SHAPE ANALYSIS OF DEFECT-LADEN DATA

by

Matthew Berger

A dissertation submitted to the faculty of
The University of Utah
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

School of Computing

The University of Utah

May 2013

The University of Utah Graduate School

STATEMENT OF DISSERTATION APPROVAL

The dissertation of **Matthew Berger**

has been approved by the following supervisory committee members:

Claudio T. Silva	, Chair	**01/24/13** Date Approved
Adam Bargteil	, Member	**03/07/13** Date Approved
Mike Kirby	, Member	**03/06/13** Date Approved
Luis Gustavo Nonato	, Member	**02/20/13** Date Approved
Gabriel Taubin	, Member	**02/11/13** Date Approved

and by **Al Davis** , Chair of

the Department of **School of Computing**

and by Donna M. White, Interim Dean of The Graduate School.

ABSTRACT

Shape analysis is a well-established tool for processing surfaces. It is often a first step in performing tasks such as segmentation, symmetry detection, and finding correspondences between shapes. Shape analysis is traditionally employed on well-sampled surfaces where the geometry and topology is precisely known. When the form of the surface is that of a point cloud containing nonuniform sampling, noise, and incomplete measurements, traditional shape analysis methods perform poorly. Although one may first perform reconstruction on such a point cloud prior to performing shape analysis, if the geometry and topology is far from the true surface, then this can have an adverse impact on the subsequent analysis. Furthermore, for triangulated surfaces containing noise, thin sheets, and poorly shaped triangles, existing shape analysis methods can be highly unstable. This thesis explores methods of shape analysis applied *directly* to such defect-laden shapes.

We first study the problem of surface reconstruction, in order to obtain a better understanding of the types of point clouds for which reconstruction methods contain difficulties. To this end, we have devised a benchmark for surface reconstruction, establishing a standard for measuring error in reconstruction. We then develop a new method for consistently orienting normals of such challenging point clouds by using a collection of harmonic functions, intrinsically defined on the point cloud. Next, we develop a new shape analysis tool which is tolerant to imperfections, by constructing distances directly on the point cloud defined as the likelihood of two points belonging to a mutually common medial ball, and apply this for segmentation and reconstruction. We extend this distance measure to define a diffusion process on the point cloud, tolerant to missing data, which is used for the purposes of matching incomplete shapes undergoing a nonrigid deformation. Lastly, we have developed an intrinsic method for multiresolution remeshing of a poor-quality triangulated surface via spectral bisection.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

First of all, I would like to thank my advisor, Claudio T. Silva, for all of his support and encouragement throughout the course of my PhD studies. Claudio has been a tremendous influence on how I choose the types of problems to solve, how to approach problems, and what makes for good research. I thank him for this, and I especially thank him for always having the confidence in me to do good research, particularly during times when I did not.

I would like to thank all of my paper co-authors for their great help and immense improvement in the development of the papers: Joshua Levine, Luis Gustavo Nonato, Valerio Pascucci, Lee Seversky, and Gabriel Taubin. I owe Gustavo much gratitude, as early on, he was essential in helping me learn how to perform research and write papers, and his high level of enthusiasm always made collaboration a lot of fun. I thank Josh for being critical of my work when I was not, for his encouragement during periods when I was doubtful, and for helping me keep my sanity by maintaining a silly sense of humor throughout everything. I also thank Lee Seversky for all of his support, always listening to my crazy ideas and enduring my complaining.

I would also like to thank everyone at the Scientific Computing and Imaging Institute for many stimulating discussions. In particular, Joel Daniels, Tiago Etiene, and Julien Tierny always provided great feedback on my projects, and my conversations with them greatly influenced the way I approach research. I also thank my other committee members, Adam Bargteil and Mike Kirby, for their valuable feedback.

I thank everyone at the Information Directorate of the Air Force Research Laboratory for their support over the years. In particular, I thank Julie Brichacek, Alex Sisti, and Jerry Dussault for their support and their patience throughout this process. I would especially like to thank Jason Moore for all of his encouragement through the years, for his willingness to always listen to and understand my ideas, and without whom I never would have pursued a PhD, let alone discover the area of geometry processing.

Last, I would like to thank my family, my parents and my brother, for their constant support, without which I could not have completed my dissertation.

This research has been supported in part by grants from the National Science Foundation (grants IIS-0905385, CNS-0855167, IIS-0844546, ATM-0835821, CNS-0751152, OCE-0424602, CNS-0514485, IIS-0513692, CNS-0524096, CCF-0401498, OISE-0405402, CCF-0528201, CNS-0551724), Fapesp-Brazil (#2008/03349-6), the Department of Energy Office of Science, AFOSR, and the Science, Mathematics, and Research for Transformation scholarship program administered by the Naval Postgraduate School.

CHAPTER 1

INTRODUCTION

Shape analysis is a fundamental tool in geometry processing. It is the process of extracting higher-level information from raw geometric representations. This form of information has use in a large number of applications, ranging from segmentation, shape classification, object retrieval, and semantic object modeling.

At its core, shape analysis relies on extracting surface properties, and then mapping these properties to higher-level objectives. Ideally, these properties satisfy certain levels of invariance which represent the semantics of the problem in which we are interested. For instance, if we are interested in categorizing objects which are all invariant to isometries, then some properties we may want to measure include Gaussian curvature, geodesics, and the heat kernel – all measures which are isometry-invariant.

Often, there are strict assumptions on the type of geometric representation for which shape analysis is employed. Namely, for the aforementioned applications, the typical surface representation required is a triangulated surface mesh which contains good-quality triangles. The requirement is necessary so that the properties we wish to measure can be reliably done. For instance, if we are concerned with extracting the conformal structure of a surface, then most existing approaches require a triangulated surface where we precisely know the geometry and topology. Indeed, the range of useful information one can extract from a triangulated surface is vast [Meyer et al. 2002; Coifman and Lafon 2006; Lipman and Funkhouser 2009; Sun et al. 2009; Ben-Chen et al. 2010].

Quite often, however, the data on which we are interested in performing shape analysis fail to meet the requirements of a good-quality triangulated surface. This is a simple consequence of the data of interest: acquired real-world data. Shape analysis is most useful when applied on real shapes, as it is desirable to model and understand the physical world.

The reason for this discrepancy is the acquisition process. For a given shape, most geometry acquisition methods produce a set of range images, where each range image contains the sensed depth, and the range images are organized to produce an unstructured set of points. Three-dimensional acquisition is quite often highly imperfect, where the acquired geometry is defect-laden. These defects can range from nonuniform sampling, noisy measurements, misalignment in the consolidation of scans, and incomplete measurements. This form of the geometry is typically unsuitable on which to perform shape analysis.

The process of *reconstruction* is to take this set of points and produce a continuous surface representation, often a triangulated surface, which best approximates the sensed shape in both geometry and topology, and best handles such defects. One can use the output of reconstruction for the purposes of shape analysis, but if the geometry and topology of the reconstructed surface is far away from the original shape, this can be highly problematic for further processing. Furthermore, assuming that the reconstructed surface is correct, it may still be ill-suited for analysis as the triangles may be of poor quality, where *remeshing* the given surface mesh is necessary.

1.1 Thesis Contributions

The key point of this dissertation is to employ shape analysis *directly* on such defect-laden data, in order to produce good-quality, reconstructed surface meshes from acquired data. We approach shape analysis on defect-laden data from two different perspectives. First, we consider existing analysis methods for the purposes of reconstruction and remeshing. Secondly, we develop new shape analysis methods specifically designed for the purposes of imperfect data, and their applications.

Figure 1.1 demonstrates the contributions of this work. Our goal in this scenario is to take the point cloud on the left, reconstruct the surface as shown in the middle, and remesh the reconstructed surface in a multiresolution and hierarchical fashion, as shown on the right. To reconstruct a topologically and geometrically accurate surface, a key challenge is the presence of missing data on the arm and body, due to occlusion in the acquisition process. To remesh the surface at multiple resolutions, a key challenge is the right hand's close proximity to the shoulder. The novel shape analysis approaches developed in this thesis are at the core of solving these challenging problems.

1.2 Thesis Outline

The first part of the thesis deals with a systematic understanding of surface reconstruction, via the development of a benchmark for surface reconstruction. We develop a method for modeling shapes, sampling shapes, and evaluating reconstruction algorithms, in order to depict the broad range of behavior in surface reconstruction. This provides us with insight into the types of defects which most impact reconstruction, and consequently, informs how

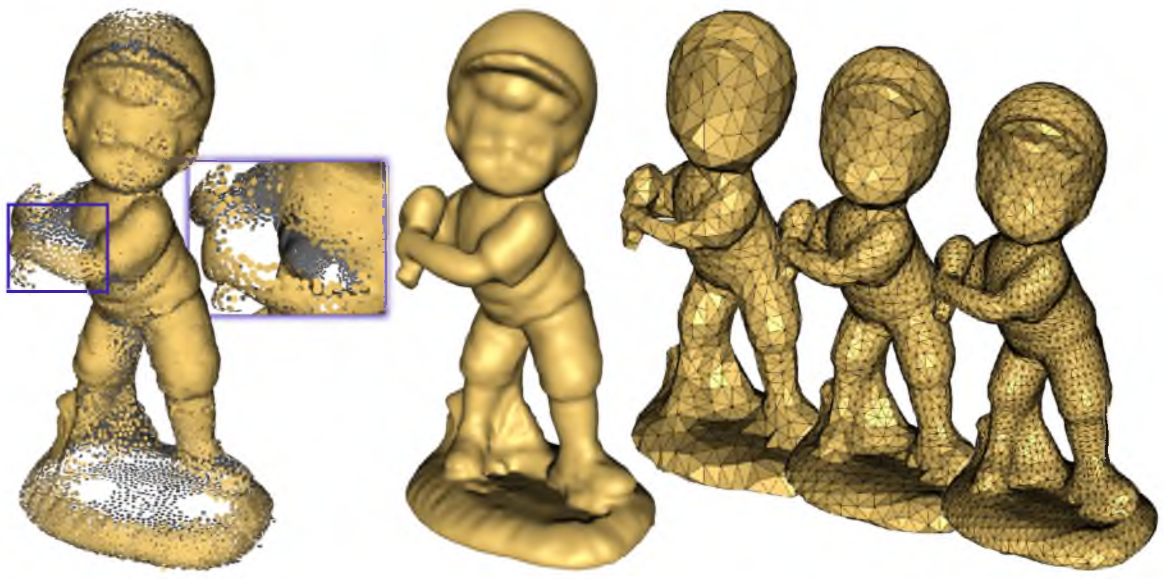


Fig. 1.1. We illustrate the contributions of employing shape analysis methods on raw, defect-laden data. From the point cloud on the left, we develop novel shape analysis methods to correctly infer its geometry and topology, shown in the middle. From this reconstructed surface, we use existing shape analysis methods to produce a hierarchy of quality surface meshes, shown on the right.

we devise shape analysis methods for this type of data. This work has been accepted with minor revisions in Berger et al. [2013].

Next, we consider the problem of normal orientation from the perspective of shape analysis. Normal orientation is the problem of classifying normal directions as being inside or outside over a point cloud, and is a necessary first step for many surface reconstruction algorithms. Our approach is to construct harmonic functions directly on the point cloud, and utilize their global smoothness property to consistently assign normal orientation – we term this process **harmonic point cloud orientation**, published in Seversky et al. [2011]. The challenge lies in remaining robust to imperfections such as nonuniform sampling, noise, and missing data.

From a point cloud containing properly oriented normals, we next consider the problem of constructing meaningful distances in the presence of missing data. Indeed, a measure as common as geodesic distances can prove to be quite misleading in the presence of missing data, and so we seek distances which are tolerant to the undersampling. We use the medial axis as a shape prior, and define distance as the *likelihood* of two points mutually belonging to a medial ball. We term this association measure the **medial kernel**, and consider its applications for segmentation and reconstruction. This work has been published in Berger and Silva [2012a].

We next extend the medial kernel for the purposes of computing correspondences between pairs of scanned and incomplete shapes. Computing correspondences between incomplete shapes undergoing a nonrigid deformation is quite a challenge, where reconstruction may be too impractical due to the substantial missing data. In such cases, we use the medial kernel to construct a diffusion process on the point cloud, and use the resulting heat diffusion for matching the medial regions of point clouds. We term this diffusion process **medial diffusion**. This work has been published in Berger and Silva [2012b].

Last, we consider the problem of taking a poor-quality surface mesh and constructing a set of good-quality, nested surface meshes, via shape analysis. We illustrate how spectral methods may be used to hierarchically decompose a surface mesh so that uniform quality meshes, and feature adaptive meshes via wavelet methods, may both be produced. We term this decomposition the **Fiedler tree**, published in Berger et al. [2010].

CHAPTER 2

BACKGROUND

The focus of this dissertation is on surface reconstruction, shape matching, and surface remeshing of defect-laden data. In particular, our focus is on shape analysis and how it benefits these problems. In this chapter, we discuss the various aspects of shape analysis for when the input surface representation is best suited – either a well-sampled point cloud or a triangulated surface mesh. We then discuss the problems of reconstruction and matching, and in particular, the issues involved when dealing with nonuniform sampling, noise, and missing data. Last, we discuss the problem of remeshing from poor-quality triangles.

2.1 Shape Analysis

Shape analysis deals with the extraction of high-level information from the raw geometry of a surface. It can roughly be broken down into two forms: analysis of extrinsic geometry and analysis of intrinsic geometry.

2.1.1 Intrinsic Geometry

The intrinsic geometry of the surface refers to its geometry which is independent of the ambient space for which the surface may lie. In this context, shape analysis typically refers to the extraction of measures which are unique to the intrinsic geometry. Put another way, these are measures which are isometry-invariant. For instance, if a surface living in $\mathbb{R}^3$ is isometrically deformed, then although its embedding may be different, its intrinsic geometry remains the same. Common intrinsic measures of a surface are its surface area, geodesics, Gaussian curvature, and its Laplace-Beltrami operator.

Harmonic functions of a surface refer to those which lie in the kernel of the Laplace-Beltrami operator. These functions are globally smooth, as the Laplace-Beltrami operator when applied to a function can be seen as a measure of smoothness. This construction of smooth functions has a number of benefits, ranging from mesh parameterization [Desbrun et al. 2002], deformation [Au et al. 2007], and segmentation [Zheng and Tai 2010].

The eigenfunctions of the Laplace-Beltrami operator underscore a large number of applications. Taken in isolation, the eigenfunctions can be looked at as modes of varying frequencies, where the corresponding eigenvalue represents the frequency for an individual eigenfunction [Vallet and Levy 2008]. This is analogous to the complex exponentials associated with the Fourier transform for linear spaces.

The heat kernel of a surface is the fundamental solution to the heat equation, which governs how heat diffuses over the surface. As it may be computed from the eigenfunctions of the Laplace-Beltrami operator, it is also isometry-invariant. The heat kernel has use in a large number of applications, ranging from feature point detection [Sun et al. 2009], segmentation [Gebal et al. 2009], shape retrieval [Dey et al. 2010], and intrinsic symmetry detection [Ovsjanikov et al. 2010].

Diffusion distances [Coifman and Lafon 2006] are closely related to the heat kernel, as these distances are a measure of connectedness of random walks defined by the Laplace-Beltrami operator. As such, these distances have greater tolerance to topological defects than geodesic distances, and have proven useful for pose-invariant segmentation [De Goes et al. 2008].

2.1.2 Extrinsic Geometry

The extrinsic geometry of a surface refers to the ambient space in which the surface lives. In this context, shape analysis traditionally takes the form of analyzing the volume which encloses the surface. Note that the extrinsic geometry can differ from the intrinsic geometry, in that there may exist an isometric deformation of a surface which can drastically change the shape's underlying volume. Nonetheless, analyzing the extrinsic geometry of a surface can nicely complement the intrinsic analysis of a surface.

The medial axis is a very common method for extracting extrinsic measures. It is the set of points in the volume in which the number of closest points to the surface is larger than one. In particular, the medial axis transform is the subset of these points interior to the surface which, along with taking the distances as ball radii, can be used to exactly represent the volume of the surface by taking the union of balls.

For triangulated surfaces, one may extract the medial axis by computing the distance field of the surface, and taking all of the points where its gradient is discontinuous [Sud et al. 2005]. Alternatively, for well-sampled point clouds, Voronoi-based methods such as Amenta et al. [2001] and Dey and Zhao [2004] may be used, where Voronoi vertices far from the point cloud, known as “poles”, are identified as points on the medial axis.

Once extracted, the medial axis has a large number of applications. One can simplify the medial axis transform itself, wherein its resulting surface is also simplified [Tam and Heidrich 2003; Miklos et al. 2010]. The simplified representation of the medial axis, and its invariance to pose, may be used for shape retrieval [Zhang et al. 2005]. The medial axis is also suitable for volumetric segmentation [Chang and Kimia 2008], by separating the individual medial sheets and identifying points on the surface to which the sheets belong.

Although quite descriptive, the medial axis can be somewhat difficult to handle, as it is composed of a set of curves and surface sheets nontrivially intersecting. Hence, another line of work focuses on extracting skeleton curves, a simpler representation than the medial axis. Various approaches exist for skeleton extraction, ranging from contouring Reeb graphs of scalar fields [Hilaga et al. 2001], growing deformable models in the volume [Sharf et al. 2006], thinning the medial axis [Dey and Sun 2006a], and surface contraction via mean curvature smoothing [Au et al. 2008].

The advantage of the medial axis and curve skeleton approaches is their compact representation of the volume. Another set of approaches capture the volume by computing measures directly on the surface, rather than operating on an explicity geometric representation. The shape diameter function [Shapira et al. 2008] focuses on measuring the overall thickness of the volume at a given point by sampling a cone of rays in the direction of a point's normal. The method of Liu et al. [2009] extends this by constructing a volume-dependent metric on the faces of a mesh. These approaches support several applications, ranging from segmentation, salient feature point detection, as well as skeletonization itself.

2.1.3 Discussion

A drawback to the above approaches is the requirement of either a triangulated surface, or a surface which has been well-sampled. Hence, one faces a challenge in applying such intrinsic and extrinsic shape analyses to a surface which has been acquired, where the requirements of a well-sampled surface are often violated. A common approach is to first perform surface reconstruction prior to running these methods on acquired data, which we next discuss.

2.2 Surface Reconstruction

Surface reconstruction is the process of taking a set of points and recovering the original surface from which those points were measured. In particular, the representation of the recovered surface is typically one in which the geometry and topology is precisely known and faithful to the measured surface. Triangulated surfaces, as used in the majority of the above approaches, is quite commonly the target representation.

The challenges involved in producing a triangulated surface from a set of measured points stem from the acquisition process. There exists a large number of acquisition modalities, ranging from passive methods such as multiview stereo, to active methods such as laser-based optical triangulation and structured lighting. Most of these methods produce range scans which contain measurement noise, nonuniform sampling, as well as missing measurements, which can be due to the surface reflectance, occlusion, and the grazing angle at which the surface is measured. Furthermore, misalignment errors can arise from the process of registering individual range scans into a single surface. The ability to handle these imperfections is what distinguishes the various surface reconstruction algorithms.

2.2.1 Interpolating Methods

One class of reconstruction methods focuses on producing a triangulated surface which interpolates a subset of the data, that is, a subset of the input point cloud is preserved in the output. These methods are typically based on extracting a subset of the Delaunay triangulation of the point cloud, such that for every triangle retained, its dual Voronoi edge meets at the medial axis [Amenta and Bern 1999]. A variety of methods have been proposed in this vein for noise-free methods, such as the power crust algorithm [Amenta and Bern 1999] and cocone [Amenta et al. 2002]. These methods have been extended to support noisy data by using the size of medial balls as a measure of stability [Dey and Sun 2006b].

The above methods are provably good, in the sense that so long as certain sampling conditions are satisfied with respect to the medial axis, these algorithms will correctly reconstruct the surface. However, in practice, it is extremely difficult to verify these sampling conditions, and as the level of data imperfection increases, these methods tend to not degrade gracefully.

2.2.2 Approximating Methods

Another line of reconstruction algorithms relax the interpolation assumption, such that the reconstruction need only approximate the input point cloud. This provides for robust algorithms in the presence of noise, nonuniform sampling, and missing data, albeit at the expense of guarantees. These algorithms typically require a set of normals accompanying the points, such that the normals are consistently oriented according to the inside and outside of the surface.

One of the first approaches to orienting normals is that of Hoppe et al. [1992]. Unoriented normal directions are first found via PCA, while orientation is found by first fixing a single normal's orientation, and propagating it to normals of close position and direction via a minimal spanning tree. This form of orientation propagation has been extended to handle sharp features [Xie et al. 2003], as well as thin surface sheets and missing data [Huang et al. 2009]. The above methods of orientation are local, in that the propagation decision is determined only via local information. Hence, a single incorrect orientation can erroneously propagate over large regions of the point cloud.

Other methods approach normal orientation from a global perspective. The method of Liu and Wang [2010] performs a coarse, bounding reconstruction [Ohtake et al. 2005b] of the point cloud to drive a more refined normal orientation estimator. The works of Chen et al. [2010] and Chen et al. [2011] utilize point set visibility [Katz et al. 2007] in order to determine the orientation of a normal by considering whether or not a point is visible on the bounding volume of the surface. These methods make sampling assumptions on the point cloud, wherein the presence of nonuniform sampling and missing data, visibility [Katz et al. 2007] and coarse reconstruction [Ohtake et al. 2005b] can perform poorly, and consequently so does the orientation approach.

Once equipped with normals, most approximating methods aim to construct an implicit function over the volume whose zero level-set is the surface. A common approach is to employ regularized shape fitting to the point cloud. This can be performed globally in the case of RBFs [Carr et al. 2001] and locally for Multiresolution Partition of Unity methods [Ohtake et al. 2003; Nagai et al. 2009], Moving Least Squares [Alexa et al. 2003; Guennebaud and Gross 2007], and compact RBFs [Ohtake et al. 2005a]. For these methods, there is often a tradeoff between smoothness in the output and faithfulness to the input, where it can be challenging to strike the right balance.

Other approximation methods frame the problem of reconstruction as finding an *indicator function* over the volume, where points inside of the volume of the surface are assigned a value of one and all other points zero. These approaches transform this volumetric problem to one on the surface via Stokes theorem, and consequently solving for the indicator function amounts to solving the Poisson equation. The method of Kazhdan [2005] inverts the gradient operator via Fourier methods, while Kazhdan et al. [2006] directly solves the Poisson equation in the spatial domain through a hierarchical solver. This was extended in Alliez et al. [2007] to provide for a better domain decomposition via Delaunay refinement and a more robust estimation of normals. The method of Manson et al. [2008] adapted Kazhdan

[2005] in using Wavelets as the basis of choice for which to invert the gradient operator.

Although approximating methods tend to be more robust to data imperfections compared to interpolating methods, there exists situations where regularizing the problem can produce poor surface reconstructions. For instance, if one is interested in capturing the topology of the original surface, approximating methods can erroneously fill in or attach tunnels and produce extraneous components under smoothness priors.

As an alternative to smoothness priors, similar to the goals of this dissertation, other methods have employed shape analysis to properly steer surface reconstruction. The key challenge is operating on raw point clouds containing various imperfections. The method of Tagliasacchi et al. [2009] extracts a skeleton from an incomplete point cloud by employing a cylindrical prior on the output shape, associating for each point a rotationally-invariant symmetry axis in order to find its accompanying skeletal point. Resampling of the surface may be performed by sampling these cylindrical regions. Regions of the point cloud which violate the cylindrical prior can result in a poor embedding; hence, substantial postprocessing is necessary to obtain a clean skeleton. This work was extended in Li et al. [2010] by strictly enforcing cylindrical shapes through a snake deformation model, hence making their method highly robust to larger gaps of data. The work of Cao et al. [2010] supports a broader class of shapes by extending the contraction approach of Au et al. [2008] to the case of point clouds. In the presence of missing data, however, the constructed Laplacian may respect the boundary components, potentially resulting in a poor contraction.

2.3 Shape Matching

Shape matching refers to finding correspondences between a pair of shapes. It can take on many forms, depending on assumptions in the types of shapes being matched, the underlying deformation space of the shapes, and how the shapes are sampled. Here, we discuss prior work most relevant to the goals of the thesis: a single shape undergoing a nonrigid deformation, containing missing data.

In the area of nonrigid registration, computing correspondence is a key component in the process of registering scans of a deforming shape. In these scenarios, missing data frequently arise, and in order for the particular deformation model to adequately converge, it is essential to construct meaningful correspondences in the presence of imperfections.

For time-varying capture, a number of approaches exist for computing correspondences, where they tend to rely on the coherence in motion between scan frames. Most of these approaches make assumptions either on templates, the acquisition process, or initialization.

The approaches of Süßmuth et al. [2008] and Li et al. [2009] rely on apriori defined templates to construct correspondences, since one can reliably construct geodesics on the template, which should remain invariant across the scanning sequence. The methods of Popa et al. [2010] and Li et al. [2012] rely on stereo matches to initialize the dense matching of correspondences. Other approaches [Wand et al. 2009; Sharf et al. 2008] rely on point-to-plane distance correspondences, which implicitly assumes that the motion between frames is small.

For a general collection of shapes, where frame-to-frame motion coherence is lost, correspondence becomes a much harder problem. The approach of Chang and Zwicker [2008] relies on local features to extract a set of candidate correspondences. In the presence of missing data, however, it can be challenging to reliably construct local features. The methods of Li et al. [2008] and Chang and Zwicker [2009] instead rely on an initial overlap between point clouds, and consequently point-to-plane distance correspondences. A more sophisticated approach is the method of Huang et al. [2008], where local features and geodesics are used to drive spectral matching. They use a k-nearest neighbor graph to construct geodesics; hence, it is only reliable when the lack of data is consistent across scans.

There are a large number of techniques for finding correspondences between well-sampled shapes; see van Kaick et al. [2011] for an overview. The approach of Bronstein et al. [2006] applies generalized multidimensional scaling to find correspondences which best preserve geodesic distances. A deformation model is used in Zhang et al. [2008] to measure the quality of correspondences, where quality is defined in terms of deformation distortion. Möbius voting [Lipman and Funkhouser 2009] seeks to find correspondences which best preserve the conformal structure, thus allowing for a large space of deformations.

It is nontrivial to generalize the above approaches to point clouds, as they typically require a continuous surface representation. A notable exception is the method of Ovsjanikov et al. [2010], where they show how the heat kernel can be used to match nonrigid shapes, as the heat kernel is invariant to isometries. They demonstrate how their approach can be used for partial matching, as well as its insensitivity to small topological changes. Although used for meshes, the approach of Ovsjanikov et al. [2010] only requires a discretization of the Laplace-Beltrami operator, and numerous such discretizations exist for point clouds; see Belkin et al. [2009] and Luo et al. [2009].

Little work has addressed the correspondence problem in the presence of large missing data. The work of Tevs et al. [2009] and Tevs et al. [2012] uses geodesic distances and a RANSAC-like approach to find landmark correspondences, which subsequently drives a dense correspondence matcher. They employ a k-nearest neighbor graph construction to approximate geodesics; hence, they still require some coherence in the missing data for an accurate correspondence. Perhaps most similar to our work is Zheng et al. [2010], where they employ the method of Tagliasacchi et al. [2009] to build a set of skeletons, and perform correspondence on the skeleton graphs. The challenge in this work is that the skeletons might be of widely varying topology, depending on the effectiveness of Tagliasacchi et al. [2009]; hence, partial matching must be employed to bring the skeletons into correspondence.

2.4 Surface Remeshing

Surface remeshing is the problem of converting a poor-quality surface mesh into one of better quality, while still closely approximating the original surface. Quality can refer to a number of different measures, ranging from minimum angle in a triangle, the ratio between the inradius and circumradius (commonly known as the radius ratio), and the ratio between the circumradius and the shortest edge length. Our approach is focused on remeshing surfaces which contain poor-quality triangles, and potentially high levels of noise, in a multiresolution manner; hence, we limit the discussion of remeshing algorithms to such relevant works.

A common approach for generating multiresolution methods is via mesh simplification. QSlim [Garland and Heckbert 1997] is a well-known method for simplifying a mesh via edge collapses, from which a hierarchy of meshes may be generated, using a progressive mesh [Hoppe 1996] approach. Other methods operate in the ambient space of the mesh through spatial subdivision, performing simplification by collapsing vertices which belong to common grid cells [Rossignac and Borrel 1993; Schaefer and Warren 2003], or through a hybrid approach of 3D-2D spatial decomposition [Boubekeur et al. 2006]. However, such methods tend to produce poor-quality triangles as part of the simplification.

Surface parameterization is a common approach to constructing quality meshes. There exists global parameterization methods [Alliez et al. 2003] and methods which construct multiple local overlapping parameterizations [Surazhsky and Gotsman 2003], where once a parameterization is known, remeshing the surface amounts to the simpler problem of resampling a 2D domain. Although one may obtain quality multiresolution from a global parameterization via subdivision schemes, it is highly nontrivial and expensive to construct a global parameterization.

Other methods use the concept of the centroidal Voronoi diagram to remesh surfaces directly, either approximately [Valette et al. 2008] or exactly [Yan et al. 2009]. Such methods require quite expensive optimization procedures, needing many iterations to adequately converge to a quality triangulation. Furthermore, to construct a multiresolution mesh, it is necessary to run these algorithms from scratch each time for every resolution.

Delaunay refinement is a popular approach for producing provably good-quality triangle meshes. For surface meshes, the approach of Boissonnat and Oudot [2005] maintains the restricted Delaunay triangulation of a surface, where appropriate sizing functions can be used to produce quality meshes. This was extended in Cheng et al. [2007] to the case of sharp features and nonmanifold configurations. Although these approaches are greedy, in that the mesh is constructed by incremental sampling, it still remains nontrivial to extend these methods to produce multiresolution meshes. Another disadvantage of these approaches is the requirement of operating in the ambient space of the surface, where nearby surface sheets may impose strict sampling requirements.

CHAPTER 3

A BENCHMARK FOR SURFACE RECONSTRUCTION

In this chapter, we consider the establishment of a benchmark for surface reconstruction. Surface reconstruction is motivated by a large number of applications. For instance, it is a crucial first step in the recovery of nonrigid motion of time-varying geometry [Sharf et al. 2008; Li et al. 2009], and used as "ground-truth" data for multiview stereo reconstruction evaluation [Seitz et al. 2006].

There has been a vast amount of work dedicated to surface reconstruction, but to date, there lacks a sufficient means of evaluating and comparing these methods. Part of this problem stems from the data on which most approaches operate: scanned point clouds of the real world. Hence, there is a noticeable absence of ground truth in these scenarios, and it is unclear how to perform evaluation with respect to raw range data.

There are some existing approaches which produce synthetically generated point clouds from triangle meshes; hence, in these scenarios, it becomes possible to perform evaluation. Existing approaches such as Kazhdan [2005] and Manson et al. [2008] randomly sample triangle meshes to produce point clouds, while the methods of Hoppe et al. [1992] and ter Haar et al. [2005] obtain synthetic scans of a triangle mesh from ray tracing or z-buffering the mesh. While these methods may produce realistic data under the assumption of completely clean data, these approaches are insufficient for replicating common scan artifacts. Indeed, to compare reconstruction algorithms, it is essential to work with data which is, if not scanned real-world data, as-realistic-as-possible.

Evaluation methodology aside, part of the difficulty in establishing a benchmark is the large variability in point clouds. Under triangulation-based scanning, a surface may be sampled under a wide variety of conditions, producing point clouds containing many different characteristics such as noise, outliers, nonuniform sampling, and missing data. This variability is further enhanced when scan data are processed to produce an oriented point cloud, where registration and normal orientation must be performed. With all of these factors considered, it is difficult to determine the effectiveness of a surface reconstruction algorithm operating on an arbitrary point cloud; see Figure 3.1 for an illustration.

To this end, we propose a benchmark for surface reconstruction for approximating methods. In particular, we operate in a synthetic environment in order to provide quantitative results, but provide realistic data by simulating a laser-based optical triangulation scanner. Our benchmark is broken up into three main phases: surface modeling, sampling, and evaluation. See Figure 3.2 for the full pipeline.

We start off with an implicit surface. In order to minimize any potential bias inherit in our implicit surface representation, we use *integrated polygonal constraints*, and approximate an implicit surface from a triangle mesh, as detailed in Section 3.1.

We then sample this implicit surface to obtain an oriented point cloud. We simulate the process of an optical triangulation scanner in order to produce range scans. We then slightly overlap the range scans and register them via a rigid-body registration algorithm. From the registered point cloud, we then compute and orient normals for each point, producing an oriented point cloud suitable for the class of algorithms under consideration. These steps are described in more detail in Section 3.2.

From the oriented point cloud, we now run a surface reconstruction algorithm on the input. This gives us a triangle mesh, which we evaluate by comparing to the implicit surface and a dense uniformly sampled point cloud of the implicit surface. We then construct

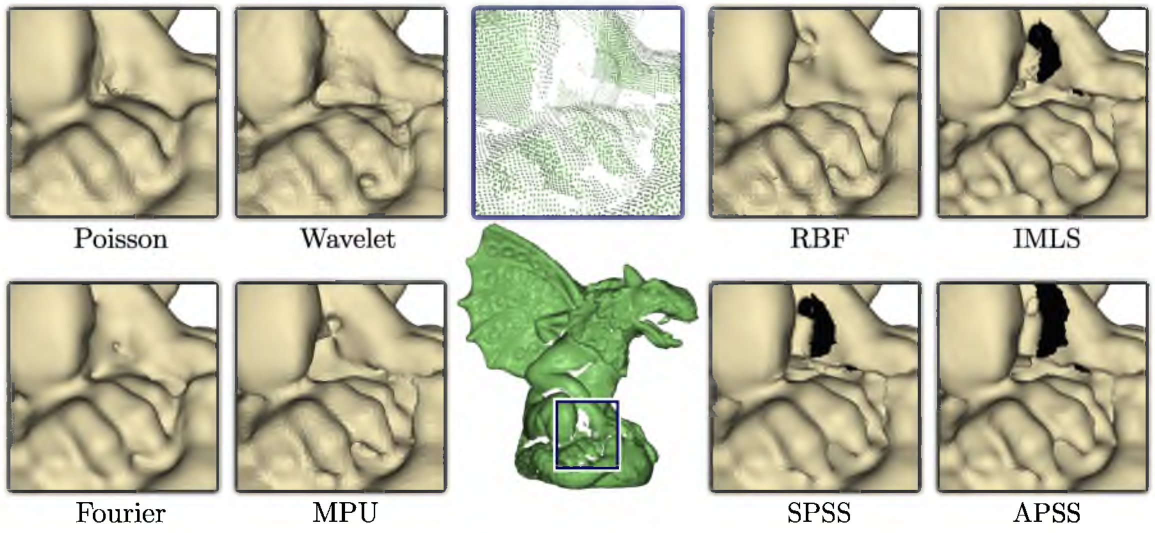


Fig. 3.1. Here, we have synthetically sampled the Gargoyle model, and ran eight separate reconstruction algorithms on this point cloud. Note the differences between the algorithms on the claw, where some algorithms over-smooth the data, while others result in spurious holes being produced. Our benchmark aims to generate such imperfect point cloud data and study these various forms of error.

