97

5.9 Experiments when $P$ is sparse for the weighted LOP algorithm from [44] and the PCD Lloyd’s algorithm. Recall that our output from Figure 7 (b) of the main text already achieves a very uniform and regular output distribution (where $L_2$-error of our output is only 0.0231). 97

5.10 The DDA power-diagram and the plots for anisotropy. Each output (as shown in this figure) has 40K points, and is obtained using a dense point cloud $P$ with 100K points. Reducing the number of output points still maintain similar DDA power diagram. However, small features in the model start to be smoothed. 100
Chapter 1: Introduction

Laplace-Beltrami operator is a fundamental geometric object for a smooth Riemannian manifold. Mathematically, it is defined as the divergence of gradient of a function defined on the manifold. Laplace operator has a lot of nice properties. For example, it is self-adjoint, positive semi-definite. As such, it admits an orthonormal eigen-system. Specifically, the eigenfunctions of Laplace operator form an orthonormal basis of square integrable functions defined on the manifold. Laplace operator uniquely determines the Riemannian metric on the manifold and is isometry invariant. It contains rich intrinsic geometric information. Once we have the Laplace operator, we can potentially retrieve much information of the input manifold.

Over the past fifteen years, a great number of spectral methods for mesh processing have been proposed in fields such as computer graphics and machine learning, relying on the eigen-analysis of the Laplace-Beltrami operator to carry out desired tasks. Successful applications span a wide area in the geometry processing domain, including mesh filtering, parametrization, pose transfer, segmentation, reconstruction, re-meshing, compression, simulation, shape matching, retrieval and so on.

Although the Laplace-Beltrami operator is defined for smooth manifold mathematically, we usually do not have the underlying manifold in practice. We only have a discrete approximation of it, sometimes as a mesh, sometimes only as a point cloud.
data. For example, in computer graphics, a physical object may be digitized by using 3-D scanning equipments. In sensor networks, quantities such as temperatures or humidity are usually collected at each individual sensor. Both scenarios produce discrete point cloud data as output. In the discrete setting, we need to approximate the Laplace operator of the underlying manifold and its spectral structure. If the input data is a mesh, several discrete Laplace operators built on the mesh structure are available. In cases where the input is only a set of sampled points with no connectivity information, mesh reconstruction can still be one option. However, reconstructing the mesh structure is typically costly for high dimensional data, and many times not necessary. As such, point cloud Laplace operator becomes a natural choice for such tasks. In my thesis work, I study the spectral point cloud processing framework. There are two aspects involved. First, given a point cloud, how can we faithfully and efficiently compute the Laplace operator and/or its related spectral structure? Second, how can we use such spectral structure for data analysis?

For all the work described in this thesis, we assume that the input data reside on a low-dimensional manifold (while the dimension of its embedded space may be high), which is believed to hold mostly for many practical applications. One standard example of such data is the space of images of an object taken under fixed lighting conditions with a rotating camera. While the dimension of the ambient space is proportional to the number of pixels which can be millions, the intrinsic dimension of the image space is only two (i.e, rotation angles of the camera). Although some experiments are done using surfaces embedded in 3D for illustration purpose, all our methods can be applied to high dimensional data with little modification.
The rest of the chapter is organized as follows: in Section 1.1 we will briefly provide some background for the Laplace operator and spectral methods. In Section 1.2, we will look at the existing discrete Laplace operators. In Section 1.3, we will give a quick review of spectral mesh processing applications. Finally, Section 1.4 provides an overview of the contributions of this thesis in both theoretical and application-based fields.

1.1 Laplace-Beltrami Operator and Its Properties

The Laplace-Beltrami operator (manifold Laplacian) is a fundamental geometric object associated to a Riemannian manifold and has many desirable properties. As the central object of this thesis, we will first recall the basics of Laplace-Beltrami operator.

Let \( f \) be a real-valued function, with \( f \in C^2 \), defined on a Riemannian manifold \( M \). The Laplace-Beltrami operator is defined as the divergence of gradient of a function defined on the manifold \([72]\):

\[
\Delta f = \text{div}(\nabla f)
\] (1.1)

where \( \text{div} \) and \( \nabla \) are the divergence and gradient operators respectively. If the underlying space is a Euclidean space \( \mathbb{R}^d \), then the Laplace-Beltrami operator is simply the summation of all the unmixed second order partial derivatives in the Cartesian coordinate:

\[
\Delta f = \sum_{i=1}^{d} \frac{\partial^2 f}{(\partial x_i)^2}
\] (1.2)
The Laplace-Beltrami operator is determined solely by the Riemannian metric \[86\] and is therefore an intrinsic property of the manifold. The heat kernel can be constructed from the eigenvalues and eigenfunctions of the Laplace-Beltrami operator, conversely, it fully determines the Riemannian metric (up to a scale).

The Laplacian eigenvalue problem is defined as

\[ \Delta f = -\lambda f. \]  

(1.3)

Since Laplace-Beltrami operator is self-adjoint and positive semi-definite \[72\], it admits an orthonormal eigen-system \((\lambda_i, \phi_i)\), which is a basis of the space of square integrable functions defined on the manifold, with \(\Delta \phi_i = \lambda_i \phi_i, \lambda_0 \leq \lambda_1 \ldots \lambda_i \leq \lambda_{i+1} \ldots \leq +\infty\). Eigenfunctions of the Laplace-Beltrami operator are analogous to Fourier harmonics. Eigenvalues intuitively give the frequency of the corresponding harmonics (eigenfunction).

For compact shapes, the spectrum of Laplace operator is discrete, that is, there exists a countable set of solutions to the equation \[1.3\]. Since the Laplace-Beltrami operator is an intrinsic property of the manifold, its spectrum is isometry invariant and can be used as a signature of the manifold \[73\]. Yet, in general, a Riemannian manifold cannot be determined solely by its spectrum. In other words, one cannot hear the shape of the drum \[70\ \ 47\]. However, one can imply important geometric properties from the spectrum. It is known that the area of two iso-spectral surfaces is equal, which implies that the area can be heard from the spectrum. The total Gaussian curvature and Euler characteristic are also determined by the spectrum. The counter-example of isospectral but non-isometric manifolds were often seem to
be somewhat artificial and appear to be exceptional. Therefore, the spectrum of the Laplace-Beltrami operator has significant discrimination power.

Properties The Laplace-Beltrami operator and its spectrum have several important properties [83].

1. Constant eigenfunctions: $\Delta f = 0$ for any $f = \text{const}$;
2. Symmetry: $\langle \Delta f, h \rangle = \langle h, \Delta f \rangle$;
3. Locality: $\Delta f(x)$ only depends on an infinitesimally small neighborhood of $x$.
4. Positive semi-definiteness: $\langle \Delta f, f \rangle \geq 0$.
5. Linear precision: $\Delta f = 0$ whenever the manifold $M$ is part of the Euclidean plane, and $f = ax + by + c$ is a linear function on the plane.
6. Maximum principle: harmonic functions (those for which $\Delta f = 0$ in the interior of $M$) have no local maxima (or minima) at interior points.

The locality property implies that the action of the Laplace-Beltrami operator is local, and a change in the value of the function $f$ at a point will not influence the value of $\Delta f$ at a distant point. The linear precision property stems from the fact that in the Euclidean case, the Laplace-Beltrami operator is the summation of all the unmixed second other derivatives, thus $\Delta f = -(f_{xx} + f_{yy}) = 0$.

1.2 Discrete Laplace Operator

In practice, we usually do not have the underlying manifold, we only have a discrete approximation of it, sometimes as a mesh, sometimes only as point cloud data. In order to compute the spectrum of the Laplace-Beltrami operator, the continuous
operator has to be approximated by some discrete Laplace operator. We assume the manifold $M$ is sampled at $N$ points $x_1, \ldots, x_N$. A function on the manifold is given at each points $f_i = f(x_i)$ for $i = 1, \ldots, N$. The discrete Laplacian is typically defined as a linear operator of the form

$$(L_M f)_i = \sum_{j=1}^{N} \omega_{ij} (f_i - f_j)$$

(1.4)

1.2.1 Properties

The properties for discrete Laplace operator and its spectrum \cite{33} are motivated from continuous case with additional geometric and physical intuition, which we will discuss in this section.

1. Symmetry: $\omega_{ij} = \omega_{ji}$. This property is motivated by the fact that real symmetric matrices exhibit real eigenvalues and orthogonal eigenvectors.

2. Locality: In case connectivity is available, weights are associated to mesh edges (1-ring support), so that $\omega_{ij} = 0$ if $i$ and $j$ do not share an edge in $M$.

3. Linear precision: $(Lf)_i = 0$ at each interior vertex whenever $M$ is straight-line embedded into the plane and $f$ is a linear function on the plane, point-sampled at the vertices of $M$.

4. Positive weights: $w_{ij} \geq 0$ whenever $i \neq j$. Additionally for each vertex $i$ there exists at least one vertex $j$ such that $\omega_{ij} \geq 0$.

5. Positive semi-definiteness: $L$ is symmetric positive semi-definite with respect to the standard inner product and has a one-dimensional kernel.
6. Convergence: $L_n \to \Delta$, in the sense that solutions to the discrete Dirichlet problem, involving $L_n$, converge to the solution of the smooth Dirichlet problem, involving $\Delta$, under appropriate refinement conditions and in appropriate norms.

The quality of a discrete Laplace operator can be judged by how many of the desired property it satisfies, with an ideal discrete Laplace operator satisfies all these properties. However, as pointed out by Wardetzky et al. [83], it is theoretically impossible to satisfy all these properties at the same time, and thus the ideal discretization scheme does not exist. This also explains the large varieties of existing discrete Laplace operator, each satisfying a subset of the above properties and is suitable to certain kind of geometric application. We will briefly review them in the next section.

### 1.2.2 Existing Discrete Laplace Operators

The discrete Laplace operators can be divided into two categories. Mesh Laplace operators can be built with the connectivity information given. Point cloud Laplace operators can be built without any connectivity information, only sampled points [87].

Existing Mesh Laplace operators can be grouped as either combinatorial mesh Laplace operators or geometric Laplace operators depending on whether they are determined by the connectivity information only or explicitly encode geometric information. Spectral structure of combinatorial mesh Laplace operators is more sensitive to the changes in mesh connectivity.
Given a mesh structure \((V, E, P)\), where \(V\) is the set of vertices, \(E\) is the set of edges and \(P\) is the set of coordinates of each point, we will first review combinatorial mesh Laplacian as follows:

1. **Graph Laplacian.** The adjacency matrix \(W\) is given by

\[
W_{ij} = \begin{cases} 
1 & \text{if } (i, j) \in E, \\
0 & \text{otherwise.}
\end{cases}
\]  

The degree matrix \(D\) is defined as

\[
D_{ij} = \begin{cases} 
d_i = |N(i)| & \text{if } i = j, \\
0 & \text{otherwise.}
\end{cases}
\]

where \(N(i)\) is the set of vertices adjacent to vertex \(i\), \(d_i\) is the degree of vertex \(i\). \(W\) and \(D\) are \(n \times n\) matrices, where \(n = |V|\). The graph Laplacian matrix \(K\) is defined as \(K = D - W\). The operator \(K\) is also known as the Kirchoff operator, as it has been encountered in the study of electrical networks by Kirchoff. In that context, the adjacency matrix \(W\) is referred as the conductance matrix. If a specific distribution of geometries is assumed, then the graph Laplacian \(K\) is optimal in terms of capturing the most spectral power for a given number of leading eigenvectors. This result is based on the idea that the spectral decomposition of a mesh signal of a certain class is equivalent to its PCA, when this class is equipped with the specific probability distribution. However, it tends to exhibit more sensitivity towards vertex degrees, resulting in artifacts in meshes reconstructed from a truncated spectrum.

2. **Tutte Laplacian** It was used by Taubin \[81\], defined as

\[
T = D^{-1}K,
\]  

8
thus

\[
T_{ij} = \begin{cases} 
1 & \text{if } i = j, \\
-1/d_i & \text{if } (i, j) \in E, \\
0 & \text{otherwise}.
\end{cases}
\] (1.8)

\(T\), while possessing useful properties, is not a symmetric matrix. However, it appears to possess more desirable properties in mesh reconstruction and when it is applied to spectral graph drawing.

3. **Normalized graph Laplacian** The symmetrized version of Tutte Laplacian is defined as:

\[
Q = D^{-1/2}K D^{-1/2},
\] (1.9)

with

\[
Q_{ij} = \begin{cases} 
1 & \text{if } i = j, \\
-1/\sqrt{d_id_j} & \text{if } (i, j) \in E, \\
0 & \text{otherwise}.
\end{cases}
\] (1.10)

This operator does not have a zero row sum. For spectral processing, the main utility of \(Q\) is to provide a symmetric matrix to facilitate the computation of eigenvectors of \(T\), as it has the same spectrum with \(T\).

If discrete mesh Laplace operators not only depend on the connectivity but also encode the geometric information, then they are called geometric mesh Laplacian as follows:

1. **Cotangent Laplacian** The original cotangent Laplace operator is defined as

\[
Cf = \sum_{j \in N(i)} \frac{1}{2}(\cot \alpha_{ij} + \cot \beta_{ij})(f_i - f_j),
\] (1.11)
where the angles $\alpha_{ij}$ and $\beta_{ij}$ are subtended by the edge $(i, j)$ as shown in Figure 1.1. If $(i, j)$ is a boundary edge, the $\cot\beta_{ij}$ term vanishes. This corresponds to imposing von Neumann boundary conditions. The formula (1.11) was obtained by noting that on a triangular face $T$ with vertices $p_i, p_j, p_k$, we have

$$\nabla \varphi_k \cdot \nabla \varphi_j = -\frac{1}{2a_T} \cot \angle p_i,$$

where $\varphi_j$ is the nodal linear basis function (“hat function”) center at $p_j$ and $a_T$ is the area of $T$. $C$ is a symmetric matrix so it possesses the important self-adjoint property. However, $Cf$ yields nodal values that represent the integral of $\Delta f$ over a neighborhood, rather than a point sample.

2. **Variant of Cotangent Laplacian** The solution to the weakness of cotangent scheme is to divide the area of the local neighborhood thus yielding values that are local spatial averages of the Laplacian [62]. If $D$ is the diagonal matrix whose entries are $A_i$, the area of $x_i$, then the Laplacian proposed is $Y = D^{-1}C$. However, $Y$ is not a symmetric matrix, so the self-adjoint property is sacrificed. By defining the inner product as a weighted inner product with the area
weight, the eigenfunctions of $Y$ are orthogonal with respect to this inner product. Cotangent scheme and its variants require meshes to have acute angles, for degenerate meshes, re-meshing may be required. Although cotangent scheme provides weak convergence (in the sense of inner product), it does not have standard point-wise convergence.

3. **FEM Laplacian** Let $\varphi_i$ be the finite element basis function. Define $B$ as $B_{ij} = \int_M \varphi_i \varphi_j da$, which is the mass matrix in finite element method. If $p_i$ and $p_j$ are neighbors, then $B_{ij}$ is $\frac{1}{12}$ of the area of the triangles adjacent to edge $(i, j)$. The diagonal entries $B_{ii}$ are $\frac{1}{6}$ of the area of the triangles adjacent of $p_i$. All other entries are 0. The FEM Laplacian is defined as $F = B^{-1}C$. This matrix is self-adjoint with respect to the inner product $(f, g)_B = f^T B g$, which also renders its eigenvectors orthogonal. Although this is an improvement upon $Y$ and $C$, $Y$ produces eigenvalues and eigenfunctions which are more robust with respect to variations in the mesh used to represent a surface.

4. **Diffusion based mesh Laplacian** This mesh Laplacian [15] is based on heat diffusion and functional approximation of the Laplace-Beltrami operator. It is defined by

$$L_t f(x_i) = \frac{1}{t(4\pi t)^{d/2}} \sum_{j \in N(i)} A_j e^{-\frac{||x_i - x_j||^2}{4t}} (f_i - f_j)$$

(1.13)

where $t$ is a parameter which indicates the time for the heat diffusion process, $d$ is the intrinsic dimension of the mesh, and $A_j$ is $\frac{1}{d+1}$ of the total volume of all d-simplices incident to the vertex $v_j$. This Laplace operator provides the first point-wise convergence result to the true Laplacian for general meshes as the input mesh approximates a smooth manifold better. This result can be
extended to high dimensional manifold. It also provides accurate approximation of the Laplace operator under various conditions, such as noisy data input, and different sampling conditions.

After summarizing the mesh Laplace operator, we will turn to point cloud Laplace operator. There are still limited work on point cloud Laplace operator as listed below. Two most important ones are the Gaussian weighted graph Laplacian and the so-called PCD-Laplacian.

1. **neighborhood graph based point cloud Laplacian** This is the first point cloud Laplace proposed in [9] forming the basis for later diffusion based mesh and PCD Laplacian framework [15, 16, 60]. Specifically, if $x_1, \ldots, x_N$ are data points on the manifold,

$$L_t f(x_i) = \frac{1}{t} [f(x_i) - \frac{1}{k} (4\pi t)^{-\frac{n}{2}} \sum_{x_j, 0 < ||x_j - x_i|| < \epsilon} e^{-\frac{||x_i - x_j||^2}{4t}} f(x_j)], \quad (1.14)$$

where $k$ is the number of neighboring points in the $\epsilon$-radius ball of $x_j$. This graph Laplacian converges to the Laplace-Beltrami operator on the manifold if the point cloud is draw from uniform distribution. For point clouds from arbitrary probability distribution, it converges to a weighted Laplacian or manifold Laplacian up to a multiplicative factor.

2. **PCD-Laplacian** In [16], Belkin et al. proposed the first discrete Laplacian for general point cloud data with convergence guarantee for a non-statistical setting. They first approximate a local tangent plane for every point. For all points within a fixed radius neighborhood of $x_i$, project them onto the tangent plane and then build the Delaunay triangulation of the projected points.
The PCD-Laplacian is obtained by substituting the area weight in Eqn 1.13 with $A_{p_j}$, where $A_{p_j}$ is the volume of all-dimensional simplices in the Delaunay triangulation in the tangent plane. Although this operator provided the first convergence result for arbitrary point cloud, it is not symmetric and have complex eigenvalues and eigenfunctions.

1.3 Spectral Methods

Spectral methods have been widely used in many research fields including graph theory, vision, and machine learning, and have recently received great attention in geometric mesh processing, from shape matching, segmentation, optimization, to parametrization and meshing. Most spectral methods for mesh processing have a basic framework in common. First, an appropriate discrete Laplace operator is calculated in the matrix form. Then an eigen-decomposition of the matrix is performed, that is, its eigenvalues and eigenvectors are computed. Resulting structures from the decomposition are employed in a problem specific manner to obtain a solution.

Existing spectral methods can be grouped to several categories based on either the discrete Laplace operator used or the eigen structures used. An extensive review in all these categories is beyond the scope of this section. For more details, please refer to [87]. Below we give a few representative example.

Since eigenfunctions form a basis for functions defined on the manifold, any function can be represented as a linear combination of Laplace eigenfunctions. A natural way to smooth a function defined on the manifold is to remove the eigenfunction components corresponding to high eigenvalues. This also applies to the manifold itself since it can be represented simply as the coordinate function [25]. Also, the isometry
invariant property of Laplace operator makes it a good candidate to compare different poses of the same shape. In shape retrieval, people use a spectral embedding to incorporate any intrinsic measure on a shape in order to obtain useful invariance properties \[66\]. Spectral embedding can also help clustering, In \[65\], the entries of the first \(k\) eigenvectors corresponding to the largest eigenvalues of a normalized affinity matrix are used to obtain the transformed coordinates of the input points. Points with high affinities tend to be grouped together in the spectral domain and simple clustering algorithm can be applied. In addition, eigenvalues of the Laplace operator form the spectrum of the manifold and can be used to estimate many important quantities of the manifold, such as manifold surface volume and total scalar curvature. Many geometric invariants of the manifold can be reconstructed from its Laplace-Beltrami operator. For example, for a 2-manifold in \(\mathbb{R}^3\), its mean curvature flow can be computed by applying the Laplace-Beltrami operator to the coordinate \(x, y, z\), considered as functions on the manifold.

Much success has already been achieved in applying a spectral framework for various geometric processing tasks when an input mesh structure is given. Nevertheless, discrete point cloud Laplace operator has only been studied recently and has limited applications so far. It would be helpful to develop a spectral point cloud processing framework to extend existing spectral mesh processing work.

1.4 Contribution

The purpose of this thesis is to study the eigen structure of discrete Laplace operator in the point cloud setting and to explore possible applications in the field of
geometry processing. In Chapters 2 \[59, 60\] we discuss the problem of integral estimation for point cloud data, based on which a new discrete Laplace operator (Voronoi Laplace operator) is proposed. In Chapter 3 \[30\] we present our data simplification work aiming to preserve the spectral structure. Later on, in Chapters 4 \[60\] and 5 we study some applications to gradient estimation and data uniformization respectively.

