UCONN LIBRARY

University of Connecticut
OpenCommons@UConn

Doctoral Dissertations | University of Connecticut Graduate School

8-23-2017

Point Cloud Model Shape Analysis

Reed Williams
University of Connecticut - Storrs, reed.m.williams@gmail.com

Follow this and additional works at: https://opencommons.uconn.edu/dissertations

Recommended Citation
Williams, Reed, "Point Cloud Model Shape Analysis" (2017). *Doctoral Dissertations*. 1592.
https://opencommons.uconn.edu/dissertations/1592

Point Cloud Model Shape Analysis

Reed M. Williams, Ph.D.

University of Connecticut, 2017

ABSTRACT

Shape analysis of point cloud surface models produces quality results and avoids the pitfalls of working with mesh models.

Shape analysis is concerned with understanding the shape of models geometrically, topologically, and relationally. This includes such applications as matching a query shape to a part in a database, grouping shapes by type, segmenting a shape into sub-shapes, and finding complementary shapes.

Traditionally, shape analysis methods have operated on solid and surface models of objects, especially tessellated models (i.e., triangular mesh surface models). Recent advances in 3D camera technology has driven demand for automatic shape analysis tools. Devices like the Microsoft Kinect are democratizing 3D sensing and such expansion of what was once an academic and industrial space is making it clear that there is a need for generally-applicable techniques which don't require expert understanding to use.

Unfortunately, mesh model methods require human expertise in order to ensure suitability for processing. Point cloud models, on the other hand, are the natural output of depth cameras and need no human post-processing to render them amenable to analysis.

Reed M. Williams, University of Connecticut, 2017

This dissertation demonstrates that it is possible to understand shape from point cloud models in ways that don't discard the broad mesh model-based shape analysis literature. Instead, I develop an understanding of how to apply a large class of existing mesh model methods directly on point cloud models without global surface reconstruction. Then I show that the results obtained by these point cloud model methods are of a quality on par with those obtained by equivalent mesh methods and can additionally avoid entire classes of mesh-specific problems (e.g., topological errors due to flawed mesh surface reconstruction). I also provide a general improvement to the large "spectral" class of shape signature algorithms on any model type.

Point Cloud Model Shape Analysis

Reed M. Williams

B.E., Vanderbilt University, 2008

A Dissertation

Submitted in Partial Fulfillment of the

Requirements for the Degree of

Doctor of Philosophy

at the

University of Connecticut

2017

Copyright by

Reed M. Williams

2017

APPROVAL PAGE

Doctor of Philosophy Dissertation

Point Cloud Model Shape Analysis

Presented by

Reed M. Williams, B.E.

Major Advisor _____
Horea Ilieş

Associate Advisor _____
Kazem Kazerounian

Associate Advisor _____
George Lykotrafitis

Associate Advisor _____
Donald Sheehy

Associate Advisor _____
Julián Norato

University of Connecticut

2017

ACKNOWLEDGMENTS

I am grateful to the National Science Foundation, General Electric, and the University of Connecticut Graduate School for providing the funding that enabled this work to go forward.

I would like to thank my advisor, Horea Ilieş, for his invaluable support, mentoring, and guidance. Without your encouragement, feedback, and unflagging positive attitude, this document wouldn't exist. I thank my associate advisors, Kazem Kazerounian, George Lykotrafitis, Don Sheehy, and Julián Norato, your advice and input has helped steer my work and made me a stronger researcher and engineer.

My lab mates provided feedback, inspiration, and commiseration throughout this journey. Radu Corcodel, Morad Behandish, Frol Periverzov, Denis Dorozhkin, Pouya Tavousi, Ata Eftekharian, thanks for coming to practice talks, letting me pick your brains, and everything else.

I am eternally grateful for the daily love and support of my amazing wife Ashlee Shaw. I can't imagine a better partner, in science or in life.

No man is an island and I could never have achieved this without the support and encouragement of my family. Special shout out to Papaw Larry, Great Papaw Fred, and Papaw Herschel. You guys didn't get to see the end of this, but you always had my back.

UConn Family, SAGE, everyone at BodyWise, you all provided much needed perspective and balance.

I couldn't have done it without all of you.

Contents

Chapter 1

Introduction

Shape analysis is a field of science and engineering which has only emerged in the past half-century but which is continuing to grow rapidly. The core questions of the field are intellectually stimulating: What is a shape? How do geometric representations have meaning? What does it mean for two shapes to be similar? Are there "natural" groupings of shape and if so, what are they? Human perception of shape is our gold standard for understanding the concept itself, but even a broadly agreed-upon definition of the word remains elusive.

Increasingly, the physical space of human existence is being mapped and digitized by 3D cameras, GPS systems, laser scanners, etc. This ever-growing volume of digital spatial information naturally makes plain the need for tools to automatically understand, sort, categorize, and match this kind of information. However, matching or sorting shapes is not as simple as matching or sorting words in a given language.

There exists a need for effective methods to perform shape analysis tasks across a broad swath of the science and engineering landscape. Tasks such as determining

how a set of proteins should be divided into smaller groups, sorting machined parts into bins, and robotic assembly of components all rely on understanding similarity and dissimilarity between shapes. The fundamental tasks sought by shape analysts are challenging and so much work has been done to date to improve our ability to meet these challenges.

1.1 Motivation

3D cameras are now being produced commercially in larger numbers and at lower cost than ever before. Such sensors provide a low entry barrier to the field of computer vision, and are allowing practically everyone to capture and integrate digital models of reality directly into their applications or engineering design processes. A depth camera generates a point cloud model, a structure which, though less 'complete' than a mesh model, provides a useful representation of real objects of engineering interest [45].

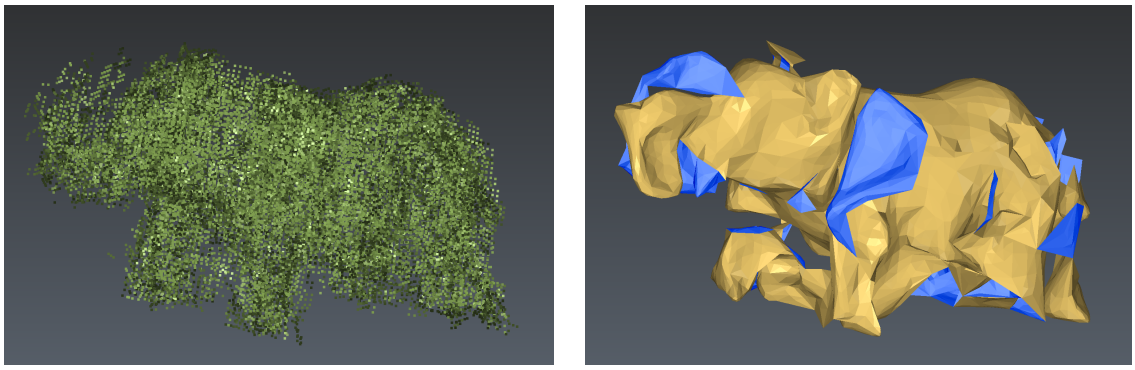


FIGURE 1.1.1: Model "006" from the CERTH/ITI dataset, an elephant figurine, showing the point cloud provided by the dataset and the poor meshing results from using the "Surface Mesh" function of 3DReshaper Meteor [3]. Note the presence of geometric and topological errors. These kinds of errors propagate downstream throughout a shape analysis procedure leading to poor overall results.

Traditionally, practical shape analysis for graphics and engineering applications has largely relied on geometric representations endowed with some kind of topological structure, especially polygonal surface meshes [35]. Although the explicit topological information borne by mesh representations lends itself to simple discrete formulations, creating a mesh from a point cloud is an *ill-posed* problem without unique solutions, and the process relies in practice on costly approximation algorithms [6]. In fact, the general surface reconstruction problem is hard with many remaining challenges [15, 16].

At the same time, automatically constructing a *valid* surface mesh (with guarantees on the geometric approximation) of a point cloud is far from being a solved problem, particularly in the presence of noise, sharp features, sampling anisotropy, and incomplete point clouds [16, 5]. (Figure 1.1.1 shows the result of a commercial software meshing algorithm on a noisy point cloud model from the CERTH/ITI Kinect Scan Dataset [1].) Consequently, the initial meshes can contain crude approximations, element degeneracies, overlaps and self-intersections, surface holes, as well as other "mesh flaws". A good review of the typical flaws in the resulting meshes is provided in [8]. Therefore, the initial meshing is almost always followed by an application-dependent mesh repair process, which introduces additional approximations that may or may not conform to the original physical model [8]. Hence, the difficulties in generating, maintaining the validity, and processing the connectivity of very large meshed models have raised justifiable questions about the utility of polygons as fundamental geometric primitives [46]. Furthermore, many important engineering applications require geometric information in higher dimensional spaces, e.g., in space-time (4D) [30] or configuration spaces (6D) [43], but meshing higher dimensional point clouds gets even harder as the dimension goes up.

It is apparent that understanding the semantics of the output of a range camera *without* requiring meshing[1] or global surface reconstruction would avoid a critical bottleneck in a host of key application contexts relying on the fundamental concept of shape similarity, including engineering design, scene recognition, digital shape reconstruction, functional co-robots, autonomous navigation, and part sorting, as well as virtual and augmented reality. At the core of shape analysis for point clouds are *compact shape descriptors* tailored to this discrete representation, as well as shape similarity, comparison, and segmentation capabilities, which are omnipresent in applications as varied as industrial product design, assistive technologies, medical diagnosis, and quality control [35].

1.2 Research Challenges and Proposed Solutions

Shape analysis techniques for mesh models almost always take advantage of the intrinsic and explicit information inherent to the mesh data structure. This allows mesh-based methods to operate quite efficiently in many ways, but it does render such methods unsuitable for application to models possessed of other data structures, especially the unstructured point cloud model. It is of interest to understand how to translate these methods from dependence on mesh structure to operation on point clouds in order to capitalize on the large and thriving mesh-model–based shape analysis literature and community.

Spectral shape signatures are a popular and useful class of shape signatures which rely on the eigensystem or "spectrum" of the Laplace operator on a surface. It is

[1] It is worth noting here that robust surface reconstruction algorithms do require *a priori* estimates of *differential operators* on the point cloud data [25] in order to construct the geometry and topology of the meshed model.

known that any given estimate of the Laplace operator can only possess some subset of the properties of the precise Laplacian [63]. The current "best" (possessing the best theoretical guarantees) Laplace operator which operates on point clouds is missing the property of symmetry [12]. This property is crucial for applications where the spectrum of the Laplace operator is required. Additionally, within the spectral shape signature literature, it is recommended that analysts truncate the eigensystem of the Laplacian for any given model, so that computations do not become excessively time-consuming. However, the recommended truncations are proposed only by their apparent effectiveness on a test set and no theoretical basis is developed.

Finally, segmenting shapes is a challenging problem. Segmenting shapes of man-made artifacts, especially without defining arbitrary primitives, is even more so.

Problem Statement

These bottlenecks and limitations lead naturally to the following problem formulation: Given a number of point cloud inputs of different intrinsic and extrinsic sizes, especially of a kind resistant to quality meshing, develop (without global surface reconstruction) spectral shape signatures and subsequent feature vectors that allow automatic grouping of the models by geometric similarity and segmentation methods which section the models into semantically-meaningful sub-shapes.

In Chapter 3, I describe a potent framework for performing shape analysis directly on point clouds that may be noisy and/or incomplete, i.e., including those obtained from engineering components and systems. To this end, I propose the Symmetric Point Cloud Laplacian (SPCL), a symmetric version of the PCDL [12], which I show retains the convergence guarantees of the PCDL, and allows us to confidently apply

physics-based "spectral" signature methods to point cloud models. The construction of the SPCL also allows us to capture an estimate of surface normal at each point in the model.

In Chapter 4, I discuss methods for segmenting shapes into meaningful features, especially those which operate by clustering shape signature values as developed in the preceding chapter directly on point cloud models. Methods of this type retain similarity information which allows further classification. I exploit this information to cluster segments by type. I also introduce a point cloud clustering method based on the Vietoris-Rips filtration, and apply this filtration on shape signature values and curvature estimates to segment point cloud models of engineering artifacts into features of engineering interest.

In Chapter 5, I investigate the dependence of spectral signature behavior on the level of eigensystem truncation and characterize the degree to which the portion of the eigensystem not considered contributes to the shape signature. I then develop a user-tunable model-adaptive eigenspectrum truncation algorithm which provides recommendations for how many eigenpairs to keep for any given model to match a desired level of eigenspectrum information capture. This allows databases to be analyzed with greater consistency and/or speed and I demonstrate significant improvements in database matching rates using this new algorithm.

Chapter 6 contains information about the algorithms I implemented to demonstrate and test my framework, the SPCL, and my point cloud spectral shape signature and segmentation methods. I discuss run times and dependencies of the results of my methods on their various parameters as well. The chapter concludes with a number of examples of the quality results obtained by applying the methods laid out in this dissertation to a variety of point cloud models, including an example of dividing

a whole database into a number of distinct subclasses and comparisons with mesh model versions of my point cloud techniques.

1.3 Summary of Contributions

The main contributions of this dissertation are:

- to propose the Symmetric Point Cloud Laplacian (SPCL) and demonstrate its symmetry, convergence properties and error bounds;
- to develop point cloud clustering tools for shape segmentation, including equivalences between intrinsic neighborhood sizes on meshes and point clouds
- to introduce a method based on the Vietoris-Rips filtration for grouping segmented sub-shapes developed from shape signatures on point cloud models;
- to formulate and implement a unified analysis framework for point cloud models that does not rely on surface reconstruction or meshing;
- to demonstrate examples of analysis and segmentations on models of real engineering artifacts as well as on models from the CERTH/ITI Dataset of Kinect-Based 3D Scans [26], which are representative of models traditionally resistant to quality meshing.
- to introduce and demonstrate a new general improvement to the robustness, consistency, and (potentially) computation speed of spectral shape signature methods, whether used on point clouds, meshes, or other arbitrary model types.

I show that the proposed techniques are robust against typical noise present in possibly incomplete point clouds, and segment them into semantically meaningful sub-shapes for point clouds scanned by depth cameras (e.g. Kinect). Furthermore, I show that the proposed framework can output the number of similar features in a given point cloud, which could be used to explore geometric factorizations of and solid model reconstruction from point cloud models. Together, this work presents a highly-automatable integrated analysis procedure for performing direct comparison and segmentation of point cloud models.

1.4 Overview of the Document

Chapter 2 provides a literature review and an overview of related works. Chapter 3 introduces the Laplacian, the SPCL, and spectral signatures for point cloud models. Chapter 4 focuses on segmentation by clustering signature values and other information directly on point clouds and clustering shape segments by geometric similarity. Chapter 5 discusses issues of eigenspectrum truncation as traditionally recommended in the spectral shape signature literature and develops my improvement of the advice provided by that literature. Chapter 6 provides algorithm details, runtime information, and a plethora of demonstrations of the results which I have obtained using the methods described in this document. Chapter 7 concludes with an overview of the contributions along with open questions and Chapter 8 provides certain appended materials.

Chapter 2

Related Work

2.1 Mesh-Model–Based Shape Analysis

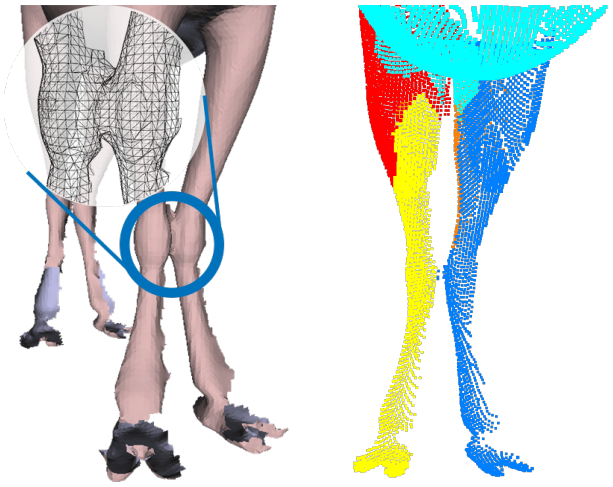
Shape analysis of mesh models is a well-established area of research. Mesh models are a commonly-encountered model type in computer graphics and possess many properties that make typical shape analysis tasks more straightforward. Mesh models consist of a number of vertices which are typically supposed to lie on the surface of the modeled object connected by edges which are intended to show explicitly the local connectivity of the surface and which should lie as close to the modeled surface as possible. This explicit connectivity information is both one of the greatest *strengths* of well-formed mesh models and one of the greatest *obstacles* to using mesh models in an automatic modeling system.

The meshing process in general takes as input a set of points and then attempts to define connections between them such that vertices are only connected if the surface which they model is best modeled by an edge which would connect those vertices.

This approximation procedure is, mathematically speaking, an *ill-posed* problem, i.e., a problem with many possible solutions given one input. A set of points can be meshed in a very large number of ways. The meshing process relies in practice on costly approximation algorithms [6]. The resultant meshes are not likely in general to have the same properties, e.g., topology. Downstream analysis results are often highly sensitive to topological noise and other noises which vary between meshes depending on sampling, meshing algorithm, smoothing, filtering, etc.

In fact, the general surface reconstruction problem is hard with many remaining challenges [15, 16]. At the same time, automatically constructing a *valid* surface mesh (with guarantees on the geometric approximation and topological validity) of a point cloud is far from being a solved problem, particularly in the presence of noise, sharp features, sampling anisotropy, and incomplete point clouds [16, 5]. Consequently, initial meshes can contain crude approximations, element degeneracies, overlaps and self-intersections, surface holes, as well as other 'mesh flaws.' A good review of the typical flaws in the resulting meshes is provided in [8].

For example, meshing the point cloud camel model of [57] with the popular RIMLS Marching Cubes [44] implemented in Meshlab [22] conjoins the legs at the knee, as shown in Figure 2.1.1(a), where the points in the cloud corresponding to two different "legs" of the camel model get relatively close to each other. This, in turn, leads to drastic topological changes in the meshed model that are not found in the original physical model. Clearly, such topological errors propagate through any mesh segmentation or geodesic algorithm developed on this mesh. At the same time, the framework I present in chapters 3 and 4 directly and robustly segments the point cloud model without producing such topological changes, as illustrated in Figure 2.1.1(b). This behavior is not specific to any one meshing algorithm. In fact, different meshing



(a) (b)

FIGURE 2.1.1: (a) The *incomplete* camel model of [57] – shown also in Figure 6.3.13 – meshed by RIMLS Marching Cubes implemented in Meshlab conjoins the legs at the knee where points get close. (b) The legs of the original point cloud remain decoupled when employing the techniques presented in this paper.

algorithms may produce meshes of the *same* point cloud having different geometric and topological properties.

Therefore, the initial meshing is almost always followed by an application-dependent mesh repair process, which introduces additional approximations that may or may not conform to the original physical model [8]. These difficulties in generating, maintaining the validity, and processing the connectivity of very large meshed models have raised justifiable questions about the utility of polygons as fundamental geometric primitives [46]. In fact, arguments supporting direct processing of point clouds in the field of CAD/CAM have appeared for some time [23, 10]. Furthermore, many important engineering applications require geometric information in higher dimensional spaces, e.g., in space-time (4D) [30] or configuration spaces (6D) [43], but meshing higher dimensional point clouds gets even harder as the dimension goes up.

2.2 Description and Similarity

Understanding the shape of a model in a way that is comparable to other models of different intrinsic or extrinsic size requires a compact, commensurable descriptor, a device known as a shape fingerprint or **signature**.

Recently, high-quality "physics-based" methods aimed at providing compact shape description (e.g., shape signatures) and similarity have been developed for triangular surface mesh models. These methods are founded in the mathematical firmament of geometry-dependent physical processes (e.g., thermal conduction). Diffusion-type processes, behavior corresponding to second-order partial differential equations in space and time, are intrinsically dependent upon the local and global geometry and topology of the continuous shapes over which they act, linking known mathematical descriptions of diffusion processes to the geodesic distances on that shape. The physics-based methods apply discrete versions of the mathematics of these processes to mesh models in order to construct shape descriptors that are used in downstream applications. Observe that reliance on pseudo-geodesic distances on the meshes results in robust signatures for noisy or incomplete models [24].

2.2.1 Spectral Shape Signatures

The physics-based methods showing such promise in mesh application (such as the Wave Kernel Signature [9], ShapeDNA [51], and Heat Kernel Signature [56]) rely on a basic descriptor of shape called the Laplace-Beltrami operator, a continuous differential operator defined on Riemannian manifolds. This past decade has seen much development in the area of discrete representations of the Laplace-Beltrami operator, which arises from and with application to the study of diffusion [38]. In mesh-based

discretizations, the "cotan method" [33] or newer, more convergent mesh Laplacian operator [11] may be used. On the one hand, shape descriptors that leverage the locally-descriptive power of the Laplacian via its eigensystem to compute a comparable description of shape are known as "spectral" methods, and are currently being explored on *mesh models* by a number of groups [56, 17, 9]. On the other hand, *direct* point cloud model-based shape analysis has not received as much attention.

Part of the reason for this discrepancy is that surface reconstruction from point cloud models (especially to mesh models) is a thriving research field [16]. However, meshing remains challenging in that it is fundamentally a mathematically ill-posed problem, which cannot even be solved without a number of assumptions (what Berger et al. call "priors" in their analysis [15]). In fact, different meshing algorithms will produce different meshes, if at all, for the same point cloud, which can impact the output of any similarity or segmentation algorithm that would process the mesh. The question also remains open as to whether a particular given point cloud model can be meshed automatically to the degree of validity required for application of various equivalent mesh-based shape analysis techniques. By avoiding global meshing, we effectively side-step this bottleneck.

Some research groups have recently introduced ad-hoc point cloud shape analysis strategies for specific applications, but the existing research has not focused on the kinds of shapes or shape analysis tasks that an engineering analyst or designer might find useful. For example, Pokrass et al. [47] developed a "bag of features"-type partial similarity method for deformable shapes based on diffusion physics, which was developed for partial similarity-based shape matching without considering semantically meaningful segmentation or internal matching. Similarly, Bronstein et al. mention in [17] that it may be possible to make use of a Laplace-Beltrami operator for point

clouds in order to perform HKS and some related analyses on them, but they do not develop the idea further beyond providing a symmetric Laplacian estimate.

Certain industrial researchers have also begun to develop point cloud-based methods for specific applications, such as GE's work [50] to fit linear and arc segments to the point cloud output of a scanner in order to measure manufactured parts against tolerances. This work, however, remains limited in scope and deals only with 2D cross-sections of 3D point clouds and a small number of primitives. Other approaches, such as deep learning-based methods, are also beginning to show promise for classifying and segmenting point clouds, though of course learning methods have limitations such as the availability of quality training data [49, 48]. SyncSpecCNN [67] proposes to combine Laplacian eigenfunctions with a convolutional neural network, operating on a 3D graph representation of scenes.

2.2.2 Discrete Laplace Operators

The Point Cloud Data Laplacian (PCDL) [12] provides one possible estimate of the Laplacian operator on point clouds with desirable theoretical guarantees. It has been shown that the PCDL converges to the true Laplace-Beltrami operator of any given shape under mild conditions on sampling. However, unlike the continuous Laplacian, the PCDL operator is *not* naturally symmetric, as required by spectral methods. Consequently, despite its theoretical guarantees, the PCDL estimate cannot be used in any application in which the innate symmetry of the operator is essential. The spectral shape signature methods require a symmetric Laplace-Beltrami estimate (a real matrix will have real eigenvalues and orthogonal eigenvectors if it is Hermitian, i.e., symmetric) in order to faithfully approximate the eigensystem of the Laplacian

[7].

2.3 Clustering and Shape Segmentation

Recent literature on mesh processing has suggested clustering signature values to segment shapes into sub-shapes. Human observers easily produce, in general, high quality shape segmentations. Indeed, the current state-of-the-art for ground truth for a given (mesh) segmentation is consensus of human observers [21]. In order to develop good automatic segmentation methods that approach human accuracy without active human participation, the recent literature has begun exploring concepts from algebraic topology.