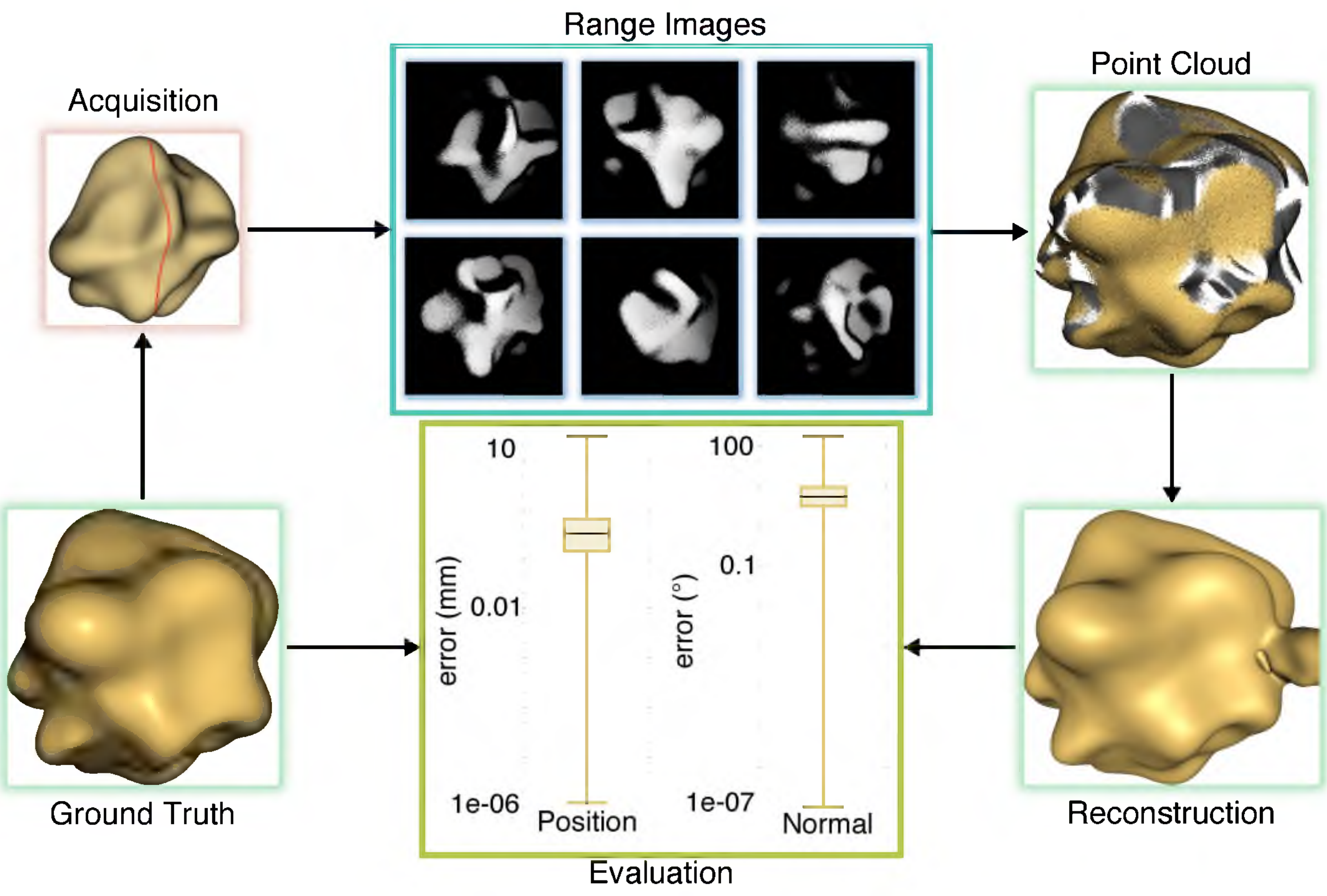


Fig. 3.2. Overview of our benchmark. First, we create an implicit representation of a surface mesh. We then sample this implicit surface by synthetically scanning the shape to obtain individual range scans, and consolidate the scans into a single oriented point cloud via registration and normal estimation. We run a reconstruction algorithm on this oriented point cloud, and compare this output to the implicit model and a dense uniform sampling of the implicit shape to obtain quantitative results.

positional and normal error metrics, demonstrated in Figure 3.2 as individual distributions of point-to-point correspondences. This is explained in more detail in Section 3.3.

In summary, we make the following contributions:

- **Realistic data**. We utilize a collection of both simple and complex shapes, where an implicit surface is used as the computational representation. We then synthetically scan the implicit surface to provide a collection of point clouds, where our scanning simulation is validated against real data.

- **Accuracy**. By employing implicit surfaces, we have a precise means of performing evaluation, in both positional and differential measures. We utilize particle systems to uniformly sample both the implicit surface and the reconstructed surface mesh, thereby minimizing any potential bias of measure from the corresponding triangulation.

- **Comprehensiveness**. The set of experiments comprise a broad range of behavior across surface reconstruction algorithms.

3.1 Surface Modeling

For modeling ground-truth data, care must be taken in the surface representation, as it impacts the rest of our pipeline. Although triangulated surfaces are popular and easy to work with, we use smooth and piecewise-smooth surfaces as ground-truth, as it benefits the sampling and evaluation phases as follows:

- **Sampling**. Our laser-based scanning simulator requires a surface equipped with a smooth normal field in order to best model an optical laser scanner. As the normal field of a triangulated surface is discontinuous between triangle faces, this surface representation can adversely impact our scanning simulator.

- **Evaluation**. The surface reconstruction algorithms under consideration assume a point cloud sampled from a smooth surface, so using a smooth surface for quantitative evaluation respects an algorithm's assumptions. Moreover, a smooth normal field permits us to reliably evaluate differential quantities in the reconstruction.

We use implicit surfaces to model smooth and piecewise-smooth surfaces, where we introduce *integrated polygonal constraints* as a mechanism for shape modeling. Namely, we create smooth and piecewise-smooth implicit surfaces by approximating triangulated surface meshes, or more generally polygon soups, through weight functions integrated over polygons. The advantages of using polygonal constraints over point constraints are twofold. First, approximation from a point cloud may produce specific forms of surface features in the presence of missing data; under polygon soup, we can ensure there is no such missing data. Secondly, identification and preservation of sharp features of a polygonal mesh is far more robust than a point cloud. This allows us to easily model smooth surfaces which contain sharp features.

3.1.1 Polygonal MPU

Our implicit representation is a straightforward extension of Multilevel Partition of Unity (MPU) [Ohtake et al. 2003] applied to polygon soup, with the main distinction of integrating weight functions over polygons. We use the weight function of Shen et al. [2004], defined for a given point $\mathbf{x} \in \mathbb{R}^3$ and for an arbitrary point on a triangle t, $\mathbf{p} \in t$:

$$w(\mathbf{x}, \mathbf{p}) = \frac{1}{\left(|\mathbf{x} - \mathbf{p}|^2 + \epsilon^2\right)^2} \tag{3.1}$$

Here, ϵ is a smoothing parameter used to ensure that w is bound above when $\mathbf{x}$ lies on t – we have set it to 0.1% of the bounding box diagonal for each shape. We may now integrate this weight function over the entire triangle t:

$$w(\mathbf{x}, t) = \int_{\mathbf{p} \in t} w(\mathbf{x}, \mathbf{p}) d\mathbf{p} \tag{3.2}$$

The quartic falloff in distance results in points which are far away from a triangle containing a low contribution to the implicit function. This falloff is necessary since the shape functions we use are roughly linear in distance; hence, w will dominate the shape function's contribution.

For evaluating Equation 3.2, Shen et al. [2004] propose a method for numerical integration. However, we derive a closed form solution for this expression. This prevents potential numerical inaccuracies caused by a quadrature scheme, which could be particularly detrimental to having a reliable benchmark. We outline the derivation in Appendix A.1.

Equipped with a mechanism for integrating weights over polygons, we proceed with MPU by fitting shape functions to a triangle soup $T = \{t_1, ..., t_n\}$. We adaptively build an octree over T, where for each octree cell, we associate with it a sphere whose radius is the length of the diagonal of the cell. We then gather all triangles which are contained in, or overlap the sphere, and fit a shape function to those triangles.

In practice, we use linear functions for our shape functions, where for each cell i, we associate the function $g_i(\mathbf{x}) = \mathbf{x}^{\mathrm{T}}\mathbf{n}_i + b_i$. For all triangles which belong to the sphere of cell i, $T_i \subset T$, we fit the shape function as follows:

$$\mathbf{n}_i = \frac{\sum_{t \in T_i} \mathbf{n}_t \int_{\mathbf{p} \in t} w(\mathbf{s}_i, \mathbf{p})\, \mathrm{d}\mathbf{p}}{\sum_{t \in T_i} \int_{\mathbf{p} \in t} w(\mathbf{s}_i, \mathbf{p})\, \mathrm{d}\mathbf{p}} \tag{3.3}$$

$$b_i = -\left\langle \frac{\sum_{t \in T_i} \int_{\mathbf{p} \in t} \mathbf{p}\, w(\mathbf{s}_i, \mathbf{p})\, \mathrm{d}\mathbf{p}}{\sum_{t \in T_i} \int_{\mathbf{p} \in t} w(\mathbf{s}_i, \mathbf{p})\, \mathrm{d}\mathbf{p}}, \mathbf{n}_i \right\rangle \tag{3.4}$$

where $\mathbf{n}_t$ is the triangle normal of t and $\mathbf{s}_i$ is the center of the sphere for cell i. Although one may use higher order shape functions under polygonal constraints, such as quadrics, we found the difference to be negligible, where the main difference is that for linear functions we require a larger number of shape functions to adequately approximate T.

The octree is built such that each cell is subdivided only if the zero set of its shape function deviates sufficiently from the sphere's triangles. If the octree cell's sphere is empty to start, then we grow the radius of the sphere out until we encompass a sufficient number of triangles (set to six in our experiments), and terminate the subdivision with its shape function. Once the octree construction is complete, we have a spherical covering of the space. We may then evaluate the implicit function at a point by blending all shape functions whose spheres contain that point:

$$f(\mathbf{x}) = \frac{\sum_i q_i(\mathbf{x}) g_i(\mathbf{x})}{\sum_i q_i(\mathbf{x})} \tag{3.5}$$

where q_i is a quadratic b-spline function centered at $\mathbf{s_i}$.

To preserve sharp features, we follow Ohtake et al. [2003] in detecting sharp features within a leaf cell and consequently applying CSG operations for exact feature preservation. In these cases, rather than using polygon soup, we instead use a manifold triangle mesh, so that sharp features can be easily identified by observing dihedral angles. We then apply union and intersection operations on overlapping shape functions to exactly preserve the sharp feature, where we support edges and corners containing a maximum degree of four.

3.1.2 Benchmark Shapes

We have modeled shapes specific to our two sets of experiments. Our first set of experiments consists of complex shapes, and so we have modeled five shapes containing different types of complexities. See Figure 3.3 for these shapes. The Gargoyle model contains details of various feature sizes, ranging from the bumps on the bottom to the ridges on its wings. The Dancing Children model is of nontrivial topology, containing tunnels of different sizes, in addition to having many varying features such as the rim of

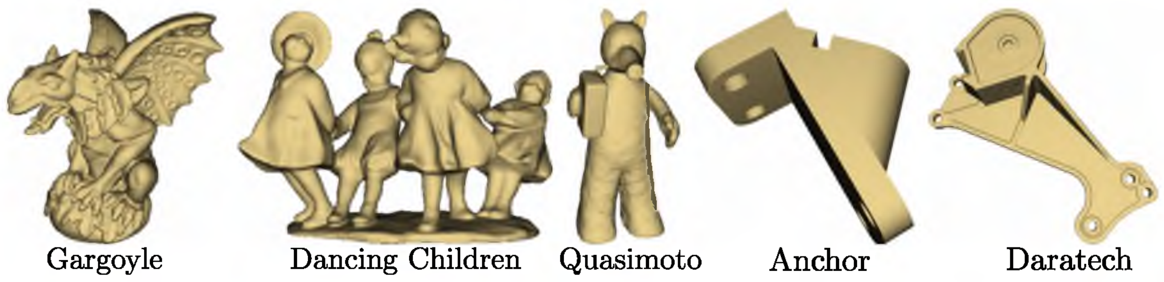


Fig. 3.3. Complex shapes created via our Polygonal MPU scheme. In our experiments, these shapes are utilized by performing synthetic range scanning under a wide variety of typical use-case scan parameters. This class of shapes contains many interesting characteristics for scanning, such as multiple scales of detail, nontrivial topology, and sharp features.

the hat on the left child and wrinkles in the cloth. The Quasimoto model is representative of a shape containing articulated parts, such as arms, legs, and head. The Anchor model contains sharp features, moderately-sized tunnels, as well as a single deep concavity. Lastly, the Daratech model contains sharp features, small tunnels, as well as thin surface sheets.

We note that the origin of these triangulated surfaces has a slight implication on the rest of the benchmark. That is, some of these models were scanned and consequently reconstructed to produce a triangle mesh. This has two consequences: the models must be visible from the perspective of a scanner, and polygonal MPU may inherit some of the smoothness properties of the particular reconstruction algorithm. While we did not notice any particular bias due to the latter, the visibility requirement is consistent with how we synthetically sample the models, namely through an optical scanner. Hence, it is still possible to sample all parts of a surface with our scanning simulator.

The second set of experiments utilizes simple shapes which may be sampled in a controlled manner. See Figure 3.4 for these shapes. The Bumpy Sphere contains smooth features at varying scales. The Spiral shape is primarily composed of a thin cylindrical

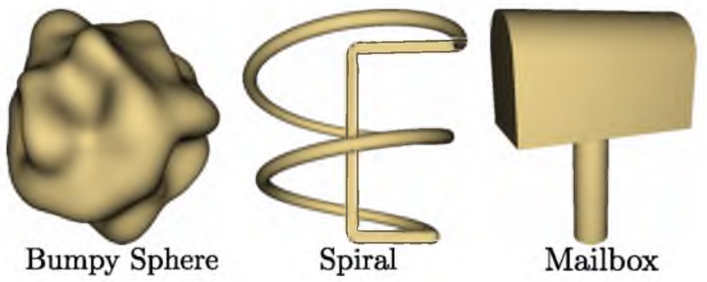


Fig. 3.4. Simple shapes created via our Polygonal MPU scheme. In our experiments, these shapes are scanned in a precise a manner in order to replicate specific scanning difficulties, such as sparsity, missing data, and noise.

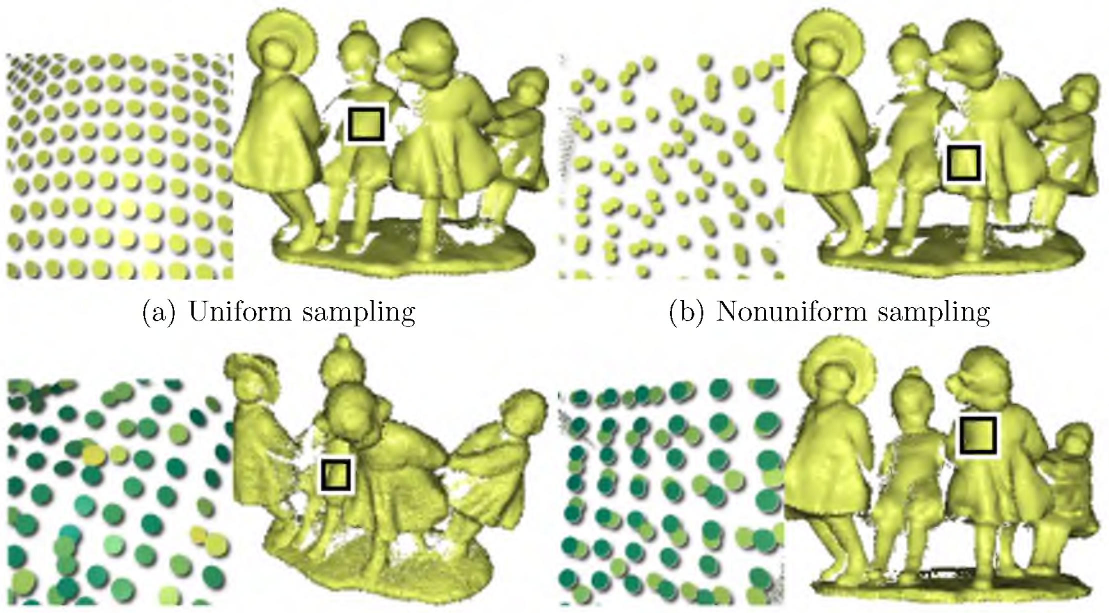
feature. Lastly, the Mailbox consists of straight and curved sharp features alike, while also remaining simple enough to sample in a controlled setting.

3.2 Sampling

The intent of our sampling scheme is to replicate the acquisition process of a triangulation-based scanner, in order to produce realistic point clouds. To this end, sampling is composed of three intermediate stages: synthetic range scanning, registration, and orientation. See Figure 3.5 for an illustration of our synthetic scanner's capability in replicating such properties.

3.2.1 Synthetic Range Scans

We simulate the acquisition of range scans by modeling a basic optical laser-based triangulation scanning system. Such scanning systems suffer from *random error* and *systematic error*. Random error is due to physical constraints, such as noise in the laser, variations in the reflectance due to surface materials, and nonlinear camera warping. Systematic error is the result of imprecise range measurement due to the peak detection algorithm. Our range scans are generated by synthesizing random error, while reproducing systematic error by



(a) Uniform sampling (b) Nonuniform sampling

(c) Noisy data (d) Misaligned scans

Fig. 3.5. Common characteristics of 3D scans. These point clouds were generated using our synthetic scanner, illustrating our capability to replicate common scan properties. In the noise and misalignment insets, we have color mapped the points by their distance away from the implicit shape, with yellow being far and green being close.

performing standard peak detection.

3.2.1.1 Random Error Synthesis

We synthesize random errors by generating a series of *radiance images*, where each image is the result of a single laser stripe projection onto the implicit surface. To this end, given a pinhole camera at position $\mathbf{c}$ and a baseline configuration, we first generate the exact range data by ray tracing the implicit surface. We reject all points that are not visible from the laser position, which is a function of the baseline distance. This provides us with a set of pixels containing geometry $P = \{\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_n\}$ and their corresponding points $X = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\}$.

We now project laser stripes onto the range geometry. We model each laser stripe projection according to a cylindrical projection, parameterized by laser position $\mathbf{l}$, field of view of the laser stripe α, and triangulation angle θ. The triangulation angle is defined with respect to an initial laser stripe plane. We may then define the laser stripe frustum as the volume enclosed by the two planes $\{\mathbf{l}, \theta - \frac{\alpha}{2}\}$ and $\{\mathbf{l}, \theta + \frac{\alpha}{2}\}$. A point is considered to be contained within the frustum if it is within positive distance to both planes. Figure 3.6 depicts a 2D illustration of this configuration, where the red points of the green curve are considered to be within the laser's frustum.

For a single laser stripe, we gather all range geometry which is contained within the stripe. This consequently defines the set of "active" pixels to which the laser stripe contributes. We then determine the noise-free radiance at pixel $\mathbf{p}_i$ due to a laser stripe at triangulation angle θ by Curless and Levoy [1995]:

$$L_\theta(\mathbf{p}_i) = |\mathbf{n}_i \cdot \omega| e^{\frac{-2.0(d(\mathbf{x}_i))^2}{\beta^2}} \tag{3.6}$$

where $\mathbf{n}_i$ is the normal of the implicit surface at $\mathbf{x}_i$, ω is the unit vector pointing towards the laser position from $\mathbf{x}_i$, $d : \mathbb{R}^3 \to \mathbb{R}$ is the closest distance to the center of the laser frustum, and β is the width of the frustum at $\mathbf{x}_i$. Here, we assume that the surface is purely diffuse; hence, the BRDF is reduced to a constant factor which we omit.

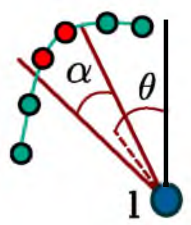


Fig. 3.6. Baseline configuration for determining points which are visible to the laser.

In practice, diffuse surfaces suffer from noise in the form of laser speckle, where surface roughness contributes to variations in the reflectance [Baribeau and Rioux 1991]. We observe that this form of noise is more significant further away from the center of the laser stripe frustum. We model this as additive noise sampled under a normal distribution, where the variance is the distance away from the center of the laser stripe:

$$\tilde{L}_\theta(\mathbf{p}_i) = L_\theta(\mathbf{p}_i) + \eta\epsilon_\sigma(\mathbf{x}_i) \tag{3.7}$$

Here, η is a user-specified noise magnitude, while ϵ is a random variable normally distributed with variance σ, the distance away from the center stripe. In addition, we also allow for smoothing of the noisy radiance image by convolving $\tilde{L}_\theta$ with a Gaussian kernel of a user-specified bandwidth.

3.2.1.2 Systematic Error

For each corrupted radiance image $\tilde{L}_\theta$, we next perform peak detection in order to find each pixel's laser stripe plane. From the laser stripe plane, depth is obtained simply by triangulation. A common assumption in many peak detection algorithms is for the radiance profile, either over space or time (triangulation angle), to be Gaussian [Curless and Levoy 1995]. However, in the presence of depth discontinuities, curved surfaces, and noise, this assumption is violated, resulting in range containing systematic errors.

To this end, we consider all radiance images $\tilde{L}_\theta$ defined for each triangulation angle $\theta \in \{\theta_1, \theta_2, ...\theta_m\}$, where m is the number of laser stripes. For each pixel, we consider its radiance profile as θ increases. We fit a Gaussian to this radiance profile via the Levenberg-Marquardt method. This Gaussian provides us with a mean, which determines the stripe plane, as well as a peak magnitude and variance, both of which are used for rejecting unconfident range data.

Please see Appendix A.2 for the full list of scanning parameters and common parameter settings.

3.2.2 Validation

It is important to verify that the range scans we are producing contain artifacts found in real scans. To this end, we validate the manner in which we generate range scans by comparing them to data acquired by commercial scanning systems. We illustrate our capability of replicating noise and missing data artifacts, which arguably have the greatest impact on surface reconstruction. We are not interested in exactly reproducing scans produced by commercial scanning systems. Most systems perform postprocessing which is far beyond the scope of our scanning simulation. Instead, we show that our scanning simulation is expressive enough to generate a range of scan artifacts, while still capable of generating artifacts of a commercial scanner under proper scan parameters. To perform validation, we use the following pipeline: model implicit surface $\rightarrow$ 3D print surface $\rightarrow$ scan printed model $\rightarrow$ register scan to implicit surface $\rightarrow$ compare to our synthetic scan.

We have manufactured the Gargoyle model via 3D printing, through the company Shapeways [Shapeways 2011]. The minimum detail at which models may be manufactured through Shapeways is 0.2mm. From this physical model, we then scan it via an optical triangulation-based scanner, namely the NextEngine scanner [NextEngine 2011]. In its finest resolution mode, termed macro mode, the scanner has a maximum accuracy of 0.127mm. For shapes in which the distance from the camera is at a specified optimum, and whose normal is approximately aligned with the camera's optical axis, we found this to be true. However, for a complex shape like the Gargoyle, as we will demonstrate, the accuracy can indeed vary and the noise magnitude becomes greater than the shape's resolution.

To compare a real scan to a synthetic scan, we first register the real scan to the implicit surface. We perform ICP under a rigid-body deformation in order to best align the real scan to the implicit surface. As the NextEngine does not provide specifics on their CCD sensor, we take the depth image and utilize the camera calibration toolbox [Bouguet 2010] to obtain the intrinsic and extrinsic camera parameters. We feed these camera parameters in to our synthetic scanning system to obtain a comparable range scan. We note that a small nonrigid deformation might be preferable to a rigid-body deformation for registration due to small nonlinear camera deformation artifacts [Brown and Rusinkiewicz 2007]. However, this adversely impacts camera calibration and hence is unsuitable for our purposes.

3.2.2.1 Noise Validation

In our scanning simulation, noise is strongly dependent on laser stripe resolution, laser stripe field of view, noise magnitude, and image smoothing bandwidth. As NextEngine does not provide these parameters for their system, to compare noise against the NextEngine scanner, we have best estimated the stripe resolution, field of view, and smoothing bandwidth, while varying the noise magnitude. See Figure 3.7 for the comparison. Note that real scanner noise is in fact anisotropic - a function of the baseline [Abbasinejad et al. 2009]. Hence, we see “bumps” which are slightly aligned with the direction of the laser projection in the NextEngine scan. Our synthetic scans demonstrate this anisotropy as well. We show that by simply tuning the noise magnitude, we are capable of producing a variety of noise profiles, wherein the NextEngine scanner is but a subset.

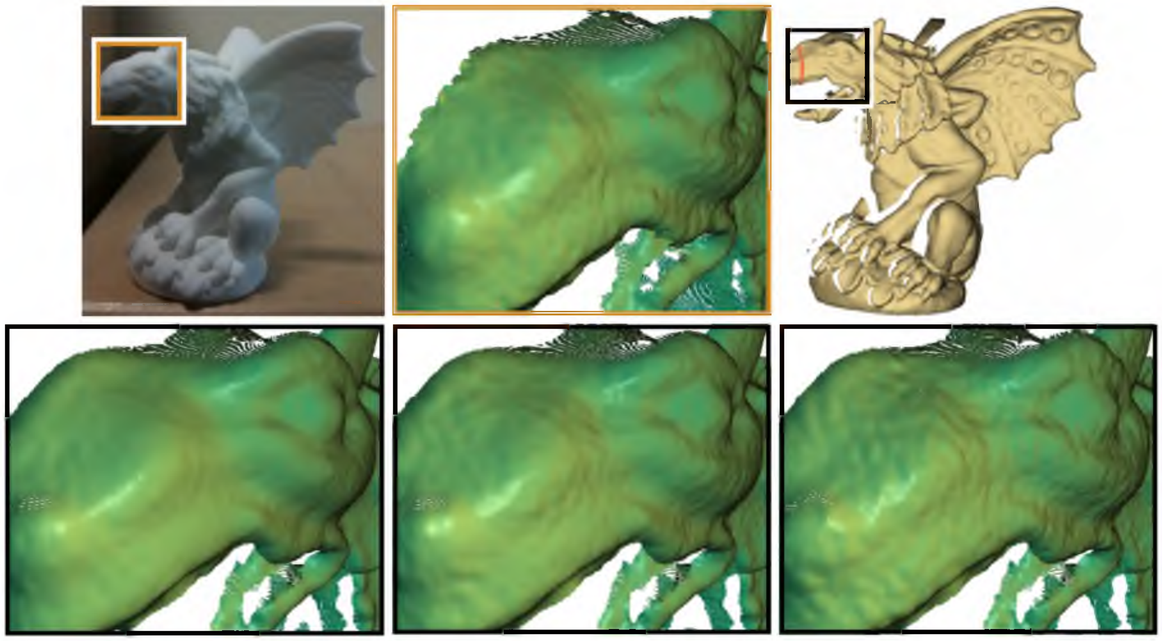


Fig. 3.7. Comparison of noise profiles between our scanning simulation in increasing noise magnitude (bottom), and a NextEngine scan (top-center). Note that real scanner noise takes the form of bumps aligned in the direction of the laser scan projection (top-right), and our synthetic noise is able to capture this anisotropic noise over varying noise magnitude.

3.2.2.2 Missing Data Validation

Missing data in a range scan are typically the result of the rejection of unconfident range data. In our scanning simulation, this is related to the peak intensity threshold, where a small peak may indicate a poor Gaussian fit. Hence, to compare missing data to the NextEngine scanner, we vary the peak intensity threshold and observe where regions of missing data exist; see Figure 3.8. As shown, the NextEngine scanner has a fixed threshold at which to reject unconfident range, while in our scanning system, this is a tunable parameter, producing varying degrees of missing data.

3.2.3 Scanning and Registration

Given that we have a means of acquiring range scans, next we must determine where to scan, and register the scans. It is extremely difficult to automate the process of positioning/orienting a scanner, as this is inherently a manual process. We assume an ideal environment wherein we place the scanner at uniformly sampled positions over the bounding sphere of the object, such that the camera is oriented to look at the object's center of mass. Note that such acquisition systems are starting to gain popularity [Vlasic et al. 2009].

From these individual range scans, we next register them into a single coordinate system. First, we overlap the scans by a parameterized amount. We then run the registration

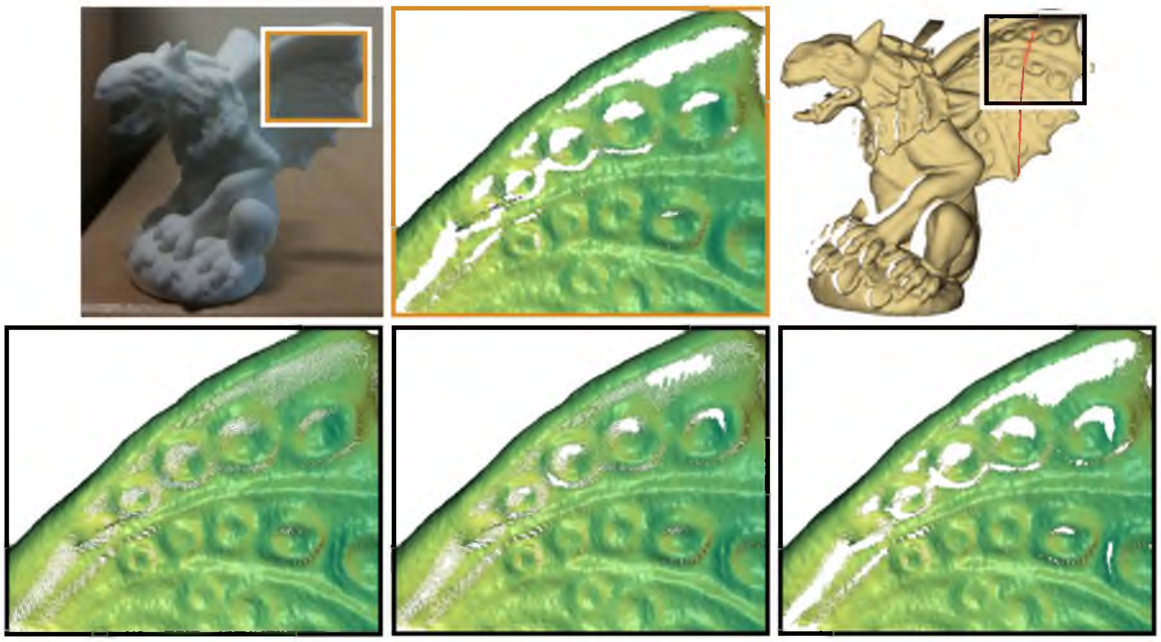


Fig. 3.8. A comparison of missing data between our scanning simulation in increasing peak threshold (bottom), and a NextEngine scan (top-center). Note the similarities in regions of missing data between our scan (bottom-right) and the NextEngine scan, chiefly due to the grazing angle at which laser strikes the surface, resulting in a low level of radiance.

algorithm of Brown and Rusinkiewicz [2007] to align the scans, which is a variant of ICP wherein a rigid-body transformation is assumed to be sufficient to align all scans. Note that the amount of overlap effectively determines the quality of the alignment. Less overlap means a poorer initialization, and the optimization process may hit an undesirable local minimum causing misalignment errors.

3.2.4 Orientation

From the registered point cloud, we must assign a normal to each point. One option is to simply use the analytical normal defined by the implicit function. However, for misaligned and noisy data, it becomes unclear what the normal should be from the implicit function. As a result, we also allow for normal orientation via the method of Hoppe et al. [1992].

Under this method, at every point, we estimate the local tangent plane via PCA, by gathering the k-nearest neighbors and extracting the eigenvectors of the covariance matrix. PCA, however, does not give orientation of the normals, and so we employ the minimum spanning tree approach of Hoppe et al. [1992] to propagate normal directions.

We note that by using this method, we may end up with noisy tangent planes due to a number of factors such as nonuniform sampling, noise, misalignment, and missing data. Moreover, normals may be oriented in the opposite direction due to these factors. However, in certain scanning situations, we may have knowledge of the scanner positions, which can be used to properly orient the normals. Hence, we allow for both options in our experiments.

3.3 Evaluation

In order to evaluate the quality of a surface mesh M output by a reconstruction algorithm against the input implicit surface Ω, we take the view of *discrete differential geometry* for defining error measures. As illustrated in Hildebrandt et al. [2006], pointwise plus normal convergence of a polyhedral surface to a smooth surface implies convergence in the metric, surface area, and Laplace-Beltrami operator. In their context, pointwise convergence is measured in terms of Hausdorff distance and normal convergence is measured as the supremum of the infinity norm over all normals. We take their basic framework and expand it to include other error measures, in order to provide a more informative evaluation.