### 1.4.1 Integral Estimation and Voronoi Laplacian

We start with a problem on how to estimate integral of a function from its values on sampled points. The continuous integral can be approximated by a discrete weighted summation and our goal is to estimate the weight for each point. We give two weighting schemes: Voronoi weighting scheme and left eigenvector weighting scheme. We show that the Voronoi weighting is convergent to the true integral under a mild sampling condition. While we do not have theoretical guarantee of convergence for the left eigenvector weighting scheme, we show it is more stable under discrete heat diffusion.

Based on the findings in the estimation of integral, we propose a new discrete Laplace operator called Voronoi-Laplacian. The new operator is symmetric, has real eigenvalues and eigenfunctions. Therefore it is suitable for many geometry processing applications. Moreover, in our experiments, the new discrete Laplace operator has better performance than previous point cloud Laplace operators.

### 1.4.2 HKS-preserving Data Simplification

Next we study the problem of data simplification. When the input data set is large, computation of its eigen-structure is expensive, limiting its applications. A natural thing to do is to sub-sample the data and compute the structure on the
decimated data. Unfortunately, this may smooth out the original data and lose important features. Here we propose a new method to simplify the data while trying to preserve its spectral property. In particular, we aim to preserve the so-called heat kernel signature, which is based on the eigen-structure and almost as informative as the eigen-structure itself. Hence we propose a novel sub-sampling technique in line with the definition of the $HKS$ which selects vertices where heat accumulates locally after a small time period.

### 1.4.3 Eigen Gradient for Point Cloud

In the remaining chapters, we shift our focus from the spectrum itself to applications of the Laplace operator. In Chapter 4, we study the problem of estimating gradient of a function defined on a point cloud data. We achieve this by computing gradients under a different metric on the manifold other than the original natural metric induced from the ambient space. The original point cloud is first transferred to an eigenspace and the computation is calculated there. It is easier to approximate gradients in the eigenspace and it is robust to noises in the input function and in the underlying manifold. More importantly, we can easily smooth the gradient field at different scales within this eigenspace framework.

### 1.4.4 Point Cloud Uniformization and Density Adaptation

In the second application, given a set of points approximating an unknown surface and a target function, the goal is to compute a set of points that is either a uniform sampling of the underlying surface or a sampling that matches the target distribution. We propose a simple and robust framework for this problem that is effective at both local refinement and global adjustment of point distribution. Our approach uses the
information encoded in the graph Laplacian that is orthogonal to the mean-curvature flow. It is simple, effective, and robust to noise and small holes present in input points. One can use it to improve the quality of sample points.
Chapter 2: Integral Estimation and Voronoi Laplacian

Integration over a domain, such as a Euclidean space or a Riemannian manifold, is a fundamental problem across a broad range of scientific areas. Its importance is perhaps best reflected by the fact that it is one of the most intensively investigated topics in numerical analysis. Indeed, data analysis routinely boils down to collecting certain properties over a domain of interest. For example, in Bayesian net, integration is one of the main components behind information inference. In graphics and visualization, histograms, typically obtained by integrating certain quantities over a domain, are widely used to characterize input shapes.

As specified in Chapter 1, in this problem we assume that the underlying domain is only accessible through a set of discrete sample points; and the function to be integrated is only given by values at these points; The input data is a low-dimensional manifold embedded in a high-dimensional space. Integral estimation from point clouds falls into the broad category of information recovery and data analysis over low-dimensional manifold structure from point clouds, which has recently received great attention from many research areas, including machine learning and computational geometry \[3, 55, 10, 56, 48\]. Reconstructing manifold structure is typically costly for high dimensional data, and many times not necessary. Our goal is to estimate
integral information faithfully from the point clouds data without reconstructing the entire underlying manifold structure.

Related work. Previously, integral estimation from point clouds is usually conducted in a statistical setting, where the input point samples are assumed to be drawn from certain probability distribution. The most popular method to estimate integral in such case is the Monte Carlo integration [40]. In its simplest form, this method simply estimates the integral as the average of the function values at sample points under the assumption that the samples are uniformly distributed. In the case where the uniform assumption is not valid, one can use density estimation technique to obtain statistically guaranteed results. The theoretical guarantee (i.e., closeness to the ground truth) of the Monte Carlo integration method is usually derived based on the Law of Large Numbers. Note that the Monte Carlo scheme only estimates the integral up to a scaling factor, which is the volume (surface volume) of the manifold. Volume estimation (which can be considered as the integral of the constant function) is itself a hard problem in high dimensions [34].

It is often much easier to sample the ambient space $\mathbb{R}^d$ according to some distribution than sampling a submanifold of it, even if the submanifold is given explicitly. Li et al. [53] convert the estimation of the area of a surface in $\mathbb{R}^3$ to the estimation of the number of the intersections of lines in $\mathbb{R}^3$ with this surface, based on the so-called Cauchy-Crofton formula [77]. Since a line in $\mathbb{R}^3$ has four independent parameters, one can uniformly sample the space of lines in $\mathbb{R}^3$ by a uniform sampling of this 4-dimensional line space. Liu et al. [57] adapt this idea to estimate the surface area directly from sample points. However, it is not clear how to extend this strategy to
the high dimensional case since the co-dimension of the sub-manifold in \( \mathbb{R}^d \) may be much bigger than one, whence the probability of a line intersecting the sub-manifold is zero.

From a geometric point of view, the most natural way to estimate integral from PCD is perhaps to first reconstruct a mesh from the sample points which approximates the (metric of the) underlying sub-manifold. Indeed, once a mesh approximation is given, it is shown in [15] that one can then use the area of the mesh elements incident to each point as the weight for this point, and compute the integral as a weighted sum. However, although efficient algorithms for converting a point cloud from surface in \( \mathbb{R}^3 \) into a mesh have been developed [3, 4, 27], the mesh construction problem is rather expensive in high dimensions. The best existing such algorithm [18, 28] takes time exponential in the dimension of the ambient space, which, in most of the applications, is much higher than the intrinsic dimension of the manifold. Hence it is highly desirable to have a scheme that directly operates on the sample points, with running time depending only mildly on the ambient dimension, which we aims at developing.

**New work.** In this chapter, we consider the problem of integral estimation from PCD in a non-statistical setting where input data is not necessarily randomly sampled. We approach it from a geometric point view, and develop a new algorithm that operate on the PCD directly, with running time mostly depending on the intrinsic dimension of the underlying manifold. Specifically, given a point cloud \( P \) sampled from a \( d \)-dimensional manifold \( M \) embedded in \( \mathbb{R}^m \), we approximate the integral of \( f \) over \( M \) by a weighted sum \( \sum_{p \in P} w(p)f(p) \), and propose two novel weighting schemes to
compute \( w(p) \)'s. Both new schemes rely on local information from each input point, and can be implemented in time exponential only in the intrinsic dimension of \( M \), which is in some sense the best one can hope for in the deterministic setting [32]. These are the first results of this sort for this important problem in a general setting.

Our first scheme is called the Voronoi weighting scheme. It reconstructs a local patch around each input point, and then computes the weight of this point based on it. Although these local patches are not consistent and can not be stitched together to form a global mesh approximating the underlying manifold, we show that they approximate the volume measure of the underlying manifold, and thus produce a convergent estimation under some mild assumption on the sample points and the integrand. We present explicit error bound of our estimation depending on the sampling conditions of input points.

As a byproduct of the Voronoi weighting scheme, we propose a new discrete Laplace operator called Voronoi-Laplacian. Previously, there have been some discrete Laplace operators for point cloud data. Two most important ones are Gaussian weighted graph Laplacian and PCD-Laplacian. However, although the former one is light weighted, it only converges to the manifold Laplacian under uniform sampling condition. While the latter one converges under a more general sampling condition, it is not symmetric and has complex eigenvalues and eigenfunctions. Our Voronoi-Laplacian is symmetric, convergent, has real eigenvalues and eigenfunctions and works better under \((\epsilon, \delta)\) sampling condition. Therefore it is more suitable for many geometry processing applications. The cost is that \((\epsilon, \delta)\) sampling condition is slightly more strict than the sampling condition of PCD-Laplacian.
The second scheme is called the *Principal Eigenvector weighting scheme*. The method falls into the same framework as the recent development of PCD-Laplace operator [16]. Although we are not able to establish the convergent result for this scheme, it preserves the global property that heat diffusion on manifolds is an integral invariant process in the discrete setting, which is an important property given that the heat diffusion has been widely used to smooth functions in many application areas.

### 2.1 Preliminaries

Consider a smooth orientable compact manifold $M$ of dimension $k$ that is isometrically embedded in some Euclidean space $\mathbb{R}^d$, and is equipped with a natural metric induced from the Euclidean metric. The *medial axis* of $M$ is the closure of the set of points in $\mathbb{R}^d$ that have at least two closest points in $M$. For any $p \in M$, the *local feature size at $p$*, denoted by $\text{lfs}(p)$, is the distance from $p$ to the medial axis. The *reach* of $M$, denoted by $\rho$, is the infimum of the local feature size at any point in $M$. In this chapter, we assume that the manifold $M$ has a positive reach. We also assume that both $d$ and $m$ are known a priori. Note many algorithms have been developed to estimate the intrinsic dimension from point cloud with guarantees [21, 22, 29, 37].

Let $P$ denote a set of sample points on $M$. We say that $P$ is an *$\varepsilon$-sampling* of $M$ if $p \in M$ for any $p \in P$, and for any point $x \in M$, there exists $q \in P$ such that $\|x - q\| \leq \varepsilon \rho$. $P$ is an *$(\varepsilon, \delta)$-sampling* of $M$ if $P$ is an *$\varepsilon$-sampling* of $M$ and that any two points in $P$ are at least $\delta \rho$ away from each other.

We use $T_p$ to denote the tangent space of $M$ at $p$, and $B(p, r)$ denotes the $d$-dimensional ball centered at $p$ with radius $r$. In what follows, we use $\|x - y\|$ to denote the Euclidean distance between two points $x$ and $y$, and $d_M(x, y)$ denote the
geodesic distance between them on $M$ for $x, y \in M$. For a point $x$ and a set $Y$, the distance between them is defined as $d(x, Y) = \inf_{y \in Y} \|x - y\|$. Integral estimation. Given a set of points $P = \{p_1, \ldots, p_n\}$, a function $f : M \rightarrow \mathbb{R}$ is represented as a vector $f = [f_1, \ldots, f_n]$, where $f_i = f(p_i)$ is its value at point $p_i$. We approximate the integral $\int_M f$ in a standard way by using a weighted sum: $\sum_i \omega_i f_i$, which can be rewritten as the vector inner product $\langle \omega, f \rangle$ for $\omega = (\omega_1, \ldots, \omega_n)$. Hence developing an integral estimation simply means to construct a weighting scheme $\omega$.

We remark that one can consider the integral $\int_M f d\nu$ as the inner product between the function $f$ and the volume measure $d\nu$. From this point of view, it is natural to approximate an integral as an inner product between vectors in the discrete setting, and the weighting vector $\omega$ should describe the volume measure at each sample point. This also implies that the weighting vector should be able to be constructed by local information, which we collect by constructing a local patch around each input point.

Local patches. Specifically, we build an approximation of the local manifold patch $M_\eta = M \cap B(p, \eta)$ for each $p \in P$ for some parameter $\eta \leq \rho/2$ as follows. Set $P_\eta = P \cap B(p, \eta)$.

1. Construct from $P_\eta$ a $d$-dimensional subspace $\tilde{T}_p$ to approximate the tangent space $T_p$ at $p$, using the algorithm in [16, 41].
2. Project the set of points $P_\eta$ onto $\tilde{T}_p$.
3. Let $\Pi$ denote the projection from $M_\eta$ onto $\tilde{T}_p$. Build the Delaunay triangulation $K_\eta$ of $\Pi(P_\eta)$ on $\tilde{T}_p$.

$K_\eta$ is a $d$-dimensional triangulation. The projection map $\Pi : M_\eta \rightarrow \tilde{T}_p$ is injective for $\eta \leq \rho/2$. Let $\Phi : \Pi(M_\eta) \rightarrow M_\eta$ be its inverse. Below we cite some known

23
results relating $T_p$, $\tilde{T}_p$ and $M$ \cite{15, 16}. In particular, Lemma 1 (3) guarantees that the volume measure on $K_\eta$ is indeed close to the measure on $M_\eta$ as the Jacobian of this projection map is close to 1. Hence we can use $K_\eta$ as a faithful approximation of $M_\eta$ and estimate weights for integral from $K_\eta$.

**Lemma 1** (\cite{15, 16}). Given a point cloud $P$ that $\varepsilon$-samples $M$, for a fixed point $p$, perform the algorithm introduced above with parameter $10\varepsilon\rho \leq \eta \leq \rho/2$.

(1) Given two points $p, q \in M$, let $d = \|p - q\| < \rho/2$. Then we have that $d \leq d_M(p, q) \leq d + O(d^3)$.

(2) The approximate tangent space $\tilde{T}_p$ is close to $T_p$ and the angle between them is $\angle(T_p, \tilde{T}_p) = O(\eta/\rho)$.

(3) The Jacobian of the map $\Pi$ and its inverse $\Phi$ at any point $x \in M_\eta$ are bounded respectively by:

$$1 - O\left(\frac{\|x - p\|^2}{\rho^2} + \frac{\eta}{\rho}\right) \leq J(\Pi)|_x \leq 1,$$

and

$$1 \leq J(\Phi)|_{\Pi(x)} \leq 1 + O\left(\frac{\|x - p\|^2}{\rho^2} + \frac{\eta}{\rho}\right).$$

Furthermore, the smallest eigenvalue of $J(\Pi)|_x$ is lower-bounded by $1 - O(\|x - p\|^2/\rho^2 + \eta/\rho)$, while the largest eigenvalue of $J(\Phi)|_{\Pi(x)}$ is upper-bounded by $1 + O(\|x - p\|^2/\rho^2 + \eta/\rho)$.

**2.2 Voronoi Scheme**

We choose and fix the parameter $\eta = c\varepsilon\rho$ for some constant $c > 10$. For each point $p \in P$, we construct the local patch and its triangulation $K_\eta$ as described before, and consider the dual Voronoi diagram of $K_\eta$ in $\tilde{T}_p$. Let $\hat{V}(p)$ denote the
Voronoi cell of \( p \) in \( \hat{T}_p \). Take the area of \( \hat{V}(p) \) as the weight of \( p \), denoted by \( \hat{A}_p \). The Voronoi weighting scheme is simply setting \( \omega_i = \hat{A}_{p_i} \), and we compute the integral as \( \langle \omega, f \rangle = \sum_i \hat{A}_{p_i} f(p_i) \).

Now let \( \text{Vor}(P) \) be the geodesic Voronoi diagram of \( P \) on \( M \). Let \( V(p) \) denote the Voronoi cell of \( p \) in \( \text{Vor}(P) \) and \( A_p \) its area. Intuitively, \( A_p \) serves as a good weighting scale for the sample point \( p \) and can be used to approximate the integral. Hence to show the theoretical guarantee of our weighting scheme, the goal is to bound the relation between \( A_p \) (which cannot be computed) with the area \( \hat{A}_p \) in the approximate tangent space \( \hat{T}_p \). Note that the Voronoi neighbors of \( p \) on \( M \) and those in \( \hat{T}_p \) are not necessarily the same, and one can in fact construct examples where they induce arbitrarily different areas for some \( \epsilon \)-sampled point sets. Hence we need to consider a sampling condition which is more restricted than the \( \epsilon \)-sampling condition — specifically, we assume that \( P \) is an \( (\epsilon, \delta) \)-sampling of \( M \), and the convergence result will be achieved as long as \( \delta = \Omega(\epsilon^{3/2-\xi}) \) for any \( \xi > 0 \) (the smaller \( \delta \) is, the more general the sampling condition is). Recall that we have chosen \( \eta = c\epsilon \rho \).

**Lemma 2.** Assume \( P \) is an \( (\epsilon, \delta) \)-sampling of \( M \) with sufficiently small \( \epsilon < 1/20 \). Consider the point \( p \in P \) and any point \( q \in P \cap B(p, \eta) \). Let \( \hat{S} \) be the bisector between \( p \) and \( \Pi(q) \) on \( \hat{T}_p \). For any \( x \in M_\eta = M \cap B(p, \eta) \) with \( d_M(x, p) = d_M(x, q) \), we have that \( d(\Pi(x), \hat{S}) = O(\epsilon^{3}\rho/\delta) \).

**Proof.** Denote \( \tilde{y} = \Pi(y) \) for any \( y \in M_\eta \) (note that \( \hat{p} = p \)). Now take an arbitrary point \( x \in M_\eta \) that is on the geodesic bisector between \( p \) and \( q \) on \( M \). Set \( L = d_M(x, p) = d_M(x, q) \). Since \( M_\eta = M \cap B(p, \eta) \), \( L = O(\eta) = O(\epsilon \rho) \) by Lemma 1 (1).

First, we show that \( \|p - \hat{x}\| = L(1 - c_1 \epsilon) \) and \( \|q - \hat{x}\| = L(1 - c_2 \epsilon) \) for some constants \( c_1, c_2 \). We show the inequality for \( \|p - \hat{x}\| \). That for \( \|q - \hat{x}\| \) is symmetric.
Specifically, let $\gamma(x_1, x_2)$ denote a minimizing geodesic path between $x_1$ and $x_2 \in M$. Now, for any point $y \in \gamma(x, p)$ or $y \in \gamma(x, q)$, it follows from $L = O(\varepsilon \rho)$ and Lemma 1 (1) that $\|y - p\| = O(\varepsilon \rho)$. Consider the projection of $\gamma(x, p)$ onto $\tilde{T}_p$ through the projection map $\Pi$, and let $\text{len}(\Pi(\gamma(x, p)))$ denote its length. We have that:

$$
\lambda_{\text{min}} \cdot d_M(x, p) \leq \text{len}(\Pi(\gamma(x, p))) \leq d_M(x, p) = L,
$$

where $\lambda_{\text{min}}$ is the smallest eigenvalue of the Jacobian of $\Pi$ at any point $y \in \gamma(x, p)$. It then follows from Lemma 1 (3) and $\|y - p\| = O(\varepsilon \rho)$ for $y \in \gamma(x, p)$, that $\lambda_{\text{min}} = 1 - O(\varepsilon)$, implying that

$$
\text{len}(\Pi(\gamma(x, p))) = (1 - a\varepsilon)L
$$

for some constant $a$. Since $\|\tilde{x} - p\| \leq \text{len}(\Pi(\gamma(x, p)))$, it then follows that $\|\tilde{x} - p\| \leq (1 - a\varepsilon)L$.

On the other hand, consider the image of the line segment $\tilde{x}p$ under the inverse map $\Phi = \Pi^{-1}$. Let $L'$ denote the length of this curve on $M$. It is bounded by the following inequality:

$$
L = d_M(x, p) \leq L' \leq \lambda_{\text{max}} \cdot \|\tilde{x} - p\|,
$$

where $\lambda_{\text{max}}$ is the largest eigenvalue of the Jacobian of $\Phi$ at any point $y$ in segment $\tilde{x}p$. Hence the length of $L'$ is $\|\tilde{x} - p\|(1 + b\varepsilon)$ for some constant $b$ by Lemma 3, which implies that

$$
\|\tilde{x} - p\| \geq L/(1 + b\varepsilon).
$$

Combining this with the earlier inequality $\|\tilde{x} - p\| \leq (1 - a\varepsilon)L$, we have that $\|\tilde{x} - p\| = (1 - c_1\varepsilon)L$ for some constant $c_1$. By a similar argument, we can show $\|\tilde{q} - \tilde{x}\| = (1 - c_2\varepsilon)L$ for some constant $c_2$. 
We now show that $d(\tilde{x}, \tilde{S}) = O(\epsilon^3 \rho/\delta)$. Let $\tilde{z}$ be the projection of $\tilde{x}$ in the bisector $\tilde{S}$, which is a $(d - 1)$-dimensional hyperplane. Note that $\tilde{x} \tilde{z}$ is parallel to $p\tilde{q}$ in the $d$-dimensional space $\tilde{T}_p$. See Figure 2.1 (a) for an illustration — by elementary calculations, we have that:

$$\|\tilde{x} - \tilde{q}\|^2 - \|\tilde{x} - \tilde{p}\|^2 = 2\|\tilde{p} - \tilde{q}\| \cdot \|\tilde{x} - \tilde{z}\|$$

$$\Rightarrow d(\tilde{x}, \tilde{S}) = \|\tilde{x} - \tilde{z}\| = \frac{\|\tilde{x} - \tilde{q}\|^2 - \|\tilde{x} - \tilde{p}\|^2}{2\|\tilde{p} - \tilde{q}\|}.$$

It then follows from earlier results that

$$d(\tilde{x}, \tilde{S}) = O(L^2 \epsilon)/\|\tilde{p} - \tilde{q}\| = O\left(\frac{\epsilon^3 \rho}{\delta}\right),$$

where the second equality uses the fact that $P$ is $\delta$-sparse and thus $\|p - \tilde{q}\| = \Omega(\delta \rho)$.