Rustamov, et al. [52] showed one example of using simple k-means clustering on their Global Point Signature to produce segmentations of shapes. Skraba, et al. [54] have proposed using persistence-based clustering of the heat kernel signature for shape segmentation. Similarly, Dey, et al. [24] examine persistence of local maxima of the HKS at several of the HKS's multiple "intrinsic scales" in order to characterize shapes for matching, which produces a segmentation of the shape as a byproduct. These methods all rely on surface mesh models and their innate data structure for their formulations. By contrast, the state-of-the-art for segmenting point cloud models remains so limited that the large-scale open-source point cloud processing project, i.e., Point Cloud Library, does not include any methods for segmenting point cloud models into semantically-meaningful sub-shapes. One brief article was published recently on the topic of feature identification from laser scan data, but that article focuses only on approximate normal-based segmentation and doesn't provide any theoretical backing

for their proposed techniques [68].

2.4 Machine Learning

Increasingly, the techniques of machine learning are being applied to permit automatic understanding of shape. At the lowest level, a sufficiently large and well-labeled training set that covers the test set space sufficiently well should permit classifications of various kinds to be performed by a neural network or deep learning algorithm. Other approaches have attempted to extract geodesic curves from models, then clustered and used to train a probabilistic model for future classification [55].

Machine learning techniques are of particular interest in the shape analysis community because they tend to be "stackable". That is, if a reasonable training set is available, one could build a machine learning structure which classifies the outputs of the algorithms described in this dissertation, allowing for better global results on some class of models. Such a system would likely resemble that of the aforementioned SyncSpecCNN [67], but with a higher quality and smaller dimension of input data.

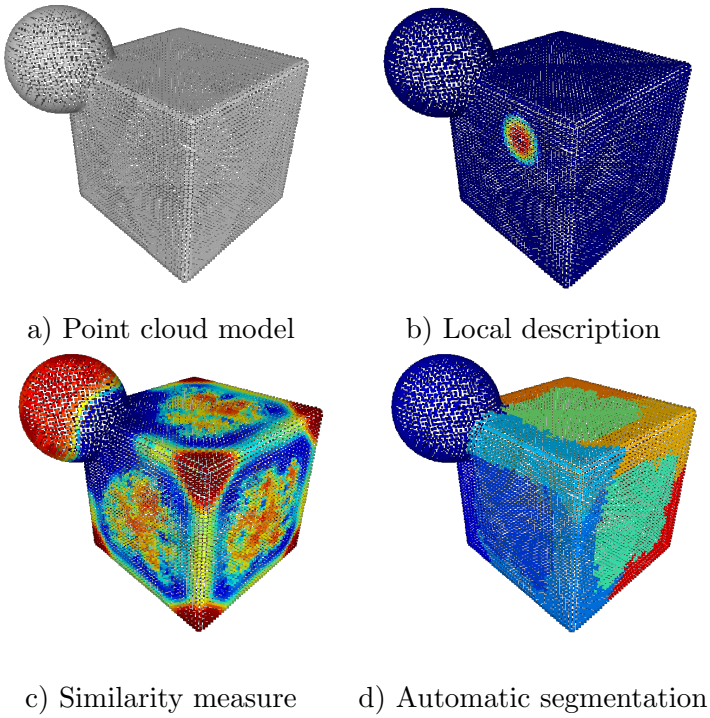
Chapter 3

The SPCL and Spectral Signatures

The existing corpora of mesh model-based shape analysis methods is large and well-established. Developing methods which allow analysts to apply these mesh methods to point clouds prevents any "re-inventing of the wheel" and provides a ready backlog of methods for a variety of applications. I take this to heart and throughout this work endeavor to create bridges between existing mesh literature and the point cloud space whenever possible, rather than blindly nosing around the design space trying to develop brand-new methods for point cloud models from scratch.

With this principle firmly in mind, I begin with an outline of the analysis procedure which I detail over the following chapters. The procedure adapts mesh-model–based methods at each step to excise reliance on the explicit structure of a mesh and allow me to recast these tools for application to point cloud models without surface reconstruction.

3.1 Shape Analysis Procedure Overview



a) Point cloud model b) Local description

c) Similarity measure d) Automatic segmentation

FIGURE 3.1.1: The stages of my point cloud model analysis procedure, from model to SPCL to an HKS vector to one of many possible segmentations.

The shape analysis procedure for point cloud models which I present in the next few chapters (and whose main steps are illustrated in Figure 3.1.1), may be understood as a series of three contributing analyses:

1. **Local shape description:** Describe the local neighborhoods on the shape via the SPCL.

2. **Shape similarity:** Evaluated by computing any spectral shape signature to allow matching and discrimination.

3. **Segmentation:** The shape is segmented by clustering signature values over

the model.

The procedure starts with the computation of a discrete estimate of the Laplace-Beltrami operator from the input point cloud produced by the output of a depth camera. This provides a local description of the input surface as well as approximate surface normals at each point in the cloud. Any spectral shape signature is then applied to the point cloud model using the guidelines we provide in section 3.3, obtaining a measure of shape similarity useful for matching and discriminating between shapes. Finally, the similarity measure and/or normal information is clustered over model neighborhoods, which are inferred from the proposed Laplacian estimate. This ensures that shape segments retain context in the form of their signature values.

This chapter introduces and discusses the Laplace operator, my symmetric point cloud discretization of the Laplace-Beltrami operator, spectral shape signatures, and how to use the SPCL to develop shape signatures on point cloud models without surface reconstruction.

3.2 The Symmetric Point Cloud Laplacian operator

The Laplace operator, or Laplacian for short, is a second-order differential operator Δf which describes the variation of a differentiable function f within a space. It is defined as the divergence of the gradient of the function

$$\Delta f = \nabla \cdot \nabla f$$

which is equivalent to[1] the sum of the unmixed second-order partial derivatives. Intuitively, this operator describes the flux of the gradient field of a function in that space. The equivalent form on a Riemannian (i.e., real, smooth, equipped with an inner product) manifold is called the Laplace-Beltrami operator:

$$\Delta_M f = tr(H(f)) \tag{3.2.1}$$

The Hessian $H(f)$ of the function is a square matrix of second-order partial derivatives that describes the local curvature of the function f over the manifold. Taking the trace of the Hessian in equation (3.2.1) keeps only the unmixed second derivatives, as in the definition of the standard Laplace operator.

This property of describing local curvature makes the Laplace-Beltrami operator of a surface a valuable tool in shape analysis. Discretizations of the Laplace-Beltrami operator for various discrete representations of a surface have been the subject of intense academic interest [63]. A recently developed discretization (and the first for point clouds) is the Point Cloud Data Laplace operator [12]. This PCDL operator is of particular interest because of its stronger than usual convergence bounds.

A native definition of the Laplace-Beltrami operator on a Riemannian manifold has the form

$$\Delta_M f = \frac{1}{\sqrt{\det(g)}} \sum_j \frac{\partial}{\partial x_j} \left(\sqrt{\det(g)} \sum_i g^{i,j} \frac{\partial f}{\partial x_i} \right) \tag{3.2.2}$$

where g is the Riemannian manifold metric and $det()$ is the determinant. The PCDL

[1] For additional information on the Laplace operator, the reader is directed to standard references such as [19].

on point cloud P at scale t has a similar form:

$$L_P^t f(p) = \frac{1}{4\pi t^2} \sum_{\sigma \in K_d} \frac{A_\sigma}{3} \sum_{q \in V(\sigma)} e^{-\delta} \left(f(p) - f(\Phi(q)) \right) \tag{3.2.3}$$

$$\text{where } \delta = ||p - \Phi(q)||^2 (4t)^{-1}$$

Here the intrinsic dimension of the manifold is 2, Φ is the projection onto an approximate local tangent plane, and A_σ and $V(\sigma)$ are the area associated to and the vertices of a given simplex σ in the local triangulation K_d on that tangent plane.

3.2.1 Construction of the PCDL/SPCL Matrix

The PCDL construction is point-wise agglomerative, echoing the summations in the manifold-native form. The operator is built row-by-row from local neighborhood estimates in reduced-dimension tangent spaces approximated from the point cloud. Figure 3.2.1 shows the creation of one such reduced-dimension tangent space and subsequent local triangulation from PCA (principal component analysis) of neighbors about the centroid of the local ball. It has been shown [12] that, given a sampling fine enough to capture the highest-curvature features of the manifold, these local neighborhood estimates approximate the actual surface to a third order term. The two lower-value eigenvalues of the PCA at each point are associated with the eigenvectors which describe the local tangent plane (for a 3D model) and the remaining eigenvector describes the normal vector to that approximate tangent plane. By recording this eigenvector for each point during SPCL construction, it is possible to obtain point normals "for free" as a byproduct of that construction procedure.

Appendix 8.1 describes Figure 3.2.1 & 3.2.2 and the construction of the Symmetric

Point Cloud Data Laplacian in greater detail.

Consider a sampling of points from a Riemannian manifold such that no point on the manifold is farther than ε from a point in the sampling. Let the reach ρ of the surface be defined as the the radius of the largest ball which can roll to touch every point on the surface [29]. This factor may be considered the "size" of the highest-curvature features of the surface. The angle between the actual tangent space to the surface and the approximate tangent space into which points are projected by the projection Φ in equation (3.2.3) is bounded to the order of $\mathrm{O}(f(\varepsilon)/\rho)$ and for points which are near one another (within $\mathrm{O}(\rho/2)$), the projected approximate tangent plane distance approximates the geodesic distance up to a third order term. As sampling becomes finer ($\lim_{\varepsilon\to 0} L_P^t f(p)$), the value of the PCDL approaches that of the Laplace-Beltrami operator. This is the essence of the convergence proofs for the PCDL [12].

3.2.2 What Makes the Laplacian Special?

The Laplace-Beltrami operator has been put to use to more than estimating curvatures. First, observe that the Laplacian on $\mathbb{R}^n$ commutes with isometries on general Riemannian manifolds, which is exactly what is needed in processes whose underlying physics are independent of position and direction, such as heat diffusion and wave propagation in $\mathbb{R}^n$. Hence, the eigensystem of the Laplace-Beltrami operator arises naturally in spectral solutions to various physical problems on these manifolds.

3.2.3 Symmetrizing the Point Cloud Data Laplacian

In order for the eigensystem of the Laplace-Beltrami estimate to be real, the estimate itself must be a real Hermitian (therefore symmetric) matrix [7]. The PCDL estimate

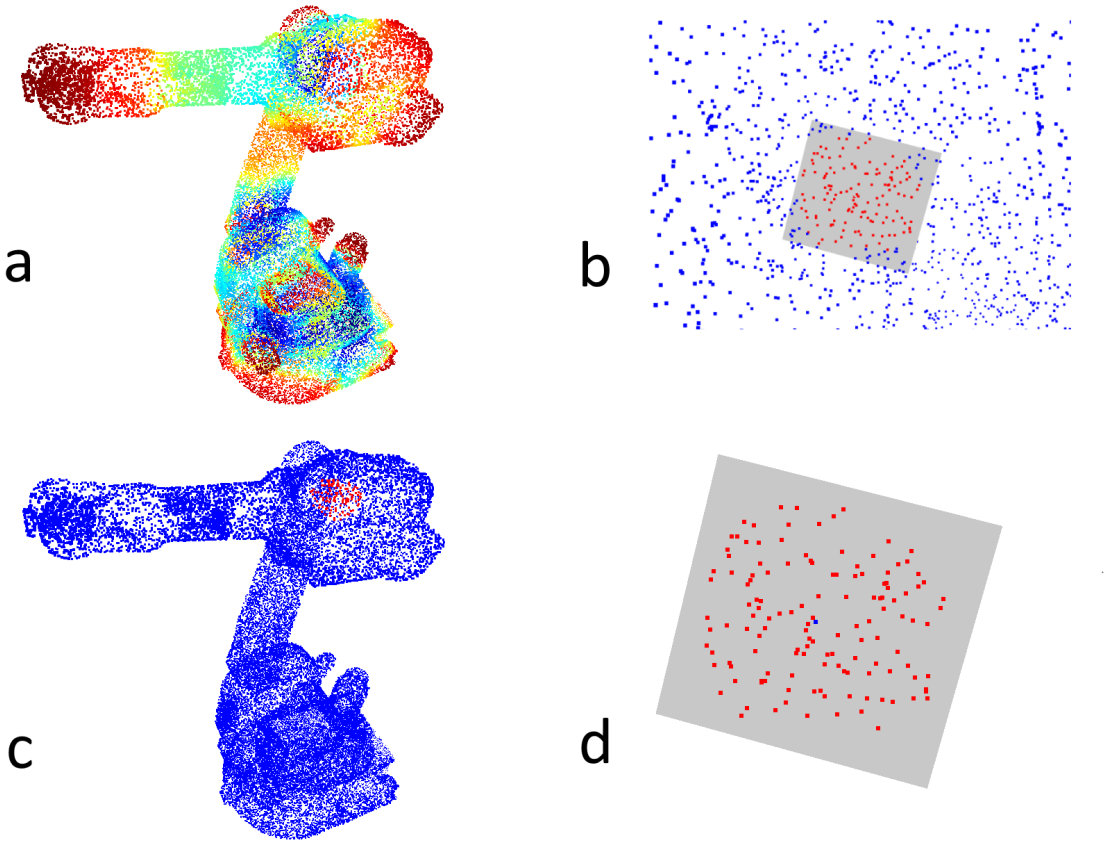


FIGURE 3.2.1: An important part of the SPCL's construction detailed in 8.1: Projecting a query point's three-dimensional noisy point cloud neighborhood into a locally-computed two-dimensional tangent space and triangulating the points in that lower-dimensional space. (a) A noised robot point cloud model. (b) A view of the tangent plane from inside the robot model (viewed from the -z direction of the tangent plane). (c) The neighborhood of a point in a corner of the third linkage. (d) The neighborhood points projected into the tangent plane.

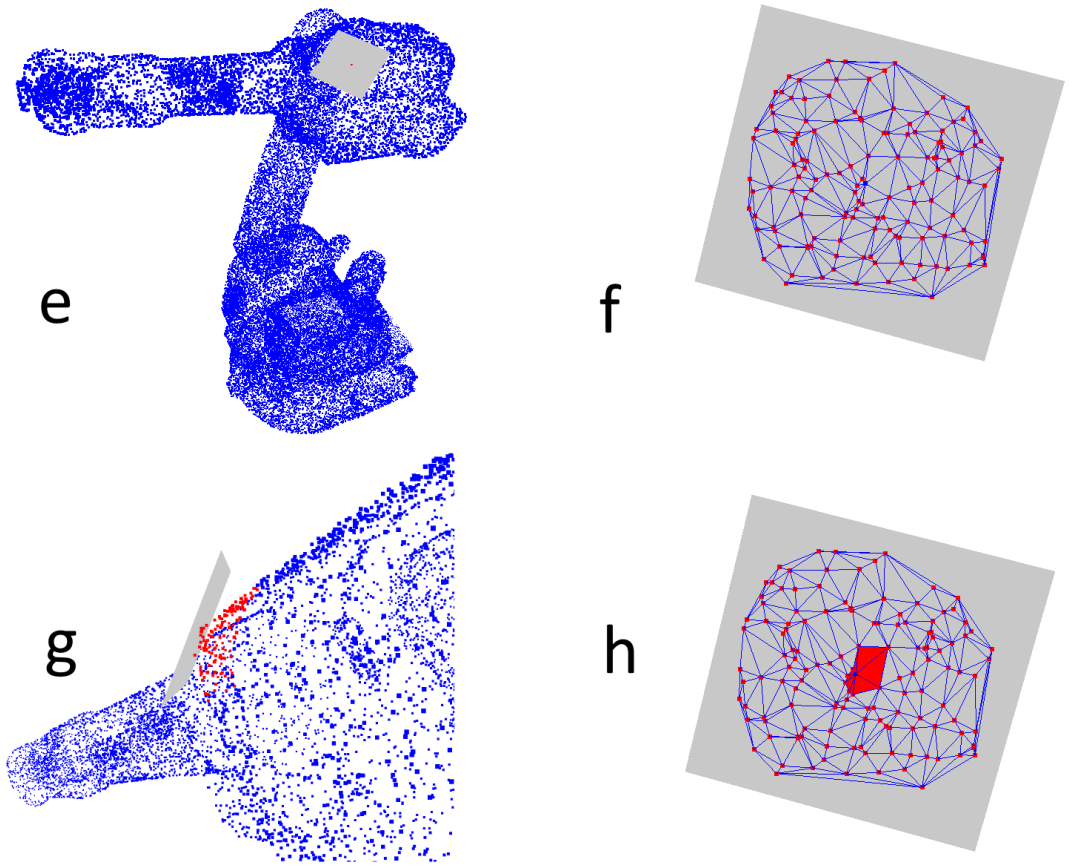


FIGURE 3.2.2: Figure 3.2.1 continued: (e) The local approximate tangent plane computed for the neighborhood. (f) A 2D local Delaunay triangulation of the projected neighborhood. (g) Another angle showing the local approximate tangent plane and the neighborhood (camera is behind robot and looking toward the end effector along the left side). (h) The triangles adjacent to the query point (the triangles whose areas sum to A_{p_i} in Equations 3.2.4 and 3.2.6) are highlighted for clarity.

of the Laplace-Beltrami operator is not symmetric for a general point cloud. Asymmetry in the construction of the PCDL arises chiefly due to the row-wise (equivalent to point-wise) computation of representative areas and point-to-point distances. The error between distance estimates in the PCDL is small, bounded by a third order term in the distance, but neighborness of a pair of points is discrete, and any discrepancy between distance measures can propagate inconsistency of neighborness (e.g., point i's row implies that point j and it are neighbors, but point j's row implies they are not).

To restore the Laplace-Beltrami operator's natural symmetry to the PCDL estimate, we include the row-point's area along with the neighbor point's area (this technique was also developed independently in [41]), then average the operator across its matrix diagonal and compute the eigensystem by the generalized eigenvalue problem. Thus, we compute the SPCL matrix and eigensystem Φ and Λ for a point cloud representing an ε-sampled 2D surface embedded in 3D space as

$$W_{i,j} = \begin{cases} -S(\varepsilon) \cdot \frac{A_{p_i} A_{p_j}}{9} \cdot e^{-\delta(\varepsilon,i,j)}, & j \neq i \\ -\sum_j W_{i,j \neq i}, & j = i \end{cases} \tag{3.2.4}$$

where

$$S(\varepsilon) = 4\left(\pi(2 \cdot \varepsilon)^4\right)^{-1}$$

$$\delta(\varepsilon, i, j) = ||p_j - p_i||^2 4^{-1} \varepsilon^{-2}$$

Then

$$\hat{W} = 2^{-1}\left(W + W^T\right) \tag{3.2.5}$$

$$\hat{W}\Phi = diag(A_{p_i}/3)\Lambda\Phi \tag{3.2.6}$$

Here, A_{p_j} is the total area of the simplices adjacent to point p_j in the local triangulation near that point. The diagonal element of each row of the SPCL is the negative sum of the other elements of that row, since the Laplace operator is by definition an averaging operator [19]. The local tangent spaces are approximated by PCA of each local ball of points about the neighborhood's centroid.

In practice, I construct $\hat{W}$ in a sparse manner in Matlab by, for each point i in the point cloud, for each point j in $B_{4\varepsilon}(i)$ the 4ε ball neighborhood of point i, recording: $\texttt{ii}(end+1) = i$, $\texttt{jj}(end+1) = j$, $\texttt{ss}(end+1) = -\frac{1}{2}S(\varepsilon) \cdot \frac{A_{p_i} A_{p_j}}{9} \cdot e^{-\delta(\varepsilon,i,j)}$, and $\texttt{ii}(end+1) = j$, $\texttt{jj}(end+1) = i$, $\texttt{ss}(end+1) = \texttt{ss}(end)$. This assigns half of the value of each computation for $\hat{W}_{i,j}$ to each of $\hat{W}_{i,j}$ and $\hat{W}_{j,i}$. After each `ss` computation, the weights assigned to each row are added to `totalweight(i)` and once all of the non-zero, non-diagonal elements of $\hat{W}$ have been assigned, for each row i, a new element $\texttt{ii}(end+1) = i$, $\texttt{jj}(end+1) = i$, $\texttt{ss}(end+1) = -\texttt{totalweight}(i)$

is added to create the negative row sum value seen in the $j = i$ portion of Equation 3.2.4. During the final step, running Matlab's `sparse(ii,jj,ss)`, the `ss` values assigned the same `ii` and `jj` pairs are summed, and the sparse matrix is created from i indices `ii`, j indices `jj`, and values `ss`, resulting in the effect described above in Equation 3.2.5 of averaging the values across the diagonal and zero-sum rows.

3.2.4 Error Bounds and Guarantees Retained by the SPCL

In each row of the PCDL, in order to improve robustness with respect to noise, the distances used between the points appearing in that row were specified as post-projection Euclidean distances in the tangent plane of the point associated with that row $d_{T_i}(\Phi(p_i), \Phi(p_j))$. It has been shown that the error between $d_{T_i}(\Phi(p_i), \Phi(p_j))$ and the geodesic distance in the manifold $d_M(p_i, p_j)$ is bounded by a third order term for points closer than $\rho/2$ [12]:

$$d_{T_i}(u, v) \leq d_M(u, v) \leq d_{T_i}(u, v) + O(d^3) \tag{3.2.7}$$

The size of neighborhood specified for construction of the local triangulations is $\lambda < \rho/4$. This means that the difference in tangent-space Euclidean distances between two points, each of which appearing in the tangent space belonging to the other (i.e., appearing in each others' rows in the operator), will be bounded by a third order term.

$$0 \leq d_M(u, v) - d_{T_i}(u, v) \leq O(d^3) \tag{3.2.8}$$

$$0 \leq d_{T_2}(u, v) - d_{T_1}(u, v) \leq O(d^3) \tag{3.2.9}$$

The error of the average of those two distance estimates will therefore also be bounded by a third order term, making the SPCL no worse an estimate than the PCDL in terms of geodesic distances approximation.

$$avg(d_{T_1}, d_{T_2}) \leq d_M(u, v) \leq avg(d_{T_1}, d_{T_2}) + O(d^3) \tag{3.2.10}$$

Similarly, the representative areas associated to points appearing in one another's tangent spaces is computed from the local triangulations developed on the projected points. A_{p_j} is the sum of triangle areas for simplices containing p_j. Each of those triangle area terms is a multiplication of two point-to-point distances in the tangent space, a term which must therefore be bound by $O(d^3)$ error from the same area in the manifold. Averaging associated areas then, just as averaging distances above, does not increase the order of the error term. Thus, the SPCL retains the error guarantees and convergence properties of the PCDL.

Normals at no cost

The construction of the SPCL requires the estimation of approximate tangent spaces (in a 3D model, planes) at each point in the sampled surface. This tangent plane is developed by way of PCA and the eigenvector associated with the least eigenvalue corresponds to the estimated normal vector of the surface at that point. Simply returning this vector value for each point as the SPCL is built provides the analyst with an approximate normal at each point at no additional computational cost. Figure 3.2.3 shows the good results of finding edges in a point cloud model by examining the maximum angle between normal vectors for the points in the local neighborhood of each point.

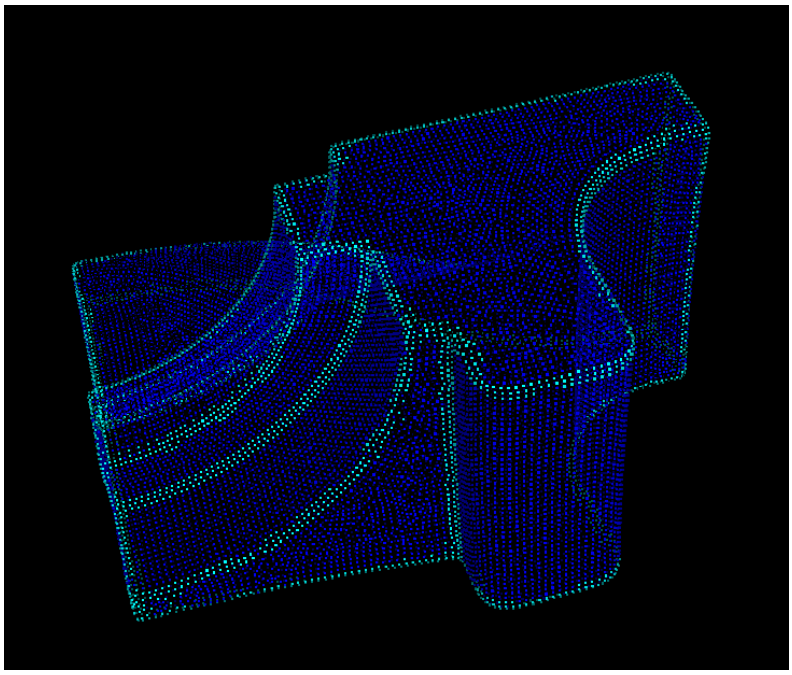


FIGURE 3.2.3: A 26525-point sampling of the fandisk model showing edges computed from point normal vector estimates obtained as a byproduct of constructing the SPCL.