3.3.1 Shortest Distance Map : $\Omega \to M$

To construct error measures, we first define the *shortest distance map*, termed Φ. This map defines, for each point on M, its closest point in Euclidean distance to Ω. More specifically, for a point $\boldsymbol{\alpha} \in \Omega$, the map $\Phi \colon \Omega \to M$ associates $\boldsymbol{\alpha} \in \Omega$ as the closest point to $\Phi(\boldsymbol{\alpha}) \in M$. We follow the approach of Hildebrandt et al. [2006] for the construction of the map:

$$\Phi(\boldsymbol{\alpha}) = \boldsymbol{\alpha} + \phi(\boldsymbol{\alpha})\mathbf{N}_\Omega(\boldsymbol{\alpha}) \tag{3.8}$$

where $\mathbf{N}_\Omega$ is the normal field over Ω and $\phi \colon \Omega \to \mathbb{R}$ is the signed distance along the normal $\mathbf{N}_\Omega(\boldsymbol{\alpha})$. So long as the Hausdorff distance of Ω and M is bound by the *reach* of Ω, or the minimal radius of all medial balls, then this construction ensures that $\boldsymbol{\alpha}$ is the point on Ω closest in distance to $\Phi(\boldsymbol{\alpha}) \in M$ [Federer 1959].

3.3.1.1 Sampling

The correspondences defined by the shortest distance map are used to construct error measures for comparing the reconstructed surface and implicit shape [Hildebrandt et al. 2006]. To obtain this in practice, we must densely sample Ω in order to obtain discretized yet precise error measures, in a similar manner to METRO [Cignoni et al. 1998]. However, we depart from METRO by employing particle systems to sample Ω, as we require not only dense samplings, but uniform samplings. A uniform sampling is essential in achieving accurate mean error measures, as a nonuniform sampling may bias certain regions of the surface in the construction of the mean.

We sample Ω by the method of Meyer et al. [2007], which minimizes an energy functional based on interparticle distances. A uniform distribution of samples is achieved by prescribing a single interparticle distance for all particles. We have empirically set the interparticle distance based on the complexity of each shape, such that all features of the shape are sufficiently sampled. See Figure 3.9 for uniform samplings of our complex shapes. If we denote P_Ω as the set of samples chosen from Ω, we build a set of ordered pairs representing shortest distance correspondences:

$$C_\Omega = \{(\mathbf{x}, \boldsymbol{\alpha}) \mid \boldsymbol{\alpha} \in P_\Omega, \mathbf{x} = \Phi(\boldsymbol{\alpha})\} \tag{3.9}$$

3.3.1.2 Correspondence Validation

If the Hausdorff distance between Ω and M exceeds the reach of Ω, then there may exist pairings in C_Ω which are not shortest distance correspondences. Namely, this situation implies that there may exist $\boldsymbol{\alpha} \in \Omega$ such that the line connecting $\boldsymbol{\alpha}$ and $\Phi(\boldsymbol{\alpha})$ crosses the medial axis of M; hence, Φ may no longer be bijective.

To handle such situations, we use the sample set P_Ω to *validate* correspondences constructed through Φ. Since P_Ω is a dense and uniformly distributed sampling of Ω, closest point queries through P_Ω serve as an upper bound in any potential error in the Φ mapping.

More specifically, for a given correspondence $\boldsymbol{\alpha}$ and $\mathbf{x} = \Phi(\boldsymbol{\alpha})$, we query the closest point to $\mathbf{x}$ in P_Ω, denoted as $\boldsymbol{\beta}$. If $|\mathbf{x} - \boldsymbol{\beta}| < |\mathbf{x} - \boldsymbol{\alpha}|$, this implies an incorrect pairing;

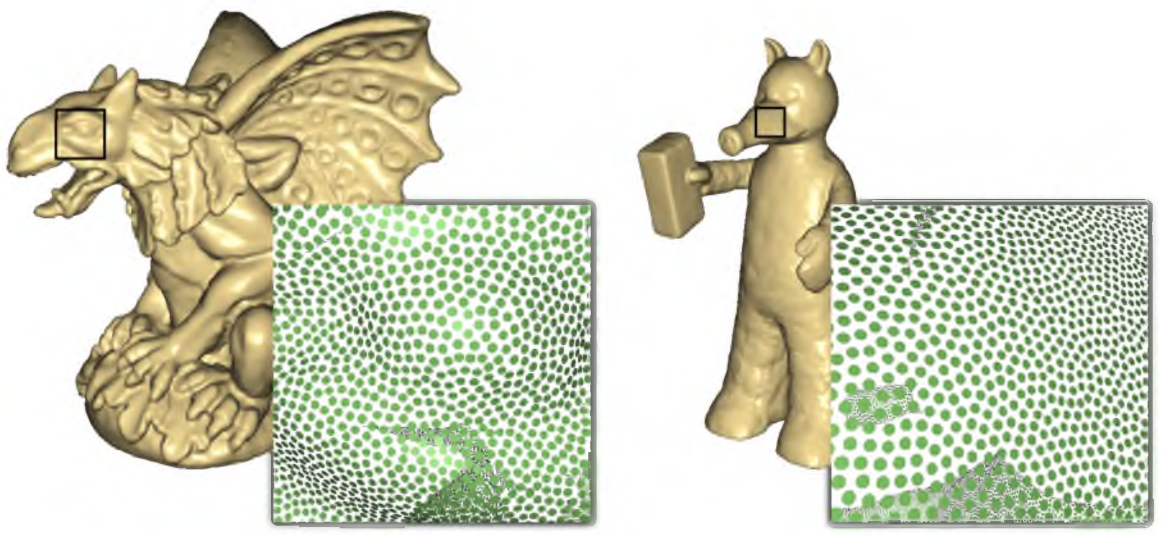


Fig. 3.9. Complex shapes sampled under particle systems. Note the high density and uniform distribution in the particles. Both of these properties are essential for obtaining precise error measures.

hence, we exclude the correspondence from C_Ω and instead add the correspondence $(\mathbf{x}, \boldsymbol{\beta})$ to C_Ω. See Figure 3.10 for a 2D illustration of the validation procedure.

3.3.2 Dual Map : $M \to \Omega$

Expanding on the work of Hildebrandt et al. [2006], we also construct a shortest distance map *dual* to Φ which we term Ψ, which considers for each point on Ω its closest point to M. There can exist points on the reconstructed mesh M which are not observed by the map Φ, and hence not considered as part of the error measurements. The mapping Ψ allows us to capture these otherwise unseen shortest distance correspondences.

To this end, we follow the methodology established in the previous section to construct $\Psi\colon M \to \Omega$, for a given $\mathbf{x} \in M$:

$$\Psi(\mathbf{x}) = \mathbf{x} + \psi(\mathbf{x})\mathbf{N}_M(\mathbf{x}) \tag{3.10}$$

where $\mathbf{N}_M$ is the normal field over M and $\psi\colon M \to \mathbb{R}$ is the signed distance along the normal $\mathbf{N}_M(\mathbf{x})$.

3.3.2.1 Sampling

Analogous to the Φ mapping, we sample M in order to construct a discrete set of correspondences for the dual map. In sampling M, we have adapted the approach of Meyer et al. [2007] to triangulated surfaces, though other methods such as Poisson disk sampling may be employed to achieve a uniform sampling [Bowers et al. 2010]. Rather than specify an interparticle distance for M in the optimization, we specify the number of particles, as the output reconstructed mesh can be arbitrarily complicated. If we denote P_M as the set

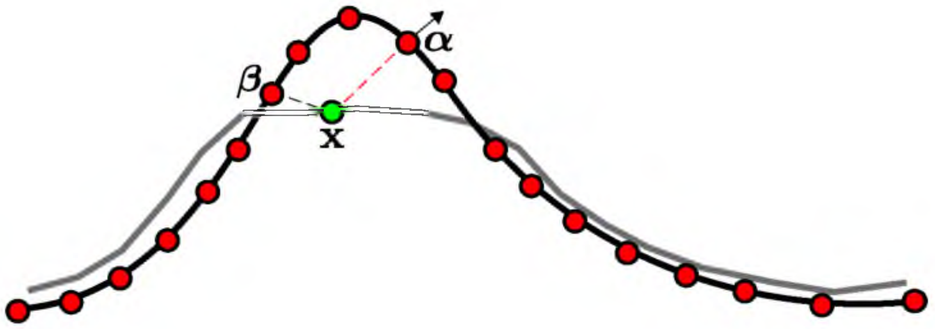


Fig. 3.10. A situation where the Φ mapping produces an incorrect shortest distance correspondence. The dashed red line indicates the normal line from $\boldsymbol{\alpha}$ to $\mathbf{x}$, giving us an inaccurate correspondence since $\boldsymbol{\beta}$ is closer to $\mathbf{x}$ than $\boldsymbol{\alpha}$. So we instead take $(\mathbf{x}, \boldsymbol{\beta})$ as a correspondence.

of samples chosen from M, we build a set of ordered pairs representing shortest distance correspondences defined by the dual map:

$$C_M = \{(\boldsymbol{\alpha}, \mathbf{x}) \mid \mathbf{x} \in P_M, \boldsymbol{\alpha} = \Psi(\mathbf{x})\} \tag{3.11}$$

3.3.2.2 Correspondence Validation

If the Hausdorff distance between Ω and M is large, then similar to the Φ mapping, the Ψ mapping may result in pairs belonging to C_M which are not shortest distance correspondences. In addition, due to the reach of M being 0, even if the Hausdorff distance is small, there may still exist incorrect pairings in C_M. We note that this mostly occurs near triangle edges where the dihedral angle is large; however, large dihedral angles are a rare occurrence in our setting since the mesh M intends to closely approximate the (piecewise-)smooth surface Ω.

In either case, we can still employ a similar validation scheme to the Φ mapping to ensure that the error in Ψ is bounded. Since the sample set P_M densely and uniformly samples M, closest point queries in P_M ensure an upper bound in the error. More specifically, for a given correspondence $\mathbf{x}$ and $\boldsymbol{\alpha} = \Psi(\mathbf{x})$, we query the closest point to $\boldsymbol{\alpha}$ in P_M, denoted as $\mathbf{y}$. If $|\boldsymbol{\alpha} - \mathbf{y}| < |\boldsymbol{\alpha} - \mathbf{x}|$, this implies an incorrect pairing; hence, we exclude the correspondence from C_M and instead add the correspondence $(\boldsymbol{\alpha}, \mathbf{y})$ to C_M.

3.3.3 Discrete Error Measures

From here, we may define a variety of discrete error measures between Ω and M. Denoting $|S| = |C_\Omega| + |C_M|$, Hausdorff distance is approximated by:

$$H(\Omega, M) = \max\left\{ \max_{(\mathbf{x},\boldsymbol{\alpha})\in C_\Omega} |\mathbf{x} - \boldsymbol{\alpha}|, \max_{(\boldsymbol{\alpha},\mathbf{x})\in C_M} |\boldsymbol{\alpha} - \mathbf{x}| \right\} \tag{3.12}$$

while mean distance is approximated by:

$$\mu(\Omega, M) = \frac{1}{|S|}\left(\sum_{(\mathbf{x},\boldsymbol{\alpha})\in C_\Omega} |\mathbf{x} - \boldsymbol{\alpha}| + \sum_{(\boldsymbol{\alpha},\mathbf{x})\in C_M} |\boldsymbol{\alpha} - \mathbf{x}| \right) \tag{3.13}$$

These measures depict error in very different ways; see Figure 3.11 for an illustration. Here, the circle is the smooth shape, while the piecewise linear curve is the approximating mesh. Hausdorff distance will be large for the pair of shapes on the left, while mean distance will be rather small, whereas for the pair of shapes on the right, mean distance will be much larger than the pair of shapes on the left, while Hausdorff distance will be less.

From these shortest distance correspondences, we have a method of measuring higher-order geometric properties, by comparing differential properties at the correspondences.

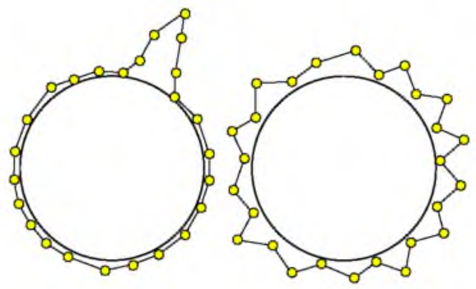


Fig. 3.11. Different forms of surface reconstruction error. On the left, Hausdorff distance is large while mean distance is small, while the opposite holds on the right.

This is analogous to defining pullbacks on Φ and Ψ. We opt to measure normal angle deviations in a similar manner to distance measures. If we denote $\gamma(\boldsymbol{\alpha}, \mathbf{x}) = \angle(\mathbf{N}_{\Omega}(\boldsymbol{\alpha}), \mathbf{N}_M(\mathbf{x}))$, the maximum and mean angle deviation of point correspondences, respectively, are:

$$H_N(\Omega, M) = \max\left\{ \max_{(\mathbf{x},\boldsymbol{\alpha})\in C_\Omega} \gamma(\boldsymbol{\alpha}, \mathbf{x}), \max_{(\boldsymbol{\alpha},\mathbf{x})\in C_M} \gamma(\boldsymbol{\alpha}, \mathbf{x}) \right\} \tag{3.14}$$

$$\mu_N(\Omega, M) = \frac{1}{|S|}\left(\sum_{(\mathbf{x},\boldsymbol{\alpha})\in C_\Omega} \gamma(\boldsymbol{\alpha}, \mathbf{x}) + \sum_{(\boldsymbol{\alpha},\mathbf{x})\in C_M} \gamma(\boldsymbol{\alpha}, \mathbf{x}) \right) \tag{3.15}$$

In practice, we take $\mathbf{N}_M$ to be triangle normals, as opposed to more sophisticated normal estimation methods [Meyer et al. 2002]. Such methods are sensitive to the triangulation and typically assume smoothness in the normal field, where in the presence of sharp features, this can result in undesirable over-smoothing.

3.3.4 Algorithms

We have chosen a wide variety of publicly available surface reconstruction algorithms to test our benchmark against. For the sake of fair comparison, we have only used algorithms which take an oriented point cloud as input, and output an approximating surface. Here, we provide a categorization and brief description of each algorithm, while also providing an abbreviation of each to help identify them in the experiments to follow.

3.3.4.1 Indicator Function

This class of algorithms reconstructs a three-dimensional solid O by finding the scalar function χ, known as the indicator function, defined in $\mathbb{R}^3$ such that:

$$\chi(\mathbf{x}) = \begin{cases} 1 & \mathbf{x} \in O \\ 0 & \mathbf{x} \notin O \end{cases} . \tag{3.16}$$

where the surface Ω is then defined by ∂O. In practice, these approaches approximate χ by operating on a regular grid or an octree, and generate Ω by isosurfacing the grid.

Poisson surface reconstruction (abbr. **Poisson**) [Kazhdan et al. 2006] solves for χ by noticing that $\nabla\chi$ should agree with the normal field $\mathbf{N}$ at ∂O. This amounts to inverting the gradient operator; hence, χ is found by solving the Poisson equation:

$$\nabla \cdot \nabla\chi = \nabla \cdot \mathbf{V} \tag{3.17}$$

where $\mathbf{V}$ is the smoothed normal field defined throughout the volume. The Poisson equation is efficiently solved only near the surface by using an adaptive multigrid solver defined on the octree built on the point cloud. Note that use of an octree may result in limited resolution over regions of missing data.

An alternative method of constructing the indicator function is to solve for it indirectly by projecting χ onto a basis, and then performing an inverse transform to obtain χ. By invoking Stokes theorem, this projection need only be performed on ∂O:

$$\int_O \nabla \cdot \mathbf{F}(p)dp = \int_{\partial O} \langle \mathbf{F}(p), \mathbf{N}(p) \rangle \, dp \tag{3.18}$$

where $\mathbf{F}$ is a vector-valued function whose divergence $\nabla \cdot \mathbf{F}$ defines the basis.

Fourier surface reconstruction (abbr. **Fourier**) [Kazhdan 2005] employs the Fourier basis as part of their solution. For efficiency, they use the Fast Fourier transform (FFT), hence requiring a regular grid and the grid resolution being a power of two. However, use of a regular grid has its benefits when faced with missing data, as their is no loss of resolution.

Wavelet surface reconstruction (abbr. **Wavelet**) [Manson et al. 2008] employs a Wavelet basis for the solution of Equation 3.18. They show how one may use a Haar or a Daubechies (4-tap) basis, where in our experiments, we employ the 4-tap Daubechies basis. Due to the multiresolution structure of wavelets, they use an octree for the basis projection; hence, similar to Poisson, this method may result in limited resolution over regions of missing data.

3.3.4.2 Point Set Surfaces

Point set surfaces (PSS) are defined based on moving least squares (MLS), where a projection operator is used to define a surface by its collection of stationary points, or where the output point of the projection operator is its input point. Originally defined for unoriented points, its definition is greatly simplified when considering points equipped with normals, and may be used for surface reconstruction by considering its implicit surface definition, rather than its projection operator.

Basic PSS methods use a weighted combination of linear functions to locally define the surface at every point. Borrowing terminology from Guennebaud and Gross [2007], we use two different definitions in our experiments: simple point set surfaces (abbr. **SPSS**) [Adamson and Alexa 2003] and implicit moving least squares (abbr. **IMLS**) [Kolluri 2005]. The implicit surface definition of SPSS is:

$$f(\mathbf{x}) = \mathbf{n}(\mathbf{x})^T(\mathbf{x} - \mathbf{c}(\mathbf{x})) \tag{3.19}$$

where $\mathbf{n}$ is a weighted average of normals in a neighborhood of $\mathbf{x}$, and $\mathbf{c}$ is the weighted centroid in a neighborhood of $\mathbf{x}$. The weights used in computing the normal and the centroid are derived from a smooth, positive function $w_{\mathbf{x}}$ defined with respect to $\mathbf{x}$, which gives points closer to $\mathbf{x}$ larger influence. IMLS is defined as the implicit function:

$$f(\mathbf{x}) = \frac{\sum_i w_{\mathbf{x}}(\mathbf{p}_i)(\mathbf{x} - \mathbf{p}_i)^T \mathbf{n}_i}{\sum_i w_{\mathbf{x}}(\mathbf{p}_i)} \tag{3.20}$$

We note that IMLS is a weighted average of linear functions, whereas SPSS is a single linear function, whose centroid and normal is a weighted average of points and normals, respectively.

Algebraic point set surfaces (abbr. **APSS**) [Guennebaud and Gross 2007] uses spheres defined algebraically as the shape function. Rather than directly obtaining the implicit function at a point, APSS fits a sphere to a neighborhood of points, requiring the solution of a linear least squares system for every point. By using a higher-order function, the method can be more robust to sparse data than SPSS and IMLS.

For our experiments, the software package provided by Gaël Guennebaud contains implementations of SPSS, IMLS, and APSS. Each PSS is evaluated over a regular grid, and the reconstructed surface is obtained by isosurfacing the zero level-set. In the software, neighborhoods used to locally fit functions are estimated at each point based on the density of the input point cloud. In the presence of missing data, this method may produce an empty neighborhood, producing holes in the output. This has an impact on evaluation, which we further discuss throughout the experiments sections.

3.3.4.3 Multilevel Partition of Unity

In our own implicit surface definition, we use a variant of Multilevel Partition of Unity (MPU) applied to polygon soup, and so we refer to Section 3.1.1 for details about the overall approach, noting that the construction of MPU with points is quite similar to that of polygons. In our experiments, we use three variants. First, we use the original approach of Ohtake et al. [2003] (abbr. **MPU**), where linear functions are used as low-order implicits. We opted not to use the fitting of sharp features, as we found its sharp feature detection to be rather sensitive and frequently produce erroneous fits. We also use the approach of Nagai et al. [2009] (abbr. **MPUSm**), which defines differential operators directly on the MPU function, though restricted to linear functions. In doing so, diffusion of the MPU function becomes possible, resulting in a more robust reconstruction method. Lastly, we also use the method of Ohtake et al. [2005b] (abbr. **RBF**), which uses compactly-supported radial basis functions for locally-defined implicit functions in the MPU construction. For all MPU methods, a surface mesh is generated by first evaluating the MPU function over a regular grid, and isosurfacing the zero level-set to obtain the surface.

3.3.4.4 Scattered Point Meshing

The method of Ohtake et al. [2005a] (abbr. **Scattered**) is a departure from the above approaches. This method grows weighted spheres around points in order to determine the connectivity in the output triangle mesh. Quadric error functions [Garland and Heckbert 1997] are used to position points in the output mesh, which can result in a small amount of simplification in the output. Similar to the PSS methods, regions of absent data may result in holes in the output.

3.3.5 Algorithm Parameters

We provide a brief description of the most relevant parameters for each algorithm.

3.3.5.1 Resolution

As all algorithms, except Scattered, contour a grid to obtain the surface, they must contain sufficient grid resolution to adequately preserve all surface details. Our aim is to provide each algorithm with such a sufficient resolution, while maintaining fairness across algorithms which may define grids differently. To achieve this, for each implicit surface, we first determine the resolution which is necessary to extract the surface with minimal error. We find that across all shapes, a resolution of 350^3 provides for sufficient resolution to preserve surface details; hence, for the PSS and MPU methods, we have set their resolution to 350. For Fourier, the resolution at which to contour is also the resolution at which the FFT is applied. As it must be a power of two, we set it to 512 in order to reduce any smoothing resulting from the FFT.

Since Poisson and Wavelet build an octree over the point cloud, we must strike a balance between resolution to where data exist, and where data do not exist due to incomplete sampling. Although a maximum octree depth of 9 may appear most reasonable, in regions of missing data, we found this resolution to be too coarse. Note that both methods refine octree nodes if certain sampling conditions are not satisfied; hence, a larger depth can greatly improve the reconstruction in regions of missing data. For Poisson, for a given sample point at an arbitrary depth, if neighboring nodes at that depth are not included in the octree, then these nodes are refined in order to support subnode precision. Figure 3.12 illustrates Poisson for two different maximum depths, and how the additional resolution shown on the bottom improves the reconstruction in regions of missing data. For Wavelet, as we are using 4-tap Daubechies Wavelets, a local support of 4^3 samples is necessary, and so neighboring nodes not already in the octree will similarly be refined. Hence, for both methods, we set the maximum tree depth to 10, in order to strike a compromise between resolution to where data exist, while providing sufficient resolution to where data do not exist.

3.3.5.2 Noise

Algorithms tend to handle noise according to their categorization. For indicator functions, noise may be combated by splatting the points in the grid under a large bandwidth, as well as through lowering the grid resolution, effectively serving as a low pass filter. PSS methods all contain a bandwidth which determines the extent of neighborhood influence. A large bandwidth results in more points for consideration in shape fitting and hence larger data smoothing. MPU methods and Scattered all contain error thresholds for which to determine the quality of a shape fit. In the presence of noise, the tolerance may simply be increased to avoid overfitting. MPUSm also provides parameters specific to their diffusion method, for which we use author-suggested settings.

3.3.5.3 Discussion

In practice, we set an algorithm's parameters based on the characteristics of the input point cloud, namely the noise level. As the point clouds of experiments 7.1-7.3 contain a constant level of noise, we have kept all algorithm parameters fixed throughout these experiments. Though one may fine-tune an algorithm's parameters to improve its performance with respect to a particular error metric, parameter insensitivity is an important indication

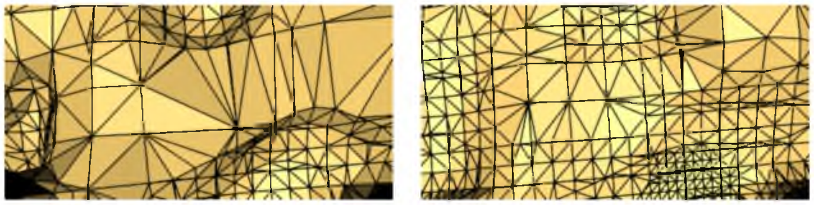


Fig. 3.12. Poisson surface reconstruction for two different maximum depths. Note that the additional resolution serves to refine regions of missing data.

of algorithmic robustness. Only in experiment 7.4, where noise varies, do we set algorithm parameters in accordance with the noise level.

3.4 Results

Our results are broken down into two main sets of experiments: one in which complex shapes are sampled under a variety of sampling settings, and one in which simple shapes are sampled under specific sampling settings. Please see Appendix A.2 for reference to the types of units used throughout the results.

We have not used the maximum angle deviation as an error measure in our experiments. By using triangle normals as the normal field over a surface mesh, this measure can be quite high even when the mesh contains low error in all other measures. As a result, in comparing algorithms, we found this error measure to be rather indistinguishable; hence, we have omitted it.

Note that it is possible for these algorithms to produce surfaces containing multiple connected components. We have decided to extract the largest connected component, in terms of surface area, as the surface for evaluation rather than all components. Unfortunately, this biases algorithms in which connected components are created far from the ground truth surface over algorithms which create additional components near the surface. Hence, in addition to the error metrics, we have provided additional information on the algorithms including the number of connected components, as well as the length of the boundary components, whether or not the surface is manifold, deviation from the true genus, and computation time.

3.4.1 Error Distributions

Our first set of experiments focuses on the performance of surface reconstruction algorithms restricted to a single shape. For a single shape, we sample it across a variety of scanner parameter settings, run all reconstruction algorithms across all point clouds, and compute error metrics for each point cloud. For each algorithm, we then aggregate the error metrics across all point clouds to obtain what we term *error distributions*.

We argue that error distributions are more effective for benchmarking reconstruction algorithms, rather than comparing algorithms with respect to a single point cloud. Each algorithm has its strengths and flaws for particular forms of data, and to sample a shape in such a way that it caters towards the strengths of certain algorithms provides an incomplete picture in the comparison of reconstruction algorithms.

To this end, we generate samples by varying scanning parameters across typical use-case settings. Namely, we vary sampling resolution, the number of range scans, the distance the camera resides from the object, peak threshold, and variance threshold. Please see Table 3.1 for the full range of parameters over all shapes. We have adapted certain parameter ranges to the specific shapes in order to ensure adequate coverage in the point clouds, and to sufficiently capture shape details. To reproduce small imperfections commonly found in range data, we introduce a constant, modest amount of noise into the laser signal. We also slightly overlap the scans and register them, causing small misalignment errors. For each point cloud, we randomly distribute camera positions uniformly on the bounding sphere of the object, rather than keeping their positions fixed.

See Figure 3.13 for the results of this experiment across all shapes, wherein the distributions take the form of box plots. The three error measures, mean distance, Hausdorff distance, and mean angle deviation, illuminate the various strengths and weaknesses of the algorithms.

3.4.1.1 Smooth Surfaces

The Gargoyle, Dancing Children, and Quasimoto shapes represent our class of shapes containing entirely smooth surface features. We find that the algorithms generally perform quite well on these shapes; however, the different error metrics point to subtle differences in performance. For instance, Wavelet tends to produce nonsmooth, rather bumpy surfaces, yet the surface tends to stay close to the surface, which is likely due to the use of wavelet bases in the presence of nonuniform or missing data. This nonsmoothness is depicted in the mean distance and angle deviation plots, yet its Hausdorff distance performance is quite

Table 3.1. The range of scanning parameters used in the error distribution experiments. Here, *res* represents the image resolution of a single range scan, *scans* is the number of scans taken, *camera dist* is the camera distance away from the center of the object, *peak* is the radiance threshold at which to reject depth, and *variance* is the variance threshold at which to reject depth.

shape	res	scans	camera dist	peak	variance
Gargoyle	250–350	7–11	75–115	0.2–0.4	0.5–0.75
DC	250–350	7–11	75–115	0.2–0.4	0.5–0.75
Quasimoto	250–350	7–11	75–115	0.2–0.4	0.5–0.75
Anchor	175–225	8–12	60–100	0.2–0.4	0.5–0.75
Daratech	250–350	8–12	75–115	0.2–0.4	0.5–0.75

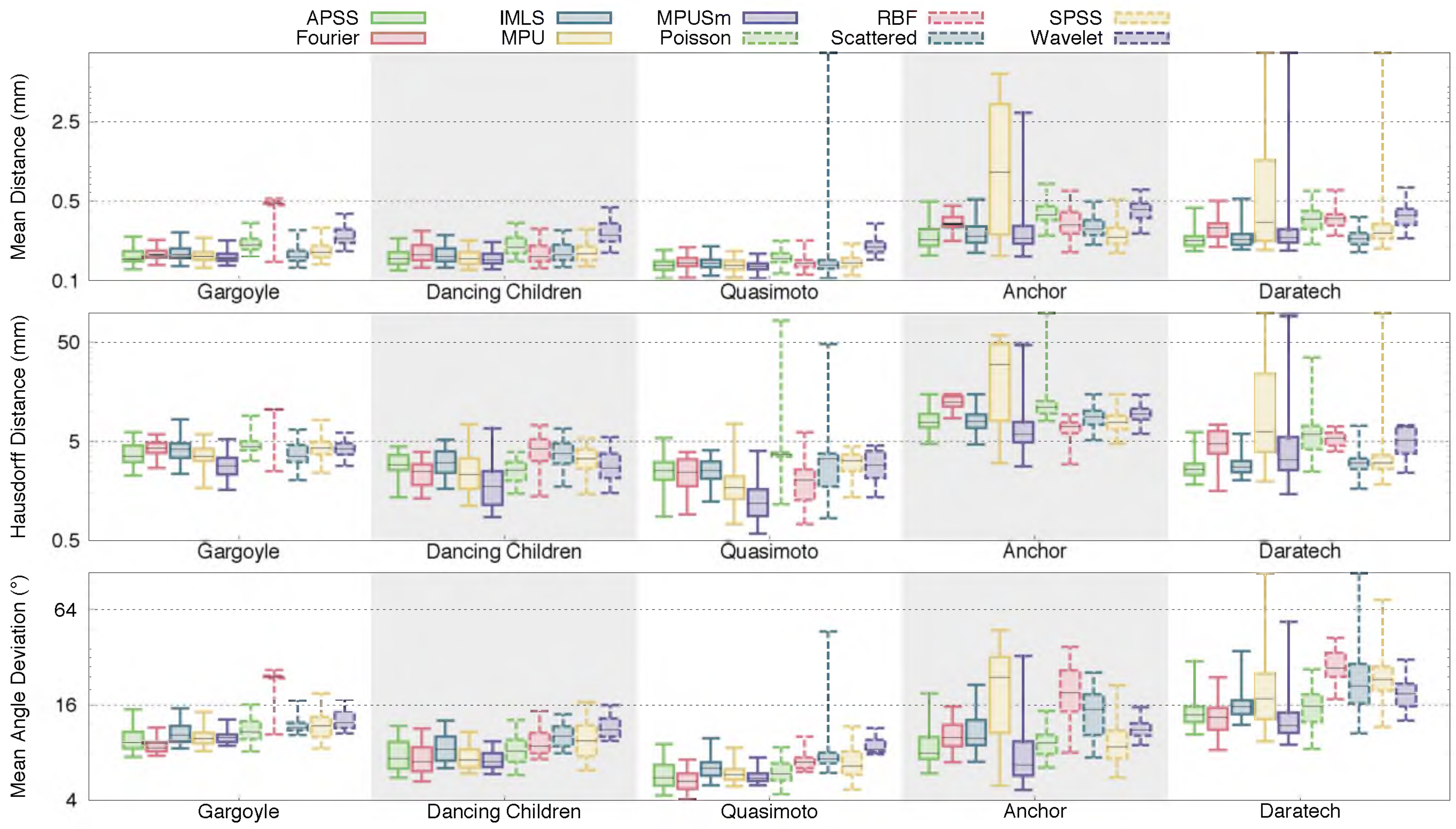


Fig. 3.13. Plots for all of the error distribution experiments. Each bar plot represents the distribution of a particular error measure for a given shape, sampled under a wide variety of scan parameters. The median provides a good indication of overall algorithmic performance for a given error measure, while the quartiles give an indication of algorithmic robustness.

competitive, indicating it never strays too far from the surface.

It is well known that Poisson and Fourier tends to oversmooth the data, and in our experiments, this is reflected in their rather large error in mean distance. However, in terms of Hausdorff distance and mean angle deviation, they perform rather well, and are fairly consistent in their performance. This indicates that these algorithms are reliable in reconstructing surfaces which do not deviate too far from the original, while also remaining close in differential quantities. We note that Fourier is more consistent than Poisson, as Poisson suffers from a lack of resolution in regions of missing data.

While RBF performed well on the Dancing Children and Quasimoto models, on the Gargoyle model, we see that it performed poorly across all metrics. The Gargoyle model is particularly difficult to sample as it has many concavities, and as shown by the lower quartile having large error across all metrics, RBF would tend to fill in the inside of the surface.

3.4.1.2 Sharp Features

The Anchor and Daratech shapes are particularly difficult to reconstruct. As these are shapes with sharp features, algorithms which only model smooth surfaces will have difficulty in reproducing sharp features. Additionally, these shapes have small topological features which are difficult to adequately scan due to occlusion. Hence, we do not necessarily expect these algorithms to perform as well on these shapes as the others, and instead, we use these shapes to measure robustness.

In observing MPU and MPUSm, we find instability in the presence of the Anchor and Daratech point clouds, where large spurious surface sheets are produced as a result of improperly fitting smooth shape functions to sharp features. However, note that the PSS methods perform much better, despite also using smooth shape functions. PSS methods fit shape functions at every point; hence, the error will be contained locally if there exists a poor fit, whereas MPU fits shape functions to the entire shape, resulting in a potentially unbounded error if a poor fit exists. Interestingly, RBF performs quite well in distance, yet has rather large error in normals. We found the RBF interpolant to remain quite close to the surface, at the expense of producing high-frequency details, hence the large normal deviations.

3.4.1.3 Topology

Overall, we find that the PSS methods and Scattered tend to perform quite well in the error metrics. However, these are also methods which produce holes in the presence of insufficient data. To depict the performance of these algorithms in terms of topology, we also show how these algorithms behave in their number of connected components, total length of boundary components, whether or not the reconstructed mesh is manifold, and the deviation from the true genus, averaged over all point clouds and shapes – see Table 3.2. As shown, Fourier and Poisson tend to outperform these methods in all categories. With respect to the PSS methods, this demonstrates that they tend not to produce topologically clean implicit functions, likely due to their local nature. Additionally, we see that Scattered produces large holes, yet all of the shapes are watertight.