Lemma 3. Given an $(\epsilon, \delta)$-sampling $P$ of $M$ with sufficiently small $\epsilon$, and $\delta = \Omega(\epsilon^{3/2})$, for any $p \in P$, we have:

$$(1 - O(\epsilon + \frac{\epsilon^3}{\delta^2})) \text{Area}_p \leq \hat{A}_p \leq (1 + O(\epsilon + \frac{\epsilon^3}{\delta^2})) \text{Area}_p.$$
Proof. Let $Q \subset P$ be the set of geodesic Voronoi neighbors of $p$ in $\text{Vor}(P)$ on $M$ and $R \subset P$ the set of points such that $\Pi(R)$ is the set of Voronoi neighbors of $p$ in $\tilde{T}_p$. Since $P$ is an $(\varepsilon, \delta)$-sampling of $M$ and the projection map $\Pi$ has a Jacobian close to 1, by choosing $\eta = c\varepsilon \rho$ for some $c$ big enough, we have that both $Q, R \subseteq P_{\eta} = P \cap M_{\eta}$. Hence $\Pi(R)$ is also the set of neighbors of $p$ in the dual Voronoi diagram of $K_{\eta}$.

Due to the sampling conditions of $P$, observe that $\|x - p\| = O(\varepsilon \rho)$ for any $x \in V(p)$, or $x \in \hat{V}(p)$, and $\|y - p\| = \Omega(\delta \rho)$ for any $y \in \partial V(p)$ or $y \in \partial \hat{V}(p)$.

For each $w \in P$, let $\tilde{S}_w$ be the bisector between $p$ and $\tilde{w}$ on $\tilde{T}_p$, which is a $(k-1)$-dimensional hyperplane. Now set $\beta = c'\varepsilon^\frac{3}{\delta^2}$ for some constant $c'$ big enough. Let $\tilde{S}_w^+ \subset \tilde{T}_p$ be the hyperplane parallel to $\tilde{S}_w$ but further away from $p$ by $\beta \|\tilde{w} - p\|$, and $\tilde{S}_w^-$ the hyperplane parallel to $\tilde{S}_w$ but closer to $p$ by $\beta \|\tilde{w} - p\|$. Let $H_w^+$ and $H_w^-$ be the halfspaces in $\tilde{T}_p$ containing $p$ bounded by $\tilde{S}_w^+$ and $\tilde{S}_w^-$ respectively. It is important to note that $\bigcap_{w \in P_{\eta}} H_w^- = \bigcap_{w \in P} H_w^-$ and $\bigcap_{w \in P_{\eta}} H_w^+ = \bigcap_{w \in P} H_w^+$, as every hyperplane is moved via the same ratio w.r.t its distance to $p$. Hence we have

$$\bigcap_{w \in R} H_w^- \subset \hat{V}(p) \subset \bigcap_{w \in R} H_w^+. \quad (2.1)$$

See Figure 2.1 (b) for an illustration, where the thick polygon is the boundary of $\hat{V}(p)$, and the shaded region is

$$\bigcap_{w \in R} H_w^+ \setminus \bigcap_{w \in R} H_w^-.$$ 

The volume of $\cap_{w \in R} H_w^-$ can be lower bounded by

$$\hat{A}(p) (1 - d\beta) = \hat{A}(p) (1 - O(\varepsilon^3 \delta^2)).$$

Similarly the volume of the outer convex region $\cap_{w \in R} H_w^+$ can be upper bounded by $\hat{A}(p)(1 + O(\varepsilon^3 \delta^2))$. 

28
Next, we claim the following, which implies that $\tilde{V}(p)$ and $\Pi(V(p))$ are sandwiched within the same region.

\[(\cap_{w \in R} H_w^-) \subset \Pi(V(p)) \subset (\cap_{w \in R} H_w^+). \tag{2.2}\]

Indeed, for any $w \in P_\eta$, note that the distance between $H_w^-$ or $H_w^+$ to $H_w$ is $\beta \|\hat{w} - p\|$, which is lower-bounded by $\Omega(\frac{\varepsilon^2}{\delta})$ as the input is an $(\varepsilon, \delta)$-sample. Hence by Lemma 2 each geodesic bisector, say the one between $w$ and $p$, is projected to the slab between $H_w^-$ and $H_w^+$. Since the map $\Pi$ is locally homeomorphic, one can show that the projection of the boundary of $V(p)$ is contained within the sandwich region $\cap_{w \in P_\eta} H_w^+ - \cap_{w \in P_\eta} H_w^-$ which is the same as $\cap_{w \in R} H_w^+ - \cap_{w \in R} H_w^-$. Eqn (2.2) thus holds.

Combined with the bounds on the volume of these intersections, we have

\[\tilde{A}_p(1 - \frac{\varepsilon^3}{\delta^2}) \leq \text{Area}(\Pi(V(p))) \leq \tilde{A}_p(1 + \frac{\varepsilon^3}{\delta^2}). \tag{2.3}\]

Furthermore, since $\|x - p\| = O(\varepsilon \rho)$ for any $x \in V(p)$, by Lemma 1 (3) we can bound the area $A_p = \text{Area}(V(p))$ by

\[\text{Area}(\Pi(V(p))) \leq A_p \leq (1 + O(\varepsilon))\text{Area}(\Pi(V(p))). \tag{2.4}\]

The lemma then follows from Eqns (2.3, 2.4).

**Theorem 1.** Given an $(\varepsilon, \delta)$-sampling $P$ of $M$ with $\varepsilon$ sufficiently small, compute $\tilde{A}_p$ for each $p \in P$ as described above. Then for any Lipschitz function $f$ we have that

\[\left| \int_M f - \sum_{p \in P} \tilde{A}_p f(p) \right| = O(\varepsilon + \varepsilon^3 / \delta^2), \]

implying that for $\delta = \Omega(\varepsilon^{3/2 - \xi})$ with any positive constant $\xi$, we have

\[\lim_{\varepsilon \to 0} \left| \int_M f - \sum_{p \in P} \tilde{A}_p f(p) \right| = 0. \]
Proof. Let $L$ be the Lipschitz constant of $f$. Note that the geodesic Voronoi cells of points in $P$ form a partition of the manifold $M$. For any point $x \in M$, let $N_P(x)$ denote its nearest neighbor in $P$. Since $d_M(x, N_P(x)) = O(\varepsilon \rho)$ for any $x \in M$, we have that

$$
\left| \int_M f dx - \sum_{p \in P} f(p) A_p \right| = \left| \int_M [f(x) - f(N_P(x))] dx \right|
\leq \int_M |f(x) - f(N_P(x))| dx
= O(\varepsilon \rho \cdot L \cdot \text{vol}(M)) = O(\varepsilon),
$$

(2.5)

where $\text{vol}(M)$ is the volume of the manifold $M$. On the other hand, by Lemma 3, we have that

$$
\left| \sum_{p \in P} f(p) A_p - \sum_{p \in P} f(p) \tilde{A}_p \right|
= \|f\|_{\infty} \sum_{p \in P} (A_p O(\varepsilon + \varepsilon^3/\delta^2)) = O(\varepsilon + \varepsilon^3/\delta^2).
$$

(2.6)

The theorem then follows from Eqns (2.5, 2.6), and the big-O notation hides constants both related to the underlying manifold $M$ and to the input Lipschitz function $f$. □

Remark. Our theoretical guarantee requires that the algorithm knows $\varepsilon \rho$, and thus can choose $\eta$ appropriately. Such information is typically hard to estimate in practice. In our implementation, we simply choose $\eta$ as a constant times the average distance between a sample point and its nearest neighbor in $P$.

2.3 Voronoi-Laplace Operator

Previously, there have been some discrete Laplace operators for point cloud data. Two most important ones are Gaussian weighted graph Laplace operator and PCD-Laplace operator [16]. Unfortunately, Gaussian weighted graph Laplacian only
converges to the manifold Laplacian under uniform sampling condition. While PCD-Laplacian converges under a more general sampling condition, it is not symmetric and has complex eigenvalues and eigenfunctions. Moreover, their eigenfunctions may not be orthonormal. Observing that the approximation of discrete Laplace operator can be written in an integral formula [11], naturally we can apply the above Voronoi weighting scheme to replace the volume form in the continuous integral. We thus modify the approach from [11] to construct the Voronoi-Laplace operator for a set of points \( P = \{ p_1, \ldots, p_n \} \). Our Voronoi-Laplace operator is symmetric, convergent and works better under \((\epsilon, \delta)\) sampling condition. The cost is that \((\epsilon, \delta)\) sampling condition is more strict than the sampling condition of PCD-Laplacian.

In [11], the authors proposed to approximate the Laplace operator in the discrete setting using its relation to the heat operator \( \mathcal{H}_t = e^{-t\Delta} \). Specifically, by Taylor expansion, we have

\[
\Delta = \lim_{t \to 0} \frac{\mathcal{H}_0 - \mathcal{H}_t}{t} = \lim_{t \to 0} \frac{I - \mathcal{H}_t}{t},
\]

where \( \mathcal{H}_0 \), the heat operator at time 0, is the identity operator \( I \). On the other hand, for any scalar function \( f \) on \( M \), it is well-known that

\[
\mathcal{H}_t f(x) = \int_M h_t(x, y) f(y) d\mu(y).
\]

Unfortunately, the analytical expression of the heat kernel \( h_t(x, y) \) is not known other than for very few cases. For example, if the underlying manifold is the Euclidean space \( \mathbb{R}^d \), then the heat kernel is \( h_t(x, y) = \frac{1}{(4\pi t)^{d/2}} e^{-\frac{||x-y||^2}{4t}} \). It turns out that for general manifold, this Gaussian kernel approximates the heat kernel well enough in the sense that the Laplace operator can be approximated via the Gaussian kernel.
More specifically, it is shown in [11] that
\[
\Delta f(x) = \lim_{t \to 0} \frac{1}{t(4\pi t)^{d/2}} \int_M e^{-\frac{\|x-y\|^2}{4t}} (f(x) - f(y))d\mu(y).
\]
(2.7)

In the discrete setting, the task is then to approximate the above integral from point clouds. Set \( G_t(x, y) = \frac{1}{t(4\pi t)^{d/2}} \). For a set of points \( P = \{p_1, \ldots, p_n\} \) randomly sampled from the uniform distribution, this can be easily achieved by using Monte Carlo integration, and computing \( \frac{1}{n} \sum_{j=1}^{n} G_t(p_i, p_j)[f(p_i) - f(p_j)] \). This approximates \( \Delta f(p_i) \) as \( t \) goes to 0 based on the Law of Large Numbers. However, in non-statistical setting, it is necessary to augment the summation with certain weights.

On the other hand, given a twice-differential function \( g : M \to \mathbb{R} \), it is shown in Section 2.2 that one can approximate \( \int_M g(y)d\mu(y) \) by \( I_t g = \sum_{i=1}^{n} g(p_i)A_i \), where \( A_i \) is the Voronoi weight defined as follows. For each point \( p_i \), take the set of points \( Q \) that are within distance \( c\varepsilon \rho \) for some constant \( c \) (say, \( c = 10 \)). Locally approximate the tangent space \( T \) at \( p_i \) by fitting the best plane through \( Q \) using the algorithm from [42]. Let \( \hat{Q} \) denote the projection of the set of points \( Q \) onto \( T \). Construct the Voronoi diagram for \( \hat{Q} \) in \( T \) and \( A_i \) is the volume (area for 2-manifolds) of the Voronoi cell containing \( p_i \). It is shown in the previous section that \( \|I_t g - \int_M g\|_\infty = O(\varepsilon + \varepsilon^3/\delta^2) \) for an \( (\varepsilon, \delta) \)-sample \( P \) with an appropriately chosen \( t \), where \( \delta \geq \varepsilon^{3/2-\xi} \) for an arbitrary value \( \xi > 0 \). This implies that \( I_t g \) converges to \( \int_M g \) as \( \varepsilon \to 0 \).

Combining these two results, we approximate \( L_t f(p_i) \) by
\[
L_f(p_i) := \sum_{j=1}^{n} G_t(p_i, p_j)A_j f(p_j).
\]
Hence the discrete Laplace operator from a set of \( n \) points is an \( n \times n \) matrix \( L \) with
\[
L[i][j] = \begin{cases} 
G_t(i, j)A_j & i \neq j \\
G_t(i, i)A_i - \sum_{j=1}^{n} G_t(i, j)A_j & \text{Otherwise}.
\end{cases}
\]
Using similar techniques from [11, 59], it is easy to show that
Theorem 2. Let the point set $P_{\varepsilon, \delta}$ be an $(\varepsilon, \delta)$-sampling of $M$. Set $t(\varepsilon) = \varepsilon^{1/\alpha}$ for an arbitrary fixed number $\alpha > 0$. Let $L^P_t$ denote the discrete Laplace operator as constructed above for point set $P$ with parameter $t$. Then for any function $f \in C^2(M)$, we have that

$$\lim_{\varepsilon \to 0} \sup_{P_{\varepsilon, \delta}} \| L^P_{t(\varepsilon)} f - \Delta f \|_{\infty} = 0,$$

where the supremum is taken over all $(\varepsilon, \delta)$-sampling of $M$ satisfying that $\delta \geq \varepsilon^{3/2 - \xi}$ for an arbitrary positive value $\xi$.

The matrix $L$ is not symmetric. However, it can be decomposed into $L = GD$, where $D$ is the diagonal matrix with $D[i][i] = A_i$, and $G[i][j] = G_h(i, j)$ for $i \neq j$, and $G[i][i] = -\frac{1}{A_i} \sum_{j \neq i} G_h(i, j) A_j$. Since $G$ is symmetric and $D$ is diagonal positive-definite, $L$ has real eigenvalues and eigenfunctions.

We compute the eigenvalues $\lambda_1, \ldots, \lambda_n$ and eigenvectors $\phi_1, \ldots, \phi_n$ of $L$ by solving the generalized eigenvalue problem $G \rho_i = \beta_i D^{-1} \rho_i$, and by observing that $\lambda_i = \beta_i$ and $\phi_i = D^{-1} \rho_i$. Indeed $GD(D^{-1} \rho_i) = G \rho_i = \beta_i(D^{-1} \rho_i)$ implies that $D^{-1} \rho_i$ is an eigenvector of $L = GD$. (We cannot compute the straightforward generalized eigenvalue problem $D \phi_i = \lambda_i G^{-1} \phi_i$ as $G$ may not be numerically invertible). The eigenvectors $\phi_i$s form an orthonormal basis with respect to the $D$-inner product, namely,

$$\langle \phi_i, \phi_j \rangle_D = \phi_i^T D \phi_j = \delta_{ij},$$

where $\delta_{ij}$ is the Kronecker delta. This is because that

$$\phi_i^T D \phi_j = (D^{-1} \rho_i)^T DD^{-1} \rho_j = \rho_i^T D^{-1} \rho_j = \delta_{ij}. \quad (2.8)$$

This is an important property because, in the continuous case, the inner product between two functions is $\langle f, g \rangle = \int_M f(x)g(x)d\mu(x)$. In the discrete setting, it is natural to replace the integral with a summation $\sum_{i=1}^n f(p_i)g(p_i)A_i = \langle f, g \rangle_D$, where $A_i$ corresponds to the volume form $d\mu(p_i)$ at point $p_i$. This area weighting is necessary
due to the non-uniform sampling of input points, and indeed, it is shown in the previous section that this discrete sum converges to the true integral for points satisfying certain sampling conditions. Hence Eqn (2.8) is the discrete analog of the fact that the Laplacian eigenfunctions are orthonormal. Similar property was established for previous mesh Laplacians as well [87].

2.4 Left Eigenvector Scheme

It is well-known that the integral of any function defined on a smooth compact manifold $M$ is preserved under the heat diffusion process. In other words, let $\mathcal{H}_t$ denote the heat operator w.r.t. time $t$. For any $f : M \rightarrow \mathbb{R}$ and any $t$, we have that $\int_M f(x)dx = \int_M \mathcal{H}_t f(x)dx$. We call this the heat-preservation property. This is a fundamental property of the heat operator, and a useful one in practice. For example, heat diffusion has been widely used to smooth functions on a manifold. The heat-preservation property means that the function will not keep decreasing to zero during the smoothing process. Now, given an $\varepsilon$-sampling $P$ of $M$, we wish to develop a discrete integral estimation scheme so that certain discrete heat diffusion is an integral invariant.

First, suppose we are given a discrete heat operator $H_t$, analogous to $\mathcal{H}_t$ in the smooth case. Being a linear operator on functions, $H_t$ can be represented as an $n \times n$ matrix, where $n$ is the number of points in $P$. The heat-preservation property means that for any $f$,

$$\langle \omega, f \rangle = \langle \omega, H_t f \rangle \Rightarrow \omega^T f = \omega^T H_t f.$$
Hence $\omega^T = \omega^T H_t$, implying that $\omega$ is necessarily a left-eigenvector of $H_t$ with eigen-value being 1. Hence we now need a good discrete heat operator, which indeed has a left-eigenvector corresponding to eigenvalue 1.

**Constructing discrete heat operator.** To this end, we construct $H_t$ using a similar algorithm as the one proposed in [16] to approximate the so-called Laplace-Beltrami operator $\Delta$, which is connected to the heat operator by the standard relation $H_t = e^{-t\Delta}$ [75]. Specifically, first, we modify the construction of the local patch (presented in Section 2.1) around each sample point $p_i$ slightly: In Step (1), we approximate the tangent space at $p_i$ using sample points that are within the ball $B(p_i, 10\varepsilon \rho)$ around $p_i$. In Step (2) and (3), we use a different neighborhood size $\eta$ which is a small constant. We then construct the Delaunay triangulation $K_\eta(p_i)$ for the projection of points from $P_\eta = P \cap B(p_i, \eta)$ in the approximate tangent space at $p_i$. Let $P_{\eta/2}(p_i)$ denote the set of sample points within the ball $B(p_i, \eta/2)$. We now define a weighting value $A_j^i$ as follows:

$$A_j^i = \begin{cases} \frac{1}{d+1} \sum_{\sigma} \text{Area}(\sigma) & \text{if } p_j \in P_{\eta/2}(p_i) \\ 0 & \text{otherwise,} \end{cases} \quad (2.9)$$

and the summation is taken over all $k$-simplices $\sigma$ incident to $\Pi(p_j)$ in the local patch $K_\eta(p_i)$ at $p_i$.

Finally, we fix the “time” parameter $t = \Omega(\frac{1}{\varepsilon^2 + \xi})$ for any constant $\xi > 0$. We now construct the discrete Heat operator $H_t$, which is also an $n \times n$ matrix, by setting the $j$th element in the $i$th row as:

$$H_t[i, j] = \frac{G_{ij} A_j^i}{\sum_{k=1}^n G_{ik} A_k^i} , \quad \text{where} \quad G_{ij} = e^{-\frac{\|p_i - p_j\|^2}{4t}}.$$
In other words, $H_t f(p_i)$ is a convex combination of all $f(p_j)$s where $p_j$ is within distance $\eta/2$ to $p_i$. The weight of each $f(p_j)$ is based on its area in the local triangulation $K_\eta(p_i)$ and the distance from $p_j$ to $p_i$.

**Properties of $H_t$.** Intuitively, the above construction follows the idea first proposed in [12] to use the Gaussian kernel $G_t(x, y) = \frac{1}{(4\pi t)^{d/2}} e^{-\|x-y\|^2/4t}$ to approximate the so-called heat kernel $h_t(x, y)$. It also relates to the discrete Laplace operator $L_t$ constructed in [16] by $H_t = I - tD^{-1}L_t$, where $D$ is a diagonal matrix with

$$D[i][i] = \frac{1}{(4\pi t)^{d/2}} \sum_{k=1}^{n} G_{ik} A^k_i.$$ 

This intuitively approximates the first order Taylor expansion of $H_t = e^{-t\Delta}$. We can show that $\|H_t f - H_t f\|_\infty = O(t^\alpha)$ for any Lipschitz function $f$, where $\alpha$ is some positive constant. Hence for small $t$, the discrete heat operator is close to the heat operator. The proof is straightforward but quite technical, following from several results and procedures in [12, 13]. We thus omit it from this extended abstract.

In general, since $H_t$ is not symmetric, it may have complex eigenvalues and eigenvectors. In our case, however, since the row sum of $H_t$ is 1, and all elements in the matrix are non-negative, it is easy to check that the constant vector $[1, 1, \ldots, 1]^T$ is a right-eigenvector of $H_t$ corresponding to eigenvalue 1. Furthermore, since $H_t$ is an averaging operator, the largest possible eigenvalue of $H_t$ is 1. Hence we obtain the following.

**Lemma 4.** The maximum eigenvalue of $H_t$ is 1, and its corresponding right and left eigenvectors are real-valued.

**Principal Eigenvector weighting scheme.** Since the left-eigenvector $\mu$ of $H_t$ with eigenvalue 1 is real valued, we can now set the weighting vector $\omega = \mu$. Obviously,
\langle \omega, f \rangle = \langle \omega, H_t f \rangle$ for any $f$. Unfortunately, it is not clear whether one can bound the relation between $\langle \omega, f \rangle$ and $\int_M f(x)dx$. However, if we can compute a global mesh from input point cloud that approximates the underlying manifold $M$ in same way as defined in [15], and use the global mesh to set up $A^j_i$ instead of using local patches, then, it can be shown that

$$\lim_{t \to 0} \left| \langle \omega, f \rangle - \frac{1}{vol(M)} \int_M f(x)dx \right| = 0,$$

where $vol(M)$ represents the volume of the manifold $M$.