From description to similarity

The local description of a surface provided by the SPCL is useful to the shape analyst, but is of limited comparability. Without a way to map between neighborhoods on different shapes with different samplings, the SPCL can't be used to compare shapes between models. As we move from local description to comparable description, which will permit similarity measurements between different shapes, we look to physics to provide us with a meaningful and compact way to describe shape.

3.3 Spectral Signatures Without Mesh Structure

A shape signature is a compact representation that retains relevant information about a shape. A useful signature should retain enough information to discriminate completely between any two general shapes or classes of shapes, while allowing straightforward computations of degree of similarity and remaining a manageable size. Many shape signatures have been proposed in the literature, but have almost exclusively operated on meshes or parametric surfaces, not point clouds [52, 56, 36].

The particular class of shape signatures known as "spectral" shape signatures consists of signatures whose values are computed by reference to the spectrum of the Laplacian of a shape. Many of these signatures derive meaning by analogy to physical processes that are governed by the intrinsic geometry of the space in which they act. The spectral shape signature I chose for my investigations is the Heat Kernel Signature, although I observe that other shape signatures could be used instead within the framework proposed here and encourage analysts to use whatever signature they find best suits their application. Other options include the Wave Kernel Signature, Global Point Signature, ShapeDNA, or the Giaquinta–Hildebrandt operator [40] (though reformulating this last option would be more involved than for those dependent on the Laplace-Beltrami operator only). In each case, the use of the operator needs to be freed from any dependence on mesh structure, as we discuss and demonstrate in the following sections.

3.3.1 The Heat Kernel Signature

The Heat Kernel Signature (HKS) is a spectral shape signature founded in the physical process of heat diffusion [56]. It has a number of desirable properties: it is invariant

up to model isometry, intrinsically multi-scale, and stable under perturbations on the scale of typical depth camera noise.

In order to get a physical sense for the meaning of the HKS of a shape, consider a point source of heat applied to a point on a surface. As time passes, the heat will diffuse on the surface away from that point. The heat kernel signature's value k_t at that point is the sum total of all of the heat which has diffused away by time t. Since the Laplacian describes the flux of a vector field on a surface, it is intuitive that the Laplace operator will be of some use in this computation. Indeed, the heat equation on a manifold is defined as

$$\frac{\partial u}{\partial t} - \alpha \Delta_M u = 0 \tag{3.3.1}$$

where α is a positive constant and u is the thermal energy as a function of time and location on the surface [28].

The heat kernel is a fundamental solution to the general heat equation [34]. Consider an operator H_t that maps any initial heat distribution $u_0(x)$ on a surface onto the distribution of heat on that surface at any time t

$$H_t(u_0(x)) = u(x,t) \tag{3.3.2}$$

A unique solution to the heat equation above may be written

$$H_t(u_0(x)) = \int_M k_t(x,y) u_0(y) dy \tag{3.3.3}$$

where $k_t(x,y)$ is called the **heat kernel**. Since H_t is compact, positive semi-definite, and self-adjoint, the spectral theorem from linear algebra, allows us to recast it in

terms of its eigensystem [27]:

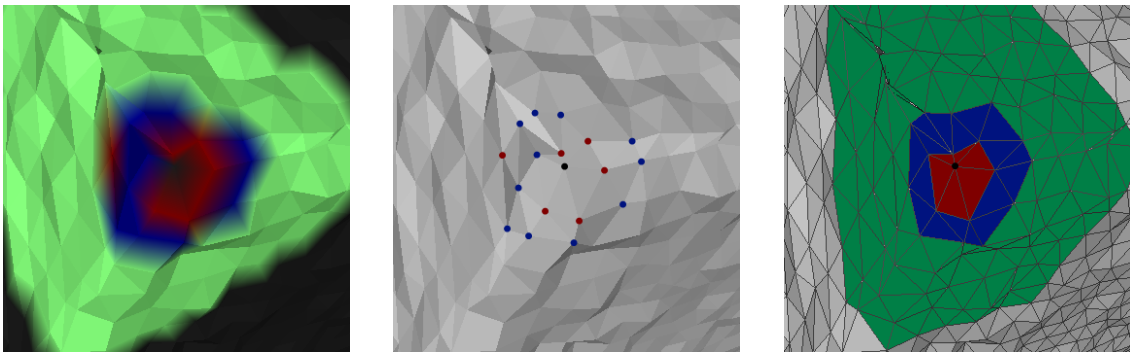
$$k_t(x,y)_H = \sum \lambda_i^H \phi_i^H(x)\phi_i^H(y). \tag{3.3.4}$$

H_t being a solution to the heat equation, it also has the form $H_t = e^{-t\Delta_M}$. The heat operator H_t therefore shares the same eigenvectors as Δ_M, and their eigenvalues are related by $\lambda_H = e^{-t\lambda_M}$. This relationship allows us to write the heat kernel in terms of the eigensystem of the Laplace-Beltrami operator as

$$k_t(x,y) = \sum e^{-\lambda_i t}\phi_i(x)\phi_i(y) \tag{3.3.5}$$

The quantity $k_t(x,y)$ may be considered equivalent to a measurement, for time t, of the amount of heat transferred from point x to point y, for some initial distribution of heat energy on the surface $u_0(x)$. Using this quantity as a measure for similarity would require mappings between each of the neighborhoods, which would be difficult or time-consuming to define between models.

The heat kernel signature (HKS) of a shape is a more compact description of a shape than the heat kernel itself; it is a restriction of the heat kernel to $k_t(x,x)$, i.e., the diagonal of the heat kernel. This restriction captures the "amount of heat" that has diffused away from point x by "time" t, and is sufficient to describe the local area of point x for the purposes of similarity [56]. Restricting the heat kernel to the "time" domain over the model reduces the computational complexity of the signature and obviates the need to develop these local mappings for similarity. This form of



a) b) c)

FIGURE 3.3.1: (a) The non-zero points of a row of the SPCL centered about the corner of a cube, colored according to segment after being clustered into three segments by SPCL value, and shown on the noisy mesh from which this point cloud is sampled for clarity. (b) The points contained in the first two segments of the clustering shown on the mesh for clarity. (c) The actual 1-, 2-, and 6-ring mesh neighborhoods around the point of interest, colored to show correspondence with the point cloud estimate of 1-, 2-, and 6-ring neighborhoods

the heat kernel on M has the eigendecomposition

$$k_t(x,x)_M = \sum e^{-\lambda_i t}\phi_i(x)\phi_i(x) \tag{3.3.6}$$

where λ and ϕ are the eigenvalues and eigenvectors of the Laplace-Beltrami operator of M. As mentioned in section 3.2, the SPCL estimate gives us a symmetric, convergent point cloud discretization of the Laplace-Beltrami operator from which to develop this signature on point cloud models.

Because of its dependence on the factor t, the HKS is innately multi-scale and may be calculated over a range of scales, which can be considered analogous to neighborhood sizes. This allows the capture of information about a surface over different degrees of locality, e.g., examining local curvature information vs. examining the global shape of the surface (extremity, global convexity/concavity, etc.).

Spectral signatures such as the HKS rely on the eigensystem of the Laplacian operator for a surface. The mesh literature has proposed that the first 300 eigenvalues and their corresponding eigenvectors are sufficient for reliable estimation of the HKS on a mesh model [56]. This simplification allows for reasonably rapid estimation of the eigensystem of the SPCL but it is important to note that it restricts the theoretical guarantees of the results to t scales which are not "too small" or "too large". The original paper introducing the HKS recommends $t_{min} = \lambda_{300}^{-1} 4\, ln10$ and $t_{max} = \lambda_{2}^{-1} 4\, ln10$ as the bounds for the t scales at which HKS can be estimated "faithfully", but the effects of pushing past these suggested boundaries are not well-explored. Indeed, in other papers [54], values of t much lower and higher than would be recommended by the guidelines mentioned above are used to no obvious ill effects. However, the minimum number of eigenvectors and eigenvalues necessary for convergence of the HKS on a model at a particular t value seem to be dependent primarily upon the complexity and sampling of the surface in question. The work presented in Chapter 5 is the result of further efforts to understand just how the number of eigenpairs used in computing a spectral shape signature effects matching results on differently sampled or sized models.

3.3.2 Feature Points and Feature Vectors

For matching shapes and for efficient signature storage, signatures are often queried for "feature points." A typical feature point from a signature is an extreme point in some way, either locally or globally, representing a location on the surface possessing some particular interesting properties. In the case of a signature whose values correspond roughly to local curvature (such as HKS at low t-scale), a feature point in

the signature may correspond to a projection from or indentation into the surface, being a point of extreme local curvature. Generally, the extreme values of a signature on a shape have shown themselves useful for discriminating between shapes and for matching [18]. Using lists of feature points, either as a group of features themselves or in combination as a "feature vector," reduces the overhead for matching by reducing the number of values that must be compared or computed.

3.3.3 Selecting Feature Points

Numerous features and feature vectors have been defined from shape signatures, and they are typically chosen experimentally to produce good results on some example set of models. For example, one can select all of the local maxima [56], a fixed number of extrema [9], or multiple values per feature point for a set of local maxima [24].

The original work proposing HKS for meshes recommended finding vertices that are locally maximal in the HKS space by examining 2-ring neighborhoods in the mesh considering those maximal points to be "feature points." Since we are dealing with point clouds rather than meshes, we have at least two options for converting this 2-ring mesh neighborhood. Specifically, we can:

1. use those points in the point cloud model that are within the n nearest neighbors of each of the n nearest neighbors of a point, which can exploit, for example, the "nearest neighborness" property of a local Delaunay triangulation;

2. exploit the neighborness information offered by SPCL, as discussed further in Chapter 4. Specifically, the points corresponding to non-zero values in a row of the SPCL are all those points which, if the point cloud were well-meshed, would

be vertices in the 6-ring neighborhood of the vertex which corresponds to that row. Those values are computed from local areas and distances from the point corresponding to the row, so for a relatively uniformly-sampled point cloud, simple k-means clustering the points by value into three clusters produces a reasonable approximation to the 1-ring neighborhood, the 2-ring neighborhood, and the 6-ring neighborhood as illustrated in Figure 3.3.1.

Additionally, it has been recently proposed [24, 54] that examining the values of a signature in terms of topological persistence may yield a high quality feature vector. Such methods are based on ideas from algebraic topology, such as the Vietoris-Rips filtration, which computes linkages between elements as a network is grown between them [32]. Dey et al. [24] proposed a feature-point selection method based on homological persistence. The method seeks to grow segments around mesh elements by a procedure that begins with the segment (initially, a single triangle) of least persistence and joins it to the region adjacent to a particular edge of that triangle. This growing and merging of regions to seek persistent feature points coincidentally produces a segmentation of the shape under examination, resulting in segments labeled under the name of the "central triangle" of highest value in each region. They observe experimentally that 15 feature points defined by their region-merging method are "usually sufficient" to differentiate between models, and then calculate the HKS values at the triangles so selected for each model at fifteen different t scales. This 15-dimensional feature vector for each feature point in the model exploits the multi-scale nature of the HKS to aid in model discrimination.

The algorithm described relies heavily on mesh structure and properties. To adapt this nxn feature vector for matching on point clouds, I reformulate the feature

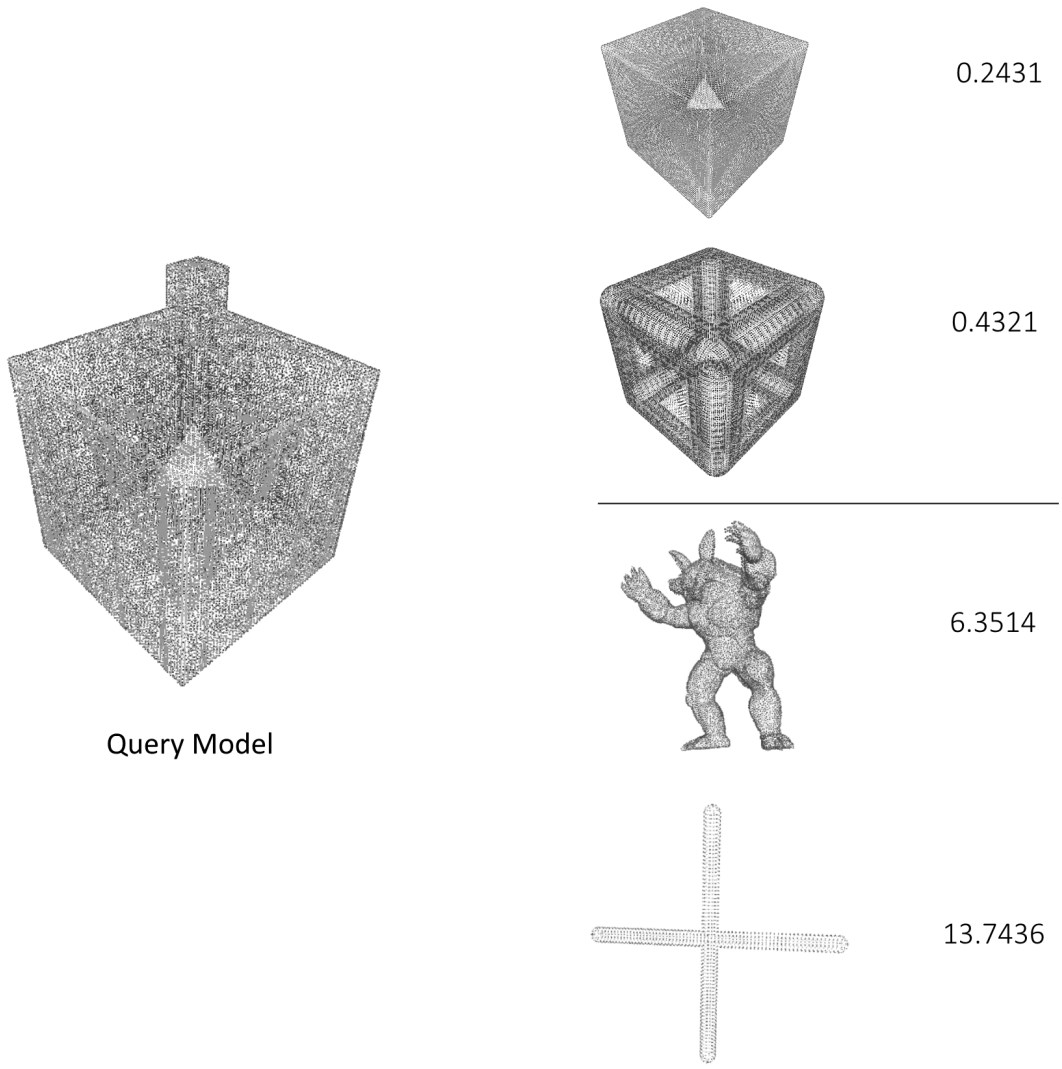


FIGURE 3.3.2: A database of 15x15 feature vectors computed from 39 point cloud models was searched for most similar feature vectors to the query model. The two best-matching non-self models and the two worst-matching models from the database are shown with the matching coefficient computed by
$d(V_1, V_2) = \sum_{v_i \in V_1} \inf_{v_j \in V_2} |\vec{v}_i - \vec{v}_j| + \sum_{v_j \in V_2} \inf_{v_i \in V_1} |\vec{v}_j - \vec{v}_i|$.

vector/segmentation algorithm proposed in [54] to produce feature points. I find a point of maximum HKS value within each region of the segmentation (at a particular t scale). This point provides a natural analogue to the "central triangle" of that

algorithm. An example of the matching results obtained by comparison of the feature vectors thus computed is shown in Figure 3.3.2. This segmentation/feature point selection method is described in Chapter 4.

3.4 Discussion

Spectral shape signatures are a powerful class of compact shape similarity descriptor. Such signatures are useful for shape matching and classification and can provide a basis for shape segmentations, as well. Traditionally, this kind of shape signature has been defined and intended for use primarily on polygonal mesh surface models. By developing a symmetric Laplace-Beltrami estimate for point cloud models, I have opened the door to using such methods directly on point cloud models of real objects, such as those provided by 3D scanning systems like the Microsoft Kinect series of depth cameras.

However, it remains unclear exactly how these signature values can be used for segmenting point cloud models when the segmentation methods that rely on these signatures were also developed on and for mesh models. Especially, it is also unclear how segmenting models of objects which are more typical of mechanical parts can be performed successfully. The underlying philosophies which have informed much of the segmentation algorithm development in shape analysis has supposed that long extensions from a central hub, as in a starfish or a person, are always the features of greatest interest in a given model. For many classes of mechanical part, such as the fandisk model (see Figure 3.4.1), that supposition does not hold. I discuss these issues further in Chapter 4.

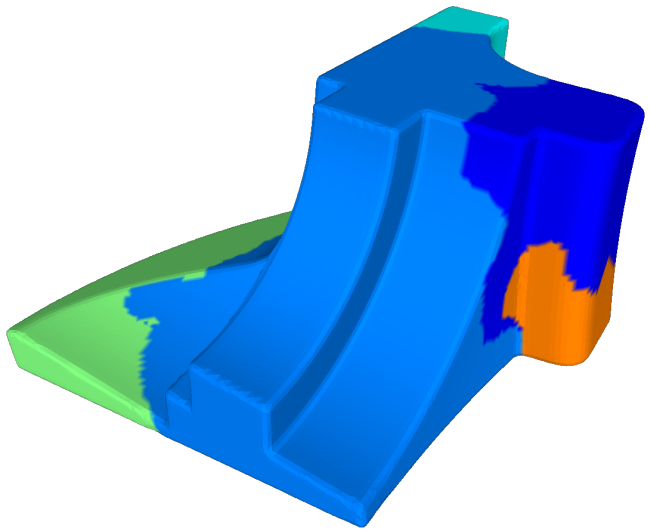


FIGURE 3.4.1: The fandisk model, an example of a difficult-to-segment model where the organic segmentation preference for dividing models into segments by projections from a central hub leads to a generally poor segmentation.

Chapter 4

Clustering and Segmentation

In the preceding chapter, I have described the Laplacian, which provides a local descriptor, and how it can be leveraged to produce various shape signatures, allowing the analyst to measure similarity and compare shapes. The final fundamental step in shape analysis of point clouds is model segmentation.

Humans are typically very good at segmenting shapes into semantically meaningful sub-shapes [13]. Therefore, a high quality automatic segmentation of an engineering model should consist of a very similar set of segments to a human segmentation of the same model [21]. Unfortunately, the decomposition of a shape into semantically meaningful "features" is a widely open problem not only because the concept of a feature is almost always context/application dependent [31], but also because the so called intersecting features still pose a significant challenge to the state of the art feature recognition methods. Therefore, in this section we aim to show that the proposed framework is capable of producing features of engineering interest directly from the point cloud models with the understanding that producing competitive geometric

decompositions remains a fruitful and important direction for further research.

A quality subdivision of a model must rely on the geometry of the model in question. The HKS (or WKS, or GPS) vector of a model provides a description of model geometry which intuitively should be of use in segmenting that shape. However, a simple 1D vector does not readily suggest the possible sub-shapes which make up a model, so we must once more take hold of mathematical tools to coax the sub-shapes from the signature. The area of topological clustering provides the necessary mathematical equipment.

4.1 Existing Methods on Meshes

Defining a general unsupervised clustering method that produces only engineering-relevant design features as clusters without matching to a finite set of primitives is a very challenging problem. In different shapes, different kinds of geometry may be relevant to a designer in different ways. In some respects, any given segmentation solution must be application-dependent for that very reason. If involute gear teeth are the most relevant subshapes to a watch designer, a segmentation method that prioritizes sharp edges or flat planes as the primary loci of segment growth will not provide the most useful information.

However, general segmentation methods are still useful for providing a "first-order" approximation of the sub-shapes which make up a given model, especially if such an approximation can be had automatically. My solution to a general automatic shape segmenter is to apply tools from algebraic topology and homology to develop a segmentation method that, with some parameter adjustments, produces segmenta-

tions separating large-scale engineering features, such as fins or holes, regardless of the shape of the model.

Guibas et al. [54] developed a shape segmentation method that uses the heat kernel signature of a shape calculated on a surface mesh model. Their method was intended for use on "deformable shapes," which could be a very broad class of models; their demonstrations, however, were on only "organic" models regularly observed under various deformations. The philosophy of geometric importance that implicitly underlies their method is that extensions from a central hub are the most relevant seeds for semantic shape segments. Specifically, the method described in [54] is specified as an examination of persistence of regions grown around HKS maxima defined over 1-ring mesh neighborhoods. In a point cloud model, there is no mesh and therefore no n-ring neighborhood structure. However, we propose to use the SPCL itself to find equivalent neighborhoods.

4.2 From Mesh Clustering to Point Cloud Clustering

The SPCL matrix described in section 3.2 may be understood as a weighted adjacency graph for the point cloud it describes. Local balls of points in sufficiently dense point clouds may be considered in some important ways analogous to n-ring neighborhoods in a mesh representation of an object. It has been shown that the Euclidean distance between the tangent space neighbors as computed in the construction of the SPCL approximates the geodesic distance between those same points on the surface to a third order error term. This means that the projection of the local neighborhood of points into the approximate tangent space is a good local approximation of the surface

under consideration. We can exploit this approximation to estimate geodesic distances locally from Euclidean distances, to estimate neighboring regions and cliques, and to find the associated representative surface areas of the points in the cloud.

Since neighbors in the local tangent spaces are neighbors in the surface, the local neighborhoods used to create the SPCL are neighborhoods in the surface represented by the point cloud model, as well. Thus, we can treat the Laplacian as a weighted neighborhood graph of the point cloud. If we ignore the weights in this graph, and consider only the connectivity information, we see that we have a matrix of neighborhood information in which the non-zero points in each Laplacian row are analogous to the points in a particular size of neighborhood in an equivalent mesh. Constructing the SPCL using the same neighborhood size suggested for the PCDL, each non-zero point in a given row in the Laplacian is a member of the 6-ring mesh neighborhood of the point corresponding to that row (see Figure 3.3.1). This correspondence holds because the neighborhood size used in computing the Laplacian rows is based on the intrinsic scale of the point cloud (which is equivalent to the scale of a mesh with vertices at those points).

Importantly, this implied structure allows us to compute locality and connectivity for regions on the surface by simple query, something simple in a complete surface mesh, but previously not straightforward on point clouds. We can also put this structure to work in defining clusterings that operate not just on the signature values on a point cloud, but locally over the surface because we can examine the way the signature values change across the local neighborhoods on the point cloud.

My persistent clustering method (See Algorithm 1) takes as inputs the point cloud model itself, the SPCL and the HKS thereof, and a user-selected scale τ. The highest ν values (I note that $\nu = 10$ seems to give good results) of the weight matrix of

the SPCL are taken as neighbors of the point corresponding to that row. That is, ν represents the approximate number of one-ring neighbors of the point in question in an equivalent mesh surface. The values of the HKS vector selected as clustering criterion are sorted in descending order. Beginning with the point in the cloud with the highest associated HKS value, points with maximum values within their neighborhood are assigned to their own cluster. Whenever a point is found to not be a maximum within its neighborhood, that point is assigned to the cluster of its highest-valued neighbor. Whenever two clusters are adjacent to a point that is being investigated, the maximum HKS value of each cluster is compared. If the difference between maximums is less than or equal to the τ selected, the points belonging to both clusters are assigned to the cluster with the highest maximum HKS value and the smaller-valued cluster is removed. Once all points have been processed, the clustering that remains is final and the algorithm terminates.

Algorithm 1 Clustering

```
function CLUSTERING(PC,SPCL,ν,HKS,τ)
    Find ν largest SPCL weight values for each row OR nearest ν neighbors for each point
    sorthks ← Sort points by HKS descending.
    Pop sorthks
    if maximum in neighborhood then
        Assign Self-Cluster
    else
        Assign highest-value adjacent cluster
    end if
    if more than one adjacent cluster then
        if max(cluster1) − max(cluster2) ≤ τ then
            Merge the smaller-valued cluster into the higher-valued cluster
        end if
    end if
end function
```

In this algorithm, we are exploiting the nearest neighborness property of the SPCL to develop segmentations of a model, such as for segmenting a model into design features for CAD model reuse and redesign. In Figure 6.3.6 I demonstrate results from my point cloud segmentation method as described in Algorithm 1. The method is formulated similarly to that introduced in [54], but excises all dependence on mesh structure in favor of *ad-hoc* neighborhoods provided by the SPCL. I consider this persistence-based segmentation method as a *0-dimensional persistent homological filtration over a restriction of the neighborhood graph induced by the SPCL to the top ten points in each SPCL row.* I use the HKS value difference between points as a distance measure in the implied graph. Defining the segmentation in this way avoids reliance on mesh structure, allowing us to meaningfully define this segmentation on a point cloud model without disregarding local connectivity information. In application, as in Section 6.3, an analyst may find it expeditious if seeking a particular number of segments *a priori* to track the births and deaths of segments in the algorithm and then sort by lifespan, then assign τ based on the number of segments you would like to survive the merging process.