3.4.2 Sparse Sampling

A common data characteristic of point clouds is sparsity. Namely, for range scan data, it is common for certain areas of the surface to be sampled less densely than others. Here, we investigate how reconstruction algorithms behave as data sparsity varies in a controlled setting. We are interested in observing how these algorithms infer the surface *between* the given input points.

In this experiment, we only vary the sampling resolution. We fix the number of scans and camera positions such that the shape is sufficiently sampled. We use the analytical normals of the surface, and no noise or misalignment. We use such clean input in order to restrict the problem to only data inference. We use the bumpy sphere as the test shape, as the coarse-scale features of the surface make data inference plausible.

Table 3.2. Additional information for experiment 1, averaged across all point clouds and shapes. Here, *comps* refers to number of connected components, *bndry* is the length of boundary components, *manifold* is whether or not a mesh is manifold, 1 being it is and 0 otherwise, *genus* refers to the amount which deviates from the actual genus, and *time* is in seconds.

algorithm	comps	bndry	manifold	genus	time
apss	47.37	140.86	0.50	1.82	36.02
fourier	1.54	0.00	1.00	0.49	28.70
imls	38.48	194.65	0.74	1.66	34.11
mpu	100.69	9.71	0.49	0.79	12.83
mpusmooth	2.88	2.93	0.91	0.67	17.83
poisson	1.54	0.44	1.00	0.63	36.83
rbf	51.73	6.30	0.82	13.55	34.78
scattered	1.90	214.21	1.00	7.47	4.48
spss	174.53	143.14	0.26	3.98	33.53
wavelet	1.35	0.04	1.00	0.71	2.13

See Figure 3.14 for plots of the experiment. MPUSm was unable to smooth its spherical covering on half of the point clouds due to the extreme sparsity, so we have omitted it from this experiment. From the distance measures, we immediately see a partitioning of the algorithms: IMLS, Poisson, SPSS, and Wavelet all tend to behave rather poorly, while the other algorithms perform well. We should certainly expect this for Poisson and Wavelet, as the resolution of the output is proportional to the input size. However, it is interesting to observe the significant improvement of APSS over IMLS and SPSS, indicating that fitting spheres under sparse data is more advantageous than trying to fit planes to the data.

We also see that Fourier demonstrates remarkable robustness to sparse data. Under very sparse data, Fourier performs best, whereas APSS, MPU, RBF, and Scattered perform rather poorly under such data, though they perform better as resolution increases. However, observe that as the sampling resolution becomes somewhat dense, the distance error in APSS, MPU, and RBF steadily decreases while Fourier remains stagnant. This is a consequence of Fourier's inherent data smoothing, whereas those algorithms which fit shape functions to the data only improve their fits as resolution increases.

3.4.3 Missing Data

Missing data will almost always be present in scanned data, simply due to concavities in the shape which cannot be reached by the scanner or insufficient scanning due to physical restraints of the scanner. In order to have a controlled setting to replicate missing data, we vary the peak threshold at which range may be rejected from consideration. We note that this is quite common for scanners, since the accuracy of the scanner suffers when the angle at which the laser line-of-sight and the normal becomes large, and the preferred option may be to reject range rather than accept outliers.

Similar to the previous experiment, here, we fix the number of scans and camera positions, and use no additive noise, in order to isolate missing data as the primary challenge in the input. We then vary the peak threshold at which to reject samples from 0.8 to 0.4, where 1 is the expected peak. We have used the bumpy sphere and mailbox shapes, in order to observe the behavior of these algorithms in the presence of missing data on both smooth and sharp features.

See Figure 3.15 for plots of the experiment. We find that all of the indicator function methods perform quite well across both shapes, with the notable exception of Wavelet failing to converge to the limit surface as missing data decreases. We credit the robustness of indicator function methods to being global methods which do not attempt to fit shape functions.

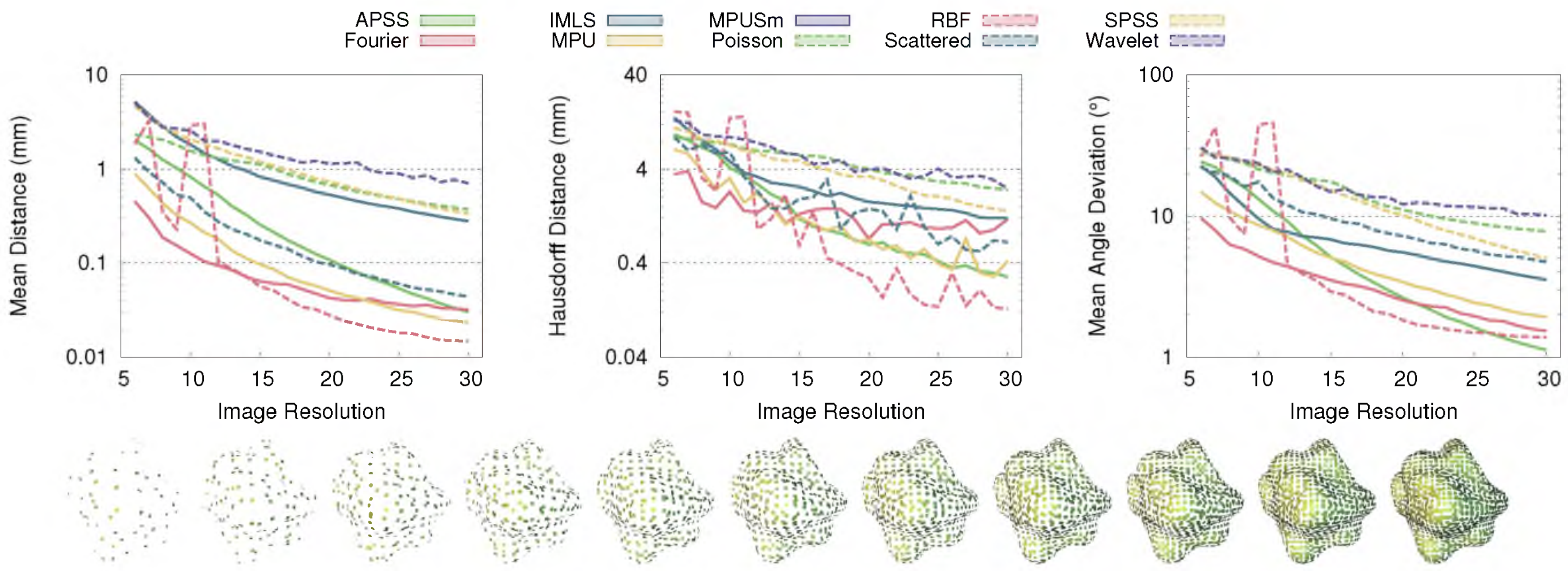


Fig. 3.14. Plots for the sparsity experiment, where we have sampled the bumpy sphere in increasing image resolution. The bottom row depicts a subset of these point clouds in decreasing sparsity. This experiment demonstrates how well these algorithms infer the surface from a sparse sampling.

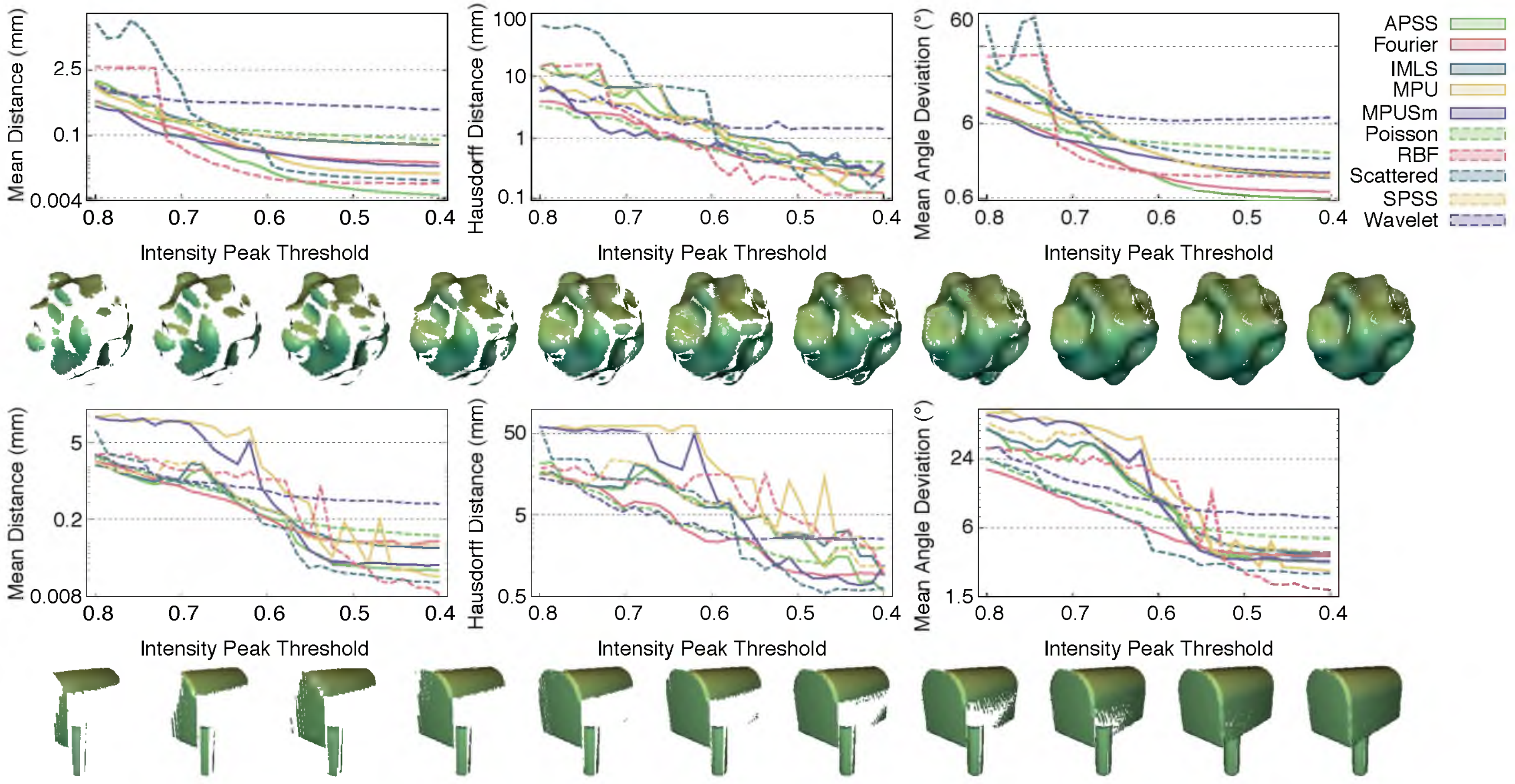


Fig. 3.15. Plots for the missing data experiments on the bumpy sphere (top row) and mailbox (bottom row). We generate missing data by varying the peak intensity threshold at which range is rejected. Note the differences in performance between the shape with smooth features and the shape with sharp features, as missing data are varied.

Indeed, methods which fit shape functions have rather erratic behavior, particularly in the mailbox shape. MPU, MPUSm, and RBF are quite unstable, producing spurious surface sheets as missing data are introduced. When a neighborhood of a sharp feature, namely an edge, is sampled on one side and not on the other, shape functions of this kind are expected to be produced. As missing data increase, the samples used for shape fitting change, which results in spurious surface sheets only occasionally appearing. This variability in the points used for shape fitting is the cause of the inconsistencies found across MPU, MPUsm, RBF, and the PSS methods as missing data increase.

Scattered and the PSS methods tend to produce holes in the presence of large missing data, due to an insufficient number of samples in these areas. These missing data are the cause for their unstable behavior in the mailbox shape, as more missing data are introduced.

3.4.4 Noise

Finally, we consider how robust reconstruction algorithms are to noise in the range data. We consider two scan parameters which have a significant impact on noise: noise magnitude and laser frustum field of view. The effect of noise magnitude is fairly clear; however, we note that the thickness of the laser plays a significant impact on outliers. The thicker the laser, the more difficult peak detection becomes at depth discontinuities, resulting in outliers.

To this end, we have taken the spiral shape and sampled it under varying noise magnitudes, and varying laser thickness. We sufficiently sample it so that missing data or sparsity are not an issue, and compute normals directly from the points, allowing for improper orientation if direction propagation is incorrect. For each algorithm and each point cloud, we also manually set the parameters to perform best, considering the scale of the noise. For the PSS and indicator function methods, such parameter settings are quite intuitive as they are based on sampling density bandwidths. However, for all other methods, a maximum error tolerance effectively determines the amount of smoothing performed, which can be quite sensitive.

See Figure 3.16 for plots of the noise experiments. Note that Fourier and Poisson, in terms of all error metrics, are quite robust in the presence of noise. This is likely due to the global nature of these methods, where smoothing the data is a natural consequence. As observed by its large variance, RBF performs rather poorly in the presence of noise. Indeed, the necessity to produce dipoles for RBF becomes especially problematic in the presence of noise and outliers.

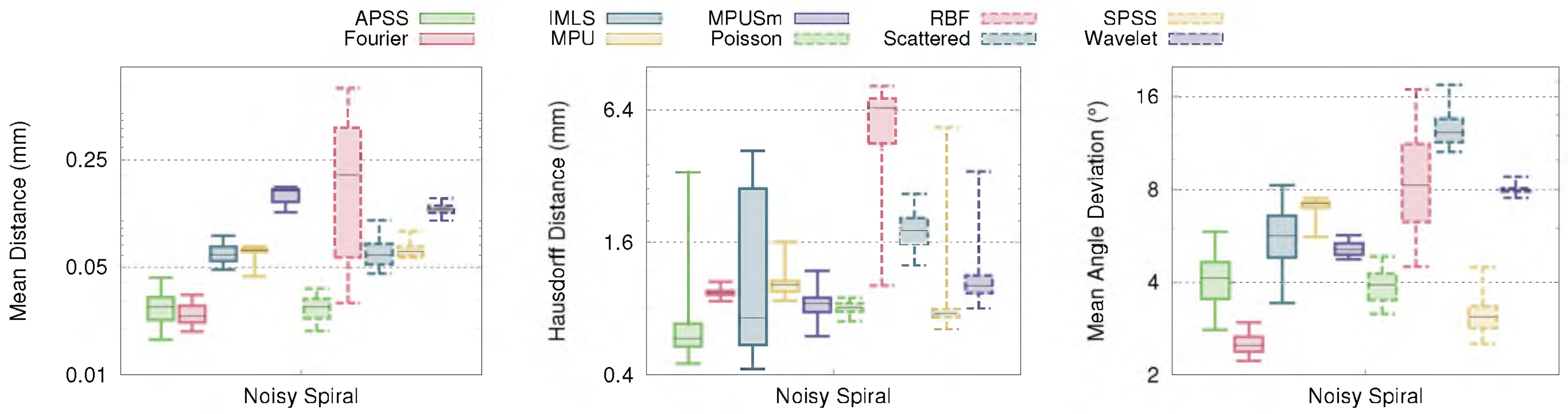


Fig. 3.16. Noise experiments for the spiral shape. Here, we vary noise level and laser thickness, and aggregate this into distributions. A small variance in a distribution is a good indication of robustness to noise.

We observe that MPU and MPUsm are somewhat robust in the presence of noise given their small variance in Hausdorff distance, though interestingly, we see significant differences between them in the two different distance measures. The smoothing performed via MPUSm tends to expand the surface outward, resulting in poor mean distance, yet it never strays too far from the surface, hence its good behavior in terms of Hausdorff distance.

The PSS methods all tend to smooth out noise and remain robust to outliers. However, far away from the surface, their behavior tends to be quite poor; see Table 3.3. They tend to produce many extraneous connected components, as well as boundary components.

3.5 Discussion

Our small-scale experiments tend to correlate well with the results of the error distribution experiments. For instance, the unstable behavior of RBF in the presence of sparse and missing data manifests itself in its unstable behavior across the gargoyle model, which is particularly difficult to adequately sample due its numerous concavities. Likewise, the behavior of MPU and to a lesser extent MPUSm in the presence of missing data on the mailbox correlates with their large variance in the Anchor and Daratech, indicative of the fact that they have trouble reconstructing sharp features. Conversely, we see that the stable behavior of Fourier in the small-scale experiments correlates well with its relatively small variance in the distribution plots.

Our experiments point toward a number of deficiencies in the state of surface reconstruction. Our results demonstrate the remarkable robustness of methods based on computation of the indicator functions, yet these methods tend to oversmooth the data, reflected in

Table 3.3. Additional information for the noisy spiral experiments, averaged across all point clouds.

algorithm	comps	bndry	manifold	genus	time
apss	221.60	0.71	1.00	0.00	50.59
fourier	1.00	0.00	1.00	0.00	27.24
imls	193.16	4.76	1.00	0.00	48.62
mpu	1.20	0.00	1.00	0.00	7.13
mpusmooth	1.08	0.06	1.00	0.00	23.08
poisson	1.00	0.00	1.00	0.00	30.90
rbf	12.48	4.69	0.92	0.30	18.90
scattered	1.08	0.00	1.00	0.44	3.11
spss	257.20	1.13	1.00	0.00	48.18
wavelet	1.00	0.00	1.00	0.00	2.26

their poor performance in mean distance across complex shapes. Developing an algorithm based on the indicator function which does not oversmooth the data would be very useful. Conversely, although MLS methods perform rather well in terms of mean and Hausdorff distance across the complex shapes, they demonstrate poor far-field behavior. We think that combining MLS methods with global constraints of some nature may rectify these issues.

Our benchmark should also prove to be useful for recent methods which resample point clouds with large missing data [Tagliasacchi et al. 2009; Cao et al. 2010; Shalom et al. 2010]. Although we have produced such point clouds in order to test robustness, it would be interesting to see how well these more recent resampling methods perform quantitatively.

All told, our benchmark consists of 351 point clouds across eight shapes, providing rich data for surface reconstruction developers. For our first set of experiments, we have 48 point clouds for each shape. Over 10 algorithms, this amounts to a total of 2400 different reconstruction outputs, and over both distance and normal correspondences, we have a total of 4800 correspondence mappings. We think that this construction of a distribution of point clouds for a given shape could be used in other areas, for instance potentially *learning* surface reconstruction, by using the point clouds and ground truth data as training data.

3.5.1 Limitations

While the surfaces in our benchmark cover a broad range of shapes, they are by no means exhaustive. As surface reconstruction becomes more specialized, such as the reconstruction of large-scale architectural buildings [Nan et al. 2010], we envision our benchmark to expand to these specific forms of surfaces. Our implicit shape representation should easily be able to accommodate other types of shapes.

Although we have generated a large variety of point cloud data with our sampling scheme, we are keeping fixed certain settings which may be worth further exploration. For instance, we assume a diffuse BRDF in the scanning simulation, where it may be interesting to consider different forms of surface reflectance, and even spatially-varying BRDFs.

Though laser-based optical triangulation scanners are quite popular, other forms of scanning may be worth simulating in order to replicate different acquisition artifacts. For instance, time-of-flight scanners contain a very distinct random noise profile and systematic error due to the emission of infrared light into the scene. Multiview stereo methods are known to produce geometry containing significant noise, as object texture, material, and lighting tend to play a more significant role in such passive methods, compared to active methods like laser-based scanning. An accurate simulation of multiview stereo would necessitate a highly photorealistic renderer, where the large space of parameters (for instance, BRDF, subsurface scattering, and lack of texture) would be interesting to explore, effectively extending the rather controlled environment of Seitz et al. [2006].

3.6 Summary

We have presented a benchmark for surface reconstruction. We have proposed novel methods for modeling and sampling smooth and piecewise-smooth shapes, as well as evaluation of reconstruction algorithms. Our extensive experiments demonstrate the quantitative behavior of many state-of-the-art surface reconstruction algorithms across a diverse range of realistic point clouds. The experiments are useful in several ways: they illustrate which algorithms are best suited for specific types of data, point out deficiencies in the current state of surface reconstruction, and indicate future work for reconstruction.

In particular, the results of the benchmark have informed many of the subsequent chapters in this thesis. The benchmark results highlight which relevant problems to focus on in processing point clouds, namely noise and missing data. Since quite often the reason for failure in existing reconstruction methods is the use of smoothness priors in the ambient space of the point cloud, this has motivated the subsequent work in developing shape analysis methods *directly* on the point cloud, rather than the ambient space. Lastly, the benchmark provides for an easy mechanism of generating point clouds and obtaining quantitative results, which we use throughout the thesis.

CHAPTER 4

HARMONIC POINT CLOUD ORIENTATION

One of the key requirements of an approximating reconstruction method is the consistent orientation of normals defined at every point. That is, for a given point cloud, at each point, we must assign the direction which the normal should point – typically, we want to assign this direction such that the normal points outside of the surface. This task is challenging for the types of shapes and point clouds that we studied in our benchmark, namely point clouds with sharp features, nearby surface sheets, noise, undersampling, and missing data.

Existing methods are sensitive to these imperfections, as the direction to choose for a normal is based on local geometric properties. Typically, such approaches use measures such as normal direction deviation [Hoppe et al. 1992; Huang et al. 2009] or curvature estimation [Guennebaud and Gross 2007], coupled with normal direction propagation [Hoppe et al. 1992] to determine orientation. An issue with these approaches is that if one or several normal directions are incorrect, then the subsequent propagation will result in large, contiguous regions of the point cloud containing the incorrect orientation. This is particularly detrimental to surface reconstruction algorithms, as they will begin to interpret the inside as the outside.

In this chapter, we propose a new method for normal orientation estimation which instead considers the problem from a more global perspective. We use globally smooth functions defined directly on the point cloud, which are inherently insensitive to data characteristics and imperfections on the point cloud. Specifically, we consider harmonic functions defined on point clouds – functions which lie in the kernel of the Laplace-Beltrami operator. Such functions are well-known to be extremely insensitive to the data characterization attributed to acquired 3D point clouds [Dey et al. 2010].

Harmonic functions are used for normal orientation by considering their gradient fields, restricted to the local tangent plane of each point. From these gradient fields, we cast the problem of normal orientation as an assignment of cross-product orderings between gradients locally at each point. Away from critical points, two functions are sufficient to fully determine the orientation, if the functions never intersect [Edelsbrunner et al. 2004]. This is analogous to the construction of a global parameterization over the point cloud, where for a pair of functions $u : \mathbb{R}^3 \rightarrow \mathbb{R}$, $v : \mathbb{R}^3 \rightarrow \mathbb{R}$, the orientation at a point $\mathbf{p}_i$ is simply defined as $\nabla u(\mathbf{p}_i) \otimes \nabla v(\mathbf{p}_i)$, where $\otimes$ denotes the vector cross product.

In practice, we are faced with point clouds with the aforementioned imperfections and nontrivial topology, which results in harmonic functions containing critical points. The presence of critical points, and in the discrete setting *critical regions*, changes how the gradient behaves locally. In such cases, it is unsuitable to determine only two functions. Instead, we produce a set of harmonic functions defined over the point cloud, resulting in a set of gradient fields. In order to avoid regions where a critical point may exist and to account for imperfections in the data, we assign a pair of gradient fields to each point. The assigned pairing is the two most correlated functions with respect to the flow of the gradient within the neighborhood of the point. Figure 4.1 shows how these local pairings of gradient fields and harmonic functions are used to determine a normal's orientation.

To summarize, the contributions of our method are as follows:

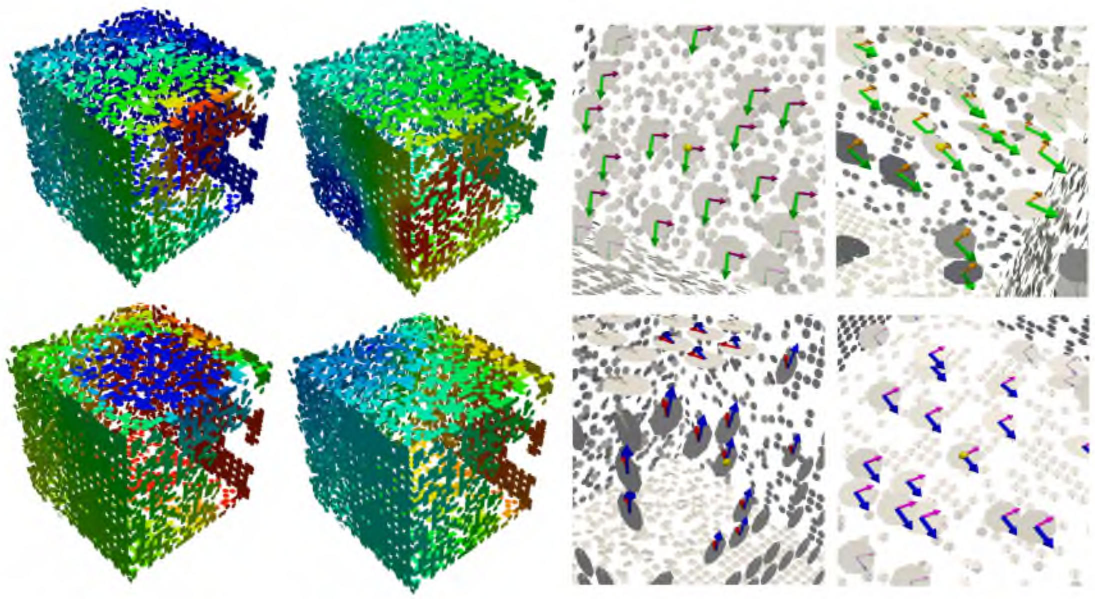


Fig. 4.1. Consistent normal orientation using the gradient fields of multiple harmonic functions. Left: Four harmonic functions defined on a point cloud of a cube, color mapped from blue to red to indicate increasing function value. Right: Paired gradient fields for four points (denoted by a yellow ball) and their neighborhoods (bold vectors). The gradient field of each function is denoted by a different colored vector. Given an ordered pairing, the normal orientation for a point is simply: $\nabla u(\mathbf{p}_i) \otimes \nabla v(\mathbf{p}_i)$, where ∇u is indicated by the longer of the two vectors.

– We demonstrate the utility of harmonic functions for robust orientation of point clouds.

– We frame the problem of orientation propagation as the assignment of cross-product orderings across smooth gradient fields.

– We show the benefit of our method by applying it to the problem of surface reconstruction for challenging point cloud data.

4.1 Harmonic Function Generation

Key to our approach is the generation of globally smooth functions defined directly on the point cloud. Harmonic functions are a natural candidate, as these functions are smooth by construction. Additionally, the formulation of the discrete Laplace-Beltrami operator [Belkin et al. 2008 2009] allows for the construction of the operator for both surface meshes and point clouds.

To this end, we employ the discretization of Luo et al. [2009] for defining the Laplace-Beltrami operator directly on a point cloud $P = \{\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_n\}$, such that the discrete Laplace Operator L is an $n \times n$ matrix

$$\mathbf{L}[i][j] = \begin{cases} G(i,j) & i \neq j \\ G(i,i) - \sum_{j=1}^{n} G(i,j) & \text{Otherwise} \end{cases} \tag{4.1}$$

where $G(i,j) = \frac{1}{4\pi h^2} e^{-\frac{||p_i - p_j||^2}{4h}}$ and h determines the kernel support size. Note, we deviate slightly from Luo et al. [2009] in the formulation of L and do not weight each entry by the Voronoi area. For noisy, nonuniform sampled data, the Voronoi cell area for a given point may not always be well-defined.

We determine h by estimating the average sample density over all points. At each point $\mathbf{p}_i$, let $r(\mathbf{p}_i)$ be the radius of the enclosing sphere of the k-nearest neighbors of $\mathbf{p}_i$, $N(\mathbf{p}_i)$, given by $r(\mathbf{p}_i) = \max_{\mathbf{q} \in N(\mathbf{p}_i)} ||\mathbf{p}_i - \mathbf{q}||$ [Pauly et al. 2008]. The average sample density over all points, h, is then given by:

$$h = \frac{1}{|P|} \sum_{i \in P} r(\mathbf{p}_i) \tag{4.2}$$

In order to ensure sufficient coverage in the presence of nonuniformly sampled data while also ensuring that L remains sparse, values of G less than s are set to 0, where s is given by $s = \min G(i,j)$ for all points p_i and their k-nearest neighbors p_j. In the presence of outliers, however, s can be very small, leading to only a few nonzero entries in each row of L. For these cases, we set s empirically.

Given that L is positive semidefinite and symmetric, we generate harmonic functions by first prescribing Dirichlet boundary conditions at two points $\mathbf{p}_i$ and $\mathbf{p}_j$, and then solve for the harmonic function u whose Laplacian is zero,

$$\begin{gathered} Lu = 0 \\ u(\mathbf{p}_i) = -1 \quad u(\mathbf{p}_j) = 1 \end{gathered} \tag{4.3}$$

We seek a collection of harmonic functions such that the following two conditions are satisfied:

– Critical points are distributed throughout the point cloud.

– Gradient fields are as-orthogonal-as-possible.

In general, developing heuristics for the placement of Dirichlet boundary constraints such that harmonic functions satisfy these conditions is highly nontrivial. Heuristics based on maximizing L_2 distance in $\mathbb{R}^3$, farthest point sampling, or some intrinsic measure will be sensitive to any noise and sampling artifacts in the point cloud data. This can introduce significant bias in the generation of gradient fields, such that large regions of the point cloud may contain near-coincident gradient fields, even if a large number of functions are used.

We instead solve for the harmonic function u by prescribing boundary constraints at two random points such that one takes on the global minimum and the other the global maximum. In this way, we can obtain a set of harmonic functions $U = \{u_1, u_2, ..., u_m\}$ by simply choosing the random placement for boundary constraints. Doing so minimizes any such bias potentially introduced by some heuristic measure, whereby simply increasing the number of functions to use increases the likelihood of satisfactory gradient fields. Figure 4.2 shows two harmonic functions with different boundary constraints.

Given that we must generate a potentially large number of harmonic functions, an efficient means of solving for Equation 4.3 is essential. Thus, rather than solving Equation 4.3 exactly under hard constraints (for instance through Lagrange multipliers), we use the method of Xu et al. [2009] and instead prescribe soft constraints,

$$(L + P)\, u = Pb \tag{4.4}$$

where P is a penalty matrix containing large values on the diagonal entries corresponding to boundary constraints i and j and zero everywhere else. Similarly, we assign the corresponding constraint values to the i^{th} and j^{th} entries of b.

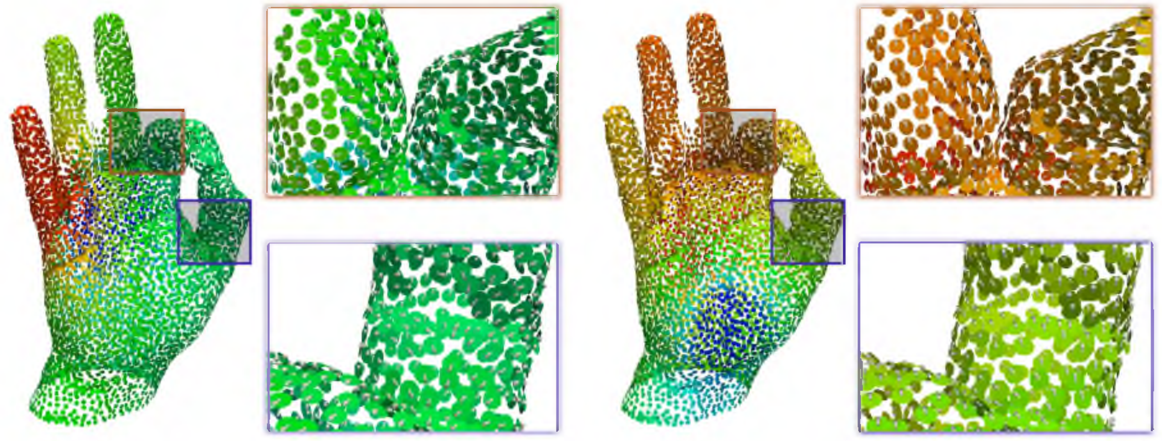


Fig. 4.2. Two different harmonic functions defined on a point cloud, and their resulting gradient fields.

By constructing the Cholesky factorization $L = GG^T$ with respect to the boundary constraints only once, we may then subsequently factorize L efficiently following the approach of Xu et al. [2009]. Namely, we forego the need to repeatedly factorize L for each new set of constraints. Instead, we use the Cholesky supernodal algorithm by updating the factorization with new boundary constraints, followed by downdating with respect to the previous boundary constraints. In this way, we have an efficient means of generating multiple harmonic functions.

4.2 Gradient Pairing

To determine normal orientation, we use the set of harmonic functions $\{u_1, u_2, ..., u_m\}$ and operate on their corresponding gradient fields $G = \{\mathbf{g}_1, \mathbf{g}_2, ..., \mathbf{g}_m\}$. Restricting the gradient fields to lie in the tangent plane of each point $\mathbf{p} \in P$ enables the consistent comparison of different gradient fields at $\mathbf{p}$. The normals are estimated for simplicity using PCA. At first glance, the use of PCA may seem to restrict the robustness of our method as in related approaches. However, we will later show that our method is extremely insensitive to noise in the estimated tangent planes.

For a function u_i and point $\mathbf{p}$, its gradient $\mathbf{g}_i$ is found numerically by using a first-order Taylor series expansion of u_i about $\mathbf{p}$

$$u_i(\mathbf{p} + \mathbf{s}) \approx u_i(\mathbf{p}) + \mathbf{g}_i{}^T\mathbf{s}. \tag{4.5}$$

From this expansion, we take N to be the set of k-nearest neighbors of $\mathbf{p}$ and find the gradient by minimizing

$$\underset{\mathbf{g}_i}{\operatorname{argmin}} \sum_{\mathbf{q} \in N} ||u_i(\mathbf{p}) + \mathbf{g}_i^T(\hat{\mathbf{q}} - \mathbf{p}) - u_i(\mathbf{q})||^2 \tag{4.6}$$

where $\hat{\mathbf{q}}$ is $\mathbf{q}$ projected to the tangent plane of $\mathbf{p}$. The optimal gradient is found by solving the normal equation associated with the least-squares system of (4.6). Each gradient is then normalized. In Figure 4.2, we show the gradient fields associated with two harmonic functions.