**Remark 1.** We remark that since the eigenvector constructed is normalized, we can only estimate the integral up to a scaling factor, which is the volume of the manifold $M$. This limitation also exists for Monte-Carlo integration based methods. The Voronoi weighting scheme approximates the integral without any scaling factor.

**Remark 2.** Note that each row of our discrete Heat operator $H_t$ is computed based on the same local patch information as in the case for Voronoi weighting scheme. Taking the left-eigenvector of this matrix in some sense connects across different local patches. Such a global connection is intuitively necessary as we wish to guarantee the heat-preservation property, which is a global behavior. On the other hand, in the Voronoi weighting scheme, we only wish to locally approximate the metric on the manifold. Hence no transformation of information across local patches is needed.

2.5 Experiment

We present two experiments separately in this chapter. The first experiment concentrates on evaluating the integral estimation. The second experiment compares
the performance of our new Voronoi-Laplace operator with previous discrete Laplace operators.

2.5.1 Performance of the Weighting Schemes for Integral Estimation

In this section, we implement the two weighting schemes as described in section 2.2 and section 2.4 and compare them with Monte Carlo integration. In Monte Carlo integration, we use kernel density estimation (KDE) Toolbox developed by Ihler and Mandel [45] to obtain the weights for input sample points and estimate the integral as weighted sum of the function values at sample points. We also show the integral estimation results using equal weights at every sample point. The experiment results show that our schemes consistently outperform Monte Carlo integration in estimation accuracy, especially when there are noise in either the integrand function or in input sample points.

Experimental setup. We consider two data sets. One is a synthetic data set where samples are drawn from a flat 2-torus \( T^2 \) which is defined parametrically as

\[
T^2(\alpha, \beta) = (\cos \alpha, \sin \alpha, \cos \beta, \sin \beta)
\]

with both \( \alpha \) and \( \beta \) ranging from 0 to \( 2\pi \). The need of such synthetic data is so that the ground truth can be computed. We have also experimented with other synthetic data, such as 2-sphere and 3-torus. The results are similar and thus we only focus on \( T^2 \) here. We perform our experiments on three types of sampling conditions of input points. To obtain a uniform-sampling of \( T^2 \), we simply uniformly sample the parameter domain (a square) since the parametrization is isometric. To achieve a non-uniform sampling of \( T^2 \), we impose a distribution of Gaussian mixture
on the parameter domain and accept a sample point with higher probability if the Gaussian mixture has a bigger value at the sample point. We also experiment with a skew-sampled point cloud of the underlying manifold that violates the $\delta$-sparsity condition. Specifically, in the skew-sampled input, half of the points are drawn from a very narrow vertical slab in the middle of the parameter domain. Finally, we also experiment with point cloud data with noise, which is produced by perturbing the coordinates of input points in the ambient space.

The second data set is a real molecular model as shown in the Figure 2.2. There are around 8000 sample points (shown as blue dots) on the surface. Since we do not know the underlying manifold, we use the underlying mesh of this model to obtain a "ground truth" — Given the function values at the vertices, we linearly interpolate the function within each mesh facet and take the integral of this piecewise linear function over the mesh as the ground truth. It is shown in [15] that this mesh-integral is close to the true integral.
We use two groups of test functions in our experiments: trigonometric functions \((\sin x, \cos x, \sin 2x, \cos 2x, \sin 3x, \cos 3x, \sin 4x, \cos 4x)\) and polynomial functions \((x, x^2, x^3)\). Appropriate constants are added to test functions to guarantee that the integral is not close to 0. We also have noisy versions of these functions by perturbing the values at each sample point randomly.

Note that except for the Voronoi weighting scheme, all other methods generate weights up to a scaling factor, which is the manifold volume. Hence to compare different schemes, we normalize the weights so that \(\sum_i w_i\) equals the area of the underlying manifold. To measure the error with the ground truth \(I\), we use relative error \(|I - I_D|/I\), where \(I_D\) is the discrete approximation. We average all the relative errors from the same group of basis functions and report that as the estimation error of a particular method.

**Accuracy and convergence.** Table 2.1 and Table 2.2 show the results for the synthetic data \(T^2\), embedded in the 4-dimensional Euclidean space. Table 2.1 includes presents results for the uniform and non-uniform sampling of \(T^2\), while Table 2.2 shows those for the skew-sampling point cloud data. The KDE toolkit provides five variants for density estimation accounting for five different ways to estimate the width of the kernel Gaussian function. Due to space limitation, in each case, we report the two KDE variants that produce the most accurate estimations. In the case of the uniform and non-uniform sampled data, we present results from the rule of thumb (rot) and the maximum smoothing principle (msp) KDE methods. In the case of skew-sampled data, we present results from the KDE (rot) and the more sophisticated likelihood cross-validation (lcv) methods.
Table 2.1 shows that both of our schemes show convergent estimation for the uniform sampled and nonuniform sampled point cloud data, as more sample points are drawn from the underlying manifold, while all methods of Monte Carlo integration do not yet. It is interesting to note that Monte Carlo integration is known to be statistical convergent, and our input point cloud data are in fact drawn from Gaussian distribution in these cases. Table 2.1 perhaps suggests that the convergent rate for Monte Carlo integration is much slower than our schemes in experiments. Our methods also achieve better estimation in general than KDE based methods.

For the skew-sampled data, Table 2.2 shows that our methods still show convergent behavior. Specifically, the Voronoi weighting scheme seems to be both accurate and convergent, even though there is no theoretical guarantee for such inputs (where the $\delta$-sparsity condition is not met).

We also remark that as the dimension of the ambient space increases, our methods return exactly same results, while the performances of KDE-based methods further deteriorate. This is partly due to the fact that our algorithm assumes that the input dimension is given. However, it is unclear how to use this information for KDE-based methods.

Table 2.5.1 shows the results for the molecular data, and similar behavior can be observed. In this case, we present the results by KDE (rot) and KDE (msp) methods, which produce best numerical estimations for this data.

**Timing.** Table 2.5.1 compares the timing of different methods. We note that the two simplest KDE methods (KDE rot and KDE msp) are very efficient. The more complicated KDE (lcv) method runs slower than the Voronoi weighting scheme. The Principal Eigenvector scheme is slowest, as it requires to first construct the discrete
<table>
<thead>
<tr>
<th></th>
<th>Voronoi</th>
<th>Eigen</th>
<th>KDE(rot)</th>
<th>KDE(msp)</th>
<th>Equal</th>
</tr>
</thead>
<tbody>
<tr>
<td>uniform</td>
<td>8 / 4 / 4</td>
<td>6 / 4 / 3</td>
<td>10 / 20 / 12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>nonuniform</td>
<td>2 / 2 / 1</td>
<td>7 / 1 / 4</td>
<td>8 / 1 / 4</td>
<td>36 / 29 / 21</td>
<td></td>
</tr>
<tr>
<td>uniform with noisy vert.</td>
<td>5 / 4 / 4</td>
<td>9 / 4 / 4</td>
<td>6 / 4 / 3</td>
<td>10 / 20 / 12</td>
<td></td>
</tr>
<tr>
<td>nonuniform with noisy vert.</td>
<td>1 / 1 / 1</td>
<td>8 / 2 / 4</td>
<td>9 / 2 / 4</td>
<td>36 / 29 / 21</td>
<td></td>
</tr>
<tr>
<td>uniform with noisy fun.</td>
<td>7 / 4 / 4</td>
<td>5 / 5 / 4</td>
<td>3 / 5 / 7</td>
<td>4 / 4 / 7</td>
<td>35 / 30 / 23</td>
</tr>
<tr>
<td>nonuniform with noisy fun.</td>
<td>7 / 4 / 4</td>
<td>5 / 5 / 4</td>
<td>3 / 5 / 7</td>
<td>4 / 4 / 7</td>
<td>35 / 30 / 23</td>
</tr>
</tbody>
</table>

Table 2.1: Relative error of integral estimation for 2500/5000/10000 points from a flat-2 torus $T^2$ embedded in $\mathbb{R}^4$. The last three columns are results based on the Monte Carlo integration, by using density estimation methods provided by the KDE toolbox, and by equal weight, respectively. All numbers are scaled by $10^{-4}$.

<table>
<thead>
<tr>
<th></th>
<th>Voronoi</th>
<th>Eigen</th>
<th>KDE(rot)</th>
<th>KDE(lev)</th>
<th>Equal</th>
</tr>
</thead>
<tbody>
<tr>
<td>uniform</td>
<td>2 / 1 / 1</td>
<td>3 / 2 / 1</td>
<td>15 / 8 / 12</td>
<td>18 / 10 / 13</td>
<td>59 / 41 / 43</td>
</tr>
<tr>
<td>nonuniform</td>
<td>3 / 3 / 2</td>
<td>8 / 7 / 2</td>
<td>18 / 38 / 28</td>
<td>22 / 42 / 31</td>
<td>50 / 85 / 75</td>
</tr>
<tr>
<td>uniform with noisy vert.</td>
<td>11 / 4 / 2</td>
<td>16 / 9 / 11</td>
<td>19 / 11 / 12</td>
<td>59 / 41 / 43</td>
<td></td>
</tr>
<tr>
<td>nonuniform with noisy vert.</td>
<td>7 / 5 / 4</td>
<td>16 / 37 / 27</td>
<td>21 / 41 / 30</td>
<td>50 / 85 / 75</td>
<td></td>
</tr>
<tr>
<td>uniform with noisy fun.</td>
<td>21 / 12 / 8</td>
<td>23 / 12 / 7</td>
<td>19 / 13 / 19</td>
<td>24 / 15 / 20</td>
<td>67 / 45 / 47</td>
</tr>
<tr>
<td>nonuniform with noisy fun.</td>
<td>26 / 11 / 6</td>
<td>30 / 12 / 8</td>
<td>28 / 36 / 31</td>
<td>32 / 40 / 34</td>
<td>62 / 80 / 77</td>
</tr>
</tbody>
</table>

Table 2.2: Relative error of integral estimation for 2500/5000/10000 skew-sampled points from flat-2 torus $T^2$ embedded in $\mathbb{R}^4$. All numbers are scaled by $10^{-4}$. 
Voronoi  Eigen  KDE(rot)  KDE(msp)  Equal
original   3     17     26     34     74
noisy vert. 3     20     27     34     74
noisy fun. 14    14     33     41     67

Table 2.3: Relative error of integral estimated by various methods for molecular data. All numbers are scaled by $10^{-4}$.

Table 2.4: Timing in seconds of various methods for 2500/5000/10000 points from flat-2 torus $T^2$ embedded in $\mathbb{R}^4$.

Laplace operator. We point out that the timing of our methods is based on a rather straightforward implementation, without any optimization. Table 2.5.1 shows that as expected, the ambient dimension has only mild effect on the running time of our proposed methods. (Note that we assume that the intrinsic dimension is known.)

Stability. To test the stability of these methods under heat diffusion smoothing, we use heat diffusion operator (as constructed by our algorithm) to smooth a function for a few times. At each iteration, we compute the integral and see how the results change. Figure 2.3 shows the trend for two test functions. The experiments are conducted on the molecular data. By definition, the integral under the Principal
Table 2.5: Timing in seconds for 2500/5000/10000 points from flat-2 torus $T^2$ embedded in $\mathbb{R}^4$, $\mathbb{R}^{20}$, $\mathbb{R}^{40}$ and $\mathbb{R}^{60}$.

<table>
<thead>
<tr>
<th>method</th>
<th>4d</th>
<th>20d</th>
<th>40d</th>
<th>60d</th>
</tr>
</thead>
<tbody>
<tr>
<td>Voronoi</td>
<td>7/18/51</td>
<td>10/28/100</td>
<td>13/40/146</td>
<td>16/53/199</td>
</tr>
<tr>
<td>Eigen</td>
<td>34/167/734</td>
<td>41/174/772</td>
<td>44/190/824</td>
<td>47/199/883</td>
</tr>
</tbody>
</table>

Figure 2.3: (a) Estimated Integral for function $f = \sin x + 1$ under heat diffusion. (b) Estimated Integral for function $f = \sin^2 x + 1$ under heat diffusion.

Eigenvector scheme is invariant; while obviously, no other method preserves integrals during this smoothing.

### 2.5.2 Performance of Voronoi-Laplace Operator

We now show some experimental results comparing our new discrete Laplace operator from point clouds, denoted by Voronoi-Laplace, with the discrete Laplace operator constructed from mesh as proposed in [15], denoted by mesh-Laplace, and with the theoretically better point-clouds Laplace operator proposed in [16], denoted by
PCD-Laplace. Here we follow the same experimental setup as [10]. The ground truth of the Laplace-Beltrami operator on the 2−sphere \( S^2 \) can be explicitly calculated [15].

Given an input function \( f \) on a manifold \( M \), and a point cloud \( P \) sampling \( M \), we evaluate the manifold Laplacian (ground truth) and our Voronoi-Laplacian at each point in \( P \), and obtain two vectors \( U \) and \( \bar{U} \) respectively. To measure the error of the Voronoi Laplacian, we consider the commonly used normalized \( L_2 \) error

\[
E_2 = \frac{||U - \bar{U}||_2}{||U||_2}.
\]

We also show the normalized \( L_\infty \) error \( E_\infty = \frac{||U - \bar{U}||_\infty}{||U||_\infty} \) since the convergence result is under \( L_\infty \) norm which is stronger than the \( L_2 \)−convergence. We compute the same error measures for PCD-Laplacian and mesh-Laplacian for comparison. As shown in Table 2.6, the performance of our new Voronoi-Laplacian is close to that of the mesh-Laplacian, and much better than the PCD-Laplacian.

<table>
<thead>
<tr>
<th>function</th>
<th>( f = x )</th>
<th>( f = x^2 )</th>
<th>( f = e^x )</th>
</tr>
</thead>
<tbody>
<tr>
<td>mesh-L</td>
<td>2.91 / 6.05</td>
<td>2.03 / 3.32</td>
<td>2.64 / 3.25</td>
</tr>
<tr>
<td>Voronoi-L</td>
<td>2.36 / 4.40</td>
<td>1.81 / 2.45</td>
<td>2.20 / 2.73</td>
</tr>
<tr>
<td>PCD-L</td>
<td>5.03 / 6.97</td>
<td>4.84 / 5.42</td>
<td>5.00 / 5.29</td>
</tr>
</tbody>
</table>

2562 points uniform sphere

<table>
<thead>
<tr>
<th>function</th>
<th>( f = x )</th>
<th>( f = x^2 )</th>
<th>( f = e^x )</th>
</tr>
</thead>
<tbody>
<tr>
<td>mesh-L</td>
<td>3.00 / 6.89</td>
<td>3.37 / 4.30</td>
<td>3.44 / 4.00</td>
</tr>
<tr>
<td>Voronoi-L</td>
<td>3.50 / 9.11</td>
<td>3.59 / 5.38</td>
<td>3.81 / 4.80</td>
</tr>
<tr>
<td>PCD-L</td>
<td>15.85/33.21</td>
<td>13.77/19.54</td>
<td>15.22/21.64</td>
</tr>
</tbody>
</table>

5000 points non-uniform sphere

<table>
<thead>
<tr>
<th>function</th>
<th>( f = x )</th>
<th>( f = x^2 )</th>
<th>( f = e^x )</th>
</tr>
</thead>
<tbody>
<tr>
<td>mesh-L</td>
<td>1.04 / 2.94</td>
<td>2.87 / 3.16</td>
<td>2.33 / 2.70</td>
</tr>
<tr>
<td>Voronoi-L</td>
<td>1.23 / 3.58</td>
<td>2.90 / 3.26</td>
<td>2.37 / 2.67</td>
</tr>
<tr>
<td>PCD-L</td>
<td>11.64/15.18</td>
<td>12.28/14.06</td>
<td>11.96/12.69</td>
</tr>
</tbody>
</table>

10000 points non-uniform sphere

Table 2.6: Normalized \( L_2/L_\infty \) error for different sampling on \( S^2 \), all numbers are scaled in percentage.
2.6 Discussions

In this chapter, we considered the problem of estimating the integral of a function over a submanifold embedded in the high dimensional space from a set of sample points. We approached the problem from a geometric point of view and developed two schemes based on the local geometric approximation of the manifold. Both schemes have a running time depending only linearly on the dimension of the ambient space. We showed that the Voronoi scheme converges to the ground truth under mild assumptions on input sample points and the integrand function. The Principal Eigenvector scheme is constructed w.r.t a discrete Heat operator, and preserves heat under the diffusion process as guided by that operator. These are the first results of this sort for integral estimation from general PCD. In our experiments, both new schemes give convergent results, and produce numerically much better estimations than the standard Monte Carlo integration methods. The Voronoi scheme usually produces more accurate results, while the Principal Eigenvector scheme has the additional property that it has an accompanying Heat operator which is an integral invariant under this scheme.

Our Voronoi scheme still requires that input points \((\varepsilon, \delta)\)-sample the underlying manifold \(M\). A natural question is whether it is possible to remove the \(\delta\)-sparsity constraint. This seems to require a more global approach to estimate error than our current local method.

Our experiments show that the Voronoi scheme provides more accurate integral estimations than statistical methods even for data uniformly sampled. This intuitively is because that the former takes advantage of the local geometry (via the local patches) to estimate a more accurate volume measure around each sample point. This raises an
interesting question: how can one use local geometry to augment traditional statistical methods to achieve more accurate numerical estimations even under the statistical setting.

We believe that ideas behind the Principal Eigenvector scheme presents an interesting step towards obtaining global behavior from local, inconsistent patches. Sometimes it is enough to approximate the local measure on the manifold. However, many times certain stitching across local patches is necessary. The general question is how to achieve that without going to the extreme of constructing a global mesh. In our case, the global connection is made by taking the left eigenvector of the Heat operator. It will be interesting to quantify precisely what this left eigenvector captures. We conjecture that it in fact encodes a consistent set of area weights for sample points, and the Principal Eigenvector scheme converges to the true integral.

Finally, from the Voronoi weighting scheme, we have developed the first point cloud Laplace operator, which is convergent and has real eigenvalues and eigenfunctions. This is achieved by modifying the previous PCD-Laplacian with the Voronoi scheme and in the same logic approximating the functional Laplacian. The real eigenvalues and eigenfunctions are appreciated in many geometry processing applications. We also showed that under this sampling condition it performs better than PCD-Laplacian.
Chapter 3: HKS-preserving Data Simplification

In the previous chapter, we studied the problem of integral estimation for functions defined on point cloud data. Based on the Voronoi weighting scheme, we proposed a new discrete Laplace operator called Voronoi-Laplacian. This new operator will potentially facilitate the investigation for point cloud processing applications, most of which relies on the eigen-decomposition of this discrete Laplace matrix. However, one issue common to all spectral methods is that the eigen-decomposition of large matrices is quite expensive, even though the discrete Laplace matrix is sparse and we typically need at most the first 300 eigenvectors. To make our algorithm scalable for large input data, one could use the shift-invert spectral transform approach as proposed in [82], which can compute the first few eigenvectors accurately much more efficiently than standard eigen-decomposition approaches. However, the time required is still large in many cases. A natural strategy is to first sub-sample the input data and then compute the spectral structure for the simplified structure.

The sub-sampling can be achieved by any mesh simplification method, such as the QSlim mesh decimation software [35] which is used in [46] to reduce the size of the input mesh. However, sub-sampling in general tends to smooth out features and it is not clear how the spectral structure changes after applying those methods. We wish to simplify it in a way that preserves the spectral structure as much as possible.
Directly relating the eigenvalues and eigenfunctions before and after the sub-sampling is difficult. Nevertheless, the state of art spectral descriptor heat kernel signature is almost as informative as the eigen-structure itself. (For more details, please see the informative theorem in [80]). Preserving $HKS$ tends to preserve the eigen-structure. Hence in this chapter we propose a novel sub-sampling technique in line with the definition of the $HKS$ which selects vertices where heat accumulates locally after a small time period.

### 3.1 Heat Kernel and Heat Diffusion

The Laplace-Beltrami operator gives rise to the partial differential equation

$$(\frac{\partial}{\partial t} + \Delta) f(t, x) = 0$$

called the heat equation. The heat equation describes the propagation of heat on the surface and its solution $f(t, x)$ is the heat distribution at a point $x$ in time $t$. The initial condition of the equation is some initial heat distribution $f(0, x)$; if $X$ has a boundary, appropriate boundary conditions must be added. The solution of the heat equation corresponding to a point initial condition $f(0, x) = \delta(x - y)$, is called the heat kernel and represents the amount of heat transferred from $x$ to $y$ in time $t$ due to the diffusion process. Using spectral decomposition, the heat kernel can be represented as

$$h_t(x, y) = \sum_{i \geq 0} e^{-\lambda_i t} \phi_i(x) \phi_i(y)$$

where $\phi_i$ and $\lambda_i$ are, respectively, the eigenfunctions and eigenvalues of the Laplace-Beltrami operator satisfying $\Delta \phi_i = \lambda_i \phi_i$ (without loss of generality, we assume $\lambda_i$ to be sorted in increasing order starting with $\lambda_0 = 0$). Since the Laplace-Beltrami
operator is an intrinsic geometric quantity, i.e., it can be expressed solely in terms of the metric of $M$, its eigenfunctions and eigenvalues as well as the heat kernel are invariant under isometric transformations of the manifold.