The clustering parameter τ allows significant flexibility in extracting segments from any given model. However, the best τ scale to choose is not a well-determined choice and this segmentation method (and any segmentation method which does not artificially constrain the number of segments found), when applied to very noisy point clouds, can sometimes produce a greater number of segments than are desirable for a particular model.

4.3 Clustering Segments by Type

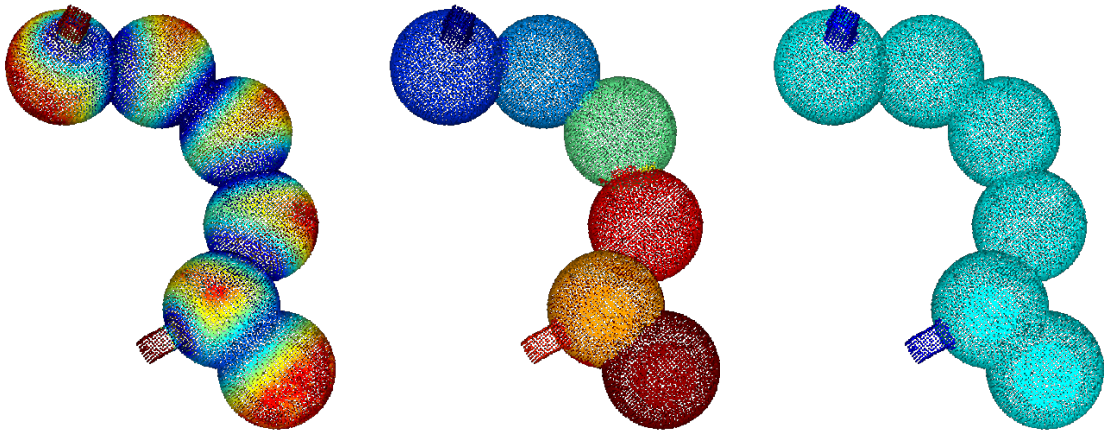


FIGURE 4.3.1: (a) The first HKS vector for this synthetic model of fused spheres with Gaussian noise is used to (b) segment the model by the method described in Algorithm 1 for a particular set of parameters, showing eight individual segments. (c) That same segmentation's clusters is then merged down by hierarchical clustering to only two clusters, showing the two distinct shapes present in the model.

The segments produced by the persistence method described above are constrained to local relevance by the neighborhood graph over which we allow the segmentation to act. This is a good thing; this constraint enforces segments to be connected in the neighborhood graph, a necessary condition for a real shape segment. However, in cases where a complex shape is made up of a smaller number of distinct shape classes, it is of interest to determine how many "types" of shapes actually comprise the model. Since my segmentation method relies on a physics-based signature and retains that distinctive physicality, we can compare the values of the maximum HKS values contained by the segments in order to group computed segments into a smaller number of "sub-shapes". A familiar topological technique once again suggests itself for this kind of comparison: the Vietoris-Rips filtration. The 0-dimensional homology of

the V-R filtration can be calculated quickly over the very small and low-dimensional space of the locally maximum HKS values of the segments returned by any other segmentation method. In fact, in this case, the V-R filtration is acting essentially as a nearest neighbor grouping algorithm acting on a scattering of points on a number line. This simple procedure can, however, produce useful results, as illustrated in Figure 6.3.11, where a segmentation showing that a number of segments detected within a shape can sometimes be reduced to show that there are some smaller number of distinct sub-shapes (or types of segment) present.

In grouping clustered segments together to produce a more relevant result, my method here is similar to that presented in Chazal et al.'s paper on persistence in Riemannian manifolds [20]; as the kinds of surfaces engineers are typically interested in for design and analysis are Riemannian, there may be additional utility to be found by combining their research with my method as described above. The 0-dimensional homology method we describe and which we consider equivalent to the mesh method of Guibas [54] is certainly useful. The 1-dimensional or even 2-dimensional homology of that same filtration may also be of interest, but these investigations are outside the scope of this dissertation.

Rather than simply guessing at the number of segmentation types present, it may be possible to use techniques such as the Clustering Balance or Clustering Gain [37] to automatically determine how many segment types to which a given segmentation should be reduced. For example, in the case of 6.3.11, the normalized Clustering Balance, as shown in Figure 4.3.2, shows a definite minimum at two segment types when investigated from two to the number of initial segments defined by the persistent segmentation.

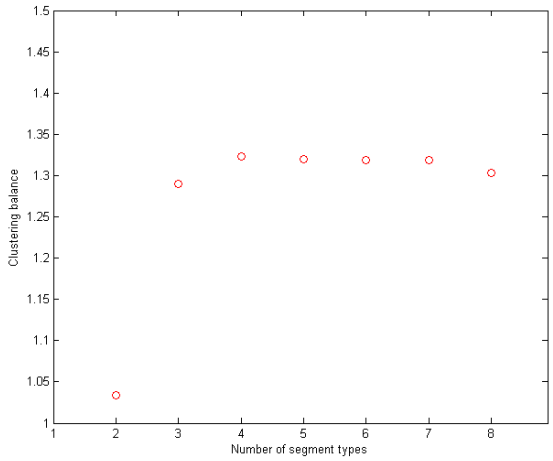


FIGURE 4.3.2: The clustering balance score for the reclustering of the mols model. Clustering the segments of the model into types does not produce a significant change in clustering balance until the number of segment types is reduced to two.

4.4 Other Clustering and Segmentation Methods

Other clustering and segmentation methods have been developed in the mesh literature. Among the most recent is the approximate convexity-based method of van Kaick et al [59]. As discussed above in section 3.3, the HKS is proportional to curvature at low t-values and encodes more global information for higher t-values. This allows segmentation methods based on the HKS to segment shapes based on local curvature, just as in [59], but also by more heuristic measures that allow segmentations to develop more naturally as in so-called "part-based" (or "higher-level") segmentations [4]. It is also important to note that the segmentation described in [59] produces segments, but does not produce the semantics of these segments. This is because approximate convexity does not contain a similarity measure. On the other hand, my segmentation relies on a specific similarity metric so that we not only produce segmented point clouds, but retain information about what shape those point clouds are. This

information could be used in downstream processing, for example for reconstructing parametrized solid models of the point cloud.

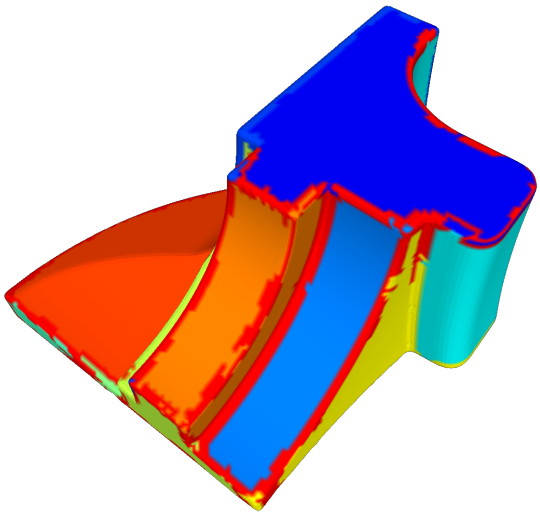


FIGURE 4.4.1: The good results attainable by segmenting the fandisk model with a method that includes explicit information about the local normals as computed by the construction of the SPCL.

Figure 6.3.9 shows an example of combining spectral signature information with explicit local curvature information for segmentations. In this example, I have used the pointwise normal information which is computed as a byproduct of local tangent space approximation during SPCL construction in reformulating the "seeded" segmentation procedure of [39].

The Heat Walk

The Heat Walk is a segmentation method which is closely related to the HKS [14]. This method uses the full-sized heat kernel in contrast with the HKS's restriction to the diagonal of that matrix. The theory is that the Heat Walk algorithm develops

knowledge about the pathways of maximum heat flow capacity on the surface and leverages that knowledge to segment the surface. The resultant segmentation is robust and stable. Since the SPCL allows computation of the elements of the HKS, we should be able to compute the Heat Walk on a surface as well.

The Heat Walk algorithm operates as follows: The heat kernel is computed for the shape, then a voting step is used to find "exemplar" points and non-exemplar points are merged into accumulator regions represented by those exemplars. Regions of the surface which are more accurately understood as dissipator regions rather than simply low-accumulation portions of accumulator regions are then identified and split off into their own region by a step which computes a difference between two quantities: the difference between value at each vertex and it's region's exemplar and the difference between the value at that vertex and a uniform distribution over the model.

In order to apply this technique to a point cloud model, the procedure is as follows:

1. Construct the SPCL of the point cloud as described in Chapter 3.

2. Compute the heat kernel signature on the point cloud. The heat kernel at scale t for points x and y, $k_t(x, y)$, is computed as

$$k_t(x, y) = \sum e^{-\lambda_i t} \phi_i(x) \phi_i(y) \tag{4.4.1}$$

   where λ_i and ϕ_i are the i^{th} eigenvalue and eigenvector of the SPCL respectively.

3. Initialize the value of the "heat potential" $s^1(x)$ for each point x to be $k_t(x, x)$.

4. Find for each point an exemplar point to represent that point. For a point x, for step $m + 1$, its exemplar $e^{m+1}(x)$ is defined as the point y which maximizes $min(k_t(x, y), s^m(y))$.

5. Update $s^{m+1}(x)$ using the set of exemplars, that is

$$s^{m+1}(x) = max\left(min(k_t(x,y), s^m(y))\right) \tag{4.4.2}$$

where $y \in e^m$, the set of exemplars at the current step.

6. Update e^{m+1} and s^{m+1} until the set e^{m+1} does not change from e^m, i.e., until the algorithm converges to one set of exemplars. This completes the merging into accumulator regions portion of the heat walk. See Figure 6.3.8b.

7. Next, to find the dissipator region, compute the probability density function (PDF) for each point p_x, for the uniform heat distribution p_U, and, for each agglomerative region i, the "mean cluster density" p_i (i.e, the PDF of the "average" point in a given accumulator region). These are defined as

$$p_x(y) = \frac{k_t(x,y)}{\sum_y k_t(x,y)} \tag{4.4.3}$$

$$p_U(y) = \frac{1}{n} \tag{4.4.4}$$

$$p_i(y) = \frac{\sum_{x \in i} k_t(x,y)}{\sum_{x \in i} \sum_y k_t(x,y)} \tag{4.4.5}$$

for each accumulator region i and where n is the number of points in the point cloud. Note that the heat kernel's values may be negative, reflecting a loss of heat from some point but PDFs should be nonnegative. It is thus necessary to shift the k_t values by their minimum prior to normalizing them as in the above PDF equations.

8. Once these are computed, for each point, compute the Kullback-Liebler diver-

gence (KLD) between the PDFs, defined as:

$$KLD(j|k) = \sum_i j(i) log \frac{j(i)}{k(i)} \tag{4.4.6}$$

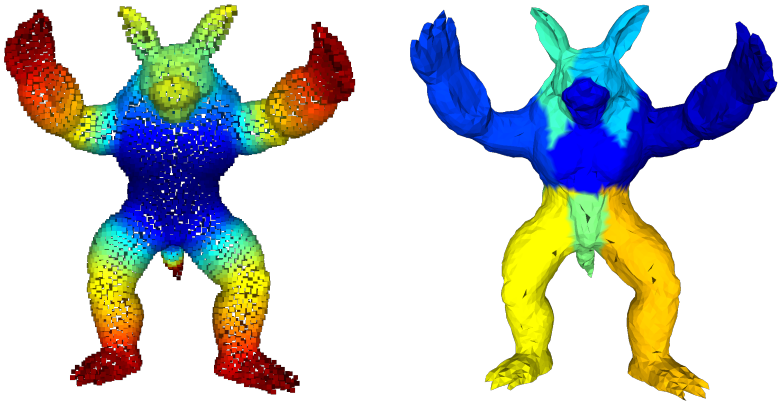
The divergence of the PDF of each point with the uniform distribution and with the average distribution for the region that point has been assigned are compared. If $KLD(p_x|p_U) < KLD(p_x|p_{i(x)})$ where $i(x)$ is the agglomerative cluster in which point x resides, then point x is reassigned to a new dissipative cluster. That is, points x with $k_t(x, y)$ distributions closer to the uniform distribution than to the distributions of the other points in its assigned cluster is mis-labeled and should instead be part of the dissipative cluster.

In implementing this algorithm, I compared the divergences for each point, then either assigned $e^{m+1}(x) = e^m(x)$ if the comparison showed the point was appropriately labeled or else assigned $e^{m+1}(x) = n+1$, defining a new dissipative region by setting those points' exemplar to the name of a non-existent point.

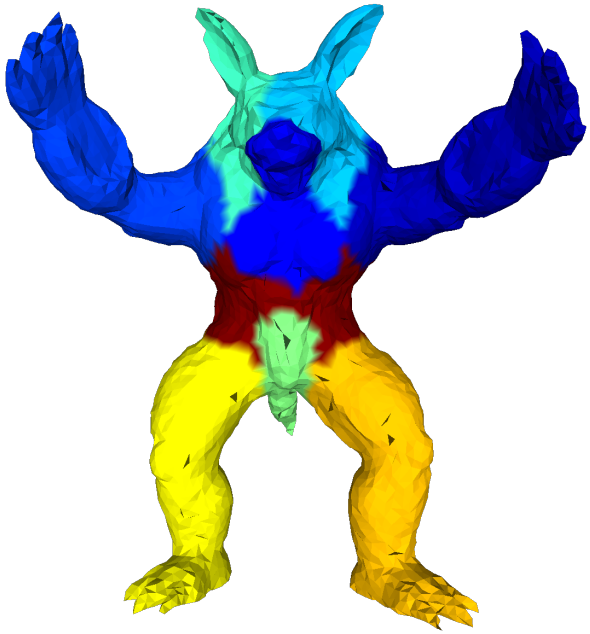
Note that although the authors of the paper claim that this segmentation method is "fast", in practice computing the entire heat kernel for even a medium-sized point cloud is time consuming and the storage requirements are daunting. Even a 10,000-point model has a heat kernel of size $10000 \times 10000 = 1E8$ floats (4E8 bytes) or doubles (8E8 bytes). In double precision, that requires storage on the order of 760MB.

Figure 6.3.8c demonstrates the suitability of the Heat Walk method to segmenting shapes especially with extremities (i.e., clear accumulators) and a single inter-accumulator dissipator region.

Additionally, since the heat walk retains, for all accumulator clusters, a record



a) b)



c)

FIGURE 4.4.2: The classic Armadillo model, first a) colored with the HKS vector $k_0.1$, b) after running the accumulator steps of the Heat Walk algorithm for $t = 0.1$ (through step 6 above), and c) after the complete Heat Walk is finished, showing accumulator regions and a dissipator region, as well, around the midsection. Only the HKS vector is plotted on the point cloud for clarity. The other two plots are onto the mesh model to aid in visualization, but the point cloud model is used for all computations.

of the exemplar point of that cluster and since each of those points possesses a heat kernel signature value at $s^1(x)$, heat walks outputs can be clustered for segment types as in Section 4.3. Figure 6.3.10 shows the results of clustering the heat walk output shown in Figure 6.3.8 with a target of five resultant cluster types.

4.5 Discussion

Clustering signature values can suggest physically-relevant feature vectors while providing candidate segmentations of models. Segmenting shapes into subshapes is an important shape analysis capability with application from finding candidates for replacing similar parts in assemblies to identifying handles for grasping for domestic robots.

I have introduced point cloud versions of several popular methods from the mesh-model–based shape analysis literature and discussed the modifications which were necessary to convert these technologies to this new model type. These discussions should encourage and enable further conversion of a large number of segmentation techniques, originally defined for use on surface meshes, to operation on point cloud models.

Chapter 5

Improving Spectral Signature Performance

Spectral shape signatures provide a high-quality similarity measure based on diffusion physics by means of the spectrum of an estimate of the Laplace-Beltrami operator for the surface of an object.

However, point cloud and mesh models often have very large intrinsic sizes and subsequently large Laplace-Beltrami estimate matrices. Recommendations from the current spectral shape signature literature are to use only a fixed number of arithmetically greatest eigenvalues and their corresponding eigenvectors in the computation of a spectral shape signature. This recommendation seems to work well, but it is not yet understood the degree to which this fixed number of eigenpairs approximates the full spectrum for the purposes of shape similarity measures or even what fixed number to use. Using a fixed number of eigenpairs for all model sizes and samplings also introduces inconsistencies between different samplings of the same shape at different intrinsic sizes and may cost unnecessary computational effort on resource-limited

systems (e.g., drones or robots).

In this chapter, I briefly examine the performance of fixed numbers of eigenpairs on approximating the spectrum of models of different sizes, propose an adaptive cutoff selection method which improves consistency between models for spectral signature use, demonstrate the method on Heat and Wave Kernel signatures (HKS and WKS), and briefly discuss the trade-off between running time and desired error or convergence properties. For more, see [66].

5.1 Adaptive Eigensystem Truncation for Spectral Shape Signatures

Understanding and comparing the shapes of parts and objects is fundamental to the design of functional structures. In the traditional of engineering practice, understanding shapes has been done intuitively by experts by means of their experience or by mathematical comparisons of simplified or representative shapes (e.g. combinations of primitives). In recent decades, methods have been developed for comparing shapes which do not rely on either disassembling shapes into representative primitives or engineering experience. Among the most useful for understanding three-dimensional shape is the class of techniques called spectral signatures. Spectral Signatures

5.1.1 Spectral Signatures

A shape signature is a compact representation of shape which retains relevant information about the shape. A useful signature should retain enough information to discriminate completely between any two general shapes or classes of shapes while

allowing straightforward computations of degree of similarity and remaining manageably sized.

A particular class of shape signatures known as "spectral" shape signatures consists of signatures whose values are computed by reference to the spectrum of the Laplacian of a shape. Many of these signatures derive meaning by analogy to physical processes which are governed by the intrinsic geometry of the space in which they act. The example spectral shape signatures I select for investigation are the Heat Kernel Signature and the Wave Kernel Signature, although any other spectral shape signature may be used instead.

5.1.2 The Heat Kernel Signature

The Heat Kernel Signature (HKS) is a spectral shape signature based on the physical process of heat diffusion [56]. It has a number of desirable properties: It is invariant up to model isometry, intrinsically multi-scale, and stable under perturbations on the scale of typical range camera noise. See Section 3.3 for a more complete description of the HKS and its basis.

The heat kernel signature (HKS) of a shape may be written

$$k_t(x,x)_M = \sum e^{-\lambda_i t}\phi_i^2(x) \tag{5.1.1}$$

where λ and ϕ are the eigenvalues and eigenvectors of the Laplace-Beltrami operator of M. Thus, the first step in calculating the HKS for a given shape must be the estimation of the Laplace-Beltrami operator of the surface of the shape.

5.1.3 The Wave Kernel Signature

To show that my methods are applicable to spectral signatures other than the Heat Kernel Signature which I use for most of the other examples and explanations in this manuscript, I demonstrate the methods on the Wave Kernel Signature [9] as well.

The Wave Kernel Signature (WKS) is a spectral shape signature based on the wave function describing the energy probability distribution of a quantum particle. It is similar to the HKS in many ways, but differs in some important respects. For example, the time parameter of the HKS is in the WKS replaced with an energy parameter and the physics described are oscillation rather than dissipation.

The WKS may be written

$$\begin{aligned} \Omega_E(x)_M =& C_E \sum e^{\frac{-(E-log\lambda_i)^2}{2\sigma^2}} \phi_i^2(x) \\ \text{where } C_E =& \left(\sum e^{\frac{-(E-log\lambda_i)^2}{2\sigma^2}} \right)^{-1} \end{aligned} \tag{5.1.2}$$

where λ and ϕ are the eigenvalues and eigenvectors of the Laplace-Beltrami operator of M, $\sigma = 7(E_{max} - E_{min})/100$, and E is the energy (similar to t in the HKS). The developers of the WKS recommend $E = [log(\lambda_1) + 2\sigma, log(\lambda_N) - 2\sigma]$ and $N = 300$ eigenvalues.

5.1.4 Laplace-Beltrami Estimate

The Laplace operator or Laplacian is a second-order differential operator Δf which describes the variation of a differentiable function f within a space. It is defined as

the divergence of the gradient of the function

$$\Delta f = \nabla \cdot \nabla f$$

which is equivalent to the sum of the unmixed second-order partial derivatives. Intuitively, this describes the flux of the gradient field of a function in that space. The equivalent form on a Riemannian (i.e., real, smooth, equipped with an inner product) manifold is called the Laplace-Beltrami operator[1].

Discretizations of the Laplace-Beltrami operator for various discrete representations of a surface have been the subject of intense academic interest [63]. For triangular surface meshes, the current state-of-the-art is the Mesh Laplace Operator [11]. For point cloud models, the equivalent estimate is the Symmetric Point Cloud Laplacian [65, 64] (see Section 3.2).

5.1.5 A Fixed Number of Eigenpairs

Computing the values of a spectral shape signature on a shape requires computing the eigensystem of the Laplace-Beltrami estimate of that shape (see Equation 3.3.6). The Laplace-Beltrami operator for an n point or n vertex model is an nn matrix. The complete eigensystem for such a model is n eigenvalues with n associated n-length eigenvectors. For typical CAD system or range scanner-generated models, n can easily be in the tens or hundreds of thousands or higher. Computing the complete eigensystem for a 200000x200000 matrix, even a sparse matrix, is an incredibly computationally intensive and time consuming process. It has been suggested and broadly accepted that the "rapid convergence" of the eigenvalues should allow for a

[1]See Section 3.2 for more on discrete versions of the Laplace-Beltrami operator

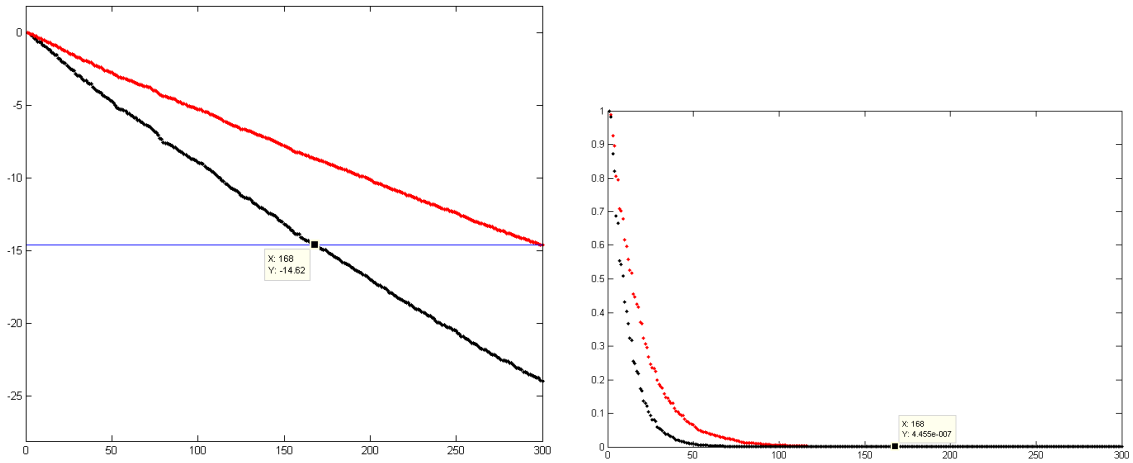


FIGURE 5.1.1: The red points in each plot correspond to the larger 39k point robot model and the black points correspond to the smaller 26k point robot model. a) The first 300 eigenvalues of each model. The blue line marks the value of the 300th eigenvalue of the larger model and the labeled point is the first point below the line in the smaller model's eigenvalues. b) e^{λ} for the first 300 eigenvalues for each model. The label marks the final value in the larger model's plot.

signature to be computed "using a moderate number of eigenvalues determined by feasible numerical computations" [51].