4.2.1 Order Assignment

Given the set of gradient fields defined at each point, we would like to obtain some relationship or ordering between the respective functions locally at each point such that the ordering can be used to propagate a given orientation to all points.

Key to propagating a consistent normal to all points is the assignment of gradient field pairings for each point $\mathbf{p} \in P$. We seek an unordered pair of gradient fields (i, j) for $\mathbf{p}$, where $\mathbf{g}_i, \mathbf{g}_j \in G$, such that the gradients in the neighborhood of $\mathbf{p}$ are as-orthogonal-as-possible and sufficiently far away from critical points – locations where the gradient vanishes. The inner product between gradient fields at a single point is a natural measure of orthogonality, but is unreliable for several reasons. First, for noisy tangent planes, this can indeed be a misleading measure. Secondly, in the vicinity of a critical point, the gradient field will exhibit nonzero divergence and hence may be unsuitable for consistent pairing. Figure 4.3 illustrates the importance of proper pairing of gradient fields.

To robustly determine optimal pairings, we consider the statistics of inner products of $\mathbf{p}$ and its local neighborhood. The following procedure is repeated for all points in P. We define η_p, as the set of points which contains point $\mathbf{p}$ and its k-nearest neighbors, $\{\mathbf{q}_1, \ldots, \mathbf{q}_K\}$. For a point $\mathbf{q}$ in η_p, we define

$$e_{ij}(\mathbf{q}) = | < \mathbf{g}_i(\mathbf{q}),\ \mathbf{g}_j(\mathbf{q}) > | \tag{4.7}$$

as our (absolute) inner product measure. The measure in (4.7) is defined for all points in η_p to obtain a distribution of inner products. For each possible gradient pairing (i, j), the mean $\mu_{ij}(\eta_p)$ and variance $\sigma^2_{ij}(\eta_p)$ of our inner product measure is computed on the distribution η_p.

The mean $\mu_{ij}(\eta_p)$ and variance $\sigma^2_{ij}(\eta_p)$ of this distribution precisely measures the two different quantities of interest. A small mean indicates orthogonality, whereas a small variance is a robust measure of smoothness and small divergence since it indicates that the two gradient fields agree in a local neighborhood.

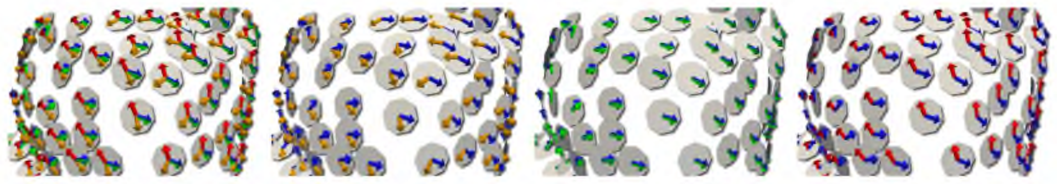


Fig. 4.3. Selection of gradient fields for consistent orientation. In (a) we show a set of four gradient fields in a local neighborhood. In (b), the orange-colored field contains a saddle point, while (c) shows nearly aligned gradients, both being rather unstable. In (d), the proper fields are selected.

Using these local statistics, we define a metric indicating the quality of a gradient pairing (i, j) at a point $\mathbf{p}$:

$$E_{ij}(\eta_p) = \alpha\sigma^2_{ij}(\eta_p) + \beta\mu_{ij}(\eta_p) \tag{4.8}$$

where α and β define the importance of the variance and mean and are set to 0.1 and 0.9, respectively. By favoring gradient field pairings which have a lower mean (more-orthogonal) than those with a lower variance (smooth), we can increase the likelihood that two fields do not cross in a neighborhood away from a critical point. Near a critical point, we observe that such regions are typically associated with a high variance and low orthogonality, as shown in Figure 4.3 (b). Using Equation 4.8, the optimal pairing $(\hat{i}, \hat{j})$ is sought which minimizes the metric over all possible combinations of gradient fields (i, j):

$$(\hat{i}, \hat{j}) = \underset{i,j}{\operatorname{argmin}} \quad E_{ij}(\eta_p). \tag{4.9}$$

Our method of selecting gradient functions does not require that the tangent planes between two points ever be compared directly. Instead, the normal orientation is estimated by comparing paired gradients within a given tangent plane and computing vector cross products. As a result, our method is robust to sharp features, missing data, nonuniform sampling, etc., provided the gradients are smooth in the neighborhood. Thus, we obtain a consistent metric that is decoupled from the geometric artifacts in a given local neighborhood.

4.3 Gradient-Based Normal Propagation

From the optimal pairings, we must now determine the correct ordering for each unordered pair of gradient functions for all points in the set. Once the ordering is found, the normal vector can be assigned for each point in the set, without the need to explicitly compare tangent planes. This assignment process is illustrated in Figure 4.4. Here, we

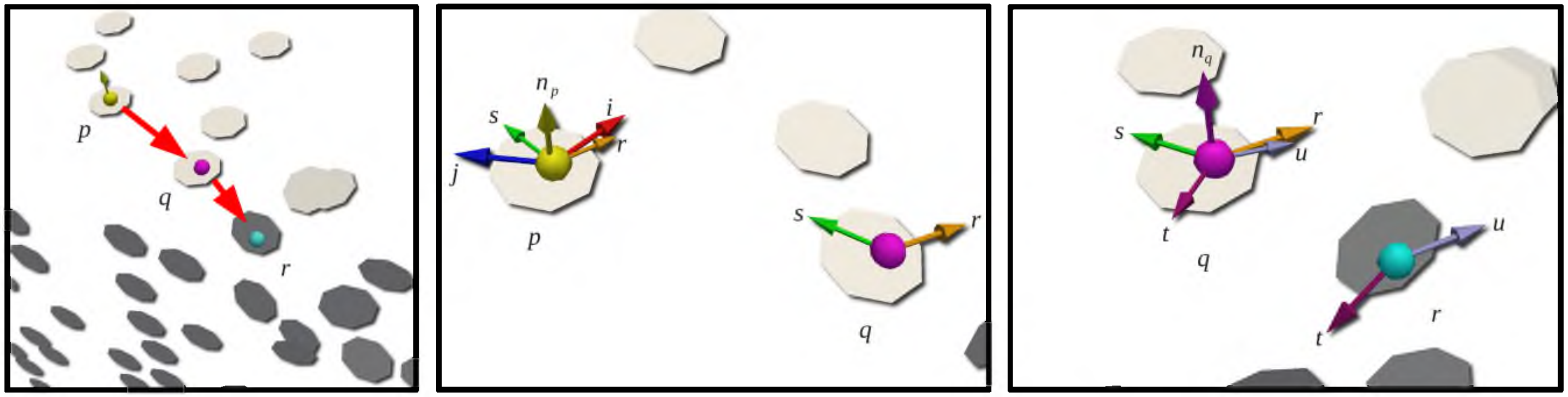


Fig. 4.4. Normal Assignment. Left: Propagation of the normal at p to q, and r. Middle: Ordering of the function pair (s, r) at q such that $\mathbf{g_r}(p) \otimes \mathbf{g_s}(p) = \mathbf{n}_p$. Right: Normal propagation over an edge. Our method is insensitive to changes in local curvature since gradient comparisons are performed w.r.t a single point's tangent plane.

show the propagation of a normal from $\mathbf{p}$ to $\mathbf{q}$ and $\mathbf{r}$ through the ordering of the optimal gradient pairs, $(\mathbf{g}_i, \mathbf{g}_j)$, $(\mathbf{g}_s, \mathbf{g}_r)$, and $(\mathbf{g}_t, \mathbf{g}_u)$, respectively.

We propagate the normal orientations over the point set by leveraging the global nature of the gradient functions. Specifically, a point $\mathbf{p}$ is used as the initial starting point and it is assumed its normal, $\mathbf{n}_p$ is correctly oriented. We then construct a minimum spanning tree (MST) and traverse the edges in the tree to propagate the given normal direction. We use the ordering of the paired gradients with respect to the known normal orientation at a point $\mathbf{p}$ to transfer the orientation to a neighboring point $\mathbf{q}$ by transferring the ordering of the gradients at $\mathbf{p}$ to $\mathbf{q}$.

Let the optimal gradient pairings for $\mathbf{p}$ and neighbor $\mathbf{q}$ be denoted (i, j) and (s, r), respectively. The cost of propagating a normal from $\mathbf{p}$ to $\mathbf{q}$ is evaluated in terms of the quality of the gradient pairing (i, j) at $\mathbf{q}$ and similarly the quality of pairing (s, r) at $\mathbf{p}$. The cost of propagating a normal from $\mathbf{p}$ to neighbor $\mathbf{q}$ is then given by

$$\text{cost}\left(\mathbf{p}, \mathbf{q}\right) = E_{ij}(\eta_q) + E_{sr}(\eta_p) \tag{4.10}$$

where $E_{ij}(\eta_q)$ and $E_{sr}(\eta_p)$ is the objective function of Equation 4.8 evaluated at the optimal gradient pair (i, j) at the point $\mathbf{q}$ and gradient pair (s, r) at the point $\mathbf{p}$, respectively. In this way, low-cost edges correspond to neighboring points whose associated gradient pairings vary smoothly with respect to the neighborhoods of both points.

Given the MST and cost function for traversing its edges, the normal is propagated from $\mathbf{p}$ to $\mathbf{q}$ by ordering the gradients at $\mathbf{q}$ to be consistent with the gradients and normal at $\mathbf{p}$. The orientation of the normal vector at $\mathbf{q}$ is found by computing the vector cross product of the gradient vectors $\mathbf{g}_s$ and $\mathbf{g}_r$ at $\mathbf{p}$ and then taking the dot product with the normal vector at point $\mathbf{p}$. If the dot product is one, then the ordering $(\mathbf{g}_s,\ \mathbf{g}_r)$ at point $\mathbf{q}$ is correct; otherwise, the ordering is switched. This criteria can be expressed as

$$n_q = \begin{cases} \mathbf{g}_s\left(\mathbf{q}\right) \otimes \mathbf{g}_r\left(\mathbf{q}\right) & < \mathbf{g}_s\left(\mathbf{p}\right) \otimes \mathbf{g}_r\left(\mathbf{p}\right), n_p > \; = 1 \\ \mathbf{g}_r\left(\mathbf{q}\right) \otimes \mathbf{g}_s\left(\mathbf{q}\right) & \text{otherwise} \end{cases} . \tag{4.11}$$

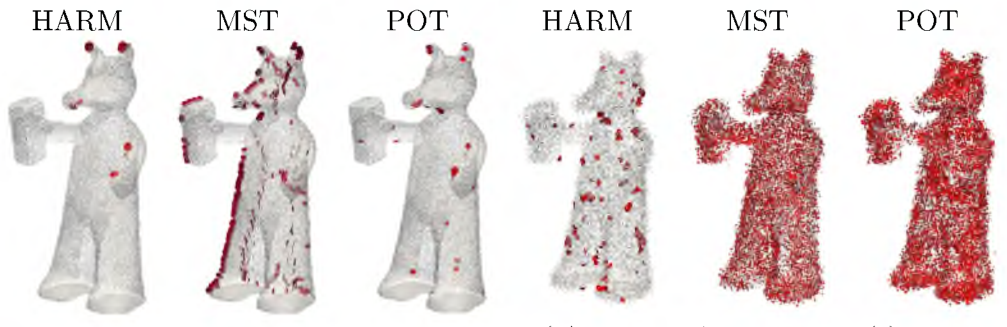
This process is illustrated for three points in Figure 4.4.

4.4 Results

We demonstrate our algorithm's ability to assign normal orientations under a wide variety of different sampling characteristics, ranging from sharp features, thin surface sheets, noise, misalignment, and missing data. In particular, we use the benchmark from Chapter 3 to generate realistic point clouds. Since we have ground truth shapes, we have a means of obtaining quantitatitve comparisons, which we measure as the ratio of incorrect normal assignments to the total number of sample points. In addition, we also perform surface reconstruction using the estimated normal orientation information and compare the reconstructed results. We have compared our algorithm (termed HARM) to the classic orientation minimum spanning tree (MST) approach of Hoppe et al. [1992], herein termed MST and to the recent work of Liu and Wang [2010], herein termed POT.

For all results, we used a fixed neighborhood of 15 point samples during all aspects of the algorithm. This includes normal orientation using PCA, to the pairing and ordering of gradient fields. In contrast, previous MST orientation tend to be sensitive to neighborhood size. The number of functions to evaluate varies across objects. As a rule of thumb, we chose to evaluate more randomly constrained functions than generally necessary to ensure that the set of functions provided a sufficient gradient field covering. For the Quasimoto, Anchor, and Daratech models, the total number of functions computed were 22, 15, and 15, respectively.

Figure 4.5 shows a comparison of our method with previous works for a smooth shape. The left side contains misalignment artifacts and missing data. Observe that MST fails in the region of misalignment as indicated by the string of inconsistent normals on the left side. The POT method tends to fail in areas of missing data, where the lack of data results in a poor spherical covering. In contrast, our algorithm succeeds in having the fewest inconsistent normals, failing only in areas of critical regions. Perhaps even more drastic a comparison is when noise is added to the model, shown on the right side. We introduce Gaussian noise with a variance of 1% of the bounding box diagonal to the 3D point cloud



(a) 15 (b) 435 (c) 40 (d) 171 (e) 12,509 (f) 13,181

Fig. 4.5. We compare normal orientation on the Quasimoto model for our method (termed HARM), MST, and POT. A red splat marks an incorrect orientation. In (a), (b), and (c) we compare against a synthetically scanned point cloud. In (d), (e), and (f), we have added 1% Gaussian noise to the point locations. The bottom row shows the number of incorrectly oriented normals.

while maintaining the original normal direction. Positional noise manifests itself in normal noise, where local methods may easily fail. We see that our method is several orders of magnitude better, illustrating our intrinsic resilience to noise in the normals.

Next, we compare to a shape containing sharp features and a modest amount of noise; see Figure 4.6. While MST is competitive with our method, we see that POT has issues in the presence of sharp features. Indeed, the control mesh generated via their spherical covering may be problematic at noisy, sharp features. As harmonic functions are insensitive to sharp features and noise, our method has little issue in handling them.

Finally, we consider a shape which contains sharp features and nearby surface sheets; see Figure 4.7. The presence of thin surface sheets poses significant challenges for local methods, as shown by the incorrect orientation by MST of entire surface patches. The harmonic functions we generate remain robust to thin surface sheets, producing few inconsistencies.

To demonstrate the impact of improper orientation, we selected another scan of the Daratech model, oriented the normals, and ran Poisson Surface Reconstruction [Kazhdan et al. 2006] on the result. See Figure 4.8 for the results. We clearly see the impact of inverting normals on a surface sheet, where MST and POT fail in different instances. Our method produces proper normals, giving us a more accurate reconstruction.

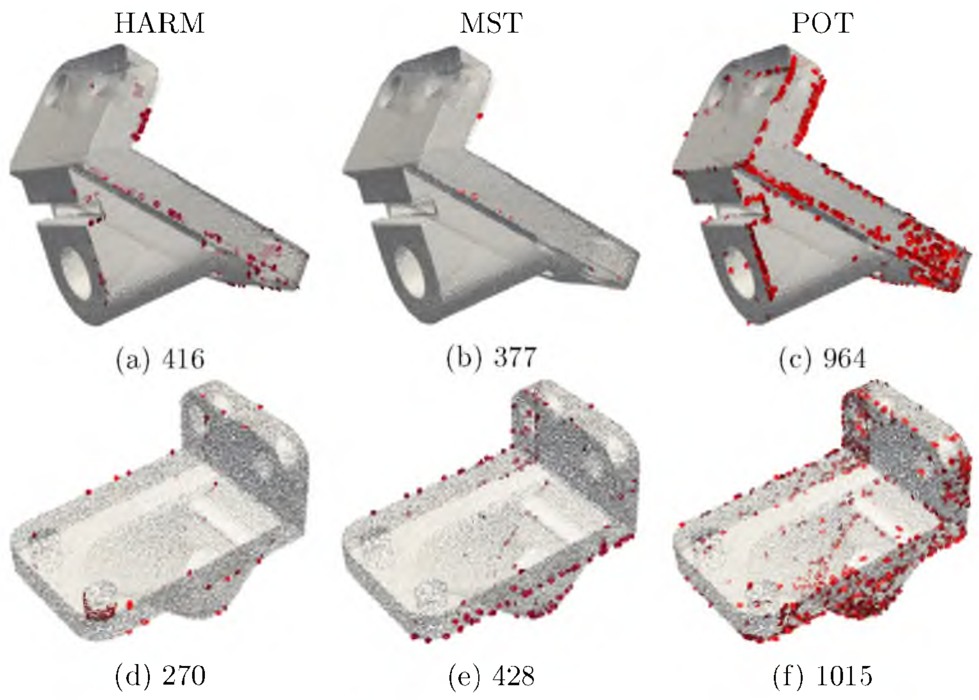


Fig. 4.6. Incorrect normal orientations for two synthetic scans of the Anchor model for the HARM, POT, MST methods. Red splats mark incorrect orientations.

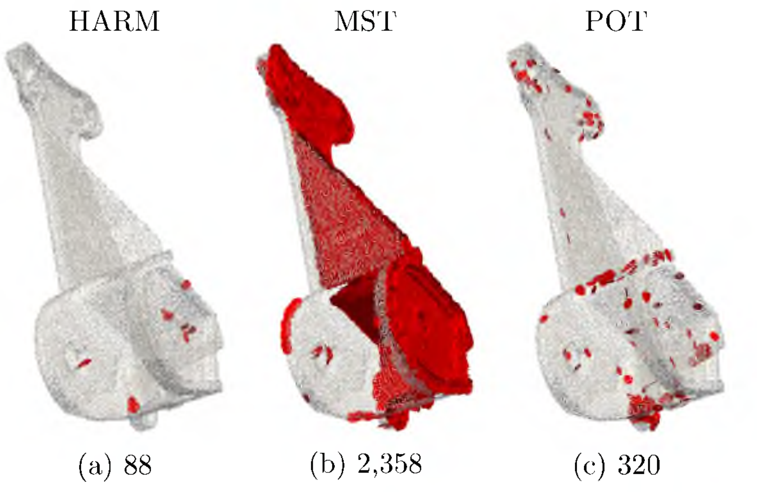


Fig. 4.7. Incorrect normal orientations for two synthetic scans of the Daratech model for the HARM, POT, MST methods. A red splat indicates an incorrect orientation.

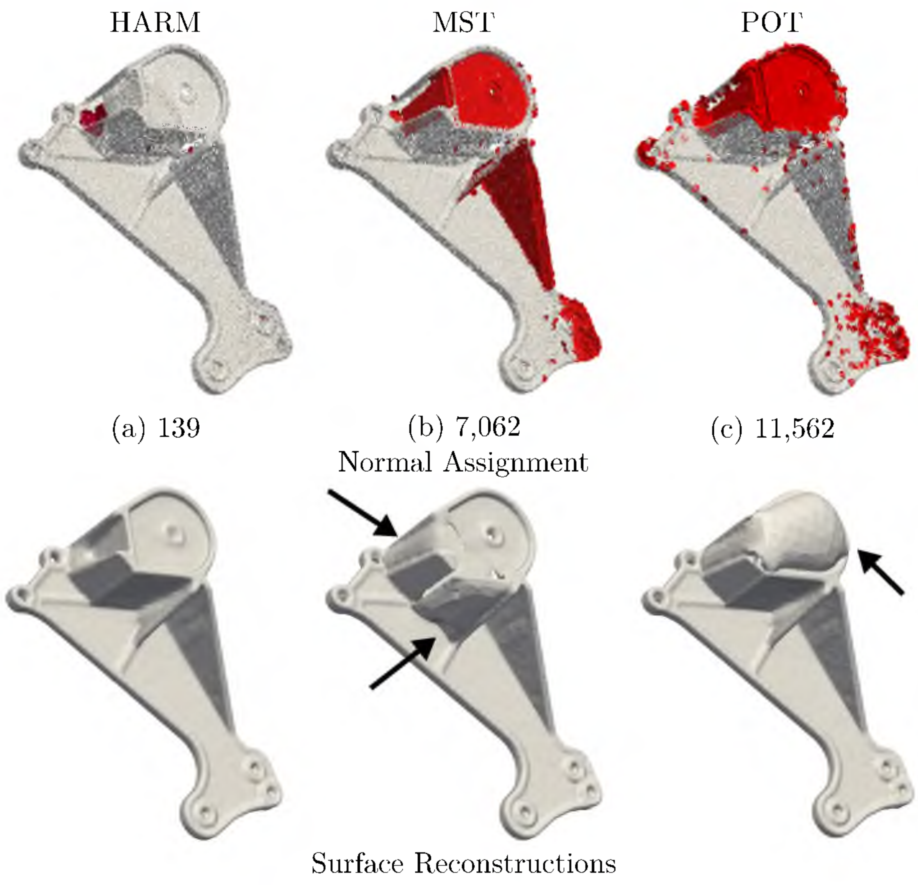


Fig. 4.8. We show the impact of normal orientation on surface reconstruction. Top: Normal estimation results for the Daratech model using the HARM, MST, and POT methods. Bottom: Poisson surface reconstruction using the estimated oriented normals.

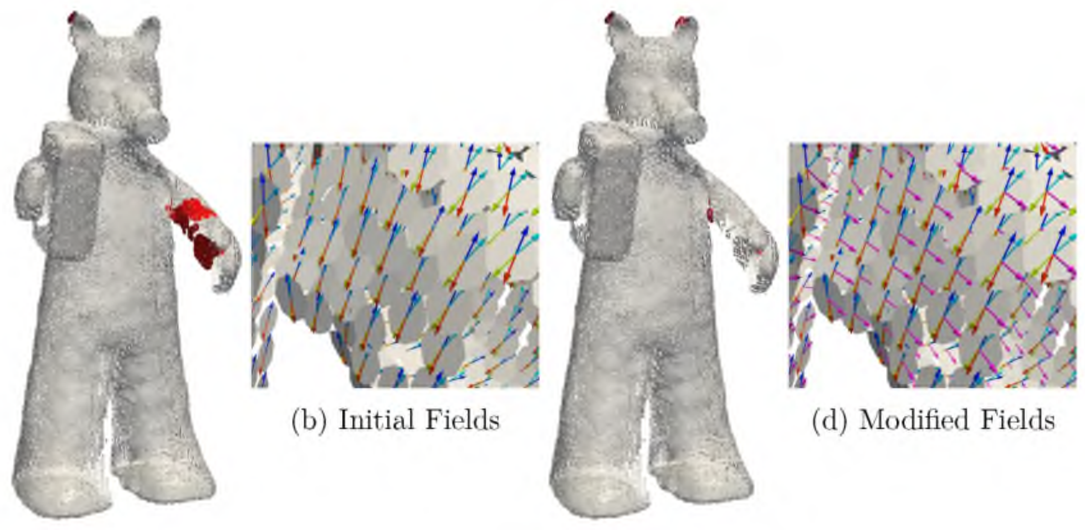
4.5 Discussion and Limitations

The strength of our method lies in its ability to orient normals by considering the global gradient flows of a set of harmonic functions. When errors do occur, they can often be traced back to the construction of the randomly generated function set.

The set of Dirichlet functions should be constructed such that for each point, there exists at least two gradients which vary smoothly and approach orthogonality in the neighborhood of every point. While increasing the number of functions can mitigate bad pairings, for certain surfaces such as highly tubular structures, some sample points may not fully satisfy this requirement under the randomly generated Dirichlet constraints. However, this can be addressed by ensuring satisfactory coverage through the placement of additional user supplied constraints interactively, as in Xu et al. [2009]. Figure 4.9 shows how additional variability can be added by placing constraints near the region of interest.

Alternatively, an automated approach may be developed by the observation that our edge cost measure defined in Equation 4.10 is quite consistent. In other words, we rarely encounter false positives or false negatives in our measure. Thus, for edges which are deemed poor pairings, we may confidently augment the cost function for normal propagation with other standard edge-cost measures, such as the work of König and Gumhold [2009].

Finally, constructing the Laplace-Beltrami operator for noisy and nonuniformly sampled point clouds is still an active area of research. While our method is shown to be robust in



(b) Initial Fields

(d) Modified Fields

(a) Initial Results

(c) Modified

Fig. 4.9. Variation in estimation due to random function generation. (a) Results using four randomly generated harmonic functions. (b) Results after adding an additional function. (c) and (d) the gradient fields before and after the addition.

the presence of noise and missing data, the construction of the Laplace-Beltrami operator can be sensitive to outliers due to the fixed global bandwidth used and in the presence of thin-surface sheets due to the k-nearest neighbor assumption.

4.6 Summary

The consistent orientation of point clouds is a critical preprocessing step for many geometry processing tasks, and in particular is necessary for the faithful reconstruction of a sampled surface in the presence of acquisition artifacts. We show how to use harmonic functions defined directly on the point cloud to robustly estimate normal orientation. We show that through the formulation of a propagation and flipping criterion, based on the gradient fields of these harmonic functions, we can achieve consistent results for point clouds exhibiting different acquisition artifacts.

The key insight of our approach is the use of globally smooth functions defined directly on point clouds for normal orientation. Note that the gradient estimation and normal propagation steps of our method only require a scalar field on the point cloud. Hence, any other set of intrinsically-defined functions may be used, such as the Laplace-Beltrami eigenfunctions, the heat kernel, and diffusion distances.

CHAPTER 5

THE MEDIAL KERNEL

In the previous chapter, we saw how to utilize intrinsic, globally smooth functions defined directly on the point cloud for normal estimation. A natural question to ask is whether we can use similar intrinsically-defined quantities for the purposes of surface reconstruction, particularly in the presence of nonuniformly sampled and incomplete data. However, the Laplace-Beltrami operator defined for point clouds assumes rather strict sampling criteria, where in the presence of missing data, quantities derived from this operator will at best respect the boundary components stemming from incomplete data. Worse yet, for undersampled nearby surface sheets, false connections may be made in the construction, resulting in a topologically incorrect representation.

In this chapter, we focus on the problem of constructing an operator directly on the point cloud which is tolerant to missing data, and useful for the purposes of reconstruction. In particular, our main goal is the construction of *distances* defined on the point cloud, wherein the distances are derived from such an operator.

To construct such a representation, we first need to find a more general invariant of a surface which is resilient to missing data. The medial axis of the surface is such an object – indeed, previous works [Tagliasacchi et al. 2009; Cao et al. 2010; Li et al. 2010] have demonstrated how to extract skeletal representations, or medial representations strictly consisting of a set of curves, in the presence of missing data. We take influence from these methods by constructing distances which adhere to the medial axis, yet we depart from the aforementioned approaches by constructing these distances *algebraically*, rather than geometrically.

We introduce the *medial kernel*, an association measure which provides for a robust construction of volume-aware distances. The kernel measure is simply the likelihood of two points lying on a common interior medial ball. From the medial kernel, we construct a random walk on the point cloud, where movement in the walk is restricted to regions containing similar medial balls. In particular, if a subset of points exclusively has a large association according to the medial kernel, then for a sufficiently large time scale, the random walk will only permit movement within this subset. Our distance construction follows as the diffusion distances [Coifman and Lafon 2006] of this random walk, where two points contain low distance if they are highly connected in terms of walking along similar medial balls.

We leverage the medial kernel for several applications – see Figure 5.1 for an overview. The distances induced by the kernel provide for a simple method of segmenting the point cloud into coherent volumetric parts. The medial kernel can be used to construct function bases, where projection onto this basis serves to average function values along medial regions. We use this for surface reconstruction in the presence of missing data. Lastly, we combine these two methods to perform reconstruction-by-parts, a reconstruction method which adheres to the volume indicated by the medial kernel.

5.1 Overview

Before going into the details of our approach, we present a brief 2D example illustrating the intuition behind our method; see Figure 5.2.

Consider the sampled 2D curve on the left-hand side of Figure 5.2. In observing its set of medial balls, we find that there exists a total of five – two medial balls capping the ends of the shape, and three medial balls towards the center. This information may be encoded as a correspondence matrix C, where C_{ij} is 1 if points i and j belong to the same medial ball, and 0 otherwise. Interpreting C as an adjacency graph, the block structure reveals that we have a disconnected collection of cliques, one clique for every medial ball.

Note that the medial axis can be represented through the spectral properties of C. Under a suitable orthogonal transformation, the eigenvectors of C serve as indicator functions for each medial ball, where for a given eigenvector, its nonzero function values group points which lie on the same medial ball. The nonzero eigenvalues of C represent the number of points belonging to a medial ball. If we row-normalize C to obtain $\hat{C}$, then the multiplicity of eigenvalue with magnitude 1 is the number of medial balls, and consequently the rank of C is the number of medial balls.

Now, consider a slight perturbation of this shape, composed of a set of points $P = \{\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_k\}$ with accompanying normals $N = \{\mathbf{n}_1, \mathbf{n}_2, ..., \mathbf{n}_k\}$, where the structure of the cliques is imprecise; see the middle of Figure 5.2. In this scenario, we would like to best recover the cliques and group points which contain a similar medial structure. In other words, we want to approximate the matrix C. Our approach for approximating C is to construct, for a given pair of points in P, a similarity measure representing the likelihood of

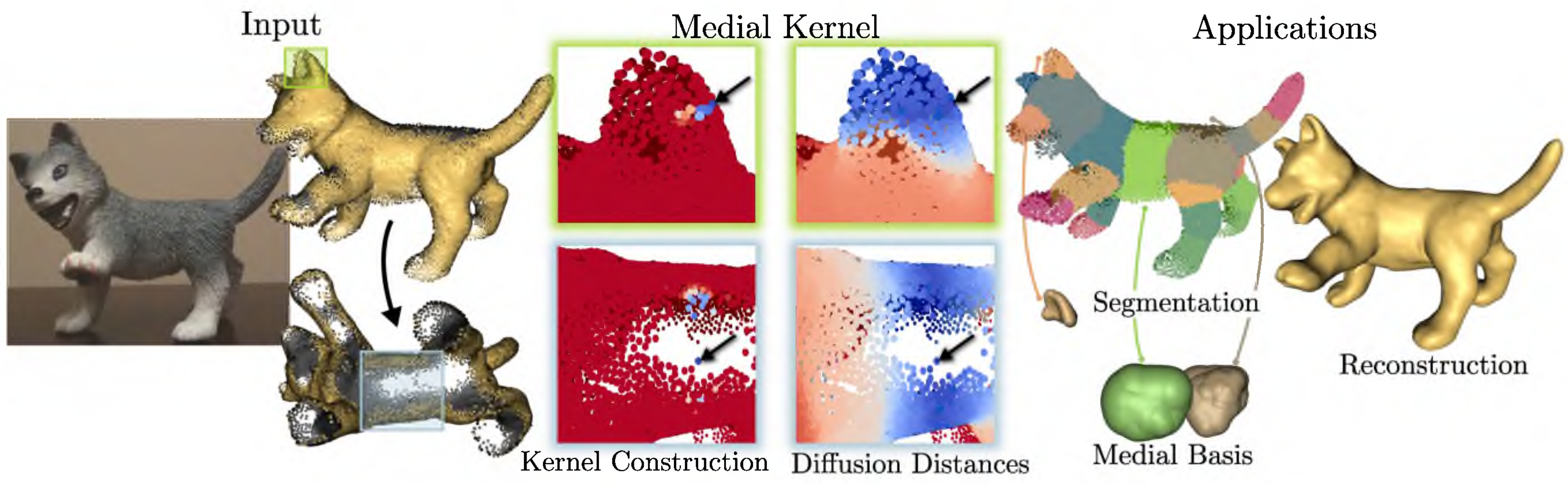


Fig. 5.1. An overview of the medial kernel. From the input point cloud (left), our main contribution is the construction of distances defined directly on the points (middle), where distance represents the likelihood of two points lying on a medial ball. Note the insensitivity to undersampling and missing data. We leverage these distances for several applications, shown on the right.

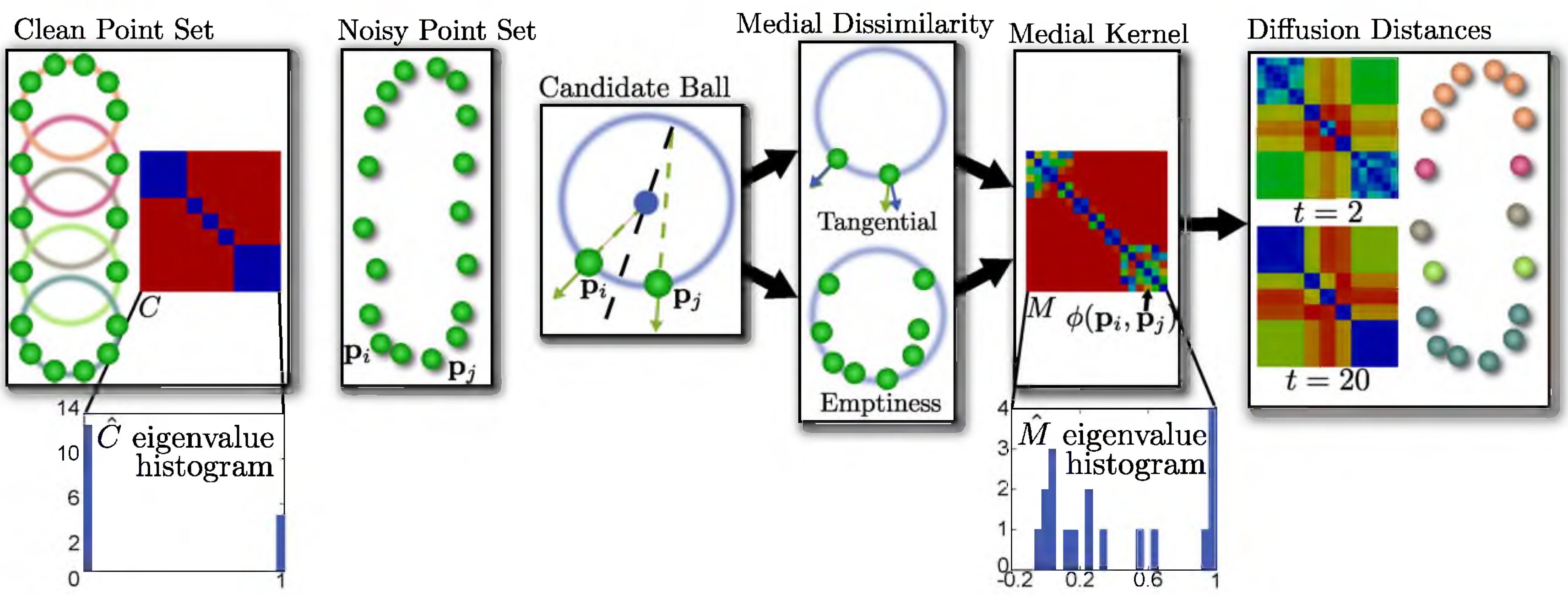


Fig. 5.2. Overview of the medial kernel construction. The clean point set and its set of medial balls illustrates the block structure we would like to recover on the noisy point set. Our medial kernel approximates such correspondences, by measuring the likelihood in which two points contain a medial ball. Note the similarities in spectra between the clean and noisy point sets. We exploit this by applying diffusion distances to the medial kernel to recover the block structure and correspondences.

these two points lying on a medial ball. We call this similarity measure the *medial kernel*, denoted $\phi : P \times P \to \mathbb{R}$.