The heat (diffusion) operator $\mathcal{H}_t$ with respect to $t$ is an operator on $L^2(M)$, the space of square integrable functions on $M$. It is defined as

$$\mathcal{H}_t f(x) = \int_M h_t(x, y) f(y) d\mu_y,$$

where $h_t(x, y)$ is the heat kernel, and $d\mu_y$ is the volume form at $y$. Given a function $f$ serving as the initial heat distribution on $M$, $\mathcal{H}_t f$ is the distribution of heat at time $t$. This intuitive physical interpretation makes the heat operator a popular tool to smooth both the manifold itself and functions defined on it. The heat operator is compact, self-adjoint, and positive semi-definite. Thus it has discrete spectrum $\rho_0 \geq \rho_1 \geq \ldots \geq 0$ with $\mathcal{H}_t \phi_i = \rho_i \phi_i$, and by the spectral theorem, the heat kernel can be written as:

$$h_t(x, y) = \sum_{i \geq 0} \rho_i \phi_i(x) \phi_i(y). \quad (3.3)$$

The largest eigenvalue $\rho_0$ is necessarily 1, because heat diffusion is an averaging process with $\int_M h_t(x, y) d\mu_y = 1$.

The heat operator is connected to the Laplace-Beltrami operator $\Delta$ of $M$ by the following relation: $\mathcal{H}_t = e^{-t\Delta}$. This implies that $\mathcal{H}_t$ and $\Delta$ share the same eigenfunctions $\phi_i$, and their eigenvalues satisfy that $\rho_i = e^{-t\lambda_i}$, where $\Delta \phi_i = \lambda_i \phi_i$. 
3.2 Heat Kernel Signature

Our algorithm relies on the Heat Kernel Signature (HKS) proposed and analyzed in [80] (a scale-invariant version was used in [36]). This shape descriptor is derived from the heat operator which we just introduced.

The heat kernel function has many nice properties; see [80] for a detailed discussion. The family of heat kernel functions uniquely defines the underlying manifold up to an isometry; thus it is very informative and a potentially good tool to construct a shape descriptor.

Specifically, Sun et al. propose the following Heat Kernel Signature (HKS) as a shape descriptor for a manifold $M$ [80].

$$HKS_t(x) = h_t(x, x) = \sum_{i \geq 0} e^{-t\lambda_i} \phi_i(x)^2.$$  \hfill (3.4)

Intuitively, it measures how much heat is left at time $t$ for the point $x \in M$ if unit amount of heat is placed at point $x$ when $t = 0$. $HKS$ inherits many nice properties of the heat kernel. It is invariant to isometric deformations, and not sensitive to noise or even slight topological changes. It is multi-scale: as $t$ gets larger, the eigenfunctions corresponding to large eigenvalues play a smaller role, hence only the main features of the shape detected by small eigenvalues matter. Most importantly, $HKS$ is almost as informative as the family of functions $h_t(., .)_{t>0}$ (see Theorem 1 in [80]). At the same time, it reduces the family of two-variables kernel functions to a family of single-variable functions, and is hence more succinct and much easier computationally.

**HKS Maxima.** It is well-known [39] that as $t \to 0$, there is an asymptotic expansion of the $HKS$ function at every point $x \in M$ of the form:

$$HKS_t(x) = h_t(x, x) = (4\pi t)^{-d/2} \sum_{i \geq 0} a_i t^i,$$
where \( a_0 = 1 \) and \( a_1 = \frac{1}{6}s(x) \) with \( s(x) \) being the scalar curvature at point \( x \). For a 2-manifold \( M \), \( s(x) \) is simply the Gaussian curvature at \( x \). Thus intuitively, the heat diffusion for small \( t \) is governed by intrinsic curvature. Heat tends to diffuse slower at points with positive curvature and faster with negative curvature. This suggests that critical points of \( HKS \) correspond to features of the shape, where maxima of \( HKS \) capture the tips of protrusions or the bottoms of concave areas. Hence, we propose to use \( HKS \) maxima as feature points for shape matching. It turns out that we can select only a handful of persistent feature points for matching.

### 3.3 Data Simplification

The heat kernel signature provide an object which we aim to preserve as much as possible during the sub-sampling process. Specifically, given a point cloud with vertex set \( V = \{v_1, \ldots, v_n\} \), we fix a small time step \( \tau \) (considered as the unit time), and construct the Voronoi-Laplace using the method described in the previous chapter. Instead of computing its eigenvectors which is costly, we construct a discrete heat operator w.r.t. time \( \tau \) by \( H_{\tau \tau} = I - \tau D^{-1}L \), where \( D \) is a diagonal matrix with \( D[i][i] = \frac{1}{\pi h} \sum_k A_k e^{-\frac{\|v_i - v_k\|^2}{4h^2}} \), and \( A_i \) is one-third of the total area of all triangles incident to vertex \( v_i \). Intuitively, \( D[i][i] \) is the total weights of all other vertices w.r.t. \( v_i \) used when computing the mesh-Laplacian. This discrete heat operator is derived by using the first order Taylor expansion of the relation \( \mathcal{H}_t = e^{-t\Delta} \).

By definition, \( HKS_t(x) = h_t(x, x) = \mathcal{H}_t \delta_x \), where \( \delta_x \) is the Dirac delta function \( \delta_x(y) = 0 \) for all \( y \neq x \) and \( \int_M \delta_x(y)dy = 1 \). Translated into the discrete setting, this means that \( HKS_{\tau}(v_i) = H_{\tau \tau} u_i[i] \), where the unit vector \( u_i = [0, \ldots, 0, 1, 0, \ldots, 0] \) is
the discrete analog of the Dirac delta function \( \delta_{v_i} \). The vector \( H_t \cdot u_i \) is the counterpart for function \( H_t \cdot \delta_{x} \). Observe that \( HKS_{\tau}(v_{i}) \) is simply the \( i \)th diagonal element \( H_{t \tau}[i][i] \) of the matrix \( H_{t \tau} \). Furthermore, \( u_i \) being the initial heat distribution, the heat distribution after \( \alpha \tau \) time is \( H_{t \alpha \tau} \cdot u_i \), implying that \( HKS_{\alpha \tau}(v_{i}) = H_{t \alpha \tau}[i][i] \).

This suggests approximating \( HKS \) values at time \( \alpha \tau \) by taking the diagonal of the matrix \( H_{t \alpha \tau} \). Computing \( H_{t \alpha \tau} \) by matrix multiplication is much more efficient for small \( \alpha \) than computing the eigen-decomposition of the Laplacian matrix. However, this method is only accurate when \( \alpha \) is small. Therefore we do not directly use this method to approximate the \( HKS \) for large \( t \) values. Instead, we take a specific small \( \alpha \) and approximate the function \( HKS_{\alpha \tau} \) as described above. \( HKS_{\alpha \tau} \) encodes the short-time behavior of heat propagation and reflects the local geometry of the input surface \( K \). We sub-sample \( K \) by keeping only the set of local maxima \( \hat{V} \) of \( HKS_{\alpha \tau} \) which tend to include all future feature points (\( HKS \) maxima for larger \( t \)). The parameter \( \alpha \) controls the resolution of simplification. This process can also be repeated.

### 3.4 One Application

Here we give one real application of this data simplification method, where it is applied in partial shape retrieval. [30]

The matching of a partial or an incomplete model against a complete one cannot rely on features that are too global. At the same time, any matching relying on only local measures becomes susceptible to noise caused by small perturbations. Therefore, we need something in between which can describe shape features at different scales. The Heat Kernel Signature (\( HKS \)) [80] bears this multi-scale property.
Figure 3.1: Given a query Armadillo model that is pose-altered, incomplete, and partially scanned, we can first compute the heat kernel signature function at a certain scale, and then extract a set of $HKS$ maxima (red dots) using persistent homology. Feature vectors computed at these maxima are then used to search for the most similar models, be it complete, partial, or incomplete, in a shape database. A few top matches are shown. The black curves are the boundary curves of either partial or incomplete models. Correspondence between segmentations of different models is shown with consistent coloring.

Given a surface $M$ with an initial distribution of unit heat concentrated at any point $x \in M$, the $HKS$ at $x$ at time (i.e., duration of the flow) $t$ provides the amount of heat retained at $x$ after heat dissipates for time $t$ according to the well-known heat equation. This heat dissipation is determined by the intrinsic geometry of $M$ and the influence of shape features on the heat flow can be controlled by regulating the time. Small time scales dissipate heat over a small region and thereby allow local features to regulate the heat whereas large time scales allow global features to exert more influence. This suggests that one can use $HKS$ at different time values to describe features at multiple scales.

The distribution-based features tend to be less discriminating for partial matching. The $HKS$ function is more likely to remain similar on a surface and its partial counterpart when $t$ is relatively small. However, a small diffusion interval tends to increase the local variation in the $HKS$ function values, and makes it more sensitive.
to noise. To counter this, the tool of persistent homology \cite{31} is brought in to identify important features.

The critical points, in particular, the maxima of $HKS$, serve as good candidates for feature points. Only persistent features are considered, which are $HKS$ maxima that persist beyond a given threshold. To compute such maxima, instead of using the standard persistent algorithm, a region merging algorithm is employed that bypasses detecting persistence values of all critical points and focuses only on eliminating those maxima that are not persistent.

The persistent feature points, together with a multi-scale $HKS$ description at each point, provide a concise yet discriminating shape descriptor for the input surface, which is also robust under near-isometric deformations and partial occlusions. In \cite{30}, experimental results show that this algorithm is able to match partial, incomplete, and pose-altered query shapes in a database of moderate size efficiently and with a high success rate. For more details of the partial shape retrieval method, see \cite{30}.

For large input data, directly computing $HKS$ is prohibitively expensive, we have to simplify the input data in order to obtain a reasonable running time. Our data simplification method introduced in the above section would lend itself to this task. Both $Qslim$-simp and $HKS$-simp can be employed. For an elephant data set with 1.5 million vertices, $Qslim$-simp takes 37 seconds and $HKS$-simp takes 1032 seconds to simplify it to 10K vertices, and then to compute its persistent $HKS$ maxima. However, we observe that $HKS$-simp tends to preserve the $HKS$ function information better. The experiment result is shown in the next section.
3.5 Experiment

To see how our algorithm approximates the HKS function, we compare it with an approach that first simplifies the input mesh using QSLIM \[35\]. Specifically, for each test model, we compare the reconstructed HKS functions by our simplification and QSLIM simplification with the “ground truth” (computed using the original mesh). The error is measured by both averaged $L_1$-norm distance and by statistical correlations (SC). (The error under the root-mean-square distance shows similar trend.) Some results are shown in Table 3.5 for different time values. All the models here originally have 30K vertices, and are simplified to 3K vertices. The unit time $\tau$ is set to be 0.0002 across all models to compute Laplacian eigenvectors. We have not reported error for larger initial meshes, as the time to compute the ground truth becomes prohibitive.

<table>
<thead>
<tr>
<th>Models</th>
<th>time $t$ used for computing $HKS$</th>
<th>(5\tau)</th>
<th>(40\tau)</th>
<th>(100\tau)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dragon</td>
<td>ours</td>
<td>4.90 / .882</td>
<td>0.22 / .998</td>
<td>0.09 / .999</td>
</tr>
<tr>
<td></td>
<td>QSLIM</td>
<td>26.55 / .363</td>
<td>0.77 / .981</td>
<td>0.33 / .999</td>
</tr>
<tr>
<td>Armadillo</td>
<td>ours</td>
<td>3.51 / .868</td>
<td>0.93 / .984</td>
<td>0.35 / .993</td>
</tr>
<tr>
<td></td>
<td>QSLIM</td>
<td>4.07 / .830</td>
<td>0.99 / .972</td>
<td>0.37 / .991</td>
</tr>
<tr>
<td>Neptune</td>
<td>ours</td>
<td>9.83 / .927</td>
<td>0.50 / .994</td>
<td>0.23 / .999</td>
</tr>
<tr>
<td></td>
<td>QSLIM</td>
<td>26.03 / .919</td>
<td>1.12 / .976</td>
<td>0.53 / .996</td>
</tr>
</tbody>
</table>

Table 3.1: Each entry: the averaged $L_1$-error / Statistical correlation between the approximated HKS and ground truth. All original meshes are of size 30K, and they are simplified to 3K by either our method or QSLIM.
3.6 Discussions

In this chapter, we proposed a data simplification method which aims to preserve the spectral property as much as possible. This approach utilizes the informative heat kernel signature. By approximating the heat diffusion process with matrix multiplication, we get a better approximation of HKS with sub-sampled data. The experimental results demonstrate the better performance over other methods.
Chapter 4: Eigen-gradient from Point Cloud Data

In the previous two chapters, we explored how to construct discrete spectral structures from point sampled data, including building the Voronoi-Laplace operator and a data simplification method which preserve as much as possible its spectral property. Once we have the spectral structure, our next direction turns to how to apply these discrete spectral structures to specific geometry processing applications. In this chapter, we introduce our first application: gradient estimation for functions defined from point cloud data.

4.1 Introduction

The gradient of a function defined on a Riemannian manifold is one of the most important differential objects in data analysis. It has been used for a broad range of applications, from partial differential equations (PDEs) in scientific computing, to modeling deformations and simulations in graphics and visualization, to feature identification in image processing, and to inference problems in machine learning. It is often the first step before further geometric quantities, such as high-order derivatives, critical points, and Morse-Smale complex, can be computed.

Very often in practice, we need to compute gradients where the input function is only available at discrete points sampled from the underlying manifold, and the
manifold is given by a mesh or simply a point cloud. While many methods have been used when the input function is defined over a mesh, computing gradients and related quantities such as critical points, of a function from a point cloud embedded in $\mathbb{R}^k$ is a non-trivial task. Furthermore, even when a mesh structure is given, being a differential operator, gradient computation is sensitive to noise both in the input function and in the underlying manifold. Hence it is highly desirable to be able to smooth gradient fields at different scales.

Given a (finite element) mesh, the simplest and most common way to compute gradient is to interpolate the input function linearly (or in higher order) within each mesh element, based on its values at vertices of the mesh. This mesh-approximation of gradient is somewhat sensitive to both the shapes of mesh elements and noise [78], as its value at a vertex $v$ depends only on the one-ring neighbors of $v$. For functions defined on a $d$-dimensional domain in $\mathbb{R}^d$, there is a rich literature in numerical analysis on the powerful finite element methods (FEMs), which can interpolate the input function non-linearly based on non-local neighborhoods. The same idea can also be extended to develop meshless finite elements methods for a point cloud sampled from a $d$-dimensional domain in $\mathbb{R}^d$. See [33] for a good survey. However, both FEMs and meshless FEMs are usually computationally inefficient in handling low-dimensional manifolds (such as surfaces) embedded in high-dimensional space, as well as changing the resolution at which we want to approximate the gradient.

In many tasks, the input is simply unstructured point sets. See [2, 38] for a good introduction of processing point clouds in graphics. For points inputs, gradients are usually estimated by solving a local optimization problem based on Taylor expansions. Specifically, let $P$ be the set of discrete points such that the function values of $f$ are
available. The gradient $\nabla f(p)$ at a point $p \in P$ can be estimated as the best vector $g^*$ minimizing the following error:

$$\sum_{q \in \text{Neighbor}(p)} w(p, q) \left| f(p) + g^T(q - p) - f(q) \right|^s,$$

where $\text{Neighbor}(p) \subseteq P$ denotes a certain set of neighboring vertices of $p$, $w(p, q)$ is a weighting function, and $s$ is a positive integer, typically 1 or 2. For example, Sibson [79] suggested to use the so-called natural neighbors of $p$ as $\text{Neighbor}(p)$, and choose $s = 1$ in the above optimization problem. This gradient estimation method is implemented in the Computational Geometry Algorithms Library (CGAL). For points sampled from a Riemannian manifold embedded in a high-dimensional space, Mukherjee et al. took a regularized version of this optimization and choose $s = 2$ [64, 85]. They provided some theoretical guarantees of the accuracy of their estimates for points randomly sampled from probabilistic distribution. To make the estimation robust to noises in the underlying manifolds, one can use moving-least-squares (MLS) surfaces to locally approximate the surface [5, 50]. To estimate gradients at a coarser scale, one can potentially take a larger neighborhood and solve the optimization problem based on more points around each point $p$. This unfortunately means that the time complexity to estimate gradients is higher when the resolution is coarser.

In this chapter, we aim at developing a unifying framework for approximating and smoothing gradients from discrete inputs that can be meshes and point clouds. Our framework is based on a novel view of considering gradients under a different metric space where the computation and smoothing are made easier, especially for point cloud inputs. The main contributions are as follows:
(1) We initiate the study of gradients under a different metric on the underlying manifold $M$, instead of the usual metric induced from the ambient space. While gradient depends on the metric, certain related quantities such as critical points are metric independent, and metric may not be essential for some gradient-related applications (such as gradient descending to find global minimum).

(2) We compute gradients by mapping the input manifold $M$ to the space spanned by the eigenfunctions of the so-called Laplace-Beltrami operator of $M$. We show that when a certain Gaussian kernel is used to approximate the Laplace operator, this mapping is an isometry up to a scaling factor. Hence we can recover the original gradients from the eigen-gradients easily.

(3) Laplacian eigenfunctions provide a natural basis for functions defined on the manifold $M$, and the mapping of $M$ to its Laplacian frequency domain provides a natural way to smooth eigen-gradients. Contrary to what is typical in the original space, computing gradients at a coarser level in the eigenspace takes less time than at a higher resolution, and it is possible to smooth both the function itself and the underlying manifold simultaneously.

Our approach can be applied to any $d$-dimensional compact Riemannian manifold embedded in $\mathbb{R}^m$. In this chapter, we focus on surfaces embedded in $\mathbb{R}^3$.

4.2 Eigen-space and Diffusion Metric

Consider a smooth $d$-dimensional, compact Riemannian manifold $M$ isometrically embedded in $\mathbb{R}^m$. Given a scalar function $f : M \to \mathbb{R}$, when we talk about its gradients, we usually implicitly assume the natural metric on $M$ induced from the Euclidean metric in the ambient space $\mathbb{R}^m$. In this chapter, we map $M$ into a spectral
domain and compute gradients under the diffusion metric associated with it. (A good exploration of general diffusion metrics can be found in [48].) Specifically, we will use the heat diffusion metric and its variant.

**Diffusion metric.** Assume that \( \lambda_0 \leq \lambda_1 \leq \ldots \) are sorted in increasing order. Note that \( \lambda_0 = 0 \) since the largest eigenvalue of \( \mathcal{H}_t \) is \( \rho_0 = 1 \). We now embed \( M \) to \( L^2(M) \) by the following diffusion map: for any \( x \in M \),

\[
\Phi(x) = [e^{-t\lambda_0/2}\phi_0(x), e^{-t\lambda_1/2}\phi_1(x), \ldots]^T.
\]

It is easy to show that \( \Phi(M) \) is indeed an embedding of \( M \). This embedding map \( \Phi \) is similar to the diffusion map in [48] and the GPS embedding in [76], but with different weights for each coordinate, and is a special case of the heat-kernel embedding studied in [17].

The eigenspace \( L^2(M) \) is a vector space, and adapts a natural Euclidean distance. For two points \( x, y \in M \), the diffusion distance between them w.r.t to \( t \), denoted by \( D_t(x, y) \), is simply the Euclidean distance between \( \Phi(x) \) and \( \Phi(y) \). It then follows from Eqn (3.3) and the fact \( \rho_i = e^{-t\lambda_i} \) that

\[
D_t^2(x, y) = \|\Phi(x) - \Phi(y)\|^2 = \sum_{i \geq 0} e^{-t\lambda_i} (\phi_i(x) - \phi_i(y))^2
\]

\[
= \sum_{i \geq 0} e^{-t\lambda_i} (\phi^2_i(x) + \phi^2_i(y) - 2\phi_i(x)\phi_i(y))
\]

\[
= h_t(x, x) + h_t(y, y) - 2h_t(x, y).
\]

An intuitive stochastic view of the diffusion distance is as follows: \( D_t(x, y) \) is small if a random walk (Brownian motion) on \( M \) from \( x \) reaches \( y \) within time \( t \) with high probability. Compared to the Euclidean or geodesic distances, the diffusion distance is stable with respect to noise and even small topological changes.
4.3 Eigen-gradient

We propose to compute gradients in the eigenspace. Below we first introduce the new eigen-gradients for smooth manifolds. We then describe how to compute it from meshes and point clouds.