Thus, in order to make a spectral signature for a typical model amenable to computation, the developers of spectral signatures have traditionally advised users to use a fixed number of eigenvalues and their associated eigenvectors for spectral signature calculation. For the Heat Kernel Signature, 300 eigenpairs is the recommendation [56]. For the Wave Kernel Signature, 300 is again the authors' preferred number [9]. The developers of the Global Point Signature used only 25 eigenvectors (though operating on decimated models of no more than 25000 vertices) [52]. The Shape Google implementation of HKS relies on only 100 eigenpairs [17]. In all of these cases the decision to compute only 25, 100, or 300 eigenpairs is justified only by experimental

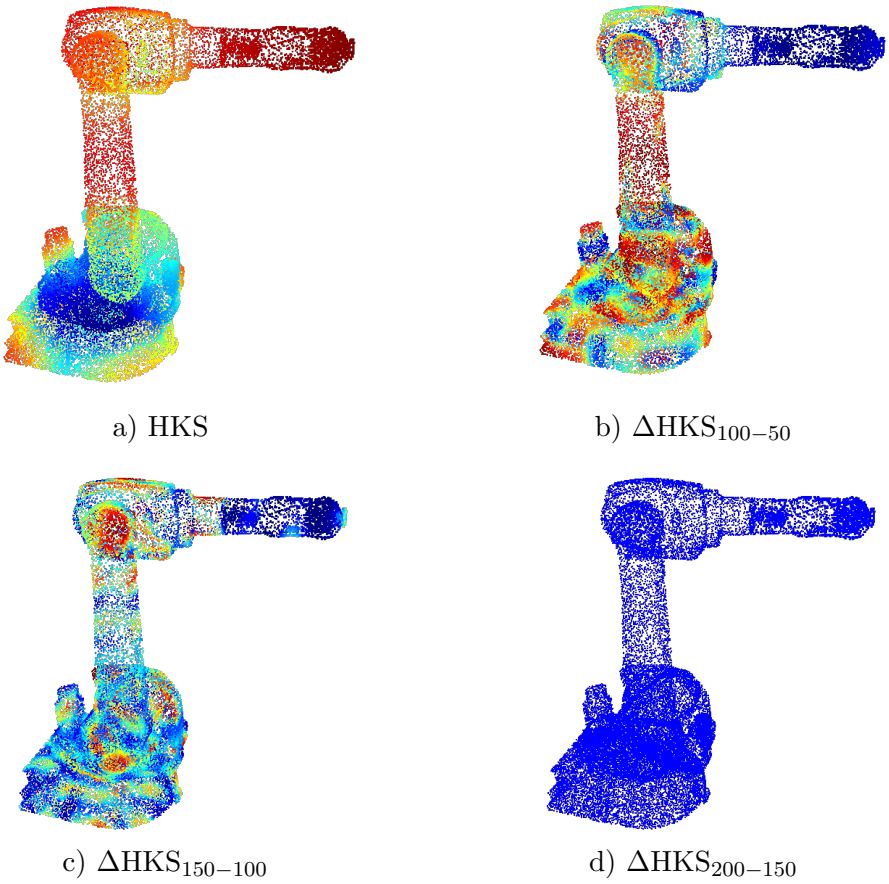
report that "it seemed to work well" for some test set. How many eigenpairs should be computed to allow the "rapid convergence" of the spectrum to converge appropriately for any given model is still an open question. Other works have suggested that some other subset of the spectrum provides better discriminatory power for particular cases, but ultimately conclude that the "first k eigenvalues" methods perform at least as well as those alternative subsets in general (see Table 2 in [42]). Additionally, alternative subsets take significant offline processing to develop for a given database [42].

5.1.6 Contribution

In this chapter, I characterize the level of approximation introduced to a typical spectral signature by fixed-number methods, discuss the limitations of these methods, and elaborate on my new tunable model-adaptive method of selecting the number of eigenpairs to use for each model in a database which helps mitigate those limitations. I also present analysis and discussion of tuning this method to adjust the balance between the computational speed and the precision of the computed signature.

5.2 Understanding the Impact of Eigenpair Cutoff

Spectral signatures are primarily used for shape similarity, comparing a query shape with the shapes of models in a database. Very rarely do all models in a database have a very similar intrinsic size. Often, even test databases contain models with an order of magnitude different numbers of vertices or points, let alone real-world examples of the kinds of databases in daily use at engineering firms and manufacturing companies.



a) HKS

b) $\Delta\text{HKS}_{100-50}$

c) $\Delta\text{HKS}_{150-100}$

d) $\Delta\text{HKS}_{200-150}$

FIGURE 5.2.1: a) The HKS vector $k_{0.03}$ for the 26k robot model sampling, scaled to a unit bounding box, computed with 200 eigenpairs. The difference between this HKS vector as computed with different numbers of eigenpairs are shown in b)-d): b) shows the difference between HKS_{100} and HKS_{50}, c) the difference between HKS_{150} and HKS_{100}, and d) the difference between HKS_{200} and HKS_{150}. In these plots, red is higher differences and blue is lower. Note that the final plot, d), shows no difference at all between the computed HKS vectors: The vectors have converged somewhere between 100 and 150 eigenpairs of information for this model at this t-scale.

For example, the CERTH/ITI Kinect scan database includes a model with 3657 points and another with 55808 points [1].

Figure 5.1.1 shows the difference in the amount of information captured by the first 300 eigenvalues for two differently-sampled models of the same object. The eigenvalues are exponents in spectral signatures, so the lower values captured by using N eigenvalues for the smaller model means capturing information not present in the larger model's N eigenpairs. Put differently, this means capturing excessive information (and therefore using excessive computational effort) if the larger model's amount of information is sufficient to the application. Two different samplings of the same model are used here rather than two different models of different sizes purely for clarity. The analogy of sufficient information holds even across models of different shapes.

5.2.1 Limitations of Fixed Number Methods

The primary limitations of these fixed-number methods are reduced precision for larger models, excessive computational effort for smaller models, and the introduction of a lack of consistency between measures which are supposed to be comparable.

In Figure 5.2.1, I offer an example of the kind of excessive computational effort which the method I introduce avoids without loss of precision. The figure shows the convergence of a particular HKS vector with respect to the number of eigenvalues and eigenvectors used in its computation is shown for a 26k point robot point cloud model. The HKS vector converges between 100 and 150 eigenpairs, as shown by the zero difference between the 150 and 200 eigenpair HKS vectors. This result implies that, for this model and sampling, at this t-value, computing any more than 150

eigenpairs is wasted computational effort. For online processing, that extra effort and the time associated with it can be the relevant factor in the timely detection of a feature or identification of an object.

5.3 A Tunable Model-Adaptive Cutoff Selection Method

Algorithm 2 Eigenvalue Cutoff Subroutine

```
function QTUNER(Laplacian,ξ,t,n_min,n_max)
    λ = eigs(Laplacian,50)
    while found ≠ TRUE do
        Fit quadratic to λ
        Use quadratic values as eigenvalue estimates to guess n_0 from ξ
        φ, λ = eigs(Laplacian,n_0 + 10)
        gap = abs(exp(t·λ(1:end-1))-exp(t·λ(2:end)))
        n = min(max(min(where gap < ξ),n_min),n_max)
        if n == length(gap)and not length(gap) == n_max  then
            set found = TRUE
        else
            if length(gap) == n_max then set found = TRUE
            end if
        end if
    end while
    return λ, ξ,n
end function
```

Instead of computing a fixed number of eigenvalues, my method instead computes a quite small user-set number of eigenvalues and then predicts approximate successive eigenvalues $\tilde{\lambda}$ by regression. These estimated eigenvalues provide a guide to what number of eigenpairs to compute for the spectral signature of the model in question. The estimated eigenvalues are examined for a point at which the contribution of the eigenvalue in question to the spectral signature is reduced below a parameter ξ (that

is, $e^{t\lambda_{n-1}} - e^{t\lambda_n} < \xi$). This allows analysts to compute different numbers of eigenvalues for different models while capturing more similar portions of the information encoded in those eigensystems.

This procedure can be performed for a given spectral signature scale (t-value), ideally the smallest t-value of interest for a given application, or with $t = 1$. Once the estimated eigenvalues are examined, a point a short distance past the estimated location of the cutoff n is selected (I choose $n + 10$ to avoid underestimating n) and the eigenvalues and eigenvectors for the model are computed up to that location. The new eigenvalues are checked to ensure that ξ has been reached; if it has not, the new set of eigenvalues just found are fed back into the quadratic estimator and the process begun again, using the additional information in the larger computed eigenvalues list to better guide the estimator. This should nearly always result in reaching ξ in a maximum of two eigensystem computations after the first 50-length computation.

The method (see Figure 2) is tunable mainly by two user-set parameters. The main tuning parameter is ξ, the cutoff difference. The cutoff difference specifies the minimum difference between pairs of subsequent eigenvalues. This parameter is where the majority of tuning for this method is performed. For our example in Figure 5.2.1, the value of $e^{\lambda_{99}} - e^{\lambda_{100}}$ was 120E-15 and the value of $e^{\lambda_{149}} - e^{\lambda_{150}}$ was 47E-21. The contribution to the HKS vector dropped more than a factor of a million across those fifty eigenvalues. The eigenvalue seed parameter I have fixed at 50 for convenience may be adjusted based on performance on an analysts system. This parameter allows the user to choose how many eigenvalues to compute before fitting the quadratic and predicting the convergence of $e^{\tilde{\lambda}}$. The user can also choose a minimum and maximum number of eigenvalues to compute for any given model and the t-value to use for the database, based on their specific spectral signature and application.

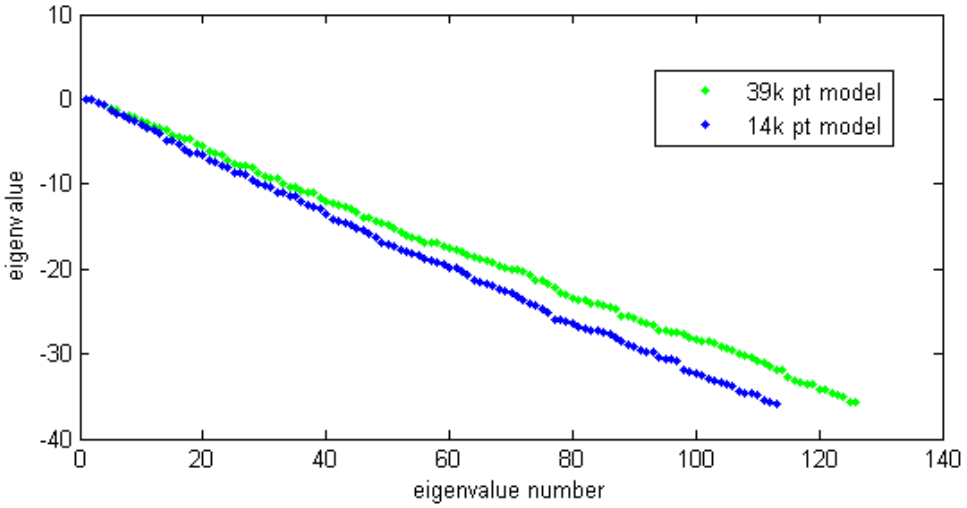
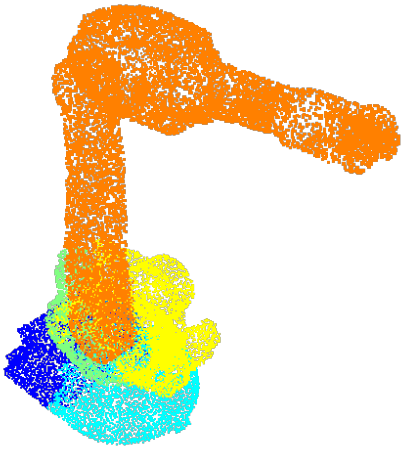


FIGURE 5.3.1: I note that spectral signatures hypothesize as part of their formulation that Laplace-Beltrami spectra will not contain repeated eigenvalues [56]. As there is no general understanding of the operation of spectral signatures outside that restriction, I do not concern myself with the possibility of high-multiplicity eigenvalues appearing at a critical juncture and interfering with the cutoff computation. The value of eigenvalues for the 14k and 39k point samplings of the robot model from the first to the cutoff number recommended by my method. Note the similar final values despite the different numbers of total eigenvalues computed to reach that point.

Setting ξ =100E-18 for the 14k point sampling of the robot model yields convergence at $n = 113$. For the 39k point sampling of the same robot model, the same ξ setting yields convergence at $n = 126$, thirteen eigenpairs further than that at which the smaller model achieved the same degree of convergence and significantly less than the 300 eigenpairs recommended in [56], but more than the 100 recommended in [17]. The eigenvalues for each of these samplings are shown in Figure 4 down to the cutoff values recommended by my algorithm with ξ =100E-18. Note that the eigenvalues reach approximately the same real value despite requiring different numbers of eigenvalues to be computed to reach that value.

Figure 5.3.2 shows a segmentation of a noisy robot model by the Wave Kernel

a)



b)

FIGURE 5.3.2: The ∼26000 point robot model segmented based on the WKS at $e = -1.4756$, using a) a fixed 300 eigenpairs as recommended in the original paper and b) using an automatically-computed 222 eigenpairs chosen by our tunable model-adaptive method. Note that colors are randomly assigned to segments and similarity of color should not be taken to mean similarity of segment between or within model.

Signature using a) 300 eigenpairs and b) an automatically-computed 222 eigenpairs chosen by my tunable adaptive selection method (using the same ξ and min and max ns as in the CERTH/ITI example). Note the similarity of the segmentations. Because the spectral signature has converged by 222 eigenvalues, the additional eigenvalues available to the $n = 300$ signature do not change the segmentation significantly. Additionally, the adaptively-tuned model required $\sim$7% less run time than the fixed-300 model.

5.3.1 Improved Consistency, Reduced Effort

This tunable adaptive cutoff method addresses the limitations of fixed-number methods discussed above. Computational effort is reduced while yielding the same effective amount of information. This permits more efficient signature development without effective loss of precision. Much larger or more complex models may require more eigenpairs than recommended by the fixed number methods to develop a similar level of convergence to smaller models in the same database.

The enhanced consistency of the spectral signature result between models of different shapes and sizes allows greater user confidence in matching candidates and segmentations based on spectral signature outputs. Sufficient inconsistency between spectral fraction used to compute a shape signature between two models in a database may lead to misidentification or misclassification of shapes. This technique helps to avoid such inconsistency.

5.3.2 Tuning for Speed vs. Precision

I additionally note that while the example above demonstrated tuning to the degree that no further convergence of the signature vector was possible with additional eigenpair computation, the nature of the tunable method allows for intentional and well-understood under-convergence. That is, if additional speed is required for some online application or computational effort must remain limited (e.g., by hardware or power requirements), a cutoff ξ may be chosen to intentionally get only enough eigenpairs to allow the degree of differentiation between shapes that your application requires. The tunable method allows this sort of designed just enough quantity of eigenpairs to be consistently specified across models of different sizes and over a range of scales.

As well, eigensystem computation speed does not scale linearly, so even if computing a consistent number of eigenpairs takes two or even three calls to the eigenvalue solver, so long as the average final number of eigenpairs is lower than would be chosen by a fixed-number method the total time to compute the spectral signature will be less. I note that nearly every model I have run reached ξ in only two calls to the eigenvalue solver (including the initial seed call of only 50 values). For example, for the first model of the CERTH/ITI database, the tunable cutoff method eigensystem call takes less than half the time of a traditional 300-pairs eigensystem call.

See Chapter 6 and especially Section 6.3.1 for an example of whole-database sorting improved by use of the tunable adaptive cutoff method.

5.4 Discussion

Spectral shape signatures are a popular class of similarity measures and have seen a great deal of use and many extensions in the literature in recent years. The number of Laplace-Beltrami eigenpairs used in computing these signatures is a critical parameter, but a parameter which had not yet been the subject of much investigation. I have discussed the limitations of fixed-number-of-eigenvalue methods for truncating model eigensystems for use in computing spectral shape signatures and developed a user-tunable method for adaptively determining required numbers of eigenpairs for different models of different shapes and scales.

My tunable adaptive method improves consistency between models of different samplings, enabling greater confidence in matching results and segmentations across large databases of different models from different scanning systems. This method can also greatly reduce computational overhead, enabling online use of techniques on systems where total computational power may be lower or resources may be in high demand, such as in autonomous systems (e.g. drones) or in real-time applications. Because this method is a modification to the general method of spectral shape signatures, the benefits obtained thereby can be combined with the advantages of any present or forthcoming published enhancements to spectral signature technology.

Once spectral signatures have been generated for a particular database, integrating additional scans may be made even more efficient by developing a function mapper to guess the cutoff number of eigenvalues for the new scan from information about the models which are already in the database. Such a function mapper could be a neural network or regression that develops an estimate of eigenvalue cutoff number n from implicit and explicit sizes of models already analyzed. This could enable closer-to-

realtime scan analysis by reducing computational overhead for mesh or point cloud models.

Chapter 6

Demonstrations and Examples

The preceding chapters have introduced a number of techniques for performing shape analysis on point cloud models without surface reconstruction. In this chapter, I provide details about the implementation with which I have tested these methods, runtime information, and a number of demonstrations on models which include both models sampled from synthetic CAD representations and models scanned from real objects with a range scanner.

Model-scale Gaussian Noise

Throughout this document, whenever I mention adding model-scale or proportional Gaussian noise to a model, I specifically mean that each point in the cloud of that model has been displaced in the x, y, and z directions by three samples drawn from a normal distribution with $\mu = 0$ and $\sigma = p\epsilon$. Recall from section 3.3 that ϵ is the average distance between points in the model's point cloud. Plots of example noise

distributions are included in Figure 6.3.14 for reference.

6.1 Computational details

I used Matlab as the primary environment for my work throughout this undertaking and the interfaces were all built in Matlab. The SPCL subroutine was built in MEX-compiled C++ based on the PCD Laplacian code of [12] which can be found at `http://web.cse.ohio-state.edu/~mbelkin/papers/pcdlaplace.tar`. The Matlab function `eigs` invokes ARPACK's C routines for solving the generalized eigenvalue problem.

I note that two factors primarily contribute to longer running times for a given model: the number of points in the model (equivalently, the sampling of the model) and the number of SPCL eigenvalues computed for the HKS calculation. As shown in Table 6.1.1, for a given model, increasing sampling density and increasing the number of computed eigenvalues both appear to increase running time near linearly. This is as we should expect: increasing the number of points in a model increases the number of rows of SPCL linearly (and each row's computation is constant in number of model points) and the increased number of SPCL rows increases the eigenvalue computation runtime, which in general is $O(n^3)$. In practice, however, for sparse matrices like the SPCL, larger numbers of points seem to effect runtime of Matlab's `eigs` function linearly.

Similarly, increasing the number of eigenvalues to be computed linearly increases the number of eigenvalue solution steps taken, the runtime for each of which depends only on number of model points. Improvements in the run times of the eigensystem

model	# pts	# eigs	SPCL time	eigs time	Total run time
Armadillo	5528	300	1.4s	9.9s	0.25 min
Armadillo	15228	300	4.0s	22.7s	0.59 min
camel	6815	300	1.6s	10.0s	0.22 min
camel	17036	300	4.15s	21.3s	0.45 min
turborotor	24503	150	8.8s	23.6s	0.59 min
turborotor	24503	300	8.7s	50.7s	1.05 min
turborotor	24503	500	8.8s	100.7s	1.91 min
turborotor	79723	300	21.2s	159.8s	3.21 min
turborotor	150163	300	42.0s	322.5s	6.46 min
robot	8446	300	2.4s	16.6s	0.34 min
robot	39889	150	10.1s	26.5s	0.73 min
robot	39889	300	10.1s	62.3s	1.35 min
c2u	4494	150	1.1s	3.12s	0.11 min
c2u	4494	300	1.1s	5.6s	0.16 min
c2u	35836	150	9.5s	35.6s	1.07 min
c2u	35836	300	9.5s	76.5s	1.78 min

TABLE 6.1.1: Running times for various models by number of points and number of eigenvalues computed. Running times measured using Matlab's `tic` and `toc` functions; eigenpairs computed with Matlab's `eigs` function. All computations performed on a Dell XPS 13 9350 with an Intel Core i5-6200U @ 2.30 GHz and 8GB RAM running Windows 10 x64.

computations can of course be improved by parallelization of the eigensystem routine, especially if implemented on the GPU.

6.2 An Overview of Parameter Dependence

There are a number of parameters that control the computations of the SPCL, HKS, and the clustering and segmentation methods I have described. As the results obtained from my methods are dependent upon careful selection of these parameters, I briefly demonstrate the effect upon the result of each parameter manipulation, in order to make applying these methods as straightforward as possible. Figure 6.2.1

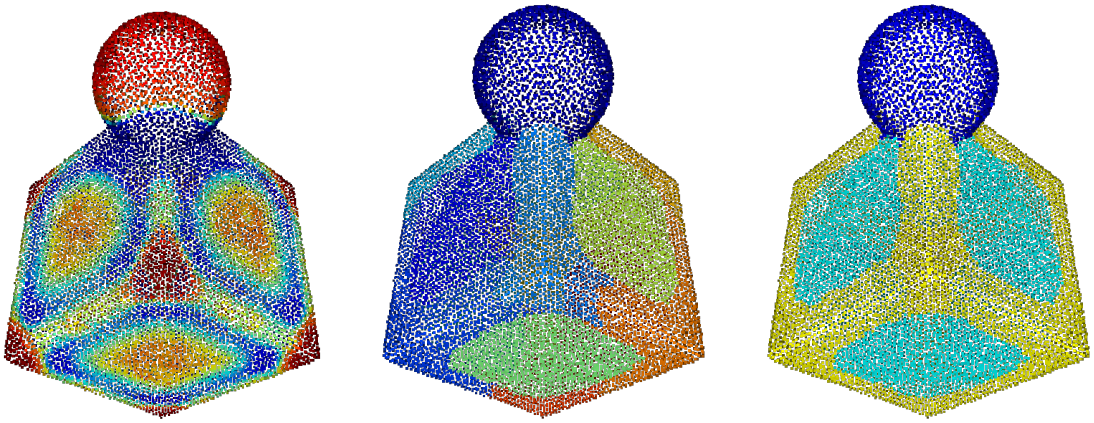
shows the synthetic 14,124 point "c4u" "ball & cube" model with model-scale Gaussian noise added and the baseline parameter values against which I will be comparing parameter changes throughout the section.

HKS t-scale

The first parameter is the t-scale(s) chosen for the HKS. This parameter determines the extent to which the HKS represents the surface either locally or globally. Figure 6.2.2 shows the effect on clustering of choosing a higher t-value (and therefore equivalently characterizing the model by a larger neighborhood around each point) for HKS computation. Note that I do not adjust the clustering parameter to find a better segmentation of the surface for this higher t-value HKS in order to show the interaction between the parameters in each step.

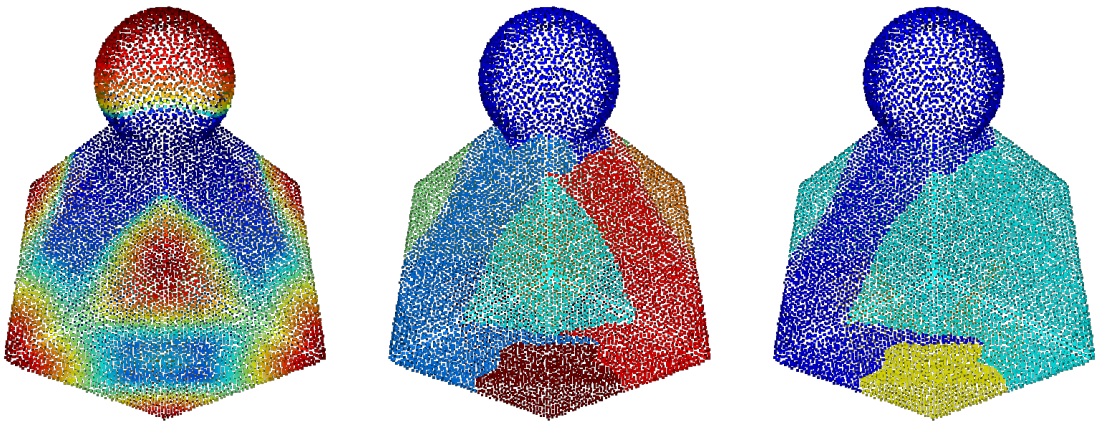
Clustering parameter τ

The method of feature vector selection and clustering introduces a second parameter, τ, which controls region merging in the agglomerative clustering procedure. Higher τ values cause more regions to merge, resulting in fewer clusters whereas lower τ values prevent regions from merging, resulting in a larger number of clusters. Figure 6.2.3 shows an example of the result of an excessively low τ on the clustering output. Notice, however, that despite a larger-than-desired number of segments (57 rather than the 14 shown in Figure 6.2.1b), the reclustered model appears very similar to that in the baseline example in Figure 6.2.1c.