Given a pair of points $\mathbf{p}_i, \mathbf{p}_j \in P$ with normals $\mathbf{n}_i, \mathbf{n}_j$, we construct the medial kernel in two steps. First, we generate a *candidate ball*, a representative medial ball for $(\mathbf{p}_i, \mathbf{p}_j)$ being equidistant to $\mathbf{p}_i$ and $\mathbf{p}_j$ and whose normals at the points are similar to $\mathbf{n}_i$ and $\mathbf{n}_j$. Next, we define a measure of *medial dissimilarity*, or how far away the candidate ball is from being medial. Following the definition of a medial ball, for a candidate ball, this is decomposed into two measures: how far from tangential with respect to $\mathbf{n}_i$ and $\mathbf{n}_j$, and how empty. Emptiness is a function of the number of points residing inside of the ball, and how close they are to the ball center. We then convert this dissimilarity measure into a similarity measure to obtain the medial kernel; see the right side of Figure 5.2.

From the medial kernel $\phi(\cdot, \cdot)$, we arrive at our approximation to C, the matrix $M : M_{ij} = \phi(\mathbf{p}_i, \mathbf{p}_j)$. Note that nonuniform sampling, positional noise, and normal noise manifest as noise in M. However, similar to Lipman et al. [2010], we find that M's row normalized matrix $\hat{M}$ largely inherits the spectral properties of $\hat{C}$. This can be seen in the eigenvalues of $\hat{M}$ where its top five eigenvalues reside near 1, and all others quickly converge to 0 – a consequence of the rank deficiency of M. This indicates the existence of five medial balls.

For any shape with a well-defined medial axis, M should exhibit rank deficiency, and we seek to define distances which respect this low rank structure. Note that the medial kernel induces a particular random walk on the point cloud, where for large time scales, points walk along similar medial regions. Moreover, a set of points which exclusively contain high associativity in the medial kernel will remain “stuck” in the walk, only moving between each other. Our distance construction follows as the measure of connectedness in this random walk: the diffusion distances [Coifman and Lafon 2006] of M; see the right side of Figure 5.2. Diffusion distances are a natural tool for recovering such a low-rank structure, in our case grouping together points which mutually contain a similar medial region. Note that unlike the eigenvectors of M, the diffusion distances are invariant to any orthogonal transformation of its eigenspaces [Lipman et al. 2010]. Observe on the far right that for $t = 20$, we recover the original block structure of C, grouping points which contain similar medial balls.

5.2 Medial Kernel Construction

Here, we describe the details of the medial kernel construction. The medial kernel associates similarity to a pair of points based on the likelihood of such points containing a medial ball. We construct this by first generating a candidate ball for the points, and then measure how far away this ball is from being a medial ball.

Our construction requires oriented normals N, where we compute normal directions from the input point set P via PCA. If P is obtained from a scanner, we use the individual scans to best orient N; otherwise, if scan information is unavailable, we propagate normal orientation via a minimal spanning tree approach.

5.2.1 Candidate Ball Generation

For points $\mathbf{p}_i$ and $\mathbf{p}_j$ with normals $\mathbf{n}_i$ and $\mathbf{n}_j$, we want its candidate ball to best represent an interior medial ball. This implies that the center $\mathbf{c}_{ij}$ lies on the bisecting plane of the points, while the normals of the ball at $\mathbf{p}_i$ and $\mathbf{p}_j$ respectively coincide with $\mathbf{n}_i$ and $\mathbf{n}_j$.

To this end, we intersect the lines formed from the points and normals against the bisecting plane to obtain intersection points $\mathbf{x}_i$ and $\mathbf{x}_j$. We discard balls if either intersection is along the positive direction of their normal, indicative of a ball lying in the exterior of the shape, or if both lines fail to intersect the bisecting plane. We would like to have the ball normals at $\mathbf{p}_i$ and $\mathbf{p}_j$ mutually satisfy $\mathbf{n}_i$ and $\mathbf{n}_j$, but at sharp features, this can produce balls of arbitrarily large radius. Figure 5.3 depicts such a situation, where the bottom point's normal line fails to intersect the bisecting plane, shown as the dashed black line. Hence, we relax this requirement by additionally considering the balls formed by the individual intersection points. This corresponds to the left point's normal intersection with the bisecting plane. So, from the points $\{\mathbf{x}_i, \frac{\mathbf{x}_i+\mathbf{x}_j}{2}, \mathbf{x}_j\}$, we take the candidate ball center $\mathbf{c}_{ij}$ as the one with minimal radius, which by construction is equidistant to $\mathbf{p}_i$ and $\mathbf{p}_j$. Such a hard constraint on point equidistance and soft constraint on normal agreement expresses our precedence for point positions over point normals, since normal estimation is often imperfect.

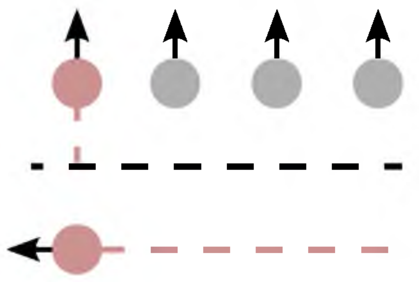


Fig. 5.3. An illustration of finding a candidate ball in the presence of sharp features. In this case, both antinormal rays may fail to intersect the bisecting plane, so we choose the antinormal ray which forms the ball of minimal radius.

5.2.2 Medial Dissimilarity

From the candidate ball, we measure its deviation from a medial ball in two measures: one measures emptiness, while the other measures how tangential.

$$\gamma(\mathbf{p}_i, \mathbf{p}_j) = \sum_{\mathbf{p} \in P} \mu(\mathbf{c}_{ij}, r_{ij}, \mathbf{p}) \tag{5.1}$$

$$\tau(\mathbf{p}_i, \mathbf{p}_j) = |\mathbf{n}_i - \mathbf{s}_i| + |\mathbf{n}_j - \mathbf{s}_j| \tag{5.2}$$

Here, $\mathbf{s}_i$ and $\mathbf{s}_j$ are the normals of the candidate ball at points $\mathbf{p}_i$ and $\mathbf{p}_j$, and μ is the *ball distance measure*, measuring how close a point $\mathbf{p}$ lies from the center of the candidate ball $\mathbf{c}_{ij}$. We would like μ to satisfy the following properties: scale-invariance, slow falloff, and computational efficiency. Scale-invariance implies that the distance measure is relative to the radius of the candidate ball. We want to prescribe a falloff to μ such that points closer to the candidate ball center contribute more, indicative of the ball deeply penetrating the surface. Lastly, μ should be defined such that its summation over all points may be performed efficiently and exactly.

To this end, we define μ as follows:

$$\mu(\mathbf{c}, r, \mathbf{p}) = \begin{cases} 1 - \left(\frac{|\mathbf{p}-\mathbf{c}|}{r}\right)^4 & \text{if } |\mathbf{p}-\mathbf{c}| < r \\ 0 & \text{otherwise} \end{cases} \tag{5.3}$$

We empirically found that this quartic falloff is suitable for penalizing points which belong in the deep interior of a candidate ball. We experimented with a quadratic falloff, but found that it failed to sufficiently penalize points which are near the center of a candidate ball, resulting in emptiness measures which erroneously indicated the potential of a medial ball.

Naively evaluating the dissimilarity measure, even using a spatial acceleration structure, can still be linear in the number of points for balls with large radius. However, note that μ can be expanded such that it is linear in $\mathbf{c}$ and powers of $\mathbf{c}$. More specifically, assume that $P_s \subseteq P$ is a set of points which reside within the candidate ball. We can thus safely rewrite γ as:

$$\gamma(\mathbf{p}_i, \mathbf{p}_j) = \sum_{\mathbf{p} \in P_s} 1 - \left(\frac{|\mathbf{p}-\mathbf{c}|}{r}\right)^4 \tag{5.4}$$

We will show that this can be written as a set of terms linear in $\mathbf{c}$ and a small set of powers of $\mathbf{c}$, thus resulting in the above summation to be constant time for *any* candidate ball in which P_s is contained, by performing a small amount of preprocessing with little memory overhead.

Note that the factors of 1 and $\frac{1}{r}$ have no bearing on the expansion and can be ignored. Consider the expansion of the quartic term $|\mathbf{p}-\mathbf{c}|^4$:

$$|\mathbf{p}-\mathbf{c}|^4 = |\mathbf{c}|^4 + |\mathbf{p}|^4 + 2|\mathbf{c}|^2|\mathbf{p}|^2 - 4\langle \mathbf{c}, \mathbf{p}\rangle(-\langle \mathbf{c}, \mathbf{p}\rangle + |\mathbf{c}|^2 + |\mathbf{p}|^2) \tag{5.5}$$

The first three terms are linear in $|\mathbf{c}|^4$, 1, and $2|\mathbf{c}|^2$, respectively. Expanding the remaining terms, we obtain:

$$-4(-\mathbf{c}^T(\mathbf{p}\mathbf{p}^T)\mathbf{c} + \langle |\mathbf{c}|^2\mathbf{c}, \mathbf{p}\rangle + \langle \mathbf{c}, |\mathbf{p}|^2\mathbf{p}\rangle) \tag{5.6}$$

Hence, we have linearity in terms of $\mathbf{c}$, and powers of $\mathbf{c}$.

Returning to the original summation, due to linearity in $\mathbf{c}$ and its various powers, we can precompute this summation with a small set of terms: $|P_s|$, $\sum|\mathbf{p}|^4$, $2\sum|\mathbf{p}^2|$, $4\sum \mathbf{p}\mathbf{p}^T$, $4\sum\mathbf{p}$, and $4\sum|\mathbf{p}|^2\mathbf{p}$. So, for a given $\mathbf{c}$, we can compute its corresponding terms: $|\mathbf{c}|^4$, $|\mathbf{c}|^2$, $\mathbf{c}$, and $|\mathbf{c}|^2\mathbf{c}$, and apply it to the precomputed summed terms to obtain the exact distance measure.

For this to be effective in practice, we must know a priori that P_s belongs to the candidate ball. We achieve this speed up by constructing a kd-tree over P, and for each node, compute the aforementioned distance measure terms, along with the bounding box of the points for that node. Then, given $\mathbf{c}$ and r, as we traverse the kd-tree if the bounding box of a tree node is entirely contained within the candidate ball, we apply the above precomputed summation to obtain the emptiness measure.

5.2.3 Medial Kernel

From the measures γ and τ, we may now define the medial kernel ϕ, effectively converting medial dissimilarity into a similarity measure:

$$\phi(\mathbf{p}_i, \mathbf{p}_j) = e^{-\left(\frac{\gamma(\mathbf{p}_i,\mathbf{p}_j)}{\sigma_e}\right)^2 - \left(\frac{\tau(\mathbf{p}_i,\mathbf{p}_j)}{\sigma_t}\right)^2} \tag{5.7}$$

where σ_e and σ_t define bandwidths for the emptiness and tangential measures, respectively. We have set $\sigma_e = 2$ and $\sigma_t = 0.7$ for all results in this chapter, unless otherwise specified. We perform this measure over all point pairs to arrive at the similarity matrix $M : M_{ij} = \phi(\mathbf{p}_i, \mathbf{p}_j)$, where each entry encodes how likely the point pair contains a medial ball.

In practice, we find most entries of M to have small magnitude – a function of the complexity of the medial axis. Hence, we set M_{ij} to 0 if $M_{ij} < 10^{-7}$, resulting in M typically being quite sparse. We use this sparsity to employ an early termination in the traversal of the kd-tree for computing the emptiness measure γ, allowing us to quickly discard point pairs which are highly dissimilar.

The matrix M can be quite noisy. For instance, since we have a hard constraint on equidistance in candidate ball generation, two adjacent points lying on a plane will result in a ball with unbounded radius, and consequently low similarity. However, if two such points mutually share other points which have a high similarity, then there is a strong likelihood that these points belong to the same medial ball.

As discussed in Section 5.1, the diffusion maps of M capture this similarity, in the form of measuring the connectedness of random walks defined via the medial kernel. To this end, consider the matrix $\hat{M}$ taken as the row-normalization of M, as suggested by Coifman and Lafon [2006] and Lipman et al. [2010]. It has an eigendecomposition of the form $\hat{M} = V \Sigma U^T$, where its eigenvalues and left/right eigenvectors are real-valued. Letting $V = [\Psi_1 \; \Psi_2 \; \cdots \; \Psi_k]$, the resulting diffusion map at point $\mathbf{p}_i$ under a time scale t is:

$$\Phi_t(\mathbf{p}_i) = \{\lambda_1^t \Psi_1(\mathbf{p}_i), \lambda_2^t \Psi_2(\mathbf{p}_i), \lambda_3^t \Psi_3(\mathbf{p}_i), ...\} \tag{5.8}$$

The diffusion distances directly follow from Φ:

$$d_t^2(\mathbf{p}_i, \mathbf{p}_j) = |\Phi_t(\mathbf{p}_i) - \Phi_t(\mathbf{p}_j)|^2 \tag{5.9}$$

Unless otherwise specified, we used a time scale of $t = 160$ for all results, which we found to be a conservative time scale as useful distances are typically achieved at smaller times. Due to the large time scale used, we found it necessary to only retain the top 300 eigenvectors, and since M is sparse, this can be computed efficiently via ARPACK.

See Figure 5.4 for several examples of diffusion distances of the medial kernel. Note how the distances relate points which have high likelihood of belonging to an underlying medial ball, for both well-sampled shapes and single range scans alike. Figure 5.5 illustrates our ability to handle the case of two nearby planar surface sheets. Note that although adjacent planar points are initially dissimilar, under a suitable time scale, we are able to capture the similarity, indicative of a medial ball lying between the surface.

Figure 5.6 shows the kernel's robustness to missing data and noise. Note that noise is both positional and normal, since we compute normals from the points via PCA. This is a particularly challenging model as the foot resides directly next to the leg of the dancer, with missing data between the two parts. As noise increases, our method is still able to associate similarity to points occupying similar volume, as points on the back of the leg contain small distance to the source point.

We note that the distances of the medial kernel do not exactly correspond to diffusion distances measured along the medial axis, namely due to two properties. First, our kernel construction results in fast diffusion for a point on the surface whose corresponding medial

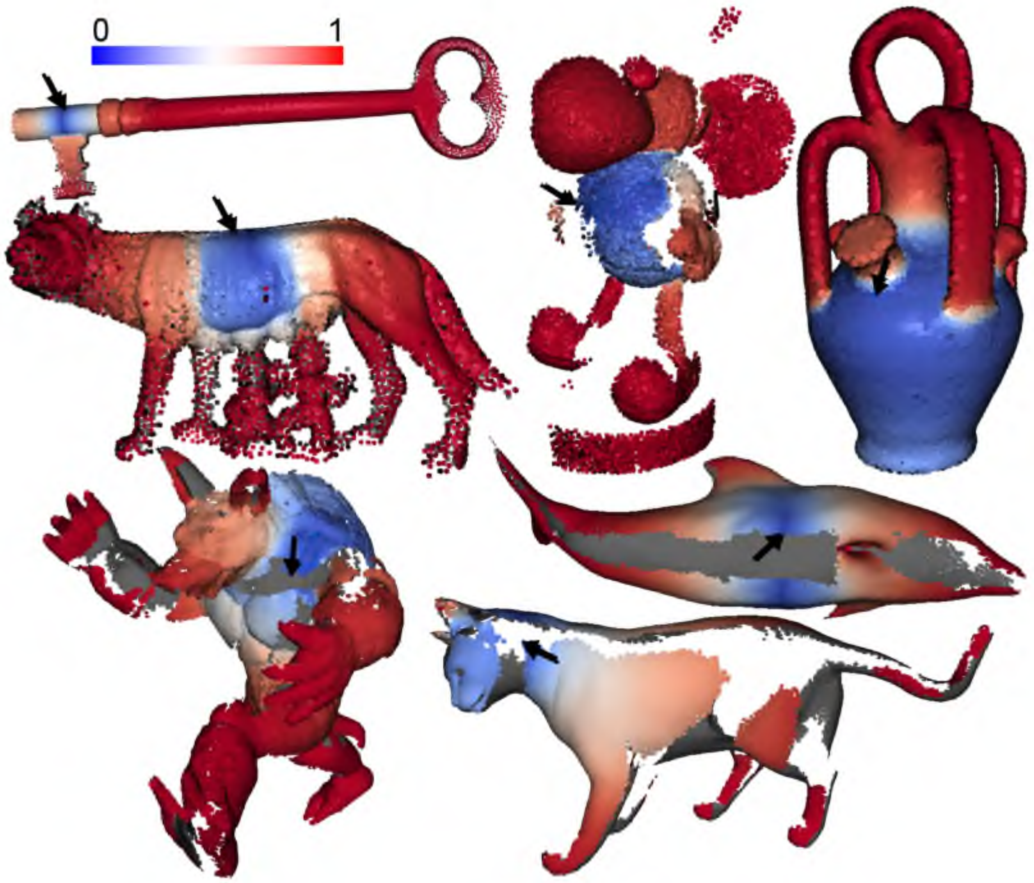


Fig. 5.4. Diffusion distances derived from the medial kernel from various source points for a variety of point clouds, ranging from fully-sampled meshes to single range scans.

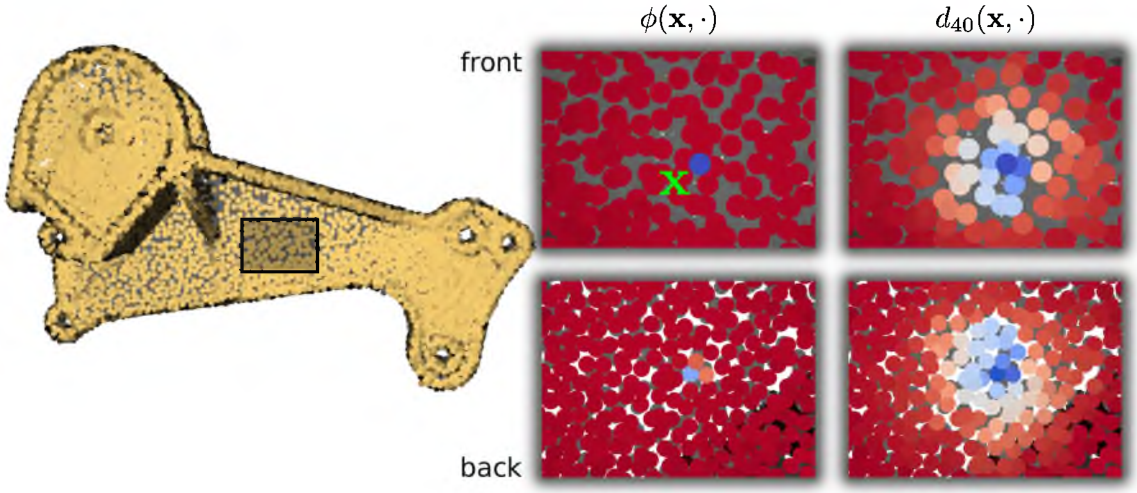


Fig. 5.5. Distances constructed on thin planar sheets. Although adjacent points to $\mathbf{x}$ are initially dissimilar, diffusion distances capture the association from the other side of the surface for suitably large times.

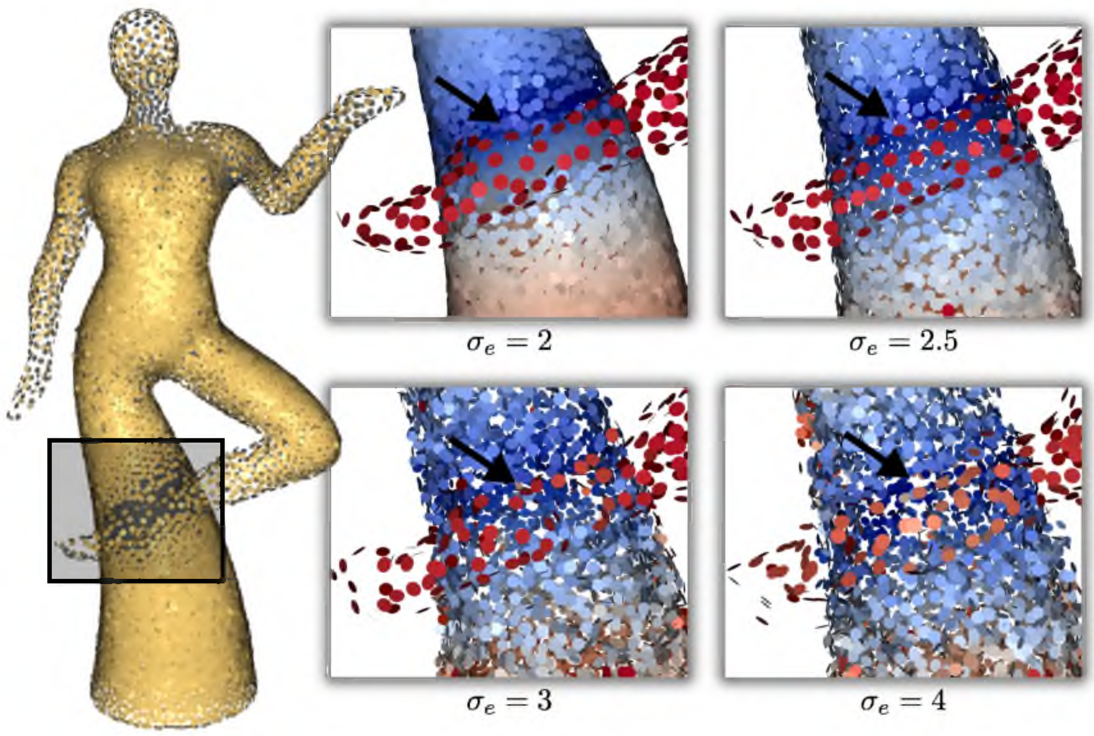


Fig. 5.6. The performance of our medial kernel under missing data and increasing noise. Note that the source point contains low distance to points occupying similar volume.

axis point contains a large number of generating surface points. For instance, a portion of a surface bounding a medial sheet contains fewer generating surface points compared to a spherical part of a surface. Secondly, in regions of negative Gaussian curvature, the medial kernel between two points can contain low similarity, regardless of how close they are in Euclidean distance, as there does not exist a candidate ball which coincides with their normals. This can result in a sharp decrease in distance between such points, and under substantial missing data, it may disconnect parts of a surface.

To illustrate this behavior, we generate a sequence of shapes smoothly deforming as a function of these two properties, and look at how the distances between two fixed points change under a large set of time scales. We have taken a cylindrical shape generated via a union of balls, and continuously deformed the shape by shrinking balls which are closer to the center of the shape. We parameterize the set of shapes by the amount in which we shrink the balls, and we uniformly sample each shape to produce a point cloud. Since we uniformly sample each shape, there will be a smaller amount of points in regions where the balls are of smaller radii. Moreover, by shrinking balls closer to the center of the cylinder, this results in a "pinching" effect, resulting in a region of negative curvature.

Figure 5.7 shows the set of shapes and the resulting distances. In order to quantitatively

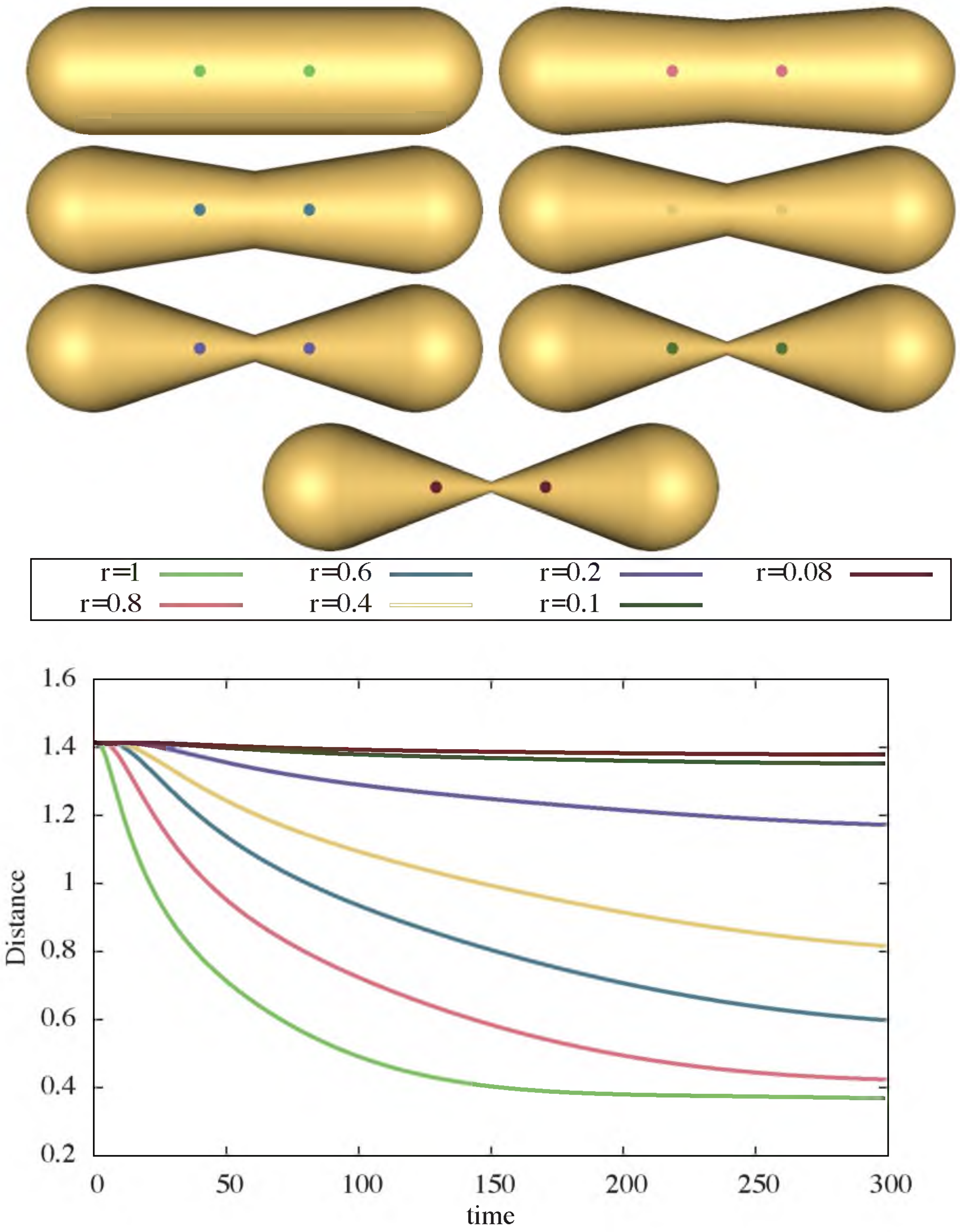


Fig. 5.7. We show how the distances between two points changes as we deform the cylinder (top-left) by shrinking its center, according to the radius r of the shape's center medial ball. Note that even though the distances are sensitive to this deformation, we nonetheless maintain connectivity between the highlighted points.

compare across different shapes, we normalize the embedding of Equation 5.8 prior to taking the distances, as suggested in Goldberg and Kim [2010]. As expected, we see that the shrinking of the balls results in distances between the two points being larger, as a function of the time scale. However, note that we are still able to retain connectivity between the points even as we nearly pinch off the surface at $r = 0.08$. As we will see in our applications, the ability to retain connectivity where the surface should be, while separating different nearby undersampled parts of a surface, permits us to robustly handle missing data.

5.3 Applications

We illustrate several applications of the medial kernel: segmenting a point cloud into volumetric parts, "medializing" functions by deriving a function basis from the medial kernel, and reconstruction-by-parts.

5.3.1 Volume-Aware Segmentation

For large time scales, the diffusion distances of the medial kernel serve to associate similarity to points which contain smoothly varying volumetric emptiness. Based on this observation, we can easily perform point cloud segmentation using the medial kernel, where points are segmented into clusters occupying similar volumes.

We achieve this segmentation by performing k-means on the diffusion maps, defined by Equation 5.8. We normalize the coordinates prior to clustering, similar to existing methods [Zelnik-Manor and Perona 2004; Solomon et al. 2011]. The resulting segmentation is not intended to be a semantic part decomposition, but rather a decomposition into simple and coherent volumetric parts. In particular, the main contribution is segmentation in the presence of missing data; see Figure 5.8 for an illustration. The segmentation properly clusters the palm into separate parts, separating it from the two fingers despite the fact that there exist no data underneath the fingers. In fact, the number of clusters has an intuitive interpretation in the context of medial kernels: we achieve an adaptive sampling of medial segments, producing more clusters for regions with smaller medial balls.

5.3.2 Medial Basis

In addition to constructing random walks, the medial kernel can also be used to define a basis from which to project functions onto, in a similar manner to Lipman et al. [2010]. In particular, powers of the matrix $\hat{M}$ correspond to a family of such bases, where for a large t, $\hat{M}^t$ serves to effectively reduce the numerical rank of $\hat{M}$. Recall that the numerical rank of $\hat{M}^t$ reflects the complexity of the shape's medial axis. The linear subspace of functions

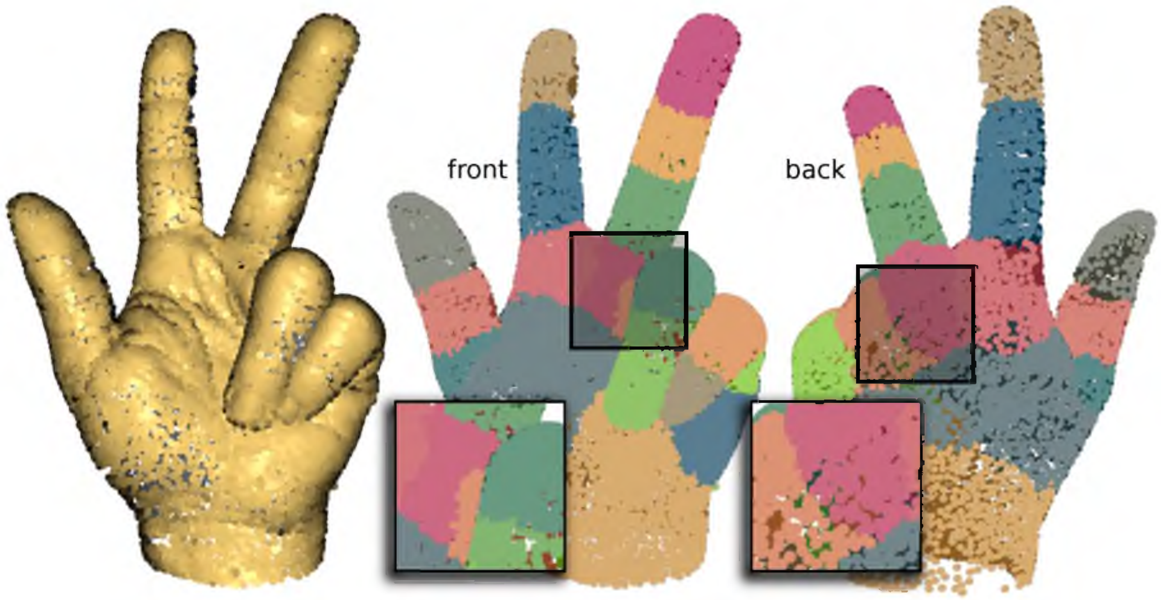


Fig. 5.8. Segmentation results on a hand point cloud (left). Note that the knuckle of the ring finger is properly associated with the palm, despite the lack of data on the palm.

spanned by $\hat{M}^t$ correspond to functions which are constant along medial balls. Hence, for an arbitrary function f, its projection onto $\hat{M}^t$ serves to diffuse function values along medial balls, in the process "medializing" f.

More specifically, from the diagonalization of $\hat{M} = V\Sigma U^T$, suppose we have the set of right eigenvectors $U = [\Theta_1 \; \Theta_2 \; \cdots \; \Theta_k]$. Powers of $\hat{M}$ may be expressed as:

$$\hat{M}^t = \sum_i \Psi_i \lambda_i^t \Theta_i^T \tag{5.10}$$

Then, the projection of f onto $\hat{M}^t$ is:

$$\hat{M}^t f = \sum_i \Psi_i \lambda_i^t \langle \Theta_i^T, f \rangle \tag{5.11}$$

In considering useful functions to medialize, we observe that one should choose functions which are naturally invariant to the medial basis, yet are initially noisy. The union of balls [Miklos et al. 2010], or the set of interior medial balls, is thus a natural candidate. This provides us with a simple yet robust method for reconstruction, which we term the Diffusion of Union of Balls (DUB).