4.3.1 Smooth Manifolds

Given an input manifold $M$ embedded in $\mathbb{R}^m$, set $\Omega = \Phi(M)$. The map $\Phi : M \rightarrow \Omega$ is a homeomorphism [17]. Thus given a function $f : M \rightarrow \mathbb{R}$, its push-forward $\mathcal{F}$ is well-defined by $\mathcal{F}(\Phi(x)) = f(x)$. Now let $g_M$ denote the natural Riemannian metric on $M$ induced from $\mathbb{R}^m$, and $g_\Omega$ the metric on $\Omega$ induced from the Euclidean distance in $L^2(M)$. Given a point $x \in M$, $\nabla_M f(x)$ is the gradient of $f$ at $x \in M$ under metric $g_M$, which we also refer to as the *original gradient* in this chapter. The *eigen-gradient of $f$ at $x$*, denoted by $\nabla_E f(x)$, is defined as the gradient of $f$ under the pullback of the metric $g_\Omega$ onto $M$.

Specifically, let $T_Mx$ and $T_\Omega x$ denote the $d$-dimensional tangent space of $M$ at $x$ and of $\Omega$ at $\Phi(x)$, respectively. Let $v = \nabla_\Omega \mathcal{F}(\Phi(x))$ be the gradient of $\mathcal{F}$ at $\Phi(x)$ in the eigenspace under metric $g_\Omega$, and $u \in T_Mx$ the pullback of the vector $v$ under the linear map $D\Phi_x : T_Mx \rightarrow T_\Omega x$; that is, $D\Phi_x(u) = v$. The eigen-gradient $\nabla_E f(x) = \frac{\|v\|u}{\|u\|}$ is simply the vector in the direction of $u$ with length $\|v\|$.

**Computation in the Eigenspace**

Since the eigenfunctions $\phi_i$s of the Laplace operator (or more generally, of a well-behaved diffusion operator) form a set of orthonormal basis for the space of square integrable function $L^2(M)$, given $f \in L^2(M)$, we can represent $f = \sum_{i \geq 0} \alpha_i \phi_i$ as a
linear combination of $\phi_i$s. By definition, we have that

$$\mathcal{F}(\tilde{x}) = \mathcal{F}(e^{\frac{t\lambda}{2}}\phi_0(x), e^{\frac{t\lambda}{2}}\phi_1(x), \ldots) = \sum_{i \geq 0} \alpha_i \phi_i(x),$$

where $\tilde{x}$ denotes $\Phi(x)$. Hence the partial derivative of $\mathcal{F}$ with respect to any eigenfunction (coordinate in $L^2(M)$) is $\partial\mathcal{F}/\partial \phi_i|_{\tilde{x}} = e^{\frac{t\lambda}{2}}\alpha_i$, which is independent of $\tilde{x}$. Now construct a vector $V$ in $L^2(M)$ as

$$V = [e^{\frac{t\lambda}{2}}\alpha_0, e^{\frac{t\lambda}{2}}\alpha_1, \ldots]^T.$$

At any point $\tilde{x} \in \Omega$, the gradient vector of $\mathcal{F}$ at $\tilde{x}$ is simply the projection of $V_x$ onto $T\Omega_x$, where $V_x$ is the vector $V$ translated to the base point $\tilde{x}$. See Figure 4.1 for an illustration.

This implies that once the vector $V$, which only depends on the input function $f$ and $M$, is given, the gradient $\nabla \mathcal{F}(\tilde{x})$ can be computed by locally approximating the tangent space at every point in the eigenspace. $\nabla_E f(x)$ can then be computed by pulling back $\nabla \mathcal{F}(\tilde{x})$ to $TM_x$. 
Smoothing Gradients

Since the eigenfunctions with high eigenvalues correspond to harmonics with high frequency, a natural way to smooth the gradients in the eigenspace is by taking the subspace of $L^2(M)$ spanned by the top $K$ Laplacian eigenfunctions with lowest eigenvalues. Specifically, consider the truncated eigenmap $\Phi_K(x) = [\phi_0(x), \phi_1(x), \ldots, \phi_K(x)]^T$ and map $M$ to $\Omega_K = \Phi_k(M)$. To compute gradients at a coarser level, we simply take a smaller $K$, compute the vector $V_K = [e^{\frac{i\lambda_0}{2}}\alpha_0, e^{\frac{i\lambda_1}{2}}\alpha_1, \ldots, e^{\frac{i\lambda_K}{2}}\alpha_K]^T$ and project it to the tangent space of $\Omega_K$ at $\Phi_K(x)$.

There are two types of smoothing simultaneously involved in the above approach: The truncation of the coefficient vector $V$ corresponds to removing the higher frequency components of the input function $f$, thus smoothing $f$; while the projection onto the subspace of $L^2(M)$ spanned by the first few eigenfunctions corresponds to mapping the manifold to its lowest frequency modes, thus removing details from $M$. In this way, gradients estimated are robust to noise both in the input function and in the underlying manifold. In practice, one can control the resolutions of two smoothing operations separately, by taking the top $K_1$ coefficients for $V$, while projecting $M$ to $\Omega_{K_2}$, for $K_1 \leq K_2$.

In order to be able to pullback the gradients in the eigenspace, it is necessary that the map $\Phi_K$ is a local homeomorphism; that is, $\Omega_K$ is an immersion of $M$ into $\mathbb{R}^K$. Indeed, it follows from Theorem 2.2.1 in [71] that this is the case for $K \geq C$, where $C$ is some constant depending only on the intrinsic properties of $M$ and it appears to be rather small in the models we test in practice (Results in [71] are much stronger. The local homeomorphism is only a corollary of them).
Gaussian Diffusion

The relation between the diffusion metric and the original metric depends on which specific diffusion operator that we use. In this chapter, we use the heat diffusion operator due to its natural physical interpretation and the nice properties its spectrum and eigenfunctions have. However, the analytical form of the heat operator is only known for very limited families of manifolds. In practice, the heat kernel \( h_t(x, y) \) is usually approximated by the following Gaussian kernel: 

\[
g_t(x, y) = \frac{e^{-\|x-y\|^2}}{4\pi^d t^{d/2}}.
\]

Indeed, \( h_t = g_t \) when the underlying manifold is the Euclidean space \( \mathbb{R}^d \). For general manifolds, Belkin and Niyogi [11, 13] gave explicit bounds of the difference between \( h_t \) and \( g_t \), as well as between the resulting diffusion operators, and used that to approximate the Laplace operator with theoretical guarantees for points randomly and uniformly sampled. In this chapter, we also use the Gaussian diffusion to approximate the heat diffusion process (via our new discrete Laplace operator that we will describe later).

The Gaussian diffusion distance relates to the Euclidean distance in the original space by the following equations. This in turns leads to an explicit relation between the eigen-gradient and the original gradient, enabling us to reconstruct the original gradients based on eigen-gradients.

**Lemma 5.** Let \( \hat{D}(x, y) \) denote the Gaussian diffusion distance between \( x \) and \( y \). Then we have that

\[
\sqrt{2t(4\pi t)^{d/2}}\hat{D}(x, y) = \|x - y\| + O(\|x - y\|^3/t).
\]
Proof. : Set $r = \|x - y\|$. By Eqn (4.1), we have
\[
\hat{D}^2(x, y) = g(x, x) + g(y, y) - 2g(x, y)
= \frac{1}{(4\pi t)^{d/2}}(2 - 2e^{-\frac{r^2}{4t}}) = \frac{1}{(4\pi t)^{d/2}}\left[\frac{r^2}{2t} + O(r^4/t^2)\right].
\]
Hence
\[
\sqrt{2t(4\pi t)^{d/2}} \hat{D}(x, y) = r\sqrt{1 + O(r^2/t)} = r + O(r^3/t).
\]

\[\square\]

Theorem 3. The eigen-gradient of $f$ is related to the gradient in the original metric space by:
\[
\nabla_E f(x) = \sqrt{2t(4\pi t)^{d/2}} \nabla_M f(x).
\]

Proof. : Lemma 3 implies that the Riemannian metric $g_M$ and the diffusion metric $g_D$ on $M$ are isometric up to a scaling factor $C = \sqrt{2t(4\pi t)^{d/2}}$. To see this, fix a point $x$ in $M$ and take any smooth curve $\gamma(s)$ through $x$ with $\gamma(0) = x$. Let $\tilde{\gamma}(s)$ denote the image of the curve $\gamma(s)$ under the map $\Phi$; that is, $\tilde{\gamma}(s) = \Phi(\gamma(s))$. Now consider the norm of the tangent of $\gamma(s)$ and of $\tilde{\gamma}(s)$ at $s = 0$ under metrics $g_M$ and $g_0$, denoted by $\|\gamma'(0)\|_M$ and $\|\tilde{\gamma}'(0)\|_0$, respectively. First note that by Lemma 3.2 in [15], given any curve $\pi(s)$ in $\mathbb{R}^n$, the length of the curve between two points that are close enough relates to the Euclidean distance between these two points by the following inequality:
\[
\text{len}(\pi(s), \pi(s')) = \|\pi(s) - \pi(s')\| + O(\|\pi(s) - \pi(s')\|^3).
\]
This, combined with Lemma 5, implies that for two points close enough on $\gamma$, the length of the curve between them and that between their images in $\tilde{\gamma}$ satisfy:

$$C \cdot \text{len}_\Omega(\tilde{\gamma}(s), \tilde{\gamma}(s'))$$

$$= \text{len}_M(\gamma(s), \gamma(s')) + O(\|\gamma(s) - \gamma(s')\|^3_M),$$

where the big-O notation hides terms in $t$ which is a constant when computing gradients. It then follows that

$$\|\gamma'(0)\|_M = \lim_{s \to 0} \frac{\text{len}_M(\gamma(0), \gamma(s))}{s} = C \cdot \lim_{s \to 0} \frac{\text{len}_\Omega(\tilde{\gamma}(0), \tilde{\gamma}(s))}{s} = C\|\tilde{\gamma}'(0)\|_\Omega.$$ 

Now, take any two vectors $u$ and $v \in T_{M_x}$ with their push-forwards $\tilde{u}, \tilde{v} \in T\Omega_x$, we have that

$$\langle u, v \rangle_M = C^2 \langle \tilde{u}, \tilde{v} \rangle_\Omega,$$

through polarization, as

$$\langle u, v \rangle_X = \langle u + v, u + v \rangle_X - \langle u - v, u - v \rangle_X,$$

where $\langle \, , \, \rangle_X$ denotes the inner product under metric $g_X$. Since the constant $C$ remains the same for all points on $M$, the map $C\Phi$ induces the same inner product as $\langle \, , \, \rangle_M$ and is thus an isometry. The claim then follows from elementary differential geometry. 

### 4.3.2 Discrete Settings

We now consider the discrete settings where the input data is a mesh $K$ with vertex set $P$, or simply a point cloud $P$, and the function values of $f$ are given at vertices $P$. Set $n = |P|$. The proposed algorithm is summarized as follows, and we explain the implementation of each step after it.
Preprocessing: Construct discrete Laplace operator $L$ and its top $N$ eigenvalues 

\{\lambda_i\} and eigenfunctions \{\phi_i\}.

**Step 1:** Compute the coefficients $\alpha_i$s where $f = \sum \alpha_i \phi_i$. Set up the vector 

\[ V = [e^{\frac{\lambda_1 t}{2}} \alpha_1, \ldots, e^{\frac{\lambda_N t}{2}} \alpha_N]^T. \]

**Step 2:** Map every vertex $x \in P$ to $\mathbb{R}^N$ by $\Phi_N(x) = [e^{-\frac{\lambda_1 t}{2}} \phi_1(x), \ldots, e^{-\frac{\lambda_N t}{2}} \phi_N(x)]^T$.

**Step 3:** For each vertex $x$, approximate the tangent space $T\Omega_x$ at $\Phi_N(x)$. Project $V$ onto $T\Omega_x$ to obtain $v_x$. Compute the pullback of $v_x$ to the tangent space $T M_x$, denoted by $u_x$. The estimated gradient at $x$ is 

\[ \|v_x\| u_x \sqrt{2(4\pi t)^{d/2}} \|u_x\|. \]

**Pre-processing**

We first construct the Voronoi-Laplace operator and do the eigen-decomposition for the Laplace matrix as described in [2,3]. We consider the construction of the discrete Laplace operator as a pre-processing step because first, it is independent of the input function given. Hence once constructed, it can be used for multiple functions, and for different resolutions of the same function. It can also be used for the same shape under isometric deformation (as will be demonstrated in our experiments in the next section). Furthermore, since the Laplace operator encodes all intrinsic geometry information, one can expect a general framework of estimating various geometric quantities directly from point clouds based on the Laplace operator constructed (see e.g [11, 59, 73] for data analysis under this framework). Our approach is simply one application of such *spectral point-cloud processing* framework.

69
Steps 1 To 3

Given a square integrable function $f$ on $M$, it can be represented as a linear combination $\sum_i \alpha_i \phi_i$ of Laplacian eigenfunctions $\phi_i$s, where $\alpha_i = \langle f, \phi_i \rangle = \int_M f \phi_i d\nu$ in the continuous setting. As mentioned above, this integral can be approximated by the $D$-inner product $\langle f, \phi_i \rangle_D$ in the discrete setting, for both point clouds and meshes, where $D$ is the area-weight diagonal matrix that we constructed above.

Implementing Step 3. For each vertex $x$, let $knn(x) = \{y_1, \ldots, y_k\}$ denotes the set of $k$-nearest neighbors of $x$ for some constant $k$ ($k$ is typically 10 in our experiments). We approximate the tangent space at $x$ or at $\Phi_N(x)$ by finding the best $d$-dimensional space fitting $knn(x)$ under least square error measure. To pull back the gradient vector $\nu_x$ from $T\Omega_x$ to $TM_x$, we need to approximate the linear map $D\Phi : TM_x \to T\Omega_x$, which is simply a $d \times d$ matrix (2 by 2 matrix for surfaces). This is achieved by finding the best $d \times d$ square matrix that minimizes the following least square error:

$$\sum_{i=1}^k \|D\Phi(u_i) - \tilde{u}_i\|^2,$$

where $u_i$ (resp. $\tilde{u}_i$) is the projection of the vector $y_i - x$ (resp. $\Phi(y_i) - \Phi(x)$) onto $TM_x$ (resp. $T\Omega_x$). Specifically, let $U$ (resp. $\tilde{U}$) denote the matrix with $u_i$s (resp. $\tilde{u}_i$) as column vectors. We have that $D\Phi = U^+\tilde{U}$ where $U^+$ is the Moore-Penrose pseudo-inverse of $U$. Since the eigen-gradient has the same magnitude as $\nu_x$, the gradient in the eigenspace. Hence this pullback operation is not necessary if we only need the magnitudes of gradients in the applications.
4.4 Experiments

We present two sets of experiments. The first set compares the eigen-gradients from point clouds with the ground truth, as well as with gradients computed by a local quadratic fitting method. We then show how to use the eigen-gradients to simplify the critical points of a function. All eigen-gradients in this section are computed directly from point clouds. In general, the choice of $t$ does not matter much as long as $t$ is much larger than the square of the average distance ($d_0$) of any point to its $k$ nearest neighbors. All experiments in this chapter use $t = 30d_0^2$.

Figure 4.2: from (a) to (e) the gradient field reconstructed from 1500, 50, 20, 10, and 5 eigenvectors respectively. in (f) and (g) the magnitude and angular error plots with their standard deviation.
4.4.1 Eigen-gradients

We sample the unit sphere $S$ non-uniformly to produce a set of about 2000 points, and consider the function $f(x, y, z) = (\sin(5x)+5)(y^3+5)e^z$ — the choice of the sphere is so that we can know the ground truth, and the input function is an arbitrary test function combining polynomials and exponential functions. Results from other test functions are similar. In the top row of Figure 4.2 we show a sequence of the gradients computed with decreasing number of eigenvectors. The plots on the bottom row show the average error between the angle and the magnitude (length) of our eigen-gradient vectors compared with the true gradient vectors. We can see that in general, few eigenvectors (with lowest eigenvalues) suffice to reconstruct the gradient faithfully (with small error). In practice, we usually compute only the top 100 eigenvectors.

In Figure 4.3 we study the performance of our eigen-gradients under noise. The eigen-gradients are computed with the top 100 eigenvectors. In addition to ground truth, we also compare our method with a discrete gradient computed from input points $P$ by a local fitting method. Specifically, for every point $p$, we take its $k$-nearest neighbors ($k = 15$ in this experiments), and locally fit a quadratic function based on these neighboring points. We refer to the gradient computed from the fitting function as Fit-gradient. This method inherently smoothes both the underlying manifold (via fitting a tangent space) and the input function (via fitting a quadratic function) but in a local way. Hence it is reasonably robust under small amount of noise. The reconstruction errors of these methods are shown in Figure 4.3 (b). In the absence of noise (function noise or surface noise), the quadratic fitting gives better results as measured by the average angular deviation and average magnitude deviation from the ground truth, however once noise is introduced the eigen-gradients method shows
improvement over the quadratic fitting method. We have also performed 15 Laplace smoothing iterations on both the input function and the input surface, and even though the Fit-gradient field appears smoother afterwards, the reconstruction error does not improve much in the case of function smoothing (7.7 degree angular, 14.95% magnitude for 3% function noise), and it gets worse in the case of surface smoothing for surface noise.

![Diagram](image)

<table>
<thead>
<tr>
<th></th>
<th>Avg. error in gradient vectors</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>angle</td>
<td>length</td>
</tr>
<tr>
<td>no noise</td>
<td>2.72</td>
<td>4.6%</td>
</tr>
<tr>
<td>1% surf-noise</td>
<td>2.98</td>
<td>8.52%</td>
</tr>
<tr>
<td>3% func-noise</td>
<td>3.51</td>
<td>6.26%</td>
</tr>
<tr>
<td>both noises</td>
<td>3.33</td>
<td>4.58%</td>
</tr>
</tbody>
</table>

Figure 4.3: top row from left to right, the ground truth gradient field for \( f(x, y, z) = (\sin(5x) + 5)(y^3 + 5)e^z \), gradient field by eigen method for 1% surface noise and 2% function noise, and gradient field by quadratic fit for the 1% surface noise and 2% function noise. The table shows the angular and magnitude errors for different noise configurations.

Our eigen-gradient is not only robust with respect to noise, but more importantly, it provides a simple way to simplify the gradient field at multiple scales, by using fewer
eigenvectors. An example is shown in Figure 4.4 where we have a molecular surface with 8952 sample points. The input function is the so-called Connolly function widely used to capture protrusions and cavities in molecules [24]. Figure 4.4 (a) shows the eigen-gradient field reconstructed using 1000 and 50 eigenvectors. Note that some small bumps (low-persistence critical points) are smoothed as we reduce the number of eigenvectors used. Figure 4.4 (b) shows the Fit-gradient fields computed by the quadratic fitting method (left) and the simplified version computed by first performing 30 iterations of Laplace smoothing operators (right).

To compare these two types of simplifications more quantitatively, we introduce a quantity to measure the smoothness of an input vector field $V = \{v(p)\}_{p \in P}$: given a sample point $p$, the local smoothness $\rho(p)$ at $p$ is the average of the dot-product between $v(p)$ and the vectors associated with each of its 15-nearest neighbors. The smoothness $\rho(V)$ of the entire vector field $V$ is $\rho(V) = \frac{1}{|P|} \sum_{p \in P} \rho(p)$. It takes the local-fitting method 30 iterations of Laplace smoothing operations (right image in Figure 4.4 (b)) to achieve a similar smoothness as the eigen-gradient field computed from 50 eigenvectors (right image in Figure 4.4 (a)). Simplifying it at larger scale will require even more number of Laplace smoothing iterations. In other words, a coarser-resolution gradient field requires more iterations of smoothing when using quadratic fitting method (which eventually is equivalent to increasing the number of neighboring points that we need to consider for each input point). Hence it requires more time. On the other hand, smoothing using our method simply means using less eigenvectors, while in the eigenspace, still the same (small) number of neighboring points are needed to compute the projection (thus the time needed is in fact slightly smaller).
4.4.2 Critical Points

Since $\Phi$ induces a homeomorphism between $M$ and $\Omega$, $D\Phi|_{x}$ can be considered as a change of coordinate system for the local chart at $x$. Hence by the Morse Lemma [63], the criticality of a point and its index are not affected.

Claim 1. Given a morse function $f : M \rightarrow \mathbb{R}$, $x \in M$ is a critical point of $f$ with index $i$ if and only if $\Phi(x)$ is a critical point of $F$ in $\Omega$ with index $i$.

When only point cloud is given, by virtue of the above claim, we can compute the critical points using solely the magnitudes of eigen-gradients. Specifically, for every point $p$, we say that it is a critical point if its eigen-gradient value is close to zero, and it has the smallest magnitude among its $k$ nearest neighbors ($k = 10$ in our experiments). The latter criteria is because points close to a critical point may also have very small gradient value.

In Figure 4.5, we consider the following function defined on the unit sphere:

$$f(p) = \sin(\theta)\sin(6\theta)\sin(2\phi),$$

where $\theta$ and $\phi$ are the spherical coordinates of $p$. All
critical points computed by using the combinatorial mesh structure (shown in (a)) were recovered from the point cloud (with 2562 points) by the eigen-gradient method (shown in (b)). The recovered critical points either coincide exactly with mesh critical points or are slightly shifted version of them with maximum shifted distance of 0.14.

For general surfaces, critical points constructed by this eigen-gradient method tends to be a smoothed version of those computed from the mesh. Further reducing the number of eigenfunctions results in a further simplification of its critical points. Figure 4.6 shows this effect for a molecular surface (with 8952 points). In some sense, those less important critical points (w.r.t. the input function which captures protrusions and cavities) are removed.