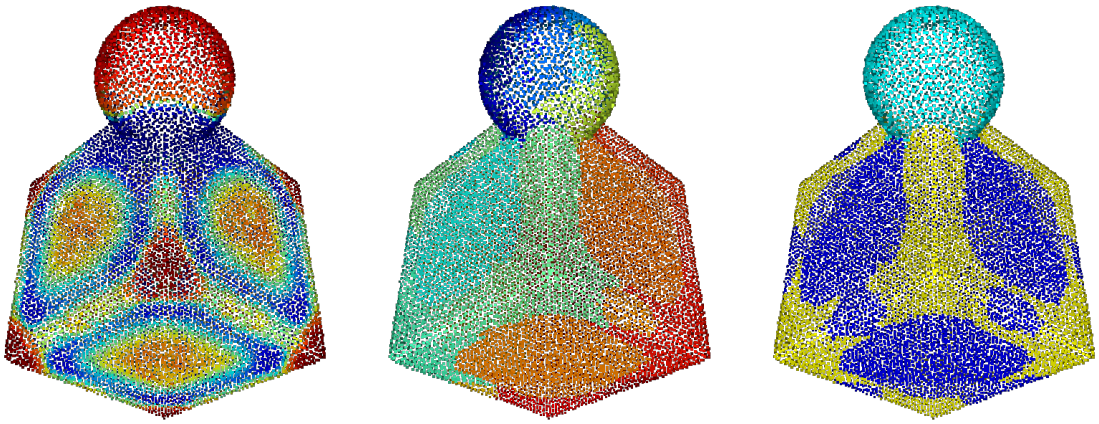
a) b) c)

FIGURE 6.2.1: (a) HKS at $t = 0.25$ with 300 SPCL eigs, (b) segmentation with $\tau = 0.007$, (c) Re-clustered segmentation with $n = 3$.



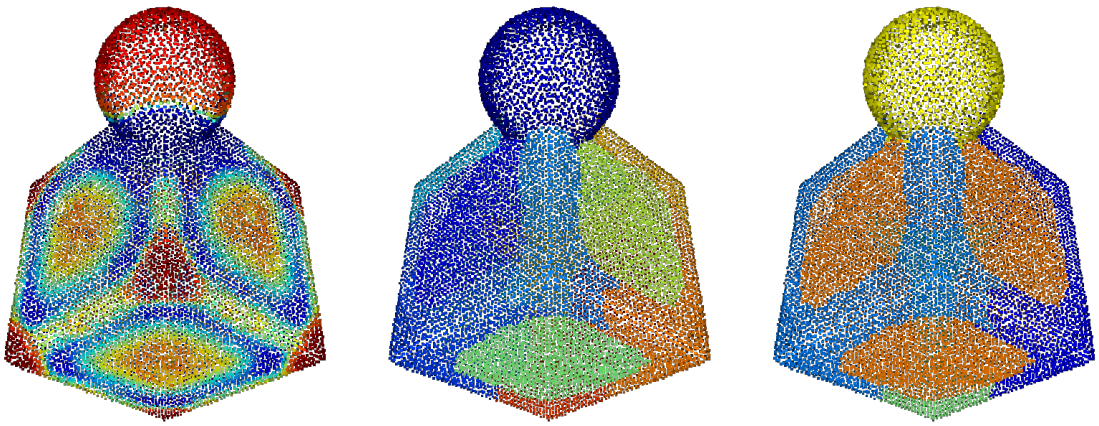
a) b) c)

FIGURE 6.2.2: (a) HKS at $t = 1.0$ with 300 SPCL eigs, (b) segmentation with $\tau = 0.007$, (c) Re-clustered segmentation with $n = 3$.



a) b) c)

FIGURE 6.2.3: (a) HKS at $t = 0.25$ with 300 SPCL eigs, (b) segmentation with $\tau = 0.002$, (c) Re-clustered segmentation with $n = 3$.



a) b) c)

FIGURE 6.2.4: (a) HKS at $t = 0.25$ with 300 SPCL eigs, (b) segmentation with $\tau = 0.007$, (c) Re-clustered segmentation with $n = 6$.

Unique cluster-type estimate

Finally, the re-clustering procedure I introduce to group segments of a model by type includes a user-defined estimate of the number of unique sub-shapes. Clearly, an incorrect estimate will cause some segments to be either grouped incorrectly into a sub-shape family with which they do not belong or else to remain self-grouped despite belonging to some multi-present family. More advanced clustering procedures which rely less on *a priori* information, such as spectral clustering [61] or clustering by reference to the gap statistic [58], or assisted by machine learning algorithms may be used to reduce this dependence of final sub-shape grouping on user input. Figure 6.2.4 shows the poorer sub-shape grouping produced by sub-shape family estimation when a higher-than-optimal estimate of the number of present sub-shape families is used.

6.3 Results of HKS and Segmentations on Several Models

In this section I demonstrate the quality results obtained by my point cloud analysis and segmentation on a variety of models beyond those discussed in the previous section. My analysis framework can be used not only to measure the similarity between whole point cloud models, but to segment these point cloud models into subsets that correspond to features. The similarity of these features can be measured in turn, allowing the individual features to then be matched with known features from a database. Furthermore, my method can in principal detect the number of distinct shape classes that make up a point cloud model, as illustrated in Figure

6.3.11, which could potentially be used as a way to explore geometric factorizations as well as solid/geometric model reconstructions of point cloud models.

Section 6.3.1 shows the effectiveness of the HKS as a similarity measure in co-categorizing a dataset of models with noise, collected by scanning with a Microsoft Kinect scanner [1, 26], into classes. Section 6.3.2 offers several examples of shape segmentations produced by my persistence-based segmentation, Section 6.3.3 discusses the Heat Walk segmentations, and Section 6.3.4 covers the curvature-aware extension of my persistence-based segmentation. Section 6.3.5 discusses some results of my work on clustering segments output by my segmentation step into "segment types". Section 6.3.6 concerns the resistance to noise demonstrated by the point cloud model HKS and my segmentation method, including topological noise in the example of the incomplete camel model. Finally, Section 6.3.7 provides additional comparisons between the results of my point cloud-based analysis and the equivalent mesh-based analysis and demonstrates the quality of results that can be obtained by processing point cloud models without surface reconstruction.

6.3.1 Similarity and Classification for the CERTH/ITI Range Scan Dataset

I present the following example on the CERTH/ITI Range Scan Dataset, which is a freely-available database of scanned point cloud models of a variety of small objects [26] produced by a Microsoft Kinect sensor with a depth resolution of about 1 cm. This demonstrates not only the effectiveness of my framework in analyzing a database of models but also its efficacy on real scanner data from a commercially-available depth camera.

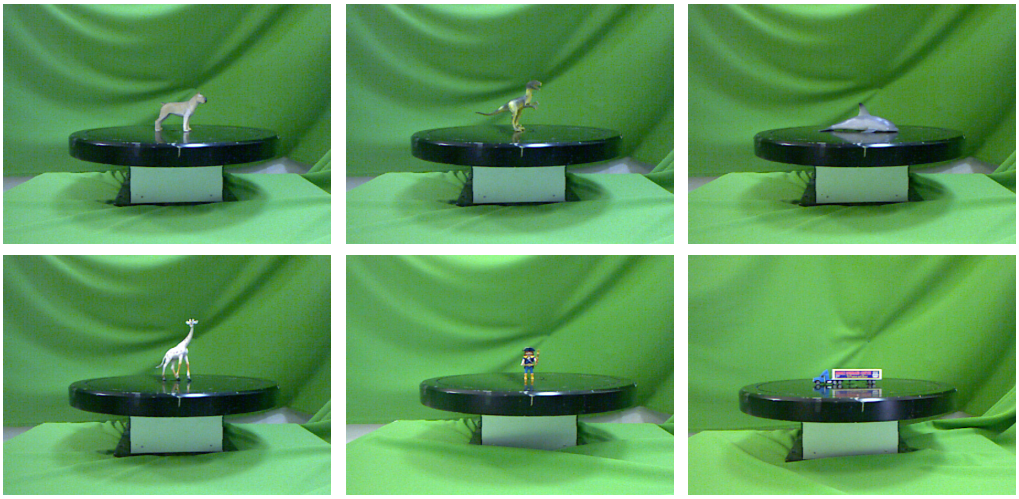


FIGURE 6.3.1: A sample of the various kinds of shapes represented in the CERTH/ITI Range Scan Database [26].

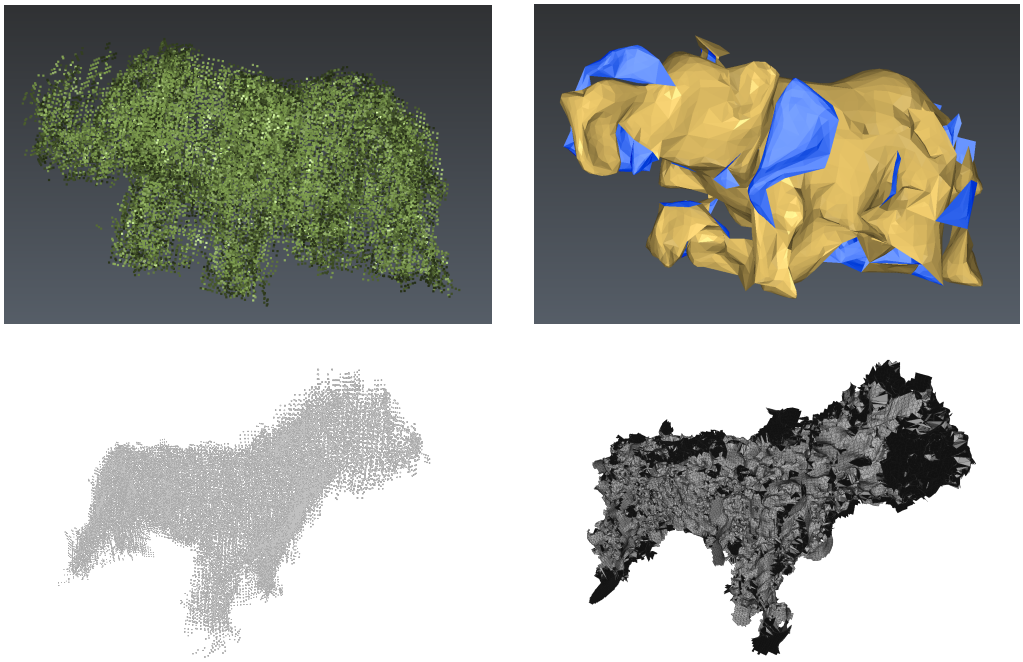


FIGURE 6.3.2: The noisy, poorly-aligned Kinect scans of the CERTH/ITI database are highly resistant to quality meshing, as these two models (with a naive meshing for each) demonstrate. The elephant mesh is produced by the Surface Mesh function of 3DReshaper Meteor. The dog mesh is produced by MeshLab's RIMLS method.

The objects in the database were scanned in eighteen different rotations of a turntable. The database provides an .XYZ file of the set of scans of each object rotated into a common coordinate system (so-called "registered" scans). Minimal cleaning has been performed to remove points outside the bounds of the turntable (i.e., background removal), but outliers remain, and the points of the aligned scans are often positioned so that a reconstructed surface through the points would result in self-intersections and other surface degeneracies. These scans, which are similar to those produced by industrial and hobby range scan systems, would be challenging to mesh without human operator intervention. Figure 6.3.1 provides an example of the various kinds of shapes represented in the database. Figure 6.3.2 shows the low-quality meshes which result from using established research and commercial meshing techniques on two representative point clouds from the dataset.

I demonstrate the efficacy of my point cloud method for grouping shapes based on their geometric similarity by running my procedure on this Kinect model database of 54 shapes. In other words, I build the shape signatures as described above and use them to measure geometric similarity between point cloud models, but without employing the segmentation techniques proposed next in Chapter 4. To this end, I remove the outlier points that remained in each model in the dataset after background removal by discarding the points belonging to non-merged segments of a given model that contain fewer than 1% of the total model points. This design is intended to prevent the discarding of true disjoint models if present while discarding unconnected patches of points from the background, turntable, or very severe alignment errors. There are more sophisticated outlier detection and removal methods (such as those in [53] and [62]) which would be suitable to use within the context of my framework, as well.

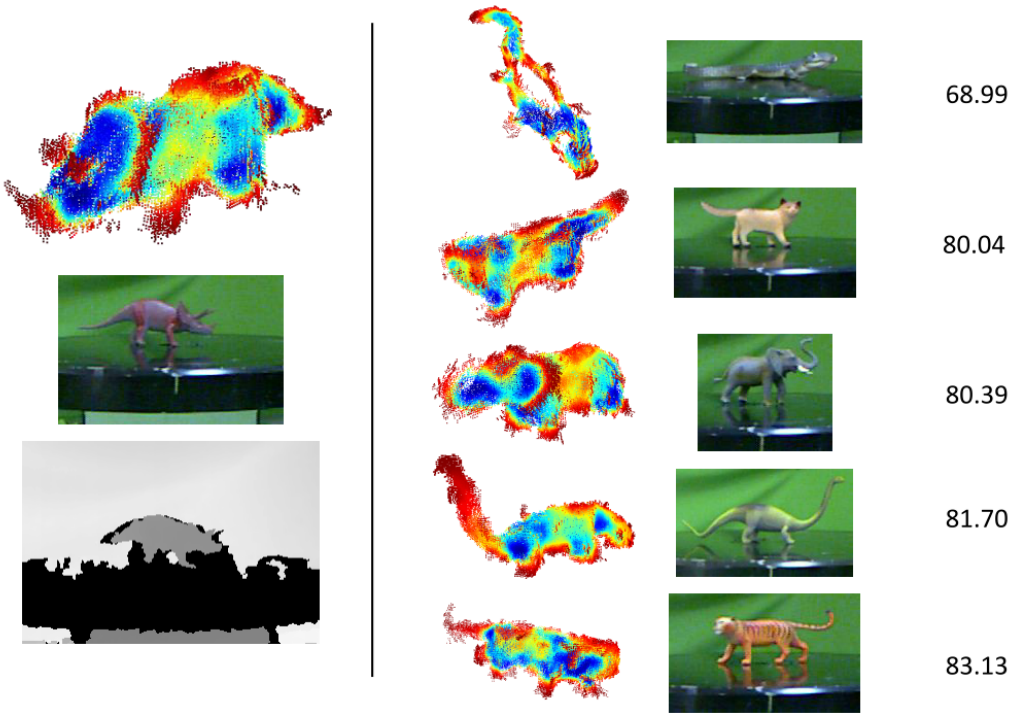


FIGURE 6.3.3: A query model from the CERTH/ITI dataset (left) and its five closest matches in the set according to matching HKS vectors from noisy point cloud models. The numbers aligned with the matching models from the database are the distance scores computed from the HKS feature vectors used to determine the best matches. Note the geometric and shape class similarities between query shape and matches.

I normalize the size of the models to a unit bounding box, and then compute the SPCL. The HKS at $t = 0.001$ was used to find 15 persistent clusters and subsequently a 15×15 feature vector was produced as described above. The HKS values comprising the feature vector were scaled to a max value of 1. Matching quality can be examined with the models representing the top matches for a given query. In Figure 6.3.3, I show an example of the top five models matched to a particular query model using my point cloud-based techniques.

The "natural" classification suggested by the CERTH/ITI dataset authors would label the query model a member of the class "dinosaurs" and only one of the top five

matches is also so classified (other categories represented include "land mammals" and "other"). However, all of the top five matches appear to belong to a slightly broader class of "quadrupedal caudate land animals", a more geometrically defined class than the geometrically variable but more deeply contextualized "dinosaurs" class.

The top-5 hit rate of my current CPU-based implementation used for the CERTH database of noisy and incomplete point cloud models is $> 61\%$. As a comparison, a top-5 hit rate of 88% is reported in [24] for a database of 50 *noiseless* and *meshed* models, in different poses and different levels of completeness. The difference in the two top-5 hit rates is due to the fact that the CERTH dataset is not only more general, but also has significantly more noise than the dataset used in [24], and that the categories given by the CERTH dataset authors are, as noted above, somewhat narrow and non-geometric.

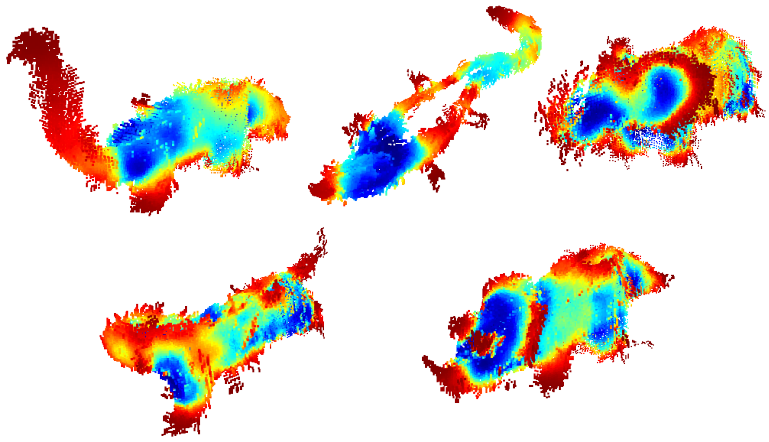


FIGURE 6.3.4: Several similar models from the CERTH database colored by one of the HKS vectors computed for the database matching procedure.

CERTH/ITI Database Matching with the Tunable Adaptive Method

As mentioned above, I have also performed this experiment using the tunable adaptive cutoff eigenvalues method. Using the same database allows for easy evaluation of improvement.

Again, minimal cleaning was performed but outlier points remain. These scans are challenging to mesh without human operator intervention. For this experiment, I use the point cloud model HKS analysis method presented in Chapter 3 to develop HKS vectors, techniques presented in Chapter 4 to extract feature vectors from the HKS vectors on the models, and then *a priori* select minimum $n = 80$, maximum $n = 500$, and cutoff ξ =100E-12. Figure 6.3.4 shows a handful of similar models from the database colored by some of the HKS vectors used.

The tunable model-adaptive selection method suggested a variety of eigenvalue truncation levels for the CERTH/ITI Kinect model database. The minimum size suggested was equal to the *a priori* minimum 80 eigenpairs for a model of 5046 points while the maximum suggestion was limited by the *a priori* maximum of 500 eigenpairs. Figure 6.3.5 shows the number of eigenpairs used in computing the heat kernel signature vectors for each model plotted against model size.

Comparing the Top-5 Hit Rate for HKS of the CERTH/ITI database using fixed 300 eigenpairs (as demonstrated above) versus using the tunable adaptive cutoff method described in Chapter 5 shows a marked improvement for the tunable adaptive cutoff method of 10% greater portion (71%) of same-category matches in the top five matches for each model. This demonstrates the importance of the enhanced consistency provided by the tunable method over a fixed number method in real application for matching and categorization.

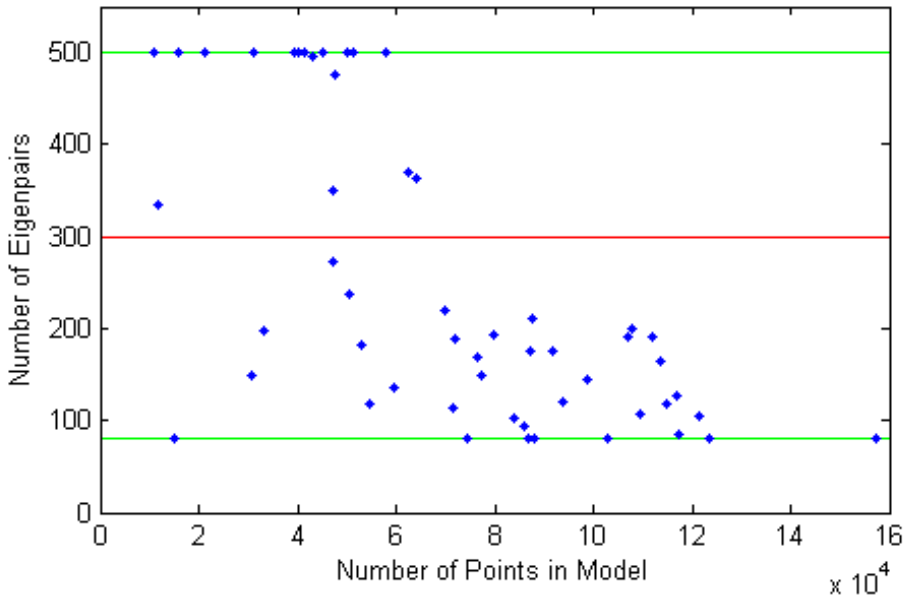
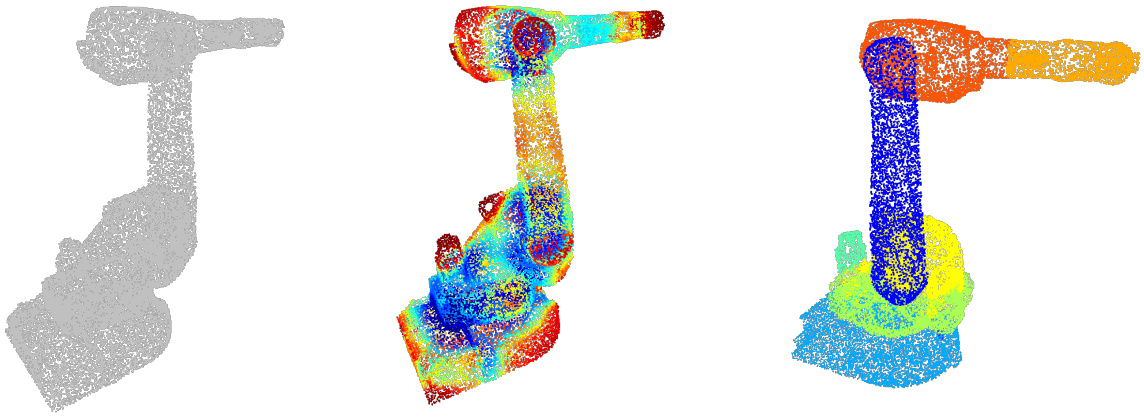


FIGURE 6.3.5: Number of eigenpairs suggested (limited to a minimum of 80 and a maximum of 500 eigenpairs) by the tunable model-adaptive cutoff method for the models in the CERTH/ITI Kinect scan database plotted by number of points in the (automatically) trimmed model. The red line at 300 represents the canonical suggested number of eigenpairs for HKS for each the model. For any model above the line, 300 values under-represents the Laplacian reducing consistency and for any model below the line, 300 values represents more computational effort or over-specificity. Note that the number of eigenpairs is not determined by model size.

6.3.2 Persistence-Based Segmentations

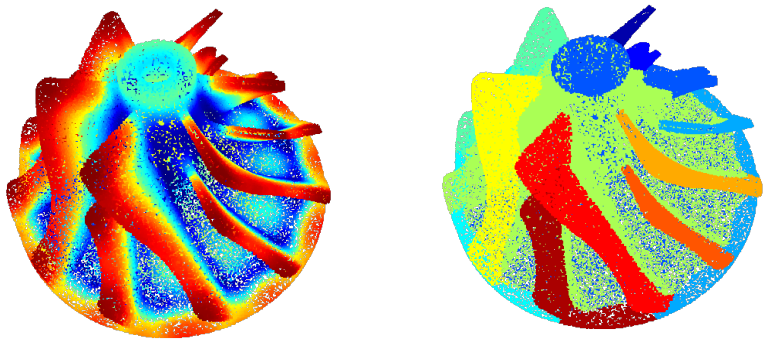
The following examples demonstrate the quality segmentation results which can be obtained using the techniques proposed in this paper.