In our approximate scenario, we derive the initial set of balls from our construction of the kernel. For a given point $\mathbf{p}_i$, we define its initial ball $[\mathbf{c}_i, r_i]$ as:

$$\mathbf{c}_i = \frac{\sum_j \mathbf{c}_{ij} \phi(\mathbf{p}_i, \mathbf{p}_j)}{\sum_j \phi(\mathbf{p}_i, \mathbf{p}_j)} \qquad r_i = \frac{\sum_j r_{ij} \phi(\mathbf{p}_i, \mathbf{p}_j)}{\sum_j \phi(\mathbf{p}_i, \mathbf{p}_j)} \tag{5.12}$$

Assuming that $\mathbf{c}$ and $\mathbf{r}$ refer to the set of ball centers and radii defined over the point set, respectively, then its diffusion over the medial basis is: $\mathbf{c}^t = \hat{M}^t\mathbf{c}$ and $\mathbf{r}^t = \hat{M}^t\mathbf{r}$.

As long as there exists sufficient evidence of a volume, we find that DUB is quite effective at preserving the overall volume of the shape; see Figure 5.9 for an illustration. We note that this method is similar to VASE [Tagliasacchi et al. 2011], in that we both rely on smoothness in the volume to diffuse a medial representation. However, we define a diffusion operator on the point cloud, rather than an intermediate mesh representation.

5.3.3 Reconstruction by Parts

Although DUB is effective when the initial set of balls is noisy, if the basis is also noisy, then the diffusion can produce undesirable results due to the contamination of dissimilar balls. The mid-left image of Figure 5.10 depicts the situation, where false positives exist between the hand and the body of the dancer, causing a tunnel to appear in the reconstruction. However, this is precisely what our segmentation method resolves: the clustering of the point cloud into coherent volumetric parts. Hence, it is natural to combine the two methods, resulting in a surface reconstruction method which performs reconstruction-by-parts. See Figure 5.10 for an illustration.

We first segment the point cloud into volumetric components via k-means. The number of clusters should be large enough so that nearby parts are separated, yet not so big that there exists an insufficient number of points to represent a volume. For most shapes, this range is typically quite large, however, and for all results in this chapter, we found $20 - 30$ segments to be sufficient. We next pad each segment out with points belonging to other

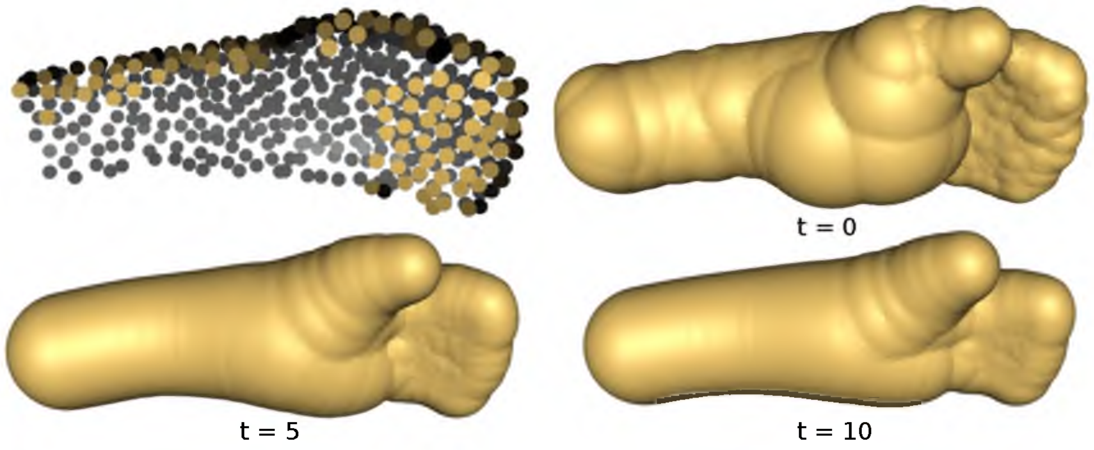


Fig. 5.9. We apply DUB to the point cloud on the top-left for times $t = 0$, $t = 5$, and $t = 10$. Note how despite the missing data, by projecting onto $\hat{M}$, we are able to sufficiently smooth out the noise.

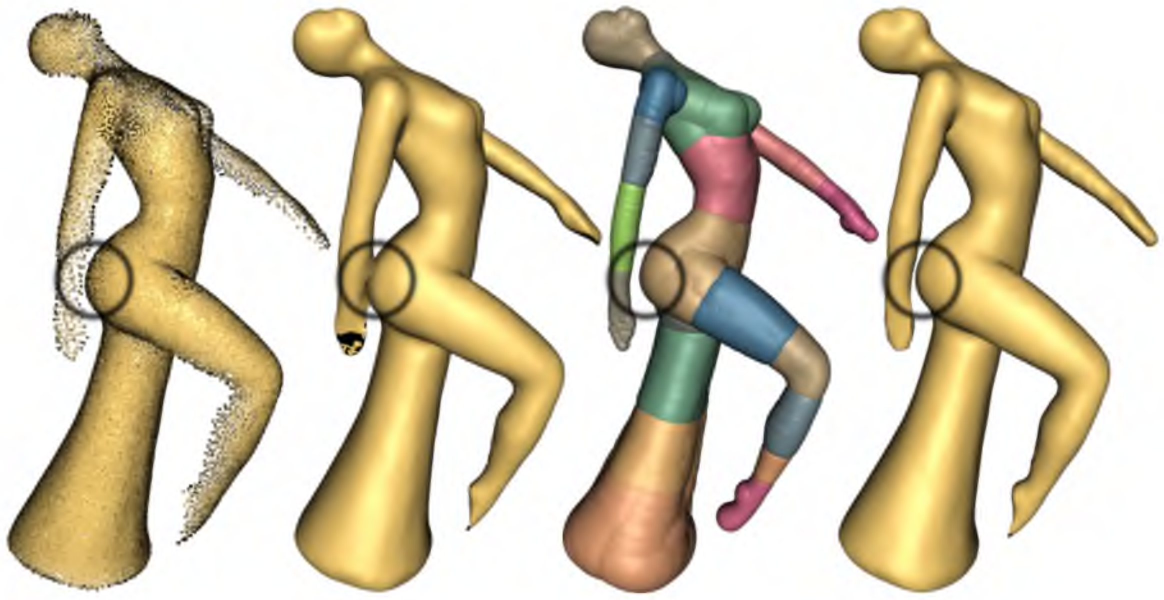


Fig. 5.10. From the point cloud on the left, we first show the reconstruction through Diffusion of Union of Balls (DUB) on the entire point cloud, followed by projection (mid-left). Note the tunnel introduced due to false positives in our kernel. By segmenting the point cloud (mid-right) and then performing reconstruction-by-parts, we produce a topologically accurate reconstruction (right).

segments which are close in terms of our medial-factored distances. We achieve this by performing a k-nearest neighbors query with respect to the diffusion map across all of the points in a segment, adding points which belong to different segments. We choose $k = 25$ in our implementation, which we have found to provide for sufficient overlap between segments.

We then apply DUB to each segment, to obtain a collection of union of balls. Our volumetric segmentation ensures that each DUB-reconstructed segment encompasses a proper volume of the shape, where we found time scale $t = 4$ to provide for a smooth yet geometrically faithful representation for each segment. We then take the union of the union of balls as our reconstructed mesh. Namely, we treat all of the union of balls as an implicit surface and isosurface to obtain the reconstructed mesh. Since we are padding each segment with points in nearby segments (with respect to medial-factored distances), there exists sufficient overlap between individual reconstructions to form a single component.

The resulting mesh is slightly shrunken due to the diffusion process. We obtain the final reconstruction by interleaving MLS projection (via Algebraic Point Set Surfaces [Guennebaud and Gross 2007]) and least squares meshes [Sorkine et al. 2005], applied to the vertices of the mesh, in a similar manner to Sharf et al. [2006]. Namely, in the first step, we use APSS to project the mesh vertices onto the point cloud. Only vertices already close to the point cloud are projected, where mesh vertices in regions of missing data remain in place. We depart from Sharf et al. [2006] by restricting the projection step to the individual clusters, in order to prevent projection issues associated with undersampling of the point cloud. More specifically, since each mesh vertex originated from a specific segment, we can associate that vertex to a point cloud segment. In gathering an epsilon ball for the projection step, we only use those points which belong to the segment. This has two benefits: it prevents points from drifting to other regions, and it allows us to use a rather large epsilon ball in the projection, hence rendering our method robust to highly nonuniform sampling.

In the second step, we then minimize an energy over the mesh which simultaneously satisfies smoothness via a Bi-Laplacian term, and a data term with respect to the original mesh. Weights favoring smoothness are given to points which are far from the point cloud. We alternate these two steps until convergence, where we found 5 iterations to be sufficient in our experiments.

5.4 Results

We compare our approach to kernel methods regarding distances and segmentation, as well as to reconstruction methods. Most point clouds used in our experiments have been acquired either through NextEngine or Kreon scanners, and consequently downsampled through farthest point sampling. No data smoothing is employed in the downsampling; we use the original points and normals.

5.4.1 Kernel Methods

We first compare the diffusion distances of our medial kernel to a more standard kernel, namely the feature-preserving kernel of Öztireli et al. [2010] which approximates the heat kernel for point clouds. Note that the heat kernel is well-known to be robust to missing data and topological shortcuts, as demonstrated for segmentation of surface meshes in De Goes et al. [2008]. For point clouds, the approach of Öztireli et al. [2010] constructs the kernel as:

$$k(\mathbf{p}_i, \mathbf{p}_j) = e^{-(\frac{|\mathbf{p}_i - \mathbf{p}_j|}{\sigma_p})^2 - (\frac{|\mathbf{n}_i - \mathbf{n}_j|}{\sigma_n})^2} \tag{5.13}$$

Originally used for its short-time behavior, we consider its long-time behavior in diffusion distances. We set σ_p to 0.02 of the bounding box diagonal, and σ_n to 0.5, a small bandwidth which heavily penalizes normal differences [Öztireli et al. 2010].

See Figure 5.11 for a comparison between distances. The Mannequin model highlights the issues with thin sheets, where although both methods contain false positives, our method

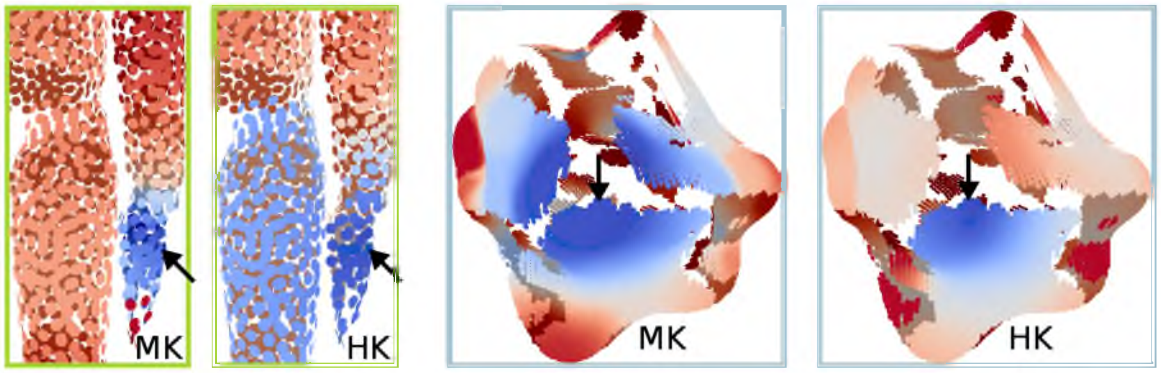


Fig. 5.11. A comparison of distances between our kernel (MK) and the kernel of Öztireli et al. [2010] (HK), both under large-time diffusion distances. Note that the kernel of Öztireli et al. [2010] can leak into other parts of the shape (left) while not adequately covering distant sheets (right). Our method properly handles both cases.

succesfully filters them out since the connectedness induced by medial balls is stronger. The Bumpy Sphere model highlights the opposite issue: points which are outside of the bandwidths of σ_p and σ_n are never connected; hence, the kernel of Öztireli et al. [2010] retains the boundary components. Our method identifies the presence of a medial ball connecting the three disparate sheets. Figure 5.12 shows how this type of identification results in a volumetric segmentation, whereas k-means applied to Öztireli et al. [2010] keeps these parts separate.

5.4.2 Comparison to Killing Vector Fields

Our method bears resemblance to the recent work on mesh segmentation via killing vector fields (KVFs) [Solomon et al. 2011]. A KVF defines an isometric self-mapping, where Solomon et al. [2011] show how the eigenfunctions of a suitable KVF energy can be

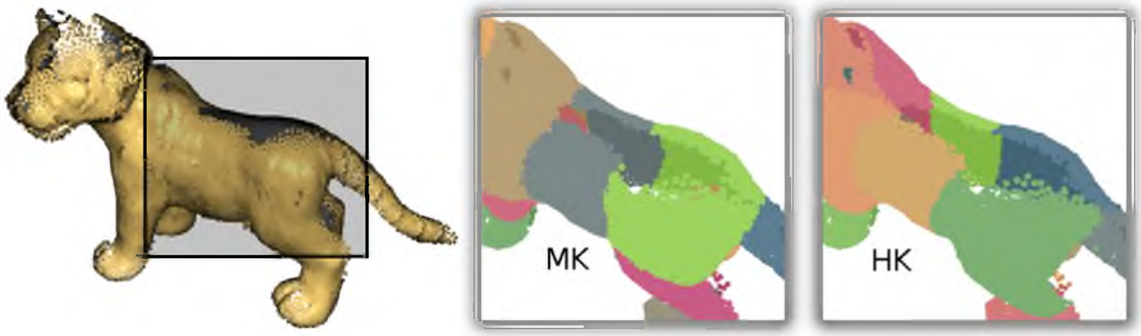


Fig. 5.12. K-means segmentation applied to our method (MK) and the kernel of Öztireli et al. [2010] (HK). Note how our segmentation captures the body of the wolf, despite the large missing data.

used to localize self-isometries for segmentation. For large time scales, our method begins to resemble local self-isometries, as the medial ball itself can be looked upon as a local transformation. Indeed, our segmentation approach of clustering (weighted) eigenvectors of a point cloud operator mirrors Solomon et al. [2011], and as Figure 5.13 shows, we obtain nearly identical segmentations for the given model. It would be interesting future work to extend Solomon et al. [2011] to the case of incomplete point clouds via our method.

5.4.3 Surface Reconstruction

We have run our reconstruction algorithm on a set of challenging acquired data, containing missing data and thin surface sheets. In these scenarios, an explicit segmentation of these regions substantially simplifies reconstruction. We show that our distances provide for a robust means of achieving this segmentation and generating a faithful reconstruction.

See Figure 5.14 for a comparison of our method with that of Fourier surface reconstruction [Kazhdan 2005], adaptive RBFs [Ohtake et al. 2005b], and smoothed MPU [Nagai et al. 2009]. One potential issue with these methods is that they employ function fitting [Ohtake et al. 2005b; Nagai et al. 2009] or variational reconstruction [Kazhdan 2005] independent of the structure of the point cloud. Hence, for thin sheets with missing data, there are no constraints on the surface produced. By segmenting the point cloud via the medial kernel, we avoid issues related to missing data and undersampling. Table 5.1 shows the genus of the resulting reconstructions. As shown, our method is able to preserve the topology of the scanned objects, whereas the other methods demonstrate highly variable behavior. Such erroneous tunnels can be quite detrimental to further processing of the reconstructed objects.

Our method is also robust to the number of segments used for reconstruction. See Figure 5.15 for an illustration of the hand point cloud reconstructed under different numbers

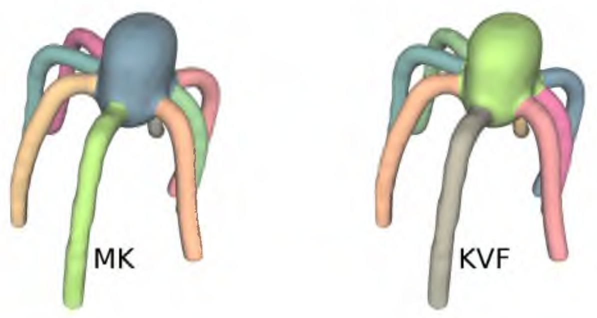


Fig. 5.13. Comparison of our method to killing vector field segmentation [Solomon et al. 2011]. Note that our method produces competitive results.

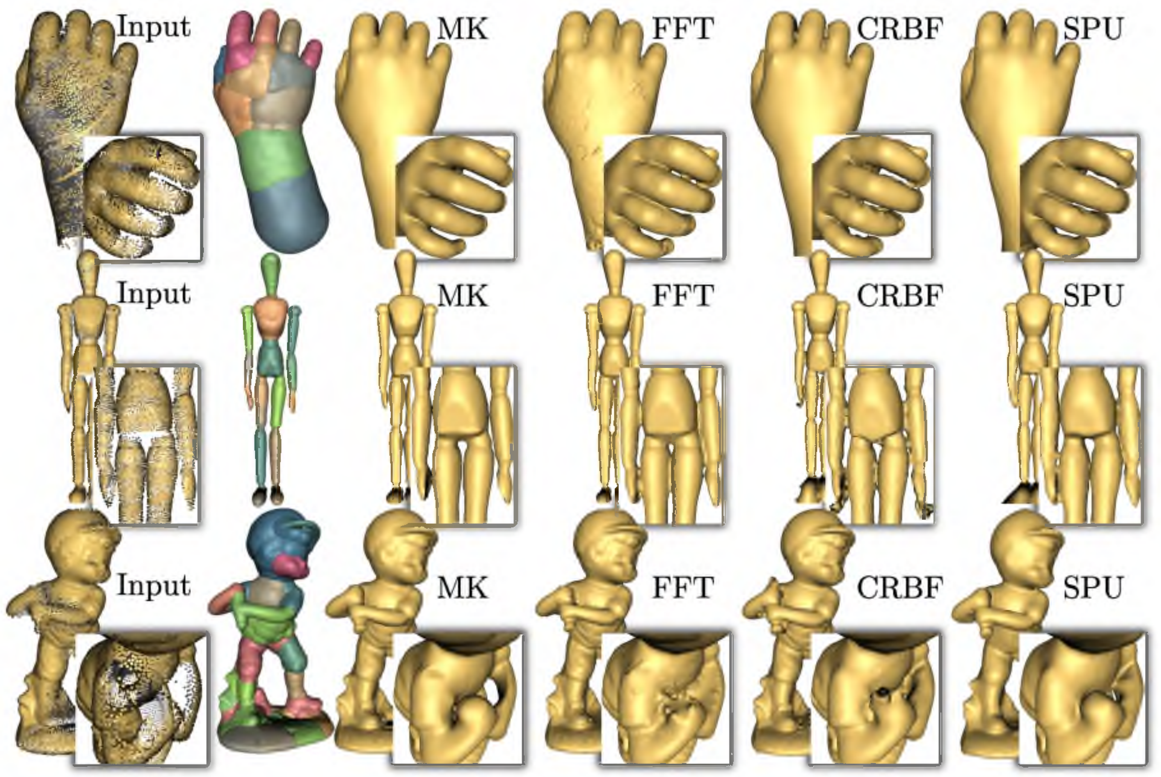


Fig. 5.14. Comparison of surface reconstruction. We first show our reconstruction by way of segmentation, and then a comparison of our method to FFT [Kazhdan 2005], CRBF [Ohtake et al. 2005b], and SPU [Nagai et al. 2009]. Since our method explicitly segments parts of the point cloud, we avoid issues related to missing data and thin surface sheets, where previous methods contain difficulties.

Table 5.1. A comparison of the genus for the reconstruction algorithms and our method.

Alg	Hand	Mannequin	Batter
MK	0	0	2
FFT	2	6	1
CRBF	8	14	2
SPU	1	9	2

of clusters. As shown, the reconstruction is largely unaffected by the different number of clusters.

5.5 Discussion and Limitations

Our medial kernel relies on sufficient evidence of a medial structure for success, so in this absence, our method will result in either isolated points or false positives. The former case does not pose much of a problem in the context of distances and segmentation, but for reconstruction, it may be difficult to construct an initial set of union-of-balls. In the

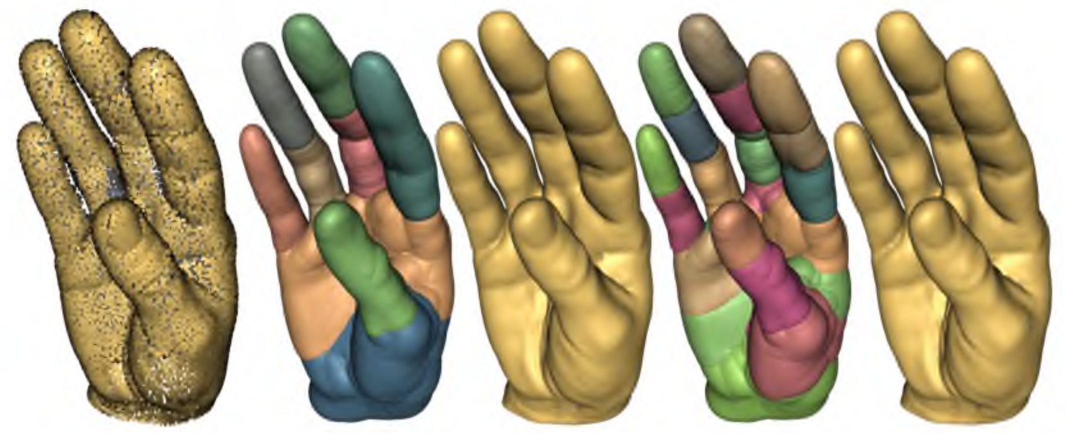


Fig. 5.15. From the point cloud on the left, we show segmentations of different cluster sizes, and the reconstructions. Note the insensitivity to the number of clusters.

latter case, false positives typically occur when neighboring parts have insufficient data to penalize candidate balls between the parts.

Our kernel construction requires oriented normals, where although it is quite robust to normal directions (see Figure 5.6), it is somewhat sensitive to inverted orientation. Our approach can tolerate a small amount of inverted orientation, but for large and continuous regions of inverted normals, we begin to interpret exterior medial balls as being interior.

Perhaps the biggest drawback to our approach is its computational complexity. To construct our measure of emptiness, we must consider, for every pair of points, the entire point set. Hence, in the worst case, the complexity is cubic in the number of points. However in practice, our acceleration scheme typically provides an order of magnitude improvement. Figure 5.16 shows computational timings as a function of point cloud size, and as shown, the kernel construction is generally quadratic. We find that our acceleration scheme is slowest when dealing with spherical parts of a shape, since for these points, the kernel measure is high and so we must sum over all other points on the part to obtain an accurate measure. This is the cause of the Batter model being so time consuming, due to the head part. However, a voting scheme analogous to Lipman et al. [2010] should considerably speed this up.

5.6 Summary

We have presented a method for constructing distances directly on point clouds which are insensitive to missing data. Our key insight is the connection between the medial axis and the inherent rank-deficiency of the correspondence matrix which associates points belonging to common medial balls. Even if there exists false positives and false negatives in

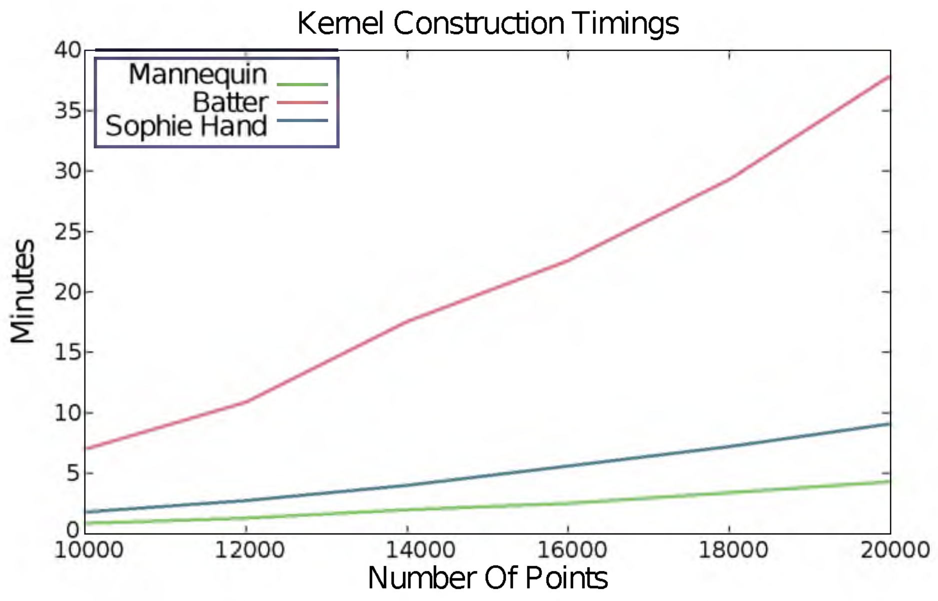


Fig. 5.16. Timings for kernel construction on the shapes used in Figure 5.14, as a function of point cloud size.

the medial kernel, the connectedness induced by the low-rank structure renders our distance construction highly robust to incomplete point clouds.

We have demonstrated how the distances can be used for a number of applications, ranging from segmentation, the construction of function bases, and surface reconstruction. In particular, we have demonstrated these applications when point clouds simultaneously contain large gaps of missing data and nearby surface sheets. In these cases, it is quite difficult to construct a notion of sampling density – indeed, our distance construction *never* makes use of local neighborhoods such as an epsilon ball or k-nearest neighbors. This is in part why our method is highly robust to missing data.

CHAPTER 6

MEDIAL DIFFUSION

In the previous chapter, we illustrated how to use the medial kernel for the purposes of surface reconstruction, and its robustness to missing data. For shapes containing substantial missing data, however, reconstruction may simply be too impractical, as additional information may be necessary. Indeed for certain tasks, reconstruction may not even be necessary, where the information inherent in the medial kernel may be sufficient.

In this chapter, we consider the problem of matching incomplete shapes which are undergoing a nonrigid deformation. The problem amounts to finding correspondences between shapes which adhere to the underlying nonrigid deformation. Nonrigid motion is fairly common in dynamic data capture scenarios, such as human and animal movement.

Although matching well-sampled shapes under nonrigid motion has been fairly well-studied [Lipman and Funkhouser 2009; Ovsjanikov et al. 2010; Tevs et al. 2011], for undersampled shapes, this is a highly nontrivial problem. For acquired point clouds, it is necessary to construct quantities which are invariant to the motion and insensitive to the lack of data for successfully matching shapes. As we have previously shown, the medial kernel is insensitive to missing data, as the medial axis is a useful shape prior. Moreover, previous work has shown [Zhang et al. 2005] that the medial axis is also invariant to pose. In fact, the work of Zheng et al. [2010] used the method of Tagliasacchi et al. [2009] to match a set of compact graph skeletons.

Our approach departs from Zheng et al. [2010] and the usage of such compact representations by instead embracing the full point cloud, in conjunction with the medial axis prior, for shape matching. Inspired by heat kernel matching [Ovsjanikov et al. 2010], we introduce *medial diffusion* for matching shapes, where matching amounts to finding points which belong to similar medial regions; see Figure 6.1. We seek correspondences between medial regions for such challenging point clouds, as there may exist a relatively small amount of surface correspondences due to missing data, where for the left shapes in Figure 6.1, this is due to the two shapes being scanned from opposing views.

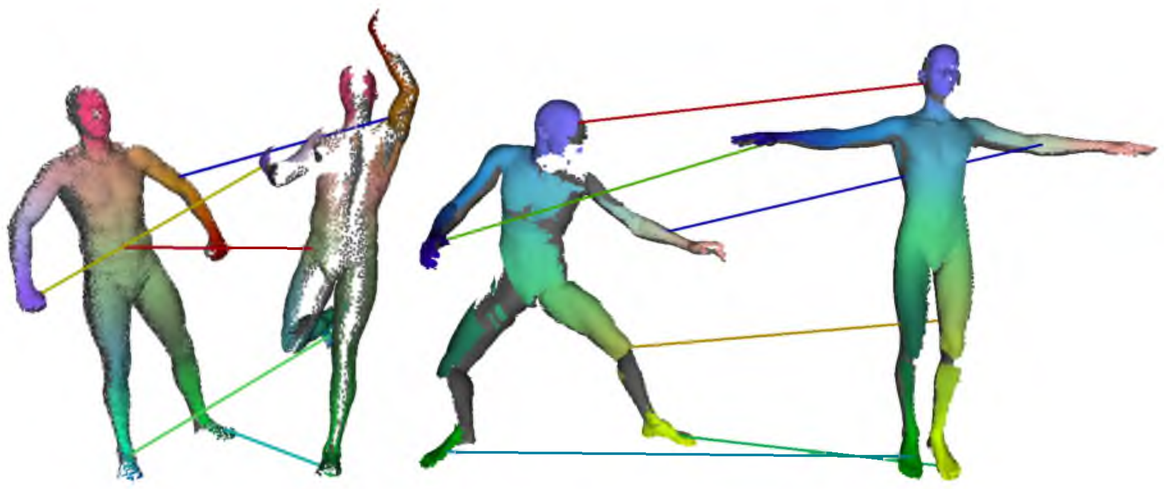


Fig. 6.1. Our approach is able to match shapes undergoing nonrigid motion which contain significant missing data. Here, we show the landmark correspondences automatically found by our algorithm, and the extrapolated dense mapping, color-mapped by the left shape's medial embedding.

Our main contribution is the construction of a Laplace operator defined with respect to the medial axis, for which its resulting heat kernel is suitable for shape matching. Key to our construction is that we define the operator *directly* on the point cloud, via the medial kernel. This direct construction has a number of advantages:

- Our diffusion more faithfully represents the shape compared to curve skeleton methods, as these are inherently lossy representations – for instance, we can capture medial sheets.
- The diffusion process is sensitive to the geometry of the surface compared to skeletal representations – for example, cylindrical regions of different radii exhibit distinct behavior.
- By working directly on the point cloud, we can easily combine other diffusion processes.

From medial diffusion, we introduce a practical algorithm for finding landmark points between shapes, and subsequently extrapolating the landmarks to a dense correspondence of medial regions. Our approach also easily extends to detecting intrinsic symmetries, or nonrigid self-transformations. We show how our method is tolerant to missing data, and improves on standard heat kernel matching of incomplete shapes.

6.1 Medial Diffusion

Our approach to constructing a diffusion process with respect to the medial axis of a shape is determined via a Laplace operator. We first discuss such a Laplace operator on a smooth surface, which we term the *Medial Laplacian*. We then detail its discretization on a point cloud, and lastly the diffusion process itself.

6.1.1 Medial Laplacian

Consider an open set O embedded in $\mathbb{R}^3$ whose boundary is the surface S. Every point $\mathbf{x} \in O$ is associated with a set of points in S which are at a smallest distance to $\mathbf{x}$. We denote this set by $\Gamma(\mathbf{x})$. The set of points $\mathbf{x} \in O$ for which $|\Gamma(\mathbf{x})| \geq 2$ comprises the medial axis of O, which we denote by M.

The medial axis is a very descriptive object, as it carries the homotopy information of O. However, it is a rather difficult object to utilize, as it is composed of a set of adjoining sheets and curves. To define a Laplace operator directly on the medial axis, one option is to construct it piecewise for each sheet and curve, and handle special cases at junctions. One issue with proceeding in this way is that we lose information with respect to the geometry of S, for instance, spherical parts of varying radii are treated equally, as are cylindrical parts of different radii.

To better capture the surface, we use Γ as a density measure. As $|\Gamma|$ is nonsmooth over M, denote $|\tilde{\Gamma}|$ as its smoothed variant, defined as:

$$|\tilde{\Gamma}(\mathbf{x})| = \int_S \exp\Big(-\frac{d(\mathbf{z}, \Pi_{\mathbf{x}}(\mathbf{z}))}{\sigma}\Big)\, \mathrm{d}\mathbf{z} \tag{6.1}$$

where $\Pi_{\mathbf{x}}(\mathbf{z}) = \min_{\mathbf{y}\in\Gamma(\mathbf{x})} d(\mathbf{y}, \mathbf{z})$, or the smallest geodesic distance d between the set $\Gamma(\mathbf{x})$ and $\mathbf{z}$. We follow the approach of Belkin and Niyogi [2005] to define a functional approximation to the Laplacian, in effect using a local Gaussian as an approximation. For a given function f defined on S, we define the Medial Laplacian Δ_M over M as:

$$\Delta_M f(\mathbf{p}) = f(\mathbf{p}) \int_M \alpha(\mathbf{p}, \mathbf{q})|\tilde{\Gamma}(\mathbf{q})|\, \mathrm{d}\mathbf{q} - \int_M f(\mathbf{q})\alpha(\mathbf{p}, \mathbf{q})|\tilde{\Gamma}(\mathbf{q})|\, \mathrm{d}\mathbf{q} \tag{6.2}$$

where $\alpha(\mathbf{p}, \mathbf{q})$ is a Gaussian parameterized by a sufficiently small time scale h:

$$\alpha(\mathbf{p}, \mathbf{q}) = e^{-\frac{|\mathbf{p}-\mathbf{q}|^2}{h}} \tag{6.3}$$

We note that this is in fact a *weighted* Laplacian [Belkin and Niyogi 2005], where $|\tilde{\Gamma}|$ is used to weight regions of the medial axis in which the number of closest points varies. Hence, it is now sensitive to the surface area of S. Note that we can also approximate Δ_M

as an integral over S itself. If we denote for a given point $\mathbf{x} \in S$ its corresponding point on the medial axis by $\hat{\mathbf{x}}$, then we define the Medial Laplacian Δ_S over S as:

$$\Delta_S f(\mathbf{p}) = f(\hat{\mathbf{p}}) \int_S \alpha(\hat{\mathbf{p}}, \hat{\mathbf{q}}) \,\mathrm{d}\mathbf{q} - \int_S f(\hat{\mathbf{q}}) \alpha(\hat{\mathbf{p}}, \hat{\mathbf{q}}) \,\mathrm{d}\mathbf{q} \tag{6.4}$$

Note that the density measure $|\tilde{\Gamma}|$ is implicitly included in the integration over S, since $|\tilde{\Gamma}(\mathbf{x})|$ is the smoothed surface area over all points $\Gamma(\mathbf{x})$.