4.5 Discussions

In this chapter, we proposed to approximate the gradients of a function defined over a point cloud (or a mesh) by computing it under the so-called diffusion distance
metric. One of the main advantages of this spectral approach is that one can simplify, at various scales, the gradient field with respect to both, the input function and the underlying manifold, in a unified and simple way. We demonstrate that our eigen-gradients can faithfully reflect true gradients, as well as effectively simplify them. We also present two preliminary applications for applying the eigen-gradients. Our results hold for points from any smooth and compact $d$-manifold embedded in $\mathbb{R}^m$. 

Figure 4.6: Left: critical points of the Connolly function computed from the mesh structure. Right: those computed from point clouds by the eigen-approach using 200 eigenvectors.
Chapter 5: Uniformization and Density Adaptation for Point Cloud Data

The final contribution of this thesis is another application, based on the Gaussian-weighted graph Laplacian and works purely in the point cloud setting. In this chapter we study the density adaptation problem that underlies many preprocessing tasks of points data. Specifically, given a set of points $P$ and $Q$ that samples an unknown surface $M$ and a target density function $g : M \rightarrow \mathbb{R}$, the goal is to adapt the set $Q$ of points that matches the target distribution $g$; The role of point cloud $P$ here is to provide an approximation of the hidden surface. The input points $Q$ does not have to be a subset of the $P$ and can either be sparser than $P$ or just $P$ itself. If the target $g$ is the uniform distribution, we also call this process the uniformization of $Q$. In practice, the target distribution may not be uniform, such as in adaptive sampling or anisotropic smoothing where it is desirable that the density of output points reflects local geometric feature sizes.

Related work. Given a set of points, projecting them onto an implicitly defined surface (such as the moving least square) is an important way to obtain point-set surface representations; e.g, [51, 1, 6]. Various further processing to “consolidate” (a term introduced in [1]) these points are proposed, including the simplification [68],
anisotropic smoothing [49] and geometry-aware resampling [67] of them. Recently, [54] introduced the Locally Optimal Projection (LOP) operator that moves a set of points \( Q \) to obtain a nice distribution for the hidden surface approximated by another set of input points \( P \). The LOP operator contains an explicit “repelling-force” term to guarantee output points are also well-separated. This algorithm can handle rather complex input data. An improved weighted-LOP operator was proposed in [44]. This, combined with a novel normal estimation and propagation algorithm, gives rise to a robust framework that can output a set of clean and evenly distributed points associated with accurate normal information.

These work typically focus on denoising, and there is often no clear requirement nor control over the density of output points, with the exception of [54] [44] which targets an uniform and even (well-separated) distribution. The current chapter considers an orthogonal problem where we aim to produce a set of points adapted to an arbitrary target distribution.

There also exists much work in refining point samples when a surface mesh is given. In particular, there has been an emerging line of work on producing points with the blue-noise property, which roughly means that these points, in addition to being randomly distributed, are also well-separated. Generating blue-noise samples for images have been extensively studied (and we omit its vast literature here). Current blue-noise sampling approaches for surface meshes usually deploys a Lloyd’s relaxation type of method [58]. In its simplest form, one Lloyd’s relaxation step computes on the input surface \( M \) the Voronoi diagram of the current set of points \( Q \). Each site in \( Q \) then moves to the centroid of its Voronoi cell so that the resulting positions are separated as far apart from one another as possible. This step is repeated till
Figure 5.1: (a): 5K input points $Q$ sampled from a random distribution far from the target distribution. The hidden surface is approximated by 50K points (not shown). (b): our output adapted to a target density distribution as shown in (c). The density map of our output is shown in (d). This is a hard case for relaxation-based methods due to the many tunnels. (e) shows the output density obtained by a PCD variant of the Lloyd relaxation method but with an input $Q'$ that is already very close to the target distribution (see Section 5.3 for explanation).

it converges. Adaptive blue-noise sampling on surfaces can be achieved by using weighted Voronoi cells and their variant \cite{7, 20, 52}.

While good blue-noise property can be achieved, it is not easy to have an accurate control over the density of output points under the Lloyd’s relaxation framework.

**Our contribution.** In this chapter, we present a simple and robust framework for this problem that is effective at both local refinement and global adjustment of points distribution. Our algorithm requires only the construction of the (Gaussian-weighted) graph Laplace operator, and is very easy to implement. Our algorithm uses certain information encoded in the graph Laplacian that is orthogonal to the mean-curvature flow. It is simple, effective, and robust to noise and small holes present in input points. One can use it to locally improve the quality of points, much similar to the wide use of the Lloyd’s relaxation type of methods. However, our algorithm works
in a purely points data setting, and can achieve both good density control and some blue-noise property (in the sense that output points are evenly spaced) even when the target distribution is far from input distribution. We believe that the simplicity and effectiveness of our algorithm will help advocate its practical usage as a preprocessing step to improve points quality.

5.1 Density Adaptation Algorithm

Assume we have a hidden surface M embedded in $\mathbb{R}^3$. Our input is (i) a set of $n$ points $P = \{p_1, \ldots, p_n\}$ sampled, not necessarily uniformly, from on and around M; (ii) another set of $m$ points $Q$, and (iii) a target density function $g : M \rightarrow \mathbb{R}^+$ (the form that $g$ is given is discussed in Section 5.1.2). The density adaptation problem aims to compute a new set of points $\hat{Q}$ that matches the target density function $g$.

The role of $P$ is to serve as a point-set approximation of the hidden surface M: Its distribution can be very different from the target distribution. $Q$ may or may not be a subset of $P$, and can be far sparser than $P$ (it is also possible that $Q = P$).

In Section 5.1.1 we describe how to move $Q$ to obtain a uniformly distributed set of points $\hat{Q}$, to illustrate the main idea. We then explain in Section 5.1.2 how to achieve a non-uniform distribution.

Gaussian-weighted graph Laplacian. We have mentioned Gaussian-weighted graph Laplacian in Section 1.2.2. It is light weighted, converges to the manifold Laplacian under the random uniform sampling condition. For general sampling condition, Gaussian-weighted graph Laplacian does not converge to the manifold Laplacian. This is an undesirable property for most applications and we turn to either Voronoi-Laplacian or PCD-Laplacian. Nevertheless, it is useful in this problem because the
difference between Gaussian-weighted graph Laplacian and the manifold Laplacian is connected to the density of the point distribution. We review Gaussian-weighted graph Laplacian in more details here.

Given a set of points \( Q = \{q_1, \ldots, q_m\} \) sampled from a surface (2-manifold) \( M \), the Gaussian-weighted (GW) graph Laplace operator \( L^Q_t \) is a linear operator such that given any function \( f : M \to \mathbb{R} \) with values available at points in \( Q \), we have:

\[
L^Q_t f(q_i) = \frac{1}{mt^2} \sum_{j=1}^{m} e^{-\frac{\|q_i - q_j\|^2}{t}} (f(q_i) - f(q_j)).
\] (5.1)

Although the above formulation involves all pairs of points \( q_i \) and \( q_j \), due to the exponential decaying effect of the Gaussian kernel, we only need to consider those \( q_j \) within \( O(\sqrt{t}) \) distance to \( q_i \) when estimating Eqn (5.1). Hence in practice, the construction of the GW-Laplace operator \( L^Q_t \) requires only the proximity graph of \( Q \).

There has been much theoretical study of this graph Laplacian and its variants. In the limit as the number of points \( m \to \infty \) and the parameter \( t \) tends to 0 at appropriate rates, and if the points \( Q \) are uniformly sampled from the manifold \( M \), then \( L^Q_t \) tends to the Laplace-Beltrami (also called the manifold Laplacian) \( \Delta_M \) for \( M \). In the case when \( Q \) is not sampled from a uniform distribution, but rather, from a distribution \( p : M \to \mathbb{R}^+ \), it turns out that [43], for any function \( f : M \to \mathbb{R} \):

\[
L^Q_t f(x) \sim \frac{\pi}{4} p(x) \Delta_M + \frac{\pi}{2} \langle \nabla p(x), \nabla f(x) \rangle
\] (5.2)

where \( \nabla \) denotes the gradient operator.

### 5.1.1 Uniformization

Consider the coordinate functions \( X, Y, \) and \( Z \), which returns the \( x \)-, \( y \)-, and \( z \)-coordinate for a point \( x \in \mathbb{R}^3 \), respectively. For any vector \( v \), note that \( \langle v, \nabla X(x) \rangle = \frac{\pi}{4} p(x) \Delta_M + \frac{\pi}{2} \langle \nabla p(x), \nabla f(x) \rangle \)
v·x (i.e., the x-component of vector v). Similarly, ⟨v, ∇Y(x)⟩ = v·y, and ⟨v, ∇Z(x)⟩ = v·z. Let \( p : M \to \mathbb{R} \) denote the density function that the current set of points \( Q \) is sampled from. Now apply \( L^Q_t \) to the three coordinate functions \( \vec{P} = [X \ Y \ Z]^t \). By Eqn (5.2) we then derive:

\[
L^Q_t \vec{P}(x) \approx \frac{\pi}{4} p(x) \Delta_M \vec{P}(x) + \frac{\pi}{2} \nabla p(x). \tag{5.3}
\]

It is known that \( \Delta_M \vec{P}(x) \) specifies the so-called mean-curvature flow: This vector lies in the normal direction \( \vec{n}(x) \) of \( M \) at \( x \), and its magnitude is the mean-curvature of \( M \) at \( x \). This motivates the Laplacian-based iterative denoising approaches for surfaces.
by repeatedly projecting a point in the direction of $\Delta_M \vec{P}(x)$, and is also
behind the mean-shift types approaches for high-dimensional data analysis.

We instead focus on the second term $\nabla p(x)$, which has not been used much so far
in applications. Being the gradient of a function on $M$, $\nabla p(x)$ lies in the tangent plane
$T_M(x)$ of $M$ at $x$, and is thus orthogonal to the first term. If $Q$ is uniformly distributed,
then the corresponding density function $p$ is a constant function, implying $\nabla p(x) = 0$
for all $x \in M$. Hence to uniformize $Q$, we aim to nullify $\nabla p(x)$. Specifically, our
algorithm, shown in Algorithm 1, will repeatedly move points in $Q$ in the direction
of the tangent projection of $L^Q_{\|} \vec{P}(x)$ (which approximates $\nabla p(x)$) till this projection
becomes zero for all points in $Q$ – at which moment, $Q$ reaches a uniform distribution.
(Another interpretation of this tangent component via kernel density estimator is
given in the next section.) Before we explain the algorithm, we remark that as

\begin{algorithm}[h]
\caption{Gradient descent uniformization algorithm}
\begin{algorithmic}
\State \textbf{input}: Two sets of points $P$ and $Q$
\State \textbf{output}: New point set $Q$ forming a uniform distribution of the hidden surface $M$ approximated by $P$
\State \textbf{begin}
\For {$t = 256t_0; t \geq 0.5t_0; t = t/2$}
\For {$k = 0; k < \text{IterNum}; k ++$}
\State \textbf{Step 1}: Compute the tangent projection $\tilde{V}_{\parallel}(Q)$ of $L^Q_{\parallel} \vec{P}(Q)$ for all
points in $Q$ ;
\State \textbf{Step 2}: Move $Q$ in direction $\tilde{V}_{\parallel}(Q)$ by step-size $\mu(k)$ ;
\EndFor
\EndFor
\textbf{end for}
\end{algorithmic}
\end{algorithm}

mentioned earlier, the role of the point set $P$ is to provide an approximation of the
hidden surface. In our algorithm, we only use $P$ to estimate tangent spaces. In
particular, given a point \( q \), we approximate the tangent plane \( T_M(q) \) at \( q \) as follows: let \( p \in P \) be the nearest neighbor of \( q \) in \( P \). \( T_M(q) \) is taken as the best plane fitting the \( k \)-nearest neighbors of \( p \) in \( P \) in the least square sense. The default value of \( k \) is 15. We also note that there is a parameter \( t \) in the Gaussian kernel of \( L_Q^t \): Intuitively, points within a region of radius \( \Theta(\sqrt{t}) \) around \( q \) will influence \( L_Q^t P(q) \). The choices of parameters involved in our algorithm often depend on this “neighborhood size” \( \sqrt{t} \).

**Step 1:** We compute \( L_Q^t P(q) \) at each point \( q \in Q \). Let \( \vec{V}_\parallel(q) \) denote the projection of \( L_Q^t P(q) \) on the estimated tangent plane \( T_M(q) \). In Algorithm 1 we use \( L_Q^t P(Q) \) (resp. \( \vec{V}_\parallel(Q) \)) to represent the set of \( L_Q^t P(q) \) (resp. the set of \( \vec{V}_\parallel(q) \)) for all \( q \in Q \).

**Step 2:** To nullify the tangent component, we move \( q \) in the direction of \( \vec{V}_\parallel(q) \). Intuitively, \( \vec{V}_\parallel(q) \) approximates the gradient \( \nabla p(q) \) of the current density function \( p \), and is pointing from a denser to a sparser region of \( Q \). By moving points in this direction, we spread them away from denser regions. The displacement vector \( \vec{V}_\parallel(q) \) is zero for all \( q \in Q \) only when \( Q \) achieves a uniform distribution.

Specifically, we move \( q \) to \( \tilde{q} = q + \mu(k)\vec{V}_\parallel(q) \), where \( \mu(k) \) is a step-size used in the \( k \)-th iteration of the inner for-loop. We now discuss how to set the step-size. In a sense, we are performing a gradient-descent type of approach to minimize the difference between the current density and target density. We use the following non-adaptive annealing strategy to balance the simplicity of our algorithm and to avoid getting stuck in local minima [74]. We set \( \mu(k) = \mu(0)/(1+|k|) \), where the initial step size \( \mu(0) \) is \( \sqrt{t}/2 \) and \( t \) is the parameter used in the graph Laplacian \( L_Q^t \). The stepszoe is constant for the first \( T \) iterations, allowing points to find the general location in that scale, before annealing (decreasing) it at a slow pace. \( T \) is a parameter that
needs to be empirically set. In our experiments, $T = 2$ provides a good tradeoff for both speed and accuracy.

After $q$ is moved to $\tilde{q} = q + \mu(k)\vec{V}_\parallel(q)$, we need to project $\tilde{q}$ back to the hidden surface. This is achieved by simply projecting $\tilde{q}$ to the estimated tangent space $T_M(q)$. This step contains a least-square fitting (of the tangent plane) and a projection, which makes our uniformization algorithm robust w.r.t. reasonable amount of noise. See Figure 5.3 for an example.

![Figure 5.3](image)

Figure 5.3: (a): Input are 5K points $Q$ over 50K dense points $P$: uniform noise of magnitude 1.5% of the diameter is added at each point. (b): Output points adapted to the target density function shown in (c). Note that these points are also smoothed by our algorithm. (d): Density map of our output points. The correlation of output density and target density is 0.99 and $L_2$-error is 0.015.

**Choice of $t$.** As discussed earlier, the parameter $t$ involved in $L_t^Q \vec{P}(q)$ roughly specifies the neighborhood size (scale) we consider around each point $q$. At the beginning when the distribution of $Q$ is still far from the target distribution, we use a large $t$ value so that the displacement of each $q$ is guided by the global distribution of points in $Q$ at this point. As the distribution of $Q$ becomes closer to the target, we use
smaller and smaller $t$ to perform local refinement. Specifically, we adjust $t$ as follows. We start with $t = 256t_0$, where $t_0$ is the square of the average distance between a point to its nearest neighbor in the input point set $Q$. We perform IterNum = 24 iterations of the non-adaptive annealing step (inner for-loop in Algorithm 1). We then reduce $t$ to its half and repeat until $t = 0.5t_0$. In a sense, $t$ also plays a role analogous to the use of “temperature” in standard simulated annealing type of approaches (see also the parameter $T$ above) for solving optimization problems. We observe that this strategy, combined with the simple non-adaptive annealing approach described above, is both efficient and effective at recovering from local minima even when the initial distribution is far from the target. An example is shown in Figure 5.1 and will be explained in more detail in Section 5.3.

Interestingly, it turns out that by reducing $t$ to be smaller than $t_0$, we can obtain well-separateness among points. Intuitively, when $t$ is smaller than $t_0$, only the immediate neighbors of $q$ will contribute when computing $L^Q_t \vec{P}(q)$ (analogous to one-ring neighbors of $q$ in a mesh); and these points need to form a somewhat regular pattern around $q$ for $\vec{V}_\parallel(q)$ to be zero. See the right figure for an illustration where all points are already projected onto the tangent plane $T_M(q)$: For small $t$, only the immediate neighbors $q_1, \ldots, q_5$ of $q$ affect $L^Q_t \vec{P}(q)$. By Eqn (5.1), each $q_i$ will contribute a vector $v_i$ which is some scaling of $q - q_i$, and $\vec{V}_\parallel(q) \approx \sum_{i=1}^5 v_i$. Intuitively, an evenly distributed $Q$ tends to form to ensure that this sum is zero for all points $q$. See also Figure 5.3 for the effect of reducing $t$.

**Boundaries.** It turns out that $L^Q_t$ behaves fundamentally different around boundaries of a surface $M$. Specifically, in contrast to Eqn (5.2), for a point $x \in \partial M$,
Figure 5.4: (a) plots the $L_2$-error of the density of intermediate point set $Q$ as the parameter $t$ decreases in Algorithm 1. The points obtained when $t = t_0$ and $t = 0.5t_0$ are shown in (b) and (c): note that at $t = t_0$, while the distribution is already close to uniform (small $L_2$-error), points are yet well-separated.

we now have:

$$L_t^Q f(x) \sim \frac{1}{\sqrt{t}} \cdot \frac{\pi^{1/2}}{2} p(x) \partial_n f(x) + o\left(\frac{1}{\sqrt{t}}\right).$$

(5.4)

where $\vec{n}$ is the outward normal at $x$: namely, $\vec{n}$ lies in the tangent plane $T_M(x)$ at $x$ and is normal to the boundary curve $\partial M$. Terms from Eqn (5.2) are now hidden in the second term above. If we plug in $\vec{P}$ as the function $f$, the first term in Eqn (5.4) becomes $\frac{\pi^{1/2}}{2\sqrt{t}} p(x) \vec{n}$, which lies in the tangent plane $T_M(x)$. In other words, the tangent projection $\vec{V}_\parallel(q)$ is now dominated by this first term. Our algorithm, by moving points in the direction of $\vec{V}_\parallel(q)$, extends points across the boundary in the tangent plane in the outward normal direction $\vec{n}$. Such extension can close small holes (missing data); see Figure 5.5 for an example. If the boundary is large (and thus may be real boundary), then such extending effect is no longer desirable. To ensure that there is no significant expanding along boundaries: for a point $q$, let its target position be $\tilde{q}$; and the nearest neighbor of $\tilde{q}$ in $P$ be $\tilde{p}$. If the distance between
\( \tilde{q} \) and \( \tilde{p} \) is greater than \( \sqrt{t_0} \), we simply project the point \( q \) to \( \tilde{p} \). Intuitively, holes of size \( \Theta(\sqrt{t_0}) \) will be sealed. Another example from raw scan data is shown in Figure 5.2.

![Figure 5.5](image)

Figure 5.5: (a) Input points and (c) output points on a human face. Small holes in (b) around nose are filled after uniformization in (d).

### 5.1.2 Non-uniform density adaptation

The algorithm to obtain a non-uniform target distribution \( g : M \rightarrow \mathbb{R} \) for points in \( Q \) follows the same framework as the case of uniformization. The only modification is that, at each iteration, we now wish to nullify \( \nabla p(q) - \nabla g(q) \) for each point \( q \). Hence instead of moving \( q \) in the direction of \( \tilde{V}(q) \), we now move it in the direction of \( \frac{2}{\pi} \tilde{V}(q) \) where \( \frac{2}{\pi} \tilde{V}(q) \) approximates \( \nabla p(q) \). Assume that we can obtain \( g(x) \) for any query point \( x \). To estimate \( \nabla g(q) \) at each \( q \in Q \), we locally fit a quadratic function in the tangent plane \( T_M(q) \) using points within \( \sqrt{t} \) distance to \( q \).

The target density function \( g : M \rightarrow \mathbb{R} \) need to satisfy the property that \( \int_M g(x) dx = 1 \). However, very often, given that \( M \) is not known, we are given an empirical target
density function \( \tilde{g} : M \to \mathbb{R} \) which is a scaled version of \( g \); that is, \( \tilde{g}(x) = c \cdot g(x) \) for some constant \( c \). (For example, \( \tilde{g}(x) = x.x^2 + x.y^2 + x.z^2 \).) In such a case, we need to estimate the constant \( c = \tilde{g}/g \).