Figure 6.3.6 shows the point cloud, HKS vector, and automatic segmentation of the ABB robot model first introduced in Figure 3.2.1. The proposed automatic persistent homological segmentation clearly delineates the point cloud into regions roughly corresponding to bulk design features that an engineer would find meaningful. Note especially the very well-defined middle rotational link and base link.



a) b) c)

FIGURE 6.3.6: (a) A point cloud Monte Carlo-sampled from an STL file of a robot courtesy of ABB [2] with model-scale noise ($\mu = \epsilon/2$, $p = 0.125$, see 6) added to each point. (b) The HKS values for at $t = 4\lambda_{300}^{-1} ln10$. (c) One of the many possible automatic segmentations of the model from that HKS vector and the SPCL of the point cloud using my method (see Algorithm 1).



a) b)

FIGURE 6.3.7: (a) The HKS vector for a turborotor model at $t = 0.001$. (b) An automatic persistent homological segmentation of the model from that HKS vector and the SPCL of the point cloud using my method (see Algorithm 1).

Figure 6.3.7 shows the HKS and persistent homological segmentation for an approximately 150,000 point turborotor-type industrial part. This point cloud is a high-quality sampling at good resolution, demonstrating the efficacy of the presented

framework on high-quality, industrial-type point cloud models. Note the good definition of the numerous fins projecting from the central cone and bore of the model.

Despite their different forms, resolutions, and constituent features, the ABB robot and turborotor models are both models of real engineering artifacts which can be segmented by the techniques introduced in this article directly from their point cloud information without either explicit connectivity or normal vector information.

6.3.3 Point Cloud Segmentation with Heat Walk

a) b) c)

FIGURE 6.3.8: A point cloud of the classic Armadillo model, first a) colored with the HKS vector $k_0.1$, b) after running the accumulator steps of the Heat Walk algorithm for $t = 0.1$ (through step 6 above), and c) after the complete Heat Walk is finished, showing accumulator regions and a dissipator region, as well, around the midsection

The core of my techniques can be extended to related methods, such as enabling the use of the Heat Walk segmenter [14] on point cloud models. Figure 6.3.8b,c show examples of the result of the Heat Walk segmentation method used for a point cloud model as I discuss in section 4.4. The near identical results (shown in Figure 6.3.15) between the point cloud model method run with the techniques I introduce in this paper and the mesh model method using traditional MeshLP, Heat Kernel, and Heat

Walks serve as an example of the usability and correspondence of my methods.

6.3.4 Curvature-Aware Segmentation

My techniques can be combined with methods which seek to improve shape segmentation accuracy for "sharp" shapes by incorporating explicit curvature information.

Figure 6.3.9 shows an example of combining spectral signature information with explicit local curvature information for segmentations. In this example, I have used the pointwise normal information which is computed as a byproduct of local tangent space approximation during SPCL construction (see Section 3.2) in reformulating the "seeded" segmentation procedure of [39].

My formulation 1) determines approximate curvature values by examining the maximum difference in normal direction between points in local neighborhoods, 2) finds sharp edges by finding vertices with particularly drastic changes of normal direction (i.e., discontinuities) in their neighborhood (see Figure 3.2.3 for an example of what this looks like on the Fandisk model), 3) performs a k-means clustering of the curvature values and treats local neighbors with the same k-means cluster value as seeds, growing those seed clusters without letting them cross sharp edges, 4) builds a region adjacency graph for the clusters based on the mean curvature of the regions and the mean curvatures of the boundaries between regions, then 5) merges clusters across the smallest graph edges in that adjacency graph (i.e., the most similar adjacent clusters on the surface) until some desired number of clusters or the minimum graph edge value is exceeds a prescribed threshold.

This example formulation underscores the flexibility of my method for point cloud shape analysis. In principal, my analysis pipeline allows any analysis technique which

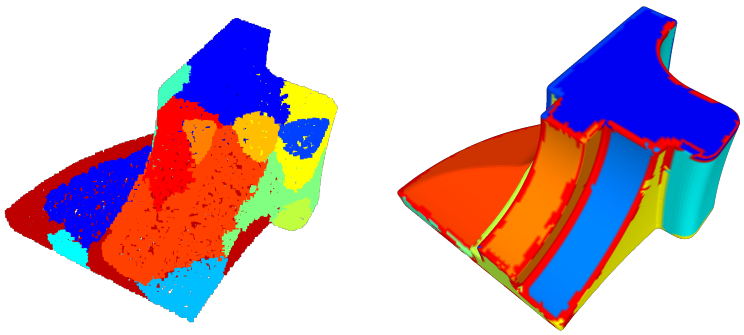


FIGURE 6.3.9: a) Poor results on the fandisk model from segmenting without understanding of the various sharp edges contrasted with b) the results possible from segmenting the fandisk model with a method that includes explicit information about the local curvature, derived from local normals as computed during the construction of the SPCL.

relies on a shape signature computed on mesh models to be converted to operate on a point cloud models.

6.3.5 Re-Clustering and Segment Type

The techniques proposed in this article can be used not only to obtain segmentations, but to group the identified segments by geometric similarity. By exploiting the similarity information (computed in the HKS) that is retained to identify the point clusters which make up the model segments, I can, with little additional cost, understand easily which segments represent similar features.

Figure 6.3.10 demonstrates the segment clustering technique of Section 4.3 on the Heat Walk segmentation from Figure 6.3.8. Recall that in this technique, the segments identified by the Heat Walk are clustered by Heat Kernel values to identify similarity between segmentation sub-shapes. Note that the limbs have all been classified as the same kind of cluster and that the three head sub-clusters have been identified as one

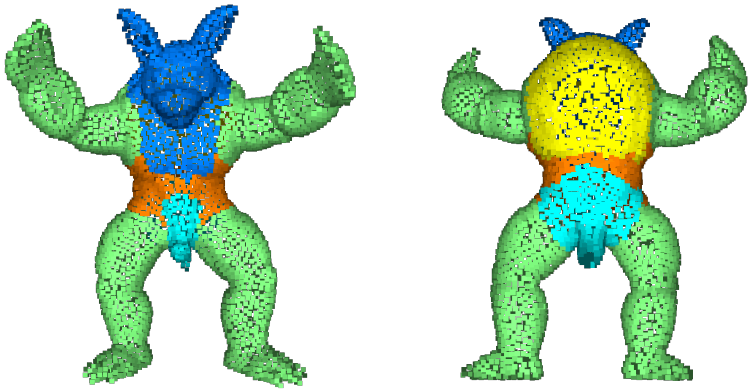


FIGURE 6.3.10: The Armadillo model with the heat walk output segments clustered by my method from Section 4.3 by initial heat potential value (i.e., heat kernel signature value) at the exemplar point for that segment. For the dissipator cluster, I assign the mean of the HKS values of the rest of the points (as that cluster is closer to an ideal dissipator than to the other clusters). Note that all of the limbs are assigned to the same segment type and all three head clusters are also assigned to their own group. These groupings may both happen in one clustering because the HKS value used to regroup the segments by type includes local and global geometry information.

cluster.

In Figure 6.3.11, the method identifies both shapes present and in fact identifies two clusters as the point at which the Clustering Balance [37] is minimized, implying that this is the "best" clustering by that metric, as shown in Figure 6.3.12.

Practically speaking, this technique can be used to identify the *number of feature types* which are present in a model. This information can be used in downstream applications where high-level semantic information is critical, such as in manufacturing planning (e.g., holes bored during in one operation, slots cut in a different operation) or robot task planning (are there handles and knobs in a part or just handles?).

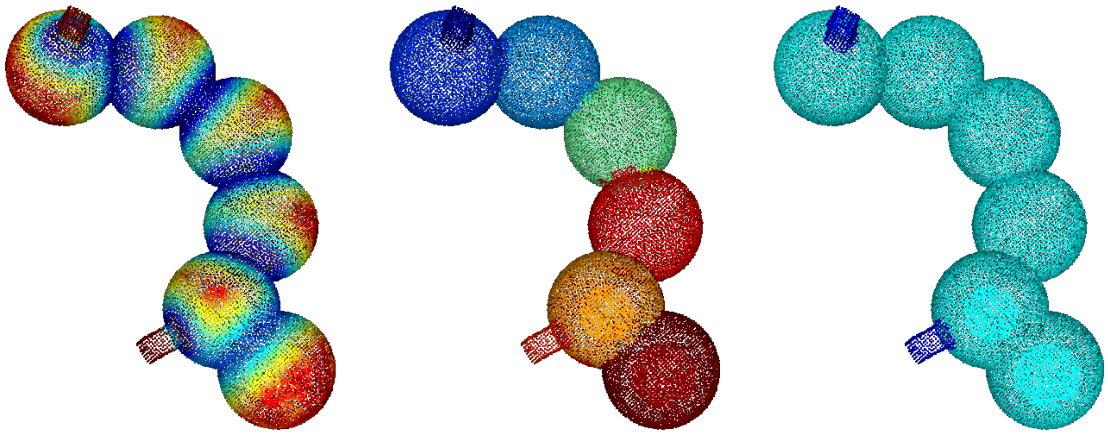


FIGURE 6.3.11: (a) The first HKS vector for this synthetic model of fused spheres with Gaussian noise ($\mu = \epsilon/2$, $p = 0.125$) is used to (b) segment the model by the method described in Chapter 4 for a particular set of parameters, showing eight individual segments. (c) That same segmentation's clusters is then merged down by hierarchical clustering to only two clusters, showing the two distinct shapes present in the model.

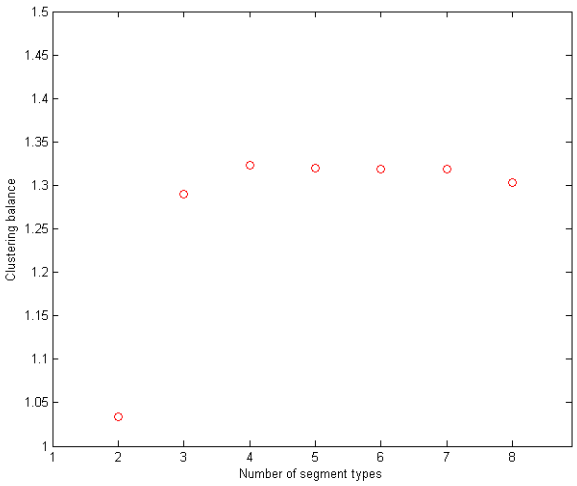


FIGURE 6.3.12: This figure shows the Clustering Balance computed for the HKS values of the segments of the fused spheres model from Figure 6.3.11 for the original eight clusters down to two clusters. Note the steep drop off to two clusters. This sharp drop in clustering balance suggests that two clusters is the "correct" number of clusters in the model.

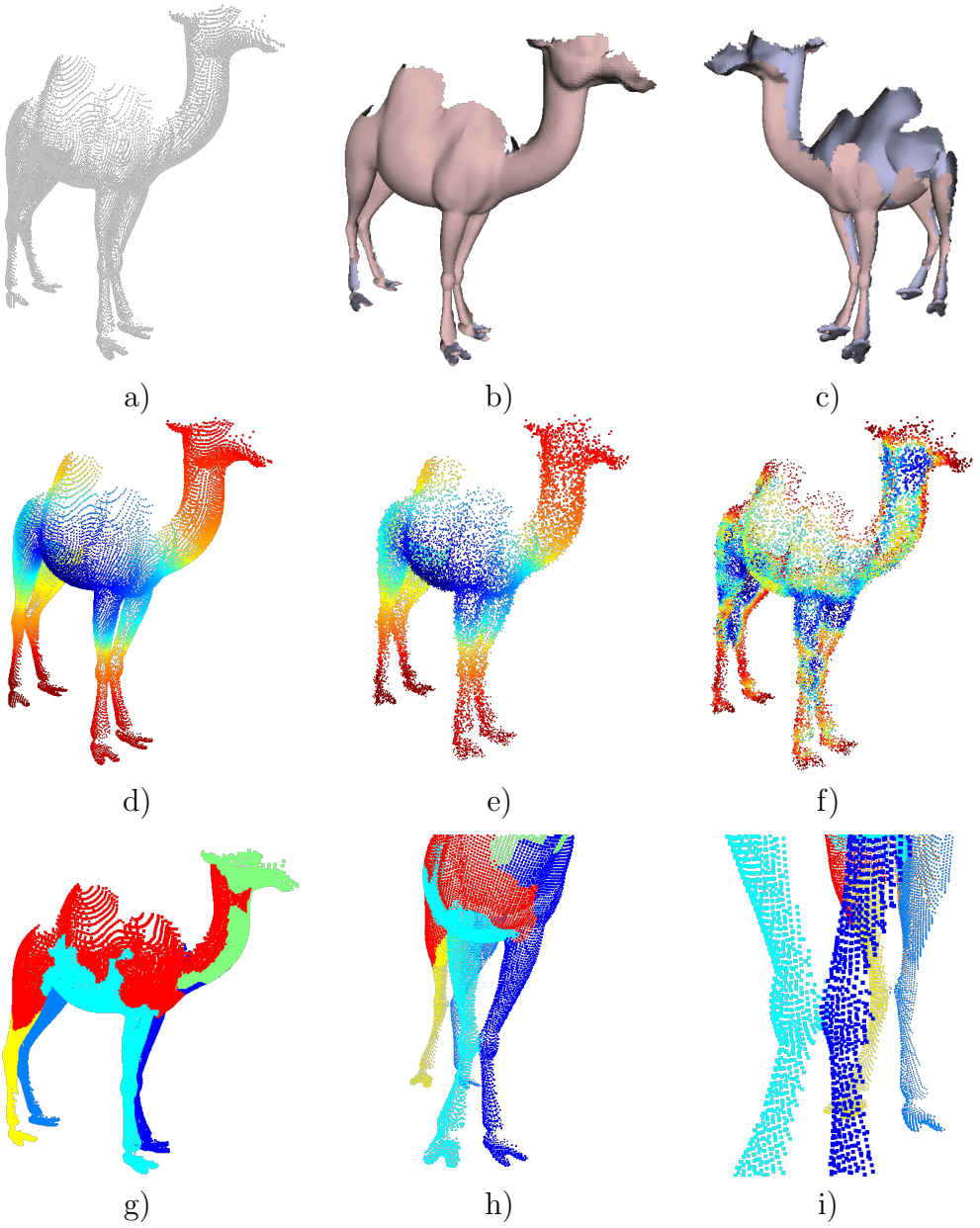


FIGURE 6.3.13: a) The point cloud of the incomplete camel model with no added noise. b) An RIMLS meshing of the point cloud. c) Another view of the RIMLS mesh to better show the severe incompleteness of the model. d) The $t = 0.1$ HKS vector for the noiseless camel model. e) The ($t = 0.1$) HKS vector for the camel model with model-scale Gaussian noise ($\mu = \epsilon/2$, $p = 0.125$) (see 6) added to the points of the model before processing. f) The HKS vector for $t = 0.001$ for the noised camel model, showing additional definition at local scales but still retaining some of the global understanding evinced by the higher t-value HKS vectors. g-i) A segmentation produced by segmenting the HKS of the camel model at $t = 0.0001$ into 7 segments automatically by the methods described in Sections 3.2,3.3,and 4.2. Note especially the separation of the legs despite topological noise at the knees.

6.3.6 Resistance to Noise and Model Incompleteness

To demonstrate the resistance to various kinds of noise these techniques provide, I present the incomplete camel model of Figure 2.1.1 with the proximal points in the knee areas and the Armadillo model with additions of Gaussian noise.

The original noiseless point cloud for the camel is highly incomplete — most of the left side of the camel model is simply missing as shown in Figure 6.3.13(c), which displays the meshed surface output by the recent RIMLS reconstruction [44] implemented in Meshlab. As I showed in Figure 2.1.1, automatic meshing solutions may produce unpredictable and unintended meshing results, which in this case is a mesh model that conjoins the legs at the knees. This is a form of topological noise. Applying my framework to the analysis of a possibly noisy and incomplete point cloud model proceeds as follows:

1. Compute the SPCL as discussed in Section 3.2 and 8.1.

2. Compute the HKS vectors of interest from the SPCL (see Figure 6.3.13(d-e) as discussed in Section 3.3. Because of the local nature of the SPCL in describing the surface and the reliance of spectral signatures like the HKS on a Laplacian, the local and global curvatures of the model are still clearly represented in spite of the large missing sections. In Figures 6.3.13(e) and 6.3.13(f), the camel point cloud model has had model-scale Gaussian noise ($\mu = \epsilon/2$, $p = 0.125$) added and the SPCL and HKS have then been computed, demonstrating the good robustness of spectral methods against scanner noise. The HKS vector t-values recommended for database matching are based on the eigenvalues computed during HKS computation (see Section 3.3), but lower and higher HKS t-values

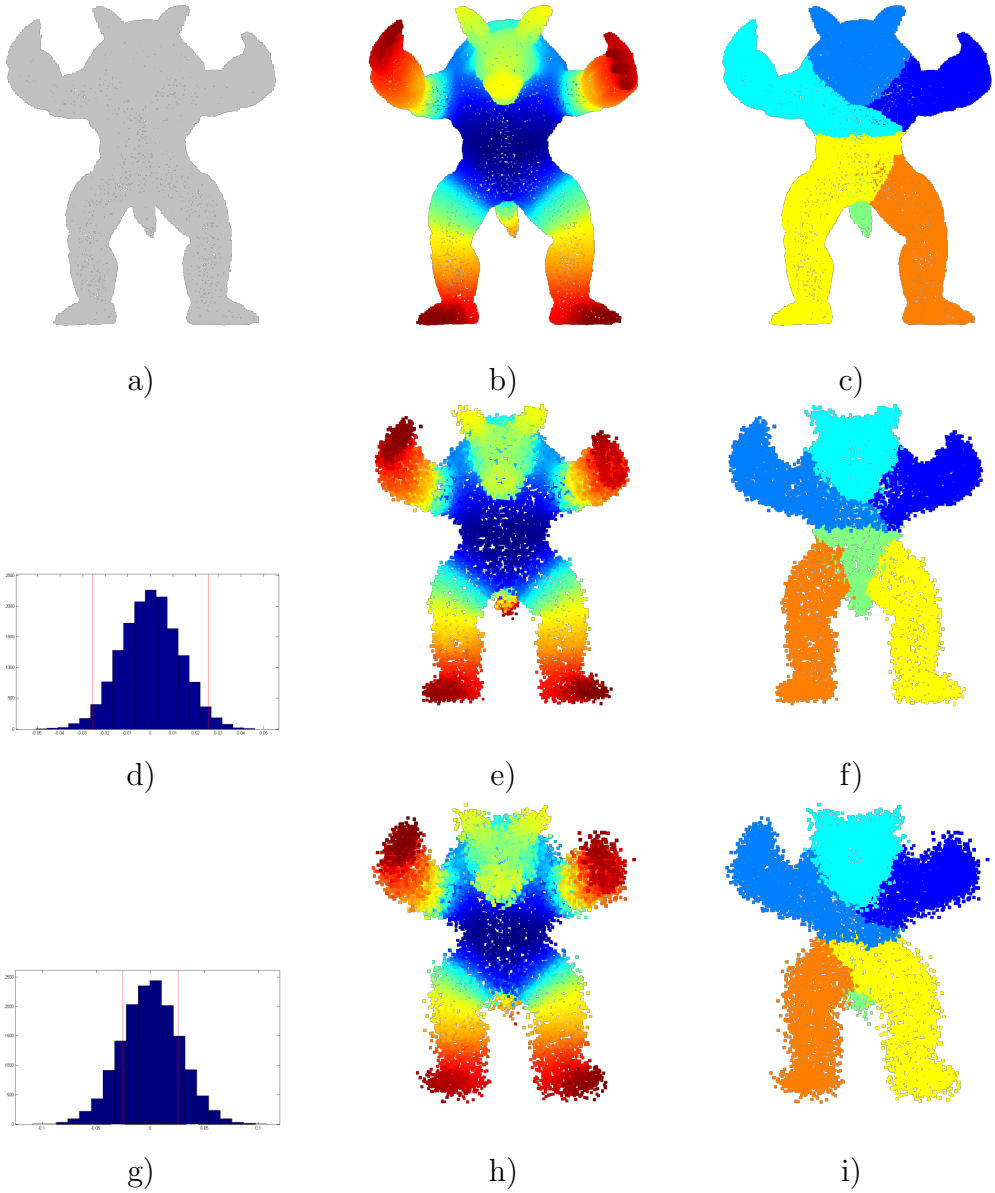


FIGURE 6.3.14: a) The point cloud of the approximately 15,000 point Armadillo model with no added noise. b) The $t = 0.1$ HKS vector for the model. c) The PD-type segmentation of the model into 7 segments. d) The noise added to the Armadillo model by a normal distribution with $\mu = 0$ and $\sigma = 0.5\epsilon$, 50% of the average distance between points in the model (see 6). The red verticals represent plus and minus the average inter-point distance. e) The $t = 0.1$ HKS vector computed on the 50% noised Armadillo PC model. f) The PD-type segmentation of this 50% noised model into 7 segments. g) The noise added to the Armadillo model by a normal distribution with $\mu = 0$ and $\sigma = \epsilon$, the average distance between points in the model. The red verticals represent plus and minus the average inter-point distance. h) The $t = 0.1$ HKS vector computed on the 100% noised Armadillo PC model. i) The PD-type segmentation of this 100% noised model into 7 segments.

are useful for emphasizing either global (e.g., ends of a shape, projections from a core) or local (e.g., smaller features, local curvatures) shape.

3. Select a target number of segments to be automatically generated by the method described in Algorithm 1. At this point my method supports a user in the selection of the number of segments; development of mechanisms to automatically detect the number of intrinsic segments of a given point cloud model are an open topic. Given a target number of segments, running the method of Algorithm 1 on the chosen HKS vector with a very high τ (say, $\tau = 1e8$) merges all of the points together into one cluster for the whole model. Examining the persistence diagram of this cluster allows one to select a proper τ to generate the desired number of model segments. My method then clusters nearby model points by their HKS values. As shown in Figure 6.3.13g-i, a seven-segment (body, head, four legs, and tail is a reasonable guess for a segmentation of a camel) segmentation produced automatically by my method at $\tau = 29$ shows distinct legs for both front and back pairs of legs.

The example of the Armadillo model which follows the camel example demonstrates the impressive resistance to sensor/geometrical noise which the HKS displays. In rows two and three of Figure 6.3.14, the point cloud of the Armadillo was noised by adding random numbers to each coordinate of each point from a normal distribution with a mean of zero and a standard deviation of approximately 50% and 100% of the average inter-point distance respectively. As can be seen in the second and third columns, despite very high levels of random noise, the HKS vectors appear virtually unchanged and the PD-type segmentations naively computed from them are also highly consistent with the non-noised result.

6.3.7 Comparing Mesh-based and Point Cloud-based Segmentations

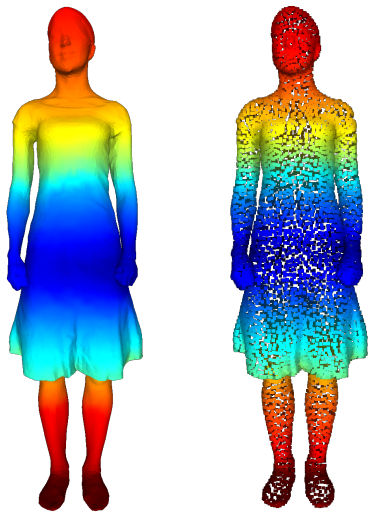
In order to demonstrate the quality results that point cloud shape analysis allows, I here compare segmentation of a single model (Armadillo) and the HKS results of the Girl Dancing Samba dataset of [60]. Additionally, I run the Girl Dancing Samba dataset with and without the tunable adaptive eigenpairs method to show the improvements that technique allows.

a) b)

FIGURE 6.3.15: The Armadillo model with the Heat Walk segmentation on a) the point cloud version of the model and b) the equivalent mesh model. Note the near identical results between model types.

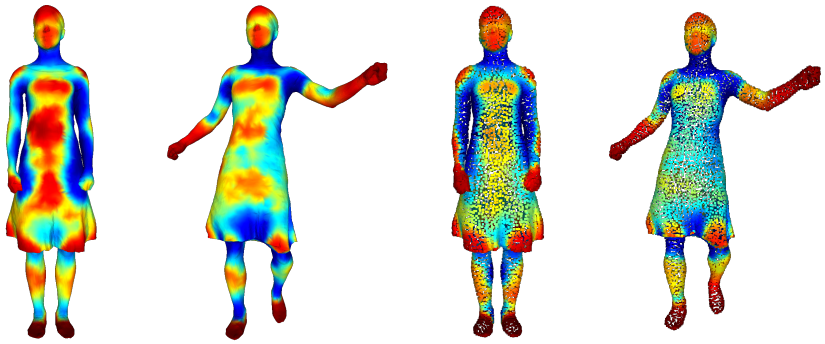
Figure 6.3.15 shows the point cloud and mesh versions of the Heat Walk segmentation for the Armadillo model. Note the high degree of similarity. In each case, every limb is segmented away from the core of the model as is the tail and each ear-dominated head segment and the midsection is identified as a dissipator region.