The integral \( \int_M g(x)dx \) can be approximated using points in \( Q \) by \( \sum_{i=1}^{m} \tilde{g}(q_i)A_i \), where \( A_i \) is the area of the hidden surface \( M \) represented by each sample point \( q_i \). Following the integral estimation in [59], we estimate \( A_i \) as follows: project the local neighbors \( Q_i \subset Q \) (points from \( Q \) within \( \sqrt{t} \) distance from \( q_i \)) onto the tangent plane \( T_M(q) \). Compute the Voronoi diagram of \( Q_i \) in \( T_M(q) \) and take the area of the Voronoi region associated to \( q_i \) as \( A_i \). This Voronoi diagram is only computed locally for a small number of points, and is fast in practice. Now we have:

\[
\sum_{i=1}^{m} g(q_i)A_i = 1 \Rightarrow \frac{1}{c} \sum_{i=1}^{m} \tilde{g}(q_i)A_i = 1 \Rightarrow c = \sum_{i=1}^{m} \tilde{g}(q_i)A_i.
\]

Note that the estimation of \( c \) needs to be done only once.

In the experiment, we show an example of geometry-aware sampling, by using the mean-curvature as the target density distribution: The mean-curvature at a point \( q \) is simply taken as the normal component of \( L_Q^\top \vec{P}(q) \) (with some smoothing to reduce the variation of the density function); recall Eqn (5.3).

### 5.2 Connection to Kernel Density Estimation

It turns out that there is an alternative way to interpret the tangent projection \( \vec{V}_\parallel(q) \) of the graph Laplacian \( L_Q^\top \vec{P}(q) \) performed on the coordinates functions \( \vec{P} = [X, Y, Z] \) via the standard kernel density estimator (KDE). Specifically, suppose current points \( Q = \{q_1, \ldots, q_m\} \) are i.i.d. sampled with respect to the density function \( p : M \to \mathbb{R} \). Now consider the following popular Gaussian kernel density estimator [61] that gives rise to an empirical density function \( \hat{g} : \mathbb{R}^3 \to \mathbb{R} \) with support being
Figure 5.6: (a) Target density and (b) our output density function as achieved by the 10K output points shown in (c).

the ambient space (which is \( \mathbb{R}^3 \) in our case): at any \( x \in \mathbb{R}^3 \), we have

\[
\tilde{g}(x) = \frac{1}{m} \sum_{i=1}^{m} K_t(x, q_i), \quad \text{where} \quad K_t(x, q_i) = \frac{1}{\pi t} e^{-\frac{\|x-q_i\|^2}{t}}.
\]

The ambient gradient of this empirical density function (w.r.t. the coordinates of the ambient space \( \mathbb{R}^3 \)) is:

\[
\nabla \tilde{g}(x) = \frac{2}{mt} \sum_{i=1}^{m} (q_i - x) K_t(x, q_i) = \frac{2}{mt} \cdot \frac{1}{\pi t^2} \sum_{i=1}^{m} \left[ e^{-\frac{\|x-q_i\|^2}{t}} (q_i - x) \right] = \frac{2}{\pi} \mathbf{L}_t^Q \mathbf{P} \mathbf{x}.
\]

(5.5)

Now if we restrict this empirical density function to the hidden surface \( M \), denoted by \( \tilde{q}' := \tilde{q}|_M \), then we have that the gradient of \( \tilde{q}' \) at a point \( x \in M \) is simply the projection of \( \nabla \tilde{g}(x) \) onto the tangent space \( T_x \) at \( x \). This projection, is exactly

\[
\frac{2}{\pi} \mathbf{N}_x \mathbf{N}(x), \quad \text{which matches what we have in Eqn (3) in the main text. In other words, the}
\]

\(^1\)The general form of the Gaussian kernel below should be \( K_t(x, y) = \frac{1}{(\pi t)^{d/2}} e^{-\frac{\|x-y\|^2}{t}} \). We choose the dimension of the Gaussian kernel to be the intrinsic dimension \( d = 2 \).
density gradient we obtained can also be viewed as the gradient of the empirical kernel density estimator using a Gaussian kernel restricted to the manifold case (instead of in the ambient space).

While the connection between the Gaussian kernel density estimator and heat diffusion process has been investigated in the Euclidean space [19], here we build the connection for the manifold case, which we believe is interesting. Eqn (5.5), combined with the theoretical studies on the Gaussian-weighted graph Laplacian, helps to provide better understanding of the ambient gradient of the kernel density estimator $\tilde{g}$. For example, by Eqn (5.5) and Eqn (3) of main text, it now becomes evident that the normal component of $\nabla \tilde{g}$ is in fact proportional to the mean-curvature flow at a point $x$ in the interior of a manifold. As another example, suppose the input domain $M$ is a singular manifold, which consists of a set of (potentially intersecting) manifolds with boundaries. Recent work in [14] shows that $L_t^Q f$ behaves drastically different around the so-called singular set, which includes the boundary of each manifold patch, non-smooth folds in each manifold patch, as well as intersections of multiple pieces of manifolds. In particular, for a point $x$ around the singular set, Eqn (2) and (3) in the main text no longer hold. Such singular behavior will now carry over to the gradient of the KDE density estimator. The projection of $L_t^Q \tilde{f}(x)$ in the tangent space at $x$ (if it can be defined) will no longer reflect the gradient of the true density function. Hence to handle domains with singularities, different strategies are necessary for points around singularities if true density is desirable.
5.3 Experiments

Several examples of obtaining uniform and non-uniform target distributions can be found in Figure 5.7. For the uniformization case, we compare our results with the weighted LOP operator approach in [44]. We note that a main advantage of our algorithm\(^2\) is that we obtain good results even when the point set \(P\) (to approximation the hidden surface) is rather sparse, including when \(P = Q\); see Figure 5.7. The algorithm of [44] typically requires that the ratio of \(\frac{|P|}{|Q|}\) is large (say \(\geq 10\)).

Furthermore, note that the weighted LOP operator approach cannot yet adapt \(Q\) to an arbitrary density distribution. Hence for the case of non-uniform target distribution, we compare our results with a variant of the Lloyd’s relaxation adapted for handling point clouds data: Specifically, for each point \(q\) in \(Q\), we approximate

\(^2\)An important part of [44] is robust normal estimation and propagation. Here we only focus on its uniformization effect.
| Size of $P$ | data  | $|Q| = 5K$ | $|Q| = 10K$ | $|Q| = 15K$ |
|-----------|-------|------------|------------|------------|
|           |       | Our Alg.   | Lloyd Input | Lloyd Output |
| 25K       | fertility | 0.998 / 0.023 | 0.881 / 0.149 | 0.888 / 0.145 | 0.998 / 0.020 | 0.908 / 0.167 | 0.929 / 0.147 |
|           | cow    | 0.998 / 0.023 | 0.887 / 0.135 | 0.895 / 0.130 | 0.998 / 0.022 | 0.920 / 0.113 | 0.934 / 0.104 |
|           | elephant | 0.988 / 0.058 | 0.932 / 0.115 | 0.923 / 0.123 | 0.990 / 0.044 | 0.954 / 0.095 | 0.954 / 0.095 |
|           | kitten  | 0.999 / 0.016 | 0.921 / 0.132 | 0.930 / 0.124 | 0.999 / 0.017 | 0.951 / 0.102 | 0.963 / 0.089 |
|           | venus   | 0.999 / 0.013 | 0.885 / 0.143 | 0.896 / 0.137 | 0.999 / 0.014 | 0.920 / 0.118 | 0.933 / 0.109 |
| 50K       | fertility | 0.998 / 0.023 | 0.881 / 0.149 | 0.888 / 0.145 | 0.998 / 0.020 | 0.908 / 0.167 | 0.929 / 0.147 |
|           | cow    | 0.998 / 0.023 | 0.887 / 0.135 | 0.895 / 0.130 | 0.998 / 0.022 | 0.920 / 0.113 | 0.934 / 0.104 |
|           | elephant | 0.988 / 0.058 | 0.932 / 0.115 | 0.923 / 0.123 | 0.990 / 0.044 | 0.954 / 0.095 | 0.954 / 0.095 |
|           | kitten  | 0.999 / 0.017 | 0.890 / 0.152 | 0.901 / 0.145 | 0.999 / 0.017 | 0.922 / 0.128 | 0.934 / 0.119 |
|           | venus   | 0.999 / 0.013 | 0.826 / 0.172 | 0.836 / 0.167 | 0.999 / 0.013 | 0.880 / 0.143 | 0.891 / 0.137 |
| 25K       | fertility | 0.998 / 0.023 | 0.881 / 0.149 | 0.888 / 0.145 | 0.998 / 0.020 | 0.908 / 0.167 | 0.929 / 0.147 |
|           | cow    | 0.998 / 0.023 | 0.887 / 0.135 | 0.895 / 0.130 | 0.998 / 0.022 | 0.920 / 0.113 | 0.934 / 0.104 |
|           | elephant | 0.988 / 0.058 | 0.932 / 0.115 | 0.923 / 0.123 | 0.990 / 0.044 | 0.954 / 0.095 | 0.954 / 0.095 |
|           | kitten  | 0.999 / 0.017 | 0.890 / 0.152 | 0.901 / 0.145 | 0.999 / 0.017 | 0.922 / 0.128 | 0.934 / 0.119 |
|           | venus   | 0.999 / 0.013 | 0.826 / 0.172 | 0.836 / 0.167 | 0.999 / 0.013 | 0.880 / 0.143 | 0.891 / 0.137 |
| 50K       | fertility | 0.998 / 0.023 | 0.881 / 0.149 | 0.888 / 0.145 | 0.998 / 0.020 | 0.908 / 0.167 | 0.929 / 0.147 |
|           | cow    | 0.998 / 0.023 | 0.887 / 0.135 | 0.895 / 0.130 | 0.998 / 0.022 | 0.920 / 0.113 | 0.934 / 0.104 |
|           | elephant | 0.988 / 0.058 | 0.932 / 0.115 | 0.923 / 0.123 | 0.990 / 0.044 | 0.954 / 0.095 | 0.954 / 0.095 |
|           | kitten  | 0.999 / 0.017 | 0.890 / 0.152 | 0.901 / 0.145 | 0.999 / 0.017 | 0.922 / 0.128 | 0.934 / 0.119 |
|           | venus   | 0.999 / 0.013 | 0.826 / 0.172 | 0.836 / 0.167 | 0.999 / 0.013 | 0.880 / 0.143 | 0.891 / 0.137 |

Table 5.1: Correlation/$L_2$ error comparisons. The “Lloyd’s Input” is already close to target distribution, while input to our algorithm is not.
its Voronoi cell by collecting the set of points from $P$, denoted by $P_q$, that have $q$ as the nearest neighbor. In each iteration, $q$ is moved to the centroid of points in $P_q$. Lloyd’s relaxation is a local-refinement method (easy to get stuck in local minima), and requires a good input distribution close to the target. Hence to improve its performance, we first obtain an adaptive subsample from the dense points $P$ using a strategy similar to the importance sampling algorithm used in blue-noise sampling, to generate a set of points $Q$ that is already close to the target distribution: we feed this $Q$ to the PCD Lloyd’s relaxation algorithm. The results on the quality of output PCDs are shown in Table 5.1, where we measure the quality of the output points by (i) the statistical correlation of the density function $\tilde{g}$ of output points with the target density $g$, and (ii) the $L_2$-error, which is simply $\|g - \tilde{g}\|_2$ by treating $g$ and $\tilde{g}$ as two $m$-vectors with $m = |Q|$. Our algorithm is able to achieve much more accurate target distribution, especially when the ratio between $|P|/|Q|$ gets smaller. Typically, the improvement of Lloyd’s over its input distribution is not significant, although it does help to improve local regularity of points. If we feed the same input (as ours) to Lloyd’s relaxation, the results are usually far worse.

Another example is shown in Figure 5.1, which is a hard case since points have to move through the many tunnels (e.g., arms and the connection between heads) to adjust their distribution. We note that the Lloyd’s relaxation fails to improve the point distribution much, even though the distribution of input points fed to it is already close to the target distribution (has a correlation of 0.91 with target distribution): After Lloyd’s relaxation, the resulting distribution has a correlation of 0.93 with target distribution. Our algorithm, on the other hand, takes in as input a quite different distribution far from target distribution, yet still achieving
0.99 correlation in the output distribution. (If we feed our input to Lloyd’s method directly, the output has only a correlation of 0.22 with the target distribution.)

**Time complexity.** Assuming that an $k$-nearest neighbor search in $n$ points takes $O(\log n)$ time, our algorithm runs in $O(|I|Q|\log |P| + \log |Q|)$, where $I$ is the number of iterations of the inner for-loop in Algorithm 1. Our current implementation, over $|P| = 25K$ points, takes 31, 81 and 169 seconds for $|Q| = 5K$, 10K, and 15K, respectively. Over $|P| = 50K$, it takes 33, 84, and 172 seconds for $|Q| = 5K$, 10K, and 15K, respectively. Its dependency on $P$ is only $\log |P|$. Hence the time increase is small when $P$ gets larger.

**Limitations.** Given the averaging nature of the graph Laplacian, our algorithm has limitations in reconstructing density functions that have sharp changes.

**Geometry-aware sampling.** In Figure 5.3, we show an example where our density adaption algorithm can be used to obtain a geometry-aware sub-sampling of input points. In particular, our target density function $g$ here is a smoothed version of the mean-curvature. To obtain the $g(x)$ value at a point $q$, we simply take the normal projection of the graph Laplacian $L_I^Q \overrightarrow{P}(q)$: recall Eqn (3) of the main text, this normal projection (the first term in Eqn (3)) is proportional to the mean-curvature flow. To reduce the variation of the density function, we average this approximated mean-curvature value at $q$ with those of its neighbors to smooth the density function. The important point to note is that the geometry-aware sub-sampling can be achieved using the same graph Laplacian framework for density adaption.

**Comparison with weighted LOP.** In Figure 5.9 we provide further comparisons between our approach and the weighted LOP approach from [44], and the
Figure 5.8: (a) (smoothed) mean curvature as target density, (b) our output density map, and (c) the output samples. The correlation between the target and our output distribution is 0.9875, and $L_2$-error of our output distribution is 0.0271.

Figure 5.9: Experiments when $P$ is sparse for the weighted LOP algorithm from [44] and the PCD Lloyd’s algorithm. Recall that our output from Figure 7 (b) of the main text already achieves a very uniform and regular output distribution (where $L_2$-error of our output is only 0.0231).
PCD Lloyd’s relaxation as described in the main text. The algorithm of [44] performs very well when the ratio $|P|/|Q|$ (between the number of dense points in $P$ and points to be moved in $Q$) is large, say $\geq 10$. However, as the size of $Q$ approaches $P$, its performance, especially in producing an even (well-separated) distribution, decreases. Similarly, the Lloyd’s relaxation method also requires the presence of a dense point set $P$. (We note, however, the scope of [44] goes beyond producing a uniform subsample. It also develops an effective and robust normal propagation framework to consolidate PCDs and can handle rather complex points data.)

**Comparison with Lloyd’s method.** In Table 5.1 we compare the accuracy of the output density distribution of our method and the PCD Lloyd’s method, as measured by the statistical correlation and $L_2$-error. Recall that for the PCD Lloyd’s method, we first perform adaptive sub-sampling of the dense points $P$ to obtain a set $Q'$ which is already close to the target distribution. In Table 5.1, “Lloyd’s Input” refers to the distribution of input points $Q$ for PCD Lloyd’s method, while “Lloyd’s Output” refers to the output distribution of Lloyd’s method. Our algorithm always achieves accurate target distribution despite that the input distribution is usually far from the output. We note that, first, Lloyd’s method also requires that the ratio between $|P|/|Q|$ is large. Hence as this ratio decreases, or as $P$ becomes sparser, its performance becomes worse. Second, we note that the improvement after performing Lloyd’s relaxation is often not major over its input distribution. However, we observe the local regularity of the points is often improved by Lloyd’s algorithm. Last, we remark that the Lloyd’s method typically converges within 30 iterations, while our algorithm will run $24 \times 10 = 240$ (as shown in Algorithm 1 of main text) iterations.
Our algorithm requires more iterations both because it takes a multi-scale (both in parameter $t$ and in the simple non-adaptive annealing) approach to avoid local minima during the optimization process and because it achieves a much more accurate target distribution.

Well-separateness. Finally, in Figure 5.10 we show the evenness of our output points using the so-called DDA (Differential domain analysis) power-diagram \[84\] for two models: the dragon and dancing children. Intuitively, this diagram shows the histogram of pairwise distances between sample points. White means high value and black means low value. A white ring of certain radius imply that many pairs of sample points share certain pairwise distance value, say $r$. The black disk enclosed by this white ring shows that very few pairs of samples have pairwise distance smaller than $r$. We note that the power-diagram of our output shows a clear white ring with dark interior, strongly indicating the evenness of the points we produce.

5.4 Discussion

In this chapter, we studied the density adaptation problem that underlies many preprocessing tasks of points of data. We proposed a simple and robust framework for this problem that is effective at both local refinement and global adjustment of points distribution. Our approach relies on the Gaussian-weighted graph Laplacian and works purely in the points setting. Interestingly, while it is well known that graph Laplacian is related to mean curvature flow and thus has denoising ability, our algorithm uses certain information encoded in the graph Laplacian that is orthogonal to the mean curvature flow. It is also as simple as any projection-based iterative denoising framework and can be potentially combined with the latter to achieve both
Figure 5.10: The DDA power-diagram and the plots for anisotropy. Each output (as shown in this figure) has 40K points, and is obtained using a dense point cloud $P$ with 100K points. Reducing the number of output points still maintain similar DDA power diagram. However, small features in the model start to be smoothed.

Denoising and density adaptation effects. Although our algorithm have good performance in general, there is no theoretical guarantee of the convergence to the target density, which remains a challenging open question.
Chapter 6: Conclusion

This thesis studied a spectral point cloud processing framework based on the Laplace-Beltrami operator. The purpose is two-fold: to study the construction of Laplace operator and its spectral structures and to apply the spectral structures to geometry processing field.

In Chapter 2, we consider the important problem of estimating integral from point cloud data under the more general non-statistical setting. We showed that the estimation based on the Voronoi scheme converges to the true integral under the so-called $(\epsilon, \delta)$-sampling condition with explicit error bound presented. This is the first result of this sort for estimating integral from general PCD. For the Principle Eigenvector scheme, although no theoretical guarantee is established, we present its connection to the heat diffusion operator and illustrate justifications behind its construction. Induced by Voronoi weighting scheme, we proposed a new discrete Laplace operator called Voronoi-Laplacian. This operator has many nice properties. It is convergent to true Laplacian under $(\epsilon, \delta)$-sampling condition, and has real eigenvalues and eigenfunctions, with eigenfunctions orthogonal to each with respect to the area weighted inner product, such it is convenient to many geometric applications.

In Chapter 3, we proposed a data simplification algorithm for large data set. Our algorithm preserve the so-called heat kernel signature. Since HKS is based on the
eigen-structure and almost as informative as the eigen-structure itself. Our algorithm tends to preserve the spectral properties of the data.

In Chapter 4, we proposed a new method to estimate gradient of a function defined on a point cloud data based on the so-called diffusion metric. It is easier to approximate gradients in the eigenspace and it is robust to noises in the input function and in the underlying manifold. More importantly, we can easily smooth the gradient field at different scales within this eigenspace framework.

In Chapter 5, we proposed a simple and robust framework for point cloud uniformization and target adaptation. Our approach used the information encoded in the graph Laplacian that is orthogonal to the mean-curvature flow. It is simple, effective, and robust to noise and small holes present in input points. One can use it to improve the quality of sample points.

Using spectral point processing framework for shape and data analysis is an important and promising direction, and has already attracted much attention. While my dissertation work has addressed several important issues involved in applying such a framework, there are still many interesting directions for future research.

Our Voronoi weight scheme for integral estimation requires that input points \((\varepsilon, \delta)\)-sample the underlying manifold \(M\). A natural question is whether it is possible to remove the \(\delta\)-sparsity constraint. This seems to require a more global approach to estimate error than our current local method. This sampling condition also applies to the induced Voronoi-Laplacian. The better performance of Voronoi-Laplacian over PCD-Laplacian in general suggests there is a good chance.

We proposed a heuristic method for data simplification based on the matrix multiplication. However, we did not give any theoretical error bound on the \(HKS\) before
and after simplification. It would be nice to have those theoretical results. More importantly, we hope these results will give us more insights on how to choose HKS features with the correct scale.

We used diffusion metric to estimate gradient of functions which is robust to noise. We will also explore other geometric quantities whose computation can be made easier and more robust by diffusion metric or by a different metric space in general. Diffusion metric is based on a parameter $t$, the scale. How to determine the right scale for a particular application is also an interesting question.

Our data uniformization and density adaptation algorithm have good performance in general, but there is no theoretical guarantee of the convergence to the target density. And it is observed in experiment that the more complicated the target function is, the worse the result. It would be nice to have an algorithm which guarantees the convergence or have some kind of explicit error bound. However, this remains an open question in general for global optimization problem. Furthermore, other application of the tangent component of the graph Laplacian also worth exploring.

Besides the possible extensions to each individual work, how to handle noise within this framework is also an important direction. While most current work can automatically tolerate certain level of noise, there is no theoretical results yet to indicate how noise affect the results quantitatively. All current work is based on the assumption that the point cloud data is drawn from some manifold. To extend the framework to include those point cloud data which is non-manifold, but with some structures would be another interesting direction.
Bibliography