Figure 6.3.16 shows the first model in the dataset colored with HKS vectors. Each model in the dataset has 9971 points in the point cloud model or 9971 vertices in the



a) b)

FIGURE 6.3.16: The first model from the Girl Dancing Samba dataset colored by HKS vector $k_{t=0.1}$ as computed on a) mesh model and b) point cloud. Note the similarity between computed HKS vectors regardless of model type.



a) b) c) d)

FIGURE 6.3.17: a-d) The 1st and 45th models from the Girl Dancing Samba dataset colored by the "first" HKS vector as computed on mesh model (a&b) and point cloud model (c&d). Note in c) the (correct) distinction of the figure's left hand compared with the mesh model in a).

mesh model. Since the 150 models in the database are all of the same physical body, a person, and the spectral signature I've chosen, the HKS, is to a large degree pose-

invariant, we should expect to see only small differences between feature vectors and very similar shape segmentations between models.

Figure 6.3.17 shows that, as we expect, different models of the same figure in different poses have highly similar HKS vectors in both mesh and point cloud space. The point cloud HKS for the first pose actually differentiates the left hand of the model to a greater degree than the mesh version does, again demonstrating the resistance of point cloud methods to topological disturbance.

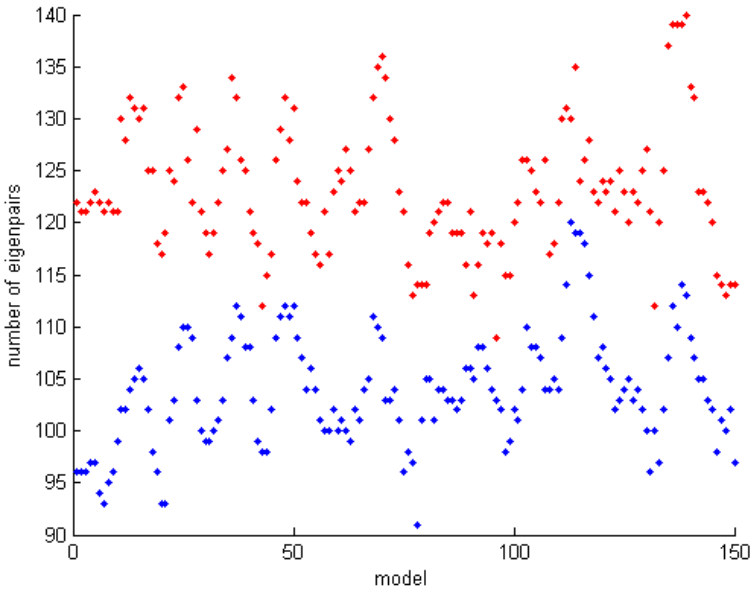


FIGURE 6.3.18: Number of eigenpairs suggested (limited to a minimum of 70 and a maximum of 600 eigenpairs) by the tunable model-adaptive cutoff method for the models in the Girl Dancing Samba database plotted by model number (since all models have the same number of points/vertices). Points in red indicate the number of eigenpairs suggested for each mesh model. Points in blue indicate the number of eigenpairs suggested for each point cloud model. Note once more that the number of eigenpairs is clearly not determined by model size.

As shown in Figure 6.3.18, the tunable adaptive version of the dataset analysis resulted in between 91 and 120 eigenpairs being used in computing the HKS for

the point cloud model version and between 109 and 140 eigenpairs being used in computing the HKS for the mesh model version. The fixed-n method runs each used 300 eigenpairs. The tunable adaptive method, therefore, reduced the length of the computed eigenspectrum for each model by more than half, reducing the amount of time taken to analyze the dataset.

Chapter 7

Conclusion

Industrial and academic interest in shape analysis continues to grow as design, analysis, animation, and engineering is increasingly performed in a digital space. Depth camera technology is becoming ubiquitous, as well, allowing anyone to easily capture dense, noisy point clouds models of real physical objects. The obvious questions of how we can automatically understand shapes from general 3D point cloud inputs have been traditionally answered by first performing a surface reconstruction on the input point cloud, followed by mesh processing. Unfortunately, meshing point clouds is itself a non-trivial, often application-dependent task with limited guarantees on the validity of the resulting mesh.

On the other hand, useful analysis methods for describing and segmenting point clouds *without surface reconstruction*, especially those of relevance to design and engineering, are quite limited in the literature. Mesh-model–based analysis methods are well-developed, yet creating a quality surface mesh from general noisy point cloud inputs automatically remains challenging.

7.1 Contributions

7.1.1 The SPCL and Spectral Shape Signatures on Point Clouds

I presented an integrated analysis procedure for shape description, similarity, and segmentation on point cloud models of real objects. I have shown that our Laplace-Beltrami estimate, the SPCL, does not worsen the error terms of the PCDL, which is known to converge in the limit to the manifold Laplacian. This, in turn, allows analysts to apply existing or develop new physics-based spectral shape signature methods directly on point cloud models. I showed that my construction provides an estimate of surface normals and the neighborhood graph implied by the SPCL, which in turn can be used to find mesh n-ring–equivalent local neighborhoods and to apply algorithms from the mesh literature (such as those for finding feature vectors) to point cloud models, affording a compact similarity-based tool for comparing the shapes represented by point clouds. Thanks to its reliance on mathematical models of physical phenomena, this comparison tool is highly robust against noise; various mesh based signatures, which now can be applied to point clouds within our framework, have even been shown to resist mis-categorizing incomplete or damaged models. I also illustrated the effectiveness of the proposed framework by analyzing a database of point cloud models output by the Microsoft Kinect, which contain defects and noise that makes them resistant to meshing.

7.1.2 Spectral Signature Clustering Tools for PC Models

The framework presented in Chapter 3, which enables the capability to perform similarity and segmentation directly on point clouds, can be adapted to most application domains that require point cloud processing, including robotics, design and manufacturing, and opens the doors to a number of engineering applications. For example, this framework can easily be applied to compare point cloud of physical artifacts to CAD model databases, irrespective of the native CAD format, obviating the need to perform solid model reconstruction or to deal with difficult CAD interoperability issues and proprietary formats. [1]

The concept of *clustering the clusters* introduced here, which provides the ability to determine the number of similar features in a given point cloud model, is critical in manufacturing planning as well as geometric reasoning. This re-clustering by signature values allows me to extract further salience information at low computational cost, and is a feature unique to segmentations that retain similarity information from the signature(s) on which they are based.

7.1.3 Spectral Signature Eigenpair Cutoff Improvements

In order to ensure effective practical application of spectral shape signatures to databases with models of various intrinsic sizes, samplings, and surface complexities as are often encountered in real-world and industrial applications, I investigated the effect of eigensystem truncation point on spectral signature performance. I de-

[1] Models for which a Hermitian Laplace-Beltrami estimate is available can have spectral signatures computed on them in the same way we here describe for point cloud models; for those that do not or for which implementing code for such procedure would require excessive effort or time for some group, a Monte Carlo sampling of the surface allows the point cloud methods we present to be used to make comparisons.

veloped a tunable cutoff algorithm to allow more consistent and rapid computation of comparable feature vectors for all spectral signature methods on any model type to address this challenge.

7.1.4 Demonstrating the Utility of PC Model Spectral Shape Analysis

Chapter 6 includes results of my methods demonstrated across a broad selection of point cloud models. In addition, to facilitate real-world application of this new analysis procedure, I examined the manner in which results depend on the parameters of my methods. I presented whole-database analysis for the CERTH/ITI database of Kinect scans, a database resistant to meshing, and compared the results of mesh and point cloud analysis solutions on the Girl Dancing Samba database. I compared the quality of results of the fixed-number-of-eigenpairs spectral signature methods on meshes and point clouds with the tunable adaptive eigenpairs method on both model types, as well.

7.2 Open Issues and Future Directions

Theoretically speaking, this work provides a general purpose point cloud analyzer, although there are several challenges that remain to be addressed by the field, including: automatically selecting an appropriate number of segments for a given model; a better understanding of how to automatically set the available parameters which influence the similarity and segmentation performance; and finally GPU acceleration of the eigensystem and clustering computations.

Additionally, the problems of defining "good" segmentations in general and specifically for general models of engineering relevance, especially without defining *a priori* primitives to limit your design space, remain open and challenging. I discussed the sometimes-contradictory philosophies of sub-shape relevance for natural and designed shapes. This difference of philosophy still lacks a unifying theory to understand either an underlying central philosophy or else how to easily understand which approach to apply in any given situation.

Chapter 8

Appendix

8.1 Practical SPCL Construction

In order to make practical construction of the SPCL clear, Figure 3.2.1 shows visualizations of the steps that take place to build each row of the SPCL's W and A matrices. We have selected the 37254th point in the model; the steps that follow are thus those undertaken to build the 37254th row of the W matrix and the 37254th diagonal element of the A matrix.

Figure 3.2.1 a) shows the robot point cloud model we choose for our demonstration with the points of the model colored by an HKS vector computed for it. Figure 3.2.1 b) shows the local neighborhood given by finding point membership in the ball of radius six times the average inter-point distance of the point cloud (approximate reach). Figure 3.2.1 c) shows the approximate local tangent space for the neighborhood. This is computed by principle component analysis (PCA) of the points in the neighborhood. Figure 3.2.1 d) is simply an additional view of the tangent space

on the model, for clarity. In Figure 3.2.1 e), the view is moved inside of the robot model, showing the tangent space and model points from the negative Z axis direction of the tangent plane. In Figure 3.2.1 f), the points belonging to the neighborhood have been projected into the tangent plane, rendering the neighborhood two dimensional. Distances in this reduced dimension projected space are used in the SPCL computation.

Figure 3.2.1 g) shows the Delaunay triangulation of the projected neighborhood. Note that this local 2D Delaunay is very fast as it only acts on some 100-200 points and in two-dimensions. This triangulation is used to approximate surface connectedness for the purpose of the Laplacian operator (which is, in essence, a kind of neighborhood graph). The triangulation is then used, as is demonstrated in Figure 3.2.1 h), to approximate the areas of the surface of the model represented by the point cloud which is "owned" or represented by each point in the neighborhood. Each element of the row of W being constructed here includes both the areas "belonging to" the query point and the neighbor point in question. The element is in the row of the query point (here, 37254) and the column of the neighbor point. All other elements of the W matrix are 0, excepting the diagonal elements which are the row sums, as discussed above.

Accordingly, the SPCL provides a record of what points are near to one another, to what degree they are near (distance), and the relative isolation of each point (encoded by the area term). To use the analogy of a diffusion process, nearness and size define how much influence any given point has over its local area and, inversely, how much area on the surface geometry effects diffusive behavior at that point.

Bibliography

[1] *The CERTH/ITI dataset of Kinect-based 3D scans*, http://vcl.iti.gr/3d-scans/.

[2] *ABB's industrial robots materials web page*, http://new.abb.com/products/robotics/industrial-robots, 2014.

[3] *3d Reshaper Meteor*, 2016.

[4] Alexander Agathos, Ioannis Pratikakis, Stavros Perantonis, Nikolaos Sapidis, and Philip Azariadis, *3d mesh segmentation methodologies for cad applications*, Computer-Aided Design and Applications **4** (2007), no. 6, 827–841.

[5] Pierre Alliez, Giuliana Ucelli, Craig Gotsman, and Marco Attene, *Recent advances in remeshing of surfaces*, Shape analysis and structuring, Springer, 2008, pp. 53–82.

[6] Nina Amenta and Marshall Bern, *Surface reconstruction by voronoi filtering*, Discrete & Computational Geometry **22** (1999), no. 4, 481–504.

[7] Tom M Apostol, *Calculus, volume 2: Multi-variable calculus and linear algebra with applications*, vol. 10, AMC, 1969.

[8] Marco Attene, Marcel Campen, and Leif Kobbelt, *Polygon mesh repairing: An application perspective*, ACM Computing Surveys (CSUR) **45** (2013), no. 2, 15.

[9] Mathieu Aubry, Ulrich Schlickewei, and Daniel Cremers, *The wave kernel signature: A quantum mechanical approach to shape analysis*, Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on, IEEE, 2011, pp. 1626–1633.

[10] Phillip N Azariadis and Nickolas S Sapidis, *Drawing curves onto a cloud of points for point-based modelling*, Computer-Aided Design **37** (2005), no. 1, 109–122.

[11] M. Belkin, J. Sun, and Y. Wang, *Discrete Laplace operator on meshed surfaces*, Proceedings of the twenty-fourth annual symposium on Computational geometry, ACM, 2008, pp. 278–287.

[12] ———, *Constructing Laplace operator from point clouds in* $\mathbb{R}^d$, Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms, Society for Industrial and Applied Mathematics, 2009, pp. 1031–1040.

[13] Halim Benhabiles, J-P Vandeborre, Guillaume Lavoué, and Mohamed Daoudi, *A framework for the objective evaluation of segmentation algorithms using a ground-truth of human segmented 3d-models*, Shape Modeling and Applications, 2009. SMI 2009. IEEE International Conference on, IEEE, 2009, pp. 36–43.

[14] William Benjamin, Andrew Wood Polk, SVN Vishwanathan, and Karthik Ramani, *Heat walk: Robust salient segmentation of non-rigid shapes*, Computer Graphics Forum, vol. 30, Wiley Online Library, 2011, pp. 2097–2106.

[15] Matthew Berger, Joshua A Levine, Luis Gustavo Nonato, Gabriel Taubin, and Claudio T Silva, *A benchmark for surface reconstruction*, ACM Transactions on Graphics (TOG) **32** (2013), no. 2, 20.

[16] Matthew Berger, Andrea Tagliasacchi, Lee Seversky, Pierre Alliez, Joshua Levine, Andrei Sharf, and Claudio Silva, *State of the art in surface reconstruction from point clouds*, EUROGRAPHICS star reports, vol. 1, 2014, pp. 161–185.

[17] A.M. Bronstein, M.M. Bronstein, L.J. Guibas, and M. Ovsjanikov, *Shape google: Geometric words and expressions for invariant shape retrieval*, ACM Transactions on Graphics (TOG) **30** (2011), no. 1, 1.

[18] M.M. Bronstein and I. Kokkinos, *Scale-invariant heat kernel signatures for non-rigid shape recognition*, Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on, IEEE, 2010, pp. 1704–1711.

[19] Isaac Chavel, *Eigenvalues in riemannian geometry*, vol. 115, Academic press, 1984.

[20] Frédéric Chazal, Leonidas J Guibas, Steve Y Oudot, and Primoz Skraba, *Persistence-based clustering in riemannian manifolds*, Journal of the ACM (JACM) **60** (2013), no. 6, 41.

[21] Xiaobai Chen, Aleksey Golovinskiy, and Thomas Funkhouser, *A benchmark for 3d mesh segmentation*, ACM Transactions on Graphics (TOG), vol. 28, ACM, 2009, p. 73.

[22] Visual Computing Lab ISTI CNR, *Meshlab*, http://meshlab.sourceforge.net/.

[23] RJ Cripps, *Algorithms to support point-based cadcam*, International Journal of Machine Tools and Manufacture **43** (2003), no. 4, 425–432.

[24] T.K. Dey, K. Li, C. Luo, P. Ranjan, I. Safa, and Y. Wang, *Persistent heat signature for pose-oblivious matching of incomplete models*, Computer Graphics Forum, vol. 29, Wiley Online Library, 2010, pp. 1545–1554.

[25] Julie Digne and Jean-Michel Morel, *Numerical analysis of differential operators on raw point clouds*, Numerische Mathematik **127** (2014), no. 2, 255–289.

[26] Alexandros Doumanoglou, Stylianos Asteriadis, Dimitrios S Alexiadis, Dimitrios Zarpalas, and Petros Daras, *A dataset of Kinect-based 3D scans*, IVMSP Workshop, 2013 IEEE 11th, IEEE, 2013, pp. 1–4.

[27] Laurent El Ghaoui, *Optimization models and applications*, `http://livebooklabs.com/keeppies/c5a5868ce26b8125`, 2015, Livebook visited July 2015.

[28] Lawrence C. Evans, *Partial differential equations*, The American Mathematical Society, 1998.

[29] Herbert Federer, *Geometric measure theory*, Springer, 1969.

[30] Panagiotis Foteinos and Nikos Chrisochoides, *4D space-time Delaunay meshing for medical images*, Proceedings of the 22nd International Meshing Roundtable, 2014.

[31] JungHyun Han and Aristides AG Requicha, *Feature recognition from cad models*, IEEE Computer Graphics and Applications (1998), no. 2, 80–94.

[32] Jean-Claude Hausmann et al., *On the vietoris-rips complexes and a cohomology theory for metric spaces*, Ann. Math. Studies **138** (1995), 175–188.

[33] Klaus Hildebrandt, Konrad Polthier, and Max Wardetzky, *On the convergence of metric and geometric properties of polyhedral surfaces*, Geometriae Dedicata **123** (2006), no. 1, 89–112.

[34] E.P. Hsu, *Stochastic analysis on manifolds*, vol. 38, Amer Mathematical Society, 2002.

[35] Natraj Iyer, Subramaniam Jayanti, Kuiyang Lou, Yagnanarayanan Kalyanaraman, and Karthik Ramani, *Three-dimensional shape searching: state-of-the-art review and future trends*, Computer-Aided Design **37** (2005), no. 5, 509–530.

[36] Andrew E Johnson and Martial Hebert, *Using spin images for efficient object recognition in cluttered 3d scenes*, Pattern Analysis and Machine Intelligence, IEEE Transactions on **21** (1999), no. 5, 433–449.

[37] Yunjae Jung, Haesun Park, Ding-Zhu Du, and Barry L Drake, *A decision criterion for the optimal number of clusters in hierarchical clustering*, Journal of Global Optimization **25** (2003), no. 1, 91–111.

[38] Andrew V Knyazev, *Signed laplacian for spectral clustering revisited*, arXiv preprint arXiv:1701.01394 (2017).

[39] Guillaume Lavoué, Florent Dupont, and Atilla Baskurt, *A new cad mesh segmentation method, based on curvature tensor analysis*, Computer-Aided Design **37** (2005), no. 10, 975–987.

[40] Tao Liao, Xinge Li, Guoliang Xu, and Yongjie Jessica Zhang, *Secondary Laplace operator and generalized Giaquinta–Hildebrandt operator with applications on surface segmentation and smoothing*, Computer-Aided Design **70** (2016), 56–66.

[41] Yang Liu, Balakrishnan Prabhakaran, and Xiaohu Guo, *Point-based manifold harmonics*, Visualization and Computer Graphics, IEEE Transactions on **18** (2012), no. 10, 1693–1703.

[42] Simone Marini, Giuseppe Patané, Michela Spagnuolo, and Bianca Falcidieno, *Spectral feature selection for shape characterization and classification*, The Visual Computer **27** (2011), no. 11, 1005–1019.

[43] Saigopal Nelaturi and Vadim Shapiro, *Solving inverse configuration space problems by adaptive sampling*, Computer-Aided Design **45** (2013), no. 2, 373–382.

[44] A Cengiz Öztireli, Gael Guennebaud, and Markus Gross, *Feature preserving point set surfaces based on non-linear kernel regression*, Computer Graphics Forum, vol. 28, Wiley Online Library, 2009, pp. 493–501.

[45] Mark Pauly, Leif P Kobbelt, and Markus Gross, *Point-based multiscale surface representation*, ACM Transactions on Graphics (TOG) **25** (2006), no. 2, 177–193.

[46] Hanspeter Pfister and Markus Gross, *Point-based computer graphics*, IEEE Computer Graphics and Applications (2004), no. 4, 22–23.

[47] Jonathan Pokrass, Alexander M Bronstein, and Michael M Bronstein, *Partial shape matching without point-wise correspondence*, Numerical Mathematics: Theory, Methods and Applications **6** (2013), no. 01, 223–244.

[48] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, *PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space*, ArXiv e-prints (2017).

[49] Charles Ruizhongtai Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas, *Pointnet: Deep learning on point sets for 3d classification and segmentation*, CoRR **abs/1612.00593** (2016).

[50] Rajesh Ramamurthy, Kevin Harding, Xiaoming Du, Vincent Lucas, Yi Liao, Ratnadeep Paul, and Tao Jia, *Geometric and topological feature extraction of linear segments from 2D cross-section data of 3D point clouds*, SPIE Sensing Technology+ Applications, International Society for Optics and Photonics, 2015, pp. 948905–948905.

[51] Martin Reuter, Franz-Erich Wolter, and Niklas Peinecke, *Laplace–beltrami spectra as ëshape-dnaíof surfaces and solids*, Computer-Aided Design **38** (2006), no. 4, 342–366.

[52] Raif M Rustamov, *Laplace-beltrami eigenfunctions for deformation invariant shape representation*, Proceedings of the fifth Eurographics symposium on Geometry processing, Eurographics Association, 2007, global point signature, pp. 225–233.

[53] Jie Shen, David Yoon, David Shehu, and Shang-Yeu Chang, *Spectral moving removal of non-isolated surface outlier clusters*, Computer-Aided Design **41** (2009), no. 4, 256 – 267, Point-based Computational Techniques.

[54] P. Skraba, M. Ovsjanikov, F. Chazal, and L. Guibas, *Persistence-based segmentation of deformable shapes*, Computer Vision and Pattern Recognition Workshops (CVPRW), 2010 IEEE Computer Society Conference on, IEEE, 2010, pp. 45–52.

[55] Anuj Srivastava, Shantanu H Joshi, Washington Mio, and Xiuwen Liu, *Statistical shape analysis: Clustering, learning, and testing*, IEEE Transactions on pattern analysis and machine intelligence **27** (2005), no. 4, 590–602.

[56] J. Sun, M. Ovsjanikov, and L. Guibas, *A concise and provably informative multi-scale signature based on heat diffusion*, Computer Graphics Forum, vol. 28, 2009, pp. 1383–1392.

[57] Andrea Tagliasacchi, Hao Zhang, and Daniel Cohen-Or, *Curve skeleton extraction from incomplete point cloud*, ACM Transactions on Graphics (Proc. SIGGRAPH) (2009).

[58] Robert Tibshirani, Guenther Walther, and Trevor Hastie, *Estimating the number of clusters in a data set via the gap statistic*, Journal of the Royal Statistical Society: Series B (Statistical Methodology) **63** (2001), no. 2, 411–423.

[59] Oliver Van Kaick, Noa Fish, Yanir Kleiman, Shmuel Asafi, and Daniel Cohen-Or, *Shape segmentation by approximate convexity analysis*, ACM Transactions on Graphics (TOG) **34** (2014), no. 1, 4.

[60] Daniel Vlasic, Ilya Baran, Wojciech Matusik, and Jovan Popović, *Articulated mesh animation from multi-view silhouettes*, ACM Transactions on Graphics (TOG), vol. 27, ACM, 2008, p. 97.

[61] Ulrike Von Luxburg, *A tutorial on spectral clustering*, Statistics and computing **17** (2007), no. 4, 395–416.

[62] Yutao Wang and Hsi-Yung Feng, *Outlier detection for scanned point clouds using majority voting*, Computer-Aided Design **62** (2015), 31 – 43.

[63] Max Wardetzky, Saurabh Mathur, Felix Kälberer, and Eitan Grinspun, *Discrete laplace operators: no free lunch*, Symposium on Geometry processing, 2007, pp. 33–37.

[64] R. M. Williams and H. T. Ilieş, *Practical shape analysis and segmentation methods for point cloud models*, in Review.

[65] Reed M Williams and Horea T Ilieş, *Towards multi-scale heat kernel signatures for point cloud models of engineering artifacts*, Workshop on Algebraic Topology and Machine Learning at Neural Information Processing Symposium 2012, 2012.

[66] Reed M. Williams and Horea T. Ilieş, *Adaptive eigensystem truncation for spectral shape signatures*, Computer-Aided Design and Applications **14** (2017), no. 6, 770–777.

[67] Li Yi, Hao Su, Xingwen Guo, and Leonidas J. Guibas, *Syncspeccnn: Synchronized spectral CNN for 3d shape segmentation*, CoRR **abs/1612.00606** (2016).

[68] Hayato Yoshioka, Jiang Zhu, Tomohisa Tanaka, et al., *Automatic segmentation and feature identification of laser scanning point cloud data for reverse engineering*, Flexible Automation (ISFA), International Symposium on, IEEE, 2016, pp. 278–285.