The Medial Laplacian is invariant to deformations on S which isometrically preserve its medial axis, while also preserving each point's ball radius. As demonstrated in previous work [Zhang et al. 2005; Zheng et al. 2010], this property is often satisfied in real-world deformations, such as varying human pose. However, it is a smaller space of deformations than isometric deformations of S permit, whereas if we denote the Laplace-Beltrami operator of S by L_S, it is well-known that L_S is invariant to the space of all isometric deformations [Sun et al. 2009]. For instance, consider taking a sphere and as-isometrically-as-possible squashing it – the medial axes of these two shapes will clearly be very different. Nonetheless, in the presence of missing data, L_S can be very far from the Laplace-Beltrami operator of the true shape, where as we will show, Δ_S remains tolerant to missing data.

6.1.2 Point Cloud Medial Laplacian

We now illustrate our approach for discretizing the Medial Laplacian on a point cloud. Consider a point cloud $P = \{\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_n\}$ accompanied with normals $N = \{\mathbf{n}_1, \mathbf{n}_2, ..., \mathbf{n}_n\}$. Normals are either directly taken from the acquisition process, or if not available, then estimated via PCA and oriented with a minimum spanning tree approach.

One option for discretizing the Medial Laplacian is to consider its form defined directly on the medial axis – Δ_M. Yet as previously discussed, there exists numerous approaches for estimating the medial axis, and in the presence of missing data, each approach has various strengths and drawbacks; hence, it is unclear which to choose. Moreover, given a medial representation, it is highly nontrivial to estimate each point's density $|\tilde{\Gamma}|$.

Hence, we discretize Δ_S instead, for two main reasons. First, we do not need to define Γ, as it is implicitly included in the integration. Secondly, we may interpret the distance between points on the medial axis, $|\hat{\mathbf{p}} - \hat{\mathbf{q}}|$, as the dissimilarity in medial regions between $\mathbf{p}$ and $\mathbf{q}$. Hence, it is not necessary to explicitly measure this distance, but rather construct the *likelihood* of two points belonging to a common medial ball – our so-called medial kernel ϕ, developed in the previous chapter.

Recall that the medial kernel may be broken down into dissimilarity with respect to the emptiness of a candidate ball and tangential agreement:

$$\phi(\mathbf{p}_i, \mathbf{p}_j) = e^{-\left(\frac{\gamma(\mathbf{p}_i,\mathbf{p}_j)}{\sigma_e}\right)^2 - \left(\frac{\tau(\mathbf{p}_i,\mathbf{p}_j)}{\sigma_t}\right)^2} \tag{6.5}$$

where γ is the emptiness measure and τ is the tangential measure.

The measure of ϕ serves as an approximation to α; see Figure 6.2 for an illustration. The parameters of σ_e and σ_t are analogous to the time scale h, where by increasing σ_e and σ_t, we begin to associate points whose corresponding medial axis points are further away. We have found $\sigma_e = 5$ and $\sigma_t = 0.7$ to be suitable values, which we used for all results in this chapter.

To discretize Δ_S into the point cloud Medial Laplacian Δ_P, we replace α with ϕ and follow the integral estimation approach of Belkin et al. [2009]:

$$\Delta_P f(\mathbf{p}) = f(\mathbf{p}) \sum_{\mathbf{q}\in P} \phi(\mathbf{p}, \mathbf{q}) |\star \mathbf{q}| - \sum_{\mathbf{q}\in P} f(\mathbf{q})\phi(\mathbf{p}, \mathbf{q}) |\star \mathbf{q}| \tag{6.6}$$

where $|\star \mathbf{q}|$ denotes the dual surface area which $\mathbf{q}$ occupies.

Crucial to an accurate discretization is an accurate estimation of the dual area at every point. The approach of Belkin et al. [2009] defines the dual area at a point as the area formed by its local Delaunay triangulation. In the presence of missing data, this will effectively lead to the preservation of the inferred boundary components. We depart from Belkin et al. [2009] and derive a more nonlocal method, based on our candidate ball centers. The basic idea is to find a neighborhood of points which belong to a similar medial region, and construct a triangulation from these points in the spirit of Belkin et al. [2009], from which the dual area follows; see Figure 6.3 for an illustration of the method.

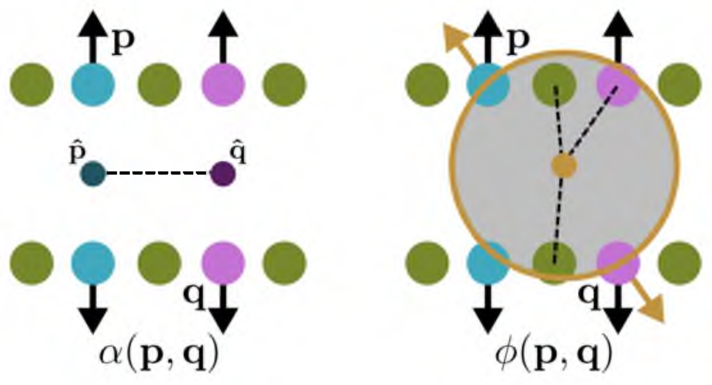


Fig. 6.2. On the left, we depict the distance between medial points α to define the Medial Laplacian Δ_S, while on the right, we show medial similarity ϕ, as an approximation to α. Note that ϕ does not require those points' corresponding medial axis points.

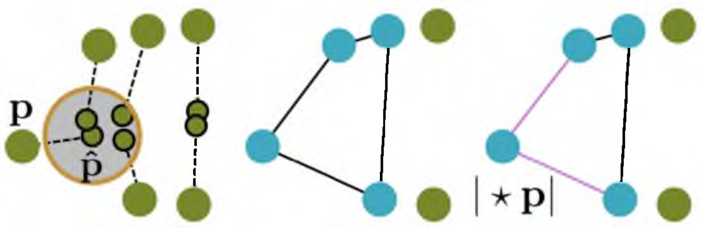


Fig. 6.3. An illustration of dual area estimation for a point p: first we gather points which belong to a similar medial neighborhood, take these points' convex hull, and use the triangles incident to p to estimate its dual area.

For each point $\mathbf{p}_i \in P$, we estimate its corresponding medial axis point $\hat{\mathbf{p}}_i$ by taking a weighted average of its candidate ball centers:

$$\hat{\mathbf{p}}_i = \frac{\sum_j \mathbf{c}_{ij}\phi(\mathbf{p}_i, \mathbf{p}_j)}{\sum_j \phi(\mathbf{p}_i, \mathbf{p}_j)} \tag{6.7}$$

We may similarly estimate its medial ball radius:

$$r_i = \frac{\sum_j r_{ij}\phi(\mathbf{p}_i, \mathbf{p}_j)}{\sum_j \phi(\mathbf{p}_i, \mathbf{p}_j)} \tag{6.8}$$

We then gather all other points $\mathbf{p}_j$ whose estimated medial axis points $\hat{\mathbf{p}}_j$ are within a small ϵ:

$$B_i = \{\mathbf{p}_j \in P \mid \ \hat{\mathbf{p}}_i - \hat{\mathbf{p}}_j| < \epsilon\} \tag{6.9}$$

where ϵ is fixed at 1.5 times the average sampling density of P. Intuitively, B_i consists of points who belong to a similar medial region of $\mathbf{p}_i$.

Next, we take the convex hull of B_i, and extract the set of triangles incident to $\mathbf{p}_i$. For concave regions, $\mathbf{p}_i$ may not reside on the convex hull; in such situations, we project all of the points to the ball formed by $(\hat{\mathbf{p}}_i, r_i)$, and then take its convex hull's triangles incident to $\mathbf{p}_i$. The dual area $|\star \mathbf{p}_i|$ follows as one-third of the area of all incident triangles.

6.1.3 Medial Diffusion

We may now construct a diffusion process from the Medial Laplacian Δ_S. The diffusion process is governed by the heat equation for a given function f defined over S:

$$\frac{\partial f(\mathbf{x}, t)}{\partial t} - \Delta_S f(\mathbf{x}, t) = 0 \tag{6.10}$$

We may then define the operator $M_t = e^{t\Delta_S}$, where $M_t f$ satisfies the heat equation for all t. Note that $M_t f$ has the effect of diffusing f along the medial axis.

We can now associate a function $m_t(\mathbf{x}, \mathbf{y})$ with M_t such that the following is satisfied:

$$M_t f(\mathbf{x}) = \int_S m_t(\mathbf{x}, \mathbf{y}) f(\mathbf{y}) \mathrm{d}\mathbf{y} \tag{6.11}$$

Intuitively, for two points $\mathbf{x}$ and $\mathbf{y}$, m_t measures the amount of heat which has diffused between the two points in time t, where heat diffusion is restricted over the medial axis; hence, we term this *medial diffusion*.

For a point cloud P, we approximate m_t through Δ_P, using the fact that m_t can be computed from the eigendecomposition of Δ_P, with $\Delta_P \Psi_i = -\lambda_i \Psi_i$:

$$m_t(\mathbf{x}, \mathbf{y}) = \sum_i e^{-\lambda_i t} \Psi_i(\mathbf{x}) \Psi_i(\mathbf{y}) \tag{6.12}$$

Analogous to the heat kernel k_t associated with the Laplace-Beltrami operator [Sun et al. 2009], m_t inherits the properties of its defining Laplacian Δ_P. For example, given a point $\mathbf{x}$, if $\mathbf{y} \in \Gamma(\hat{\mathbf{x}})$, then $m_t(\mathbf{x}, \mathbf{y})$ will be large for any time t – heat will immediately diffuse between the points. For regions containing varying ball radii, the medial diffusion will be sensitive to the surface area; see Figure 6.4 for an illustration. We see that for large cylindrical regions, the larger surface area results in fast heat dissipation; hence, m_t will be low, whereas for smaller regions, m_t will be larger.

More importantly, the diffusion is tolerant to missing data. Note that our association measure ϕ captures nonlocal relationships. For regions of missing data where at least two points indicate a medial structure, heat will diffuse in a nonlocal manner. Combined with Δ_P's insensitivity to nonrigid deformations, m_t is a useful measure for identifying similar medial regions in incomplete shapes. In Figure 6.5, we illustrate $m_t(\mathbf{x}, \mathbf{x})$ as a signature over a set of time scales t, similar to Sun et al. [2009]. Note that we are able to identify medial regions which are invariant to the pose.

Returning to our area estimation scheme, we find that dual areas can be somewhat noisy in regions which violate the medial axis prior. However, for our purposes, noisy areas are not too problematic, as we can claim the perturbation results of Sun et al. [2009]. Namely, we can write Δ_P as $\Delta_P = D^{-1}W$, where W is the symmetric weight matrix and D is the diagonal matrix containing area weights. Now, suppose that $\tilde{\Delta}_P = (D + F)^{-1}(W + E)$, where E and F are the noise weight and area matrices, respectively, with $\|E\| < \epsilon$ and $\|F\| < \alpha$ under a suitable matrix norm $\|\cdot\|$. Denoting $\tilde{M_t}$ as the heat operator of $\tilde{\Delta}_P$, then $\|M_t - \tilde{M_t}\| = O(\sqrt{\alpha} + \epsilon)$.

Intuitively, this means that the association measure between points, captured in W, has a larger impact on error than the area weights D. In our experiments, we have found that

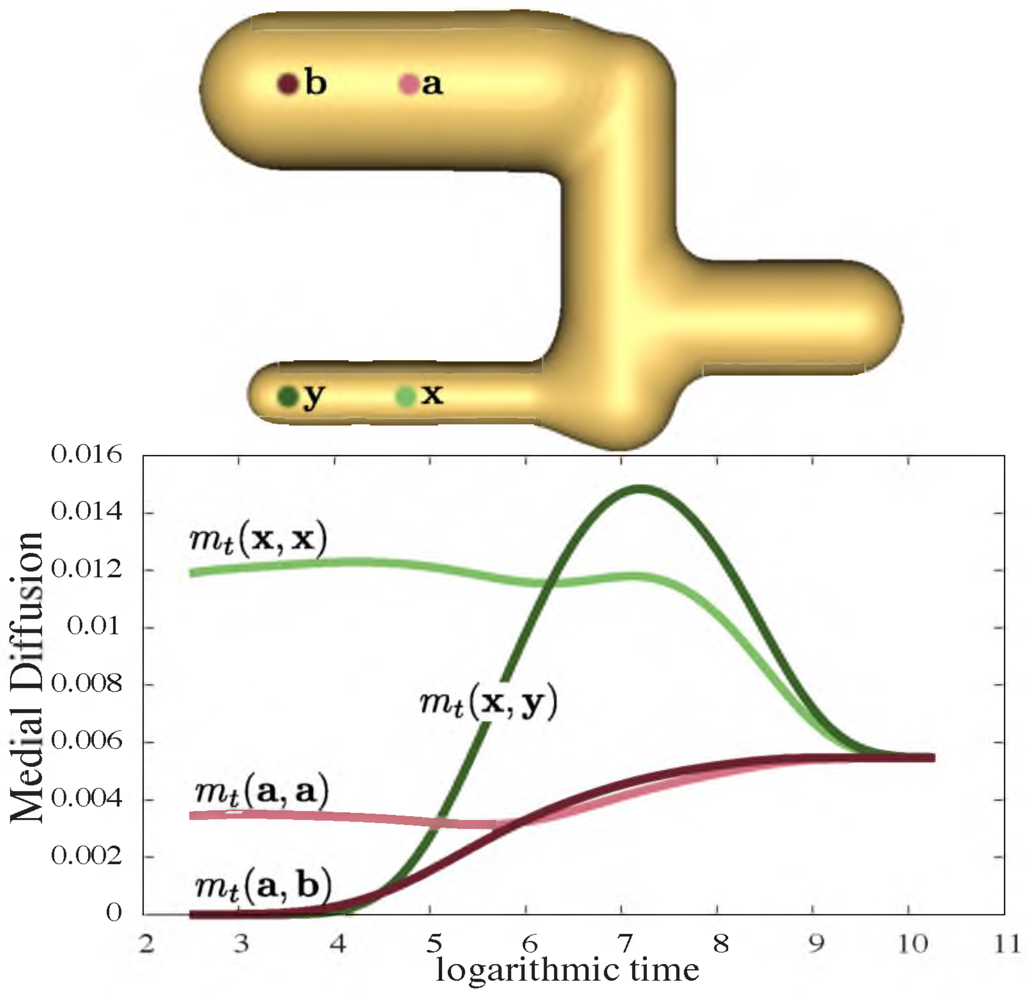


Fig. 6.4. We illustrate the behavior of medial diffusion on the shape at the top, whose corresponding medial diffusion is plotted over time. Note that the diffusion is sensitive to the volume formed by the medial balls, where at **a** heat diffuses faster than **x**; hence, it has a lower medial diffusion.

even if the estimated total surface areas between shapes are off by 10% – 15%, this has a negligible impact on our shape matching approach.

6.1.4 Combining Laplacians

An issue with medial diffusion is its behavior in regions of negative curvature, where the heat will diffuse very slowly, due to the large change in distance between medial axis points, as a function of small change in distance over the surface. This has an impact on nonrigid motion which results in significant volume change, such as regions containing negative curvature become zero or positive curvature. To address this, we can easily *combine* the standard Gaussian weight α with the medial similarity weight ϕ, so that the diffusion is less sensitive to negative curvature regions. Combining Laplace weights is common in the

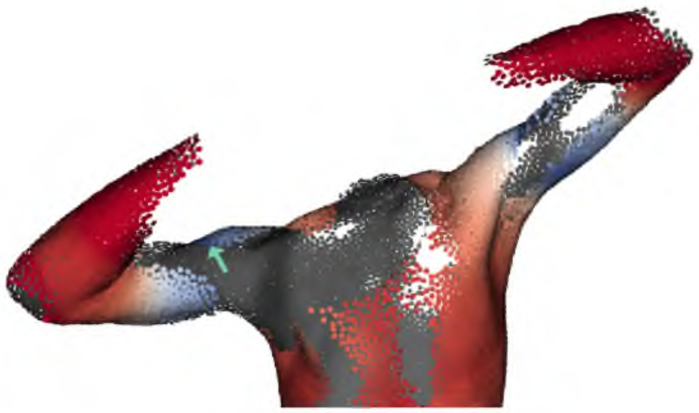


Fig. 6.5. We illustrate medial diffusion as a shape signature, similar to Sun et al. [2009], where we depict the intrinsic symmetry of the shoulders. Note the lack of data on the lower-right shoulder, whereas the lower-left shoulder contains data, yet is still recognized as having a similar signature.

spectral clustering literature, where in our situation, one may view the intrinsic geometry and the medial axis as multiple views of the same data [Zhou and Burges 2007].

To combine Laplacians, we adapt the approach of Zhou and Burges [2007]. As ϕ and α have widely varying densities, they must be suitably normalized prior to being combined. To this end, consider the weighted summations for ϕ and α:

$$d_\phi(\mathbf{p}_i) = \sum_{j \neq i} | \star \mathbf{p}_j | \phi(\mathbf{p}_i, \mathbf{p}_j) \qquad d_\alpha(\mathbf{p}_i) = \sum_{j \neq i} | \star \mathbf{p}_j | \alpha(\mathbf{p}_i, \mathbf{p}_j) \tag{6.13}$$

For a given interpolation factor η, we then combine the weights as [Belkin and Niyogi 2005]:

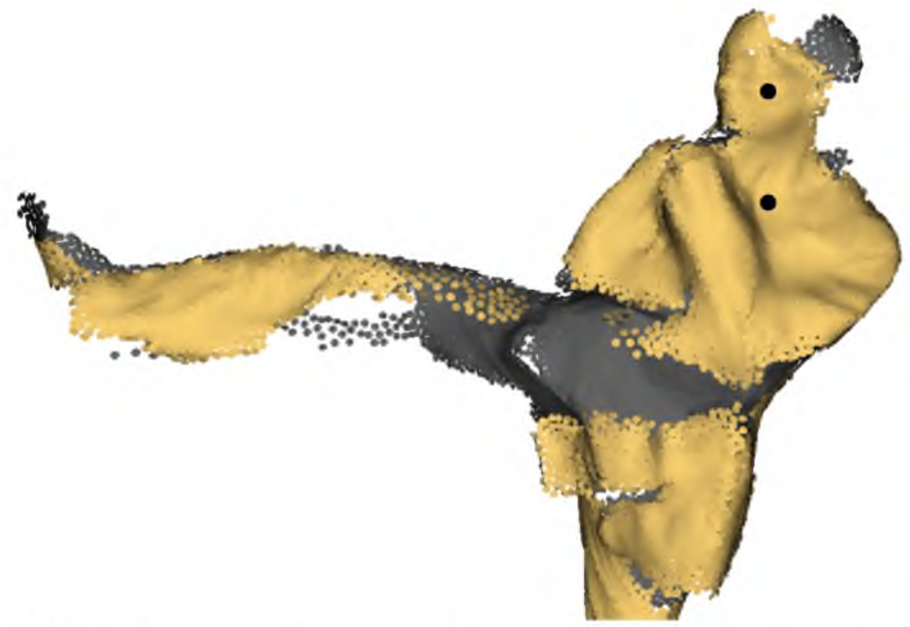
$$\sigma(\mathbf{p}_i, \mathbf{p}_j) = (1 - \eta) \frac{\phi(\mathbf{p}_i, \mathbf{p}_j)}{\sqrt{d_\phi(\mathbf{p}_i) d_\phi(\mathbf{p}_j)}} + \eta \frac{\alpha(\mathbf{p}_i, \mathbf{p}_j)}{\sqrt{d_\alpha(\mathbf{p}_i) d_\alpha(\mathbf{p}_j)}} \tag{6.14}$$

We then replace ϕ in the definition of Δ_P with σ.

We have used a value of $\eta = 0.5$ for all of our experiments. Using this setting, we find that in most regions the medial similarity term ϕ tends to dominate, and only in negative curvature regions α has an impact in the diffusion. See Figure 6.6 for an illustration of combining these weights, showing how α provides a boost in regions of negative curvature.

6.2 Shape Matching

We now describe our approach in using m_t for matching shapes. We have adapted the approach of Ovsjanikov et al. [2010], which uses the observation that, given a single landmark correspondence, k_t can be used to infer all remaining correspondences. We apply this same methodology to m_t by finding a small set of landmark correspondences, and using them to extrapolate a dense matching of medial regions. We depart from Ovsjanikov et al.



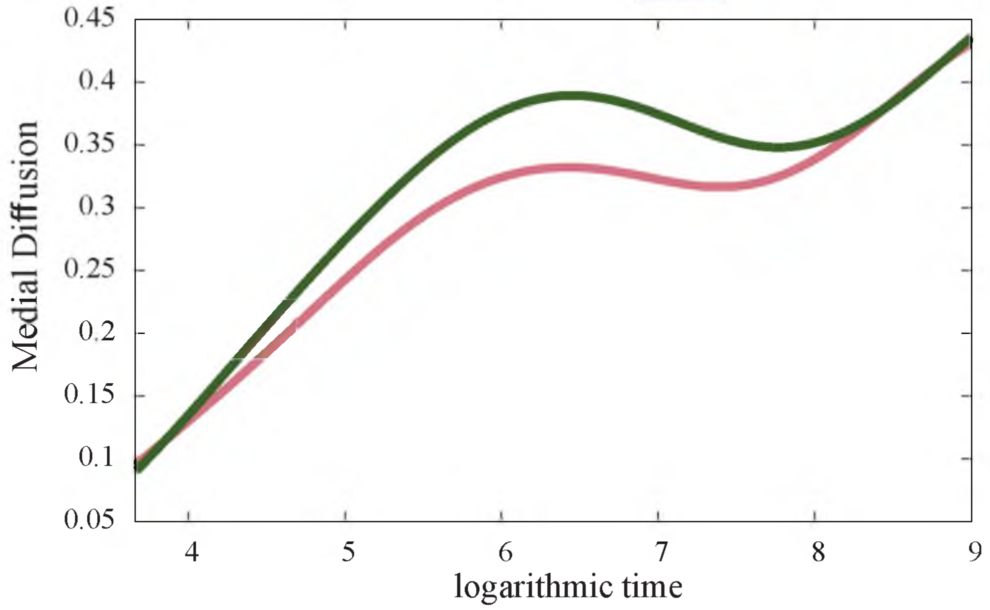


Fig. 6.6. We illustrate the effect of combining the surface-based diffusion process α with medial-based diffusion ϕ. For $\eta = 0$, or strictly using ϕ, we see that heat diffusion slows down in the region of the neck, where curvature is negative. By combining with α for $\eta = 0.5$, heat diffusion becomes less sensitive to the negative curvature.

[2010] in how we find candidate correspondence points, and how P is sampled in order to evaluate the error of a potential matching. These modifications are necessary to handle point clouds containing missing data.

Given two point clouds to be matched, P and Q, we first uniformly subsample each with respect to the medial axis. We follow the approach of Berger and Silva [2012a] by performing spectral clustering in the space of the diffusion maps formed by Δ_P,and Δ_Q, where we denote the resulting subsets as S_P and S_Q. We found 700 points to be a sufficiently fine medial representation. This results in a set of points whose corresponding medial regions are uniformly spaced.

We then choose a point $\mathbf{p}_l \in S_P$ at random, and for all points in S_Q, we choose the $\bar{\mathbf{q}}_l \in S_Q$ whose medial diffusion best matches $\mathbf{p}_l$ over all points and time scales:

$$\bar{\mathbf{q}}_l = \underset{\mathbf{q}_l \in S_Q}{\operatorname{argmin}} \sum_{\mathbf{p} \in S_P} \min_{\mathbf{q} \in S_Q} |m_\circ^P(\mathbf{p}_l, \mathbf{p}) - m_\circ^Q(\mathbf{q}_l, \mathbf{q})| \tag{6.15}$$

where $m_\circ^P(\mathbf{p}, \mathbf{q})$ represents P's medial diffusion over all times, similarly for Q. To realize this, we follow Sun et al. [2009] and logarithmically sample m over a discrete set of times, where we found 25 time scales to be sufficient, and each entry of $m_t(\mathbf{p}, \mathbf{q})$ is divided by the heat trace at t. We may then embed $m_\circ^P(\mathbf{p}, \mathbf{q})$ in a high dimensional space, and efficiently find $\mathbf{q}_l$'s minimum $\mathbf{q} \in S_Q$ through a kd-tree search, where we use the l_2 norm. To avoid searching all of S_Q for the corresponding landmark, we only consider the top 5% $\mathbf{q}_l \in S_Q$ whose signatures $m_t(\mathbf{q}_l, \mathbf{q}_l)$ are closest to $m_t(\mathbf{p}_l, \mathbf{p}_l)$. The found $\mathbf{q}_l$ is unique up to the set of points which generate the medial axis point $\hat{\mathbf{q}}_\mathbf{l}$. This redundancy is in part why our approach is robust – if the exact corresponding surface point is missing, we can instead assign a different point which generates the same medial axis point.

From this first landmark correspondence $(\mathbf{p}_l, \mathbf{q}_l)$, we greedily find additional correspondences via the same procedure, restricting the newly found correspondences to be consist with the previous ones. This can easily be accomplished by appending the previously found landmark coordinates to the new ones, as in Ovsjanikov et al. [2010]. We find new landmarks in S_P by performing a farthest-point sampling defined with respect to the diffusion map of Δ_P. This has the effect of sampling landmark points in P which are far apart in the medial axis. We denote by L the set of landmark correspondences, where we found a total of $5-6$ landmark correspondences to provide for sufficiently good results, corroborated by matching approaches [Zhang et al. 2008; Kim et al. 2011] for well-sampled shapes.

We then use the $|L|$ landmarks to extrapolate a dense set of correspondences between medial regions. This is accomplished by finding, for each $\mathbf{p} \in P$, the point $\bar{\mathbf{q}} \in Q$ in which the medial diffusion is consistent across the respective landmarks, as well as the signature $m_t(\mathbf{x}, \mathbf{x})$ [Ovsjanikov et al. 2010]:

$$\bar{\mathbf{q}} = \underset{\mathbf{q} \in Q}{\operatorname{argmin}} \sum_{l=1}^{L} |m_\circ^P(\mathbf{p}_l, \mathbf{p}) - m_\circ^Q(\mathbf{q}_l, \mathbf{q})| + |m_\circ^P(\mathbf{p}, \mathbf{p}) - m^Q(\mathbf{q}, \mathbf{q})| \tag{6.16}$$

The initial randomly chosen landmark from S_P may be a rather nondescriptive feature with respect to S_Q, resulting in a poor matching. Hence, we repeat this process (10 times in our experiments), and choose the matching which gives the lowest error in Equation 6.16, though a RANSAC-like approach [Tevs et al. 2009] could also be used.

6.3 Results

To evaluate our method, we have conducted two sets of experiments. First we measure the tolerance of our diffusion process to nonrigid motion and missing data. Secondly, we have run our matching algorithm across a set of shapes, and compared it with other similar shape matching methods.

We have used shapes in the SCAPE [Anguelov et al. 2005], TOSCA [Bronstein et al. 2008], and multiview photometric stereo (MVPS) datasets [Vlasic et al. 2009] in our experiments. For the TOSCA and SCAPE datasets, we synthetically scan the shapes using the scanner of our benchmark, as detailed in Chapter 3.

Regarding computational complexity, the largest amount of time spent in our method is constructing Δ_P. For point clouds ranging in size from 15,000–20,000, it typically takes 5-9 minutes per shape to construct the Laplacian. Hence, for all point clouds used, we have subsampled them to within this range via farthest point sampling.

6.3.1 Tolerance to Missing Data

We first evaluate the quality of our method under varying pose and missing data. One way of achieving this is to extract the medial axis of a shape, and measure the geodesic distance distortion along the medial axis. However, the medial axis is well-known to be rather unstable, and the specific medial axis simplification approach to take (see Chazal and Lieutier [2005] and Miklos et al. [2010]) is unclear.

Instead, we measure the error in m_t, in order to observe the consistency across pose and missing data. We use the SCAPE dataset, as ground truth correspondences are known. For a given SCAPE mesh, we synthetically scan it over a constant set of viewpoints. We parameterize the peak threshold at which range is accepted based on the laser intensity, giving us a controllable yet realistic means of generating missing data. For a given well-sampled rest pose P, we measure the medial diffusion error on an input point cloud Q as:

$$E(P,Q) = \frac{1}{|C||T|} \sum_{t \in T} \Big(\sum_{(\mathbf{x},\mathbf{y}) \in C} |m_t(\mathbf{x},\mathbf{y}) - m_t(f(\mathbf{x}), f(\mathbf{y}))|^2 \Big)^{\frac{1}{2}} \tag{6.17}$$

where C is a set of pairs of points uniformly sampled over Q, T is the set of logarithmic time scales, and f is the ground truth mapping function between Q and P.

See Figure 6.7 for the results. As shown, m_t remains quite stable as missing data are introduced, over varying pose. Only when large gaps of data begin to appear does the error in m_t begin to increase. At these levels, the impact of the different poses becomes evident, as the rest pose contains the lowest error.

6.3.2 Intrinsic Symmetries

The detection of intrinsic symmetries follows as a straightforward extension of the shape matching method – rather than compare two shapes, we employ m_t on the same point cloud.

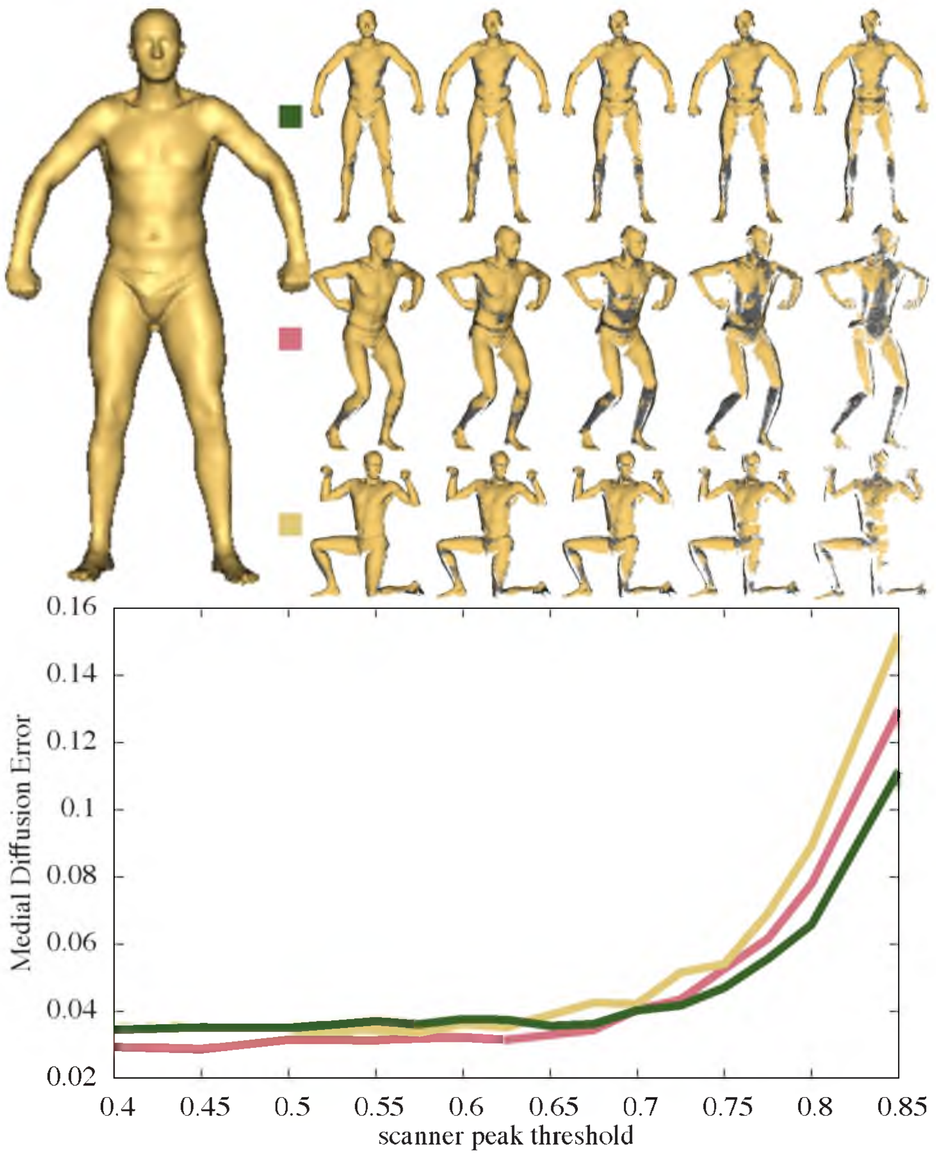


Fig. 6.7. We measure the error in medial diffusion m_t across varying missing data and pose. We show the rest pose on the upper left, and on the upper right, a subset of the poses and point clouds on which the rest pose is measured against.

Note that intrinsic symmetries in our situation refer to points along medial regions being invariant to a nonrigid self transformation. We visualize these correspondences by using the estimated medial ball centers $\hat{\mathbf{p}}_i$.

See Figure 6.8 for results on several shapes. Note that the front left leg of the cat point cloud is a separated component from the body, yet due to our method's nonlocal diffusion, we can still detect its symmetry with the front right leg.

6.3.3 Shape Matching

We now evaluate our matching approach across a set of shapes, compared with several approaches. To visualize the correspondences, for each point $\mathbf{p}_i$ on the left shape, we assign it a color based on the position of its estimated medial ball center $\hat{\mathbf{p}}_i$, and assign this same color to its corresponding point on the right shape.

Figure 6.9 shows our matching results on several shapes from the SCAPE dataset. As shown, our method is able to properly find landmark correspondences, and extrapolate a dense mapping between the shapes, despite the lack of data. In particular, note that the missing data are not consistent between the pairs of shapes as missing data occur in different regions. Our method is shown to remain highly tolerant to these imperfections.

Figure 6.10 shows matching results across several different animal categories. In the inset, we depict the similarities of the medial diffusion signature $m_t(\mathbf{x}, \mathbf{x})$ across the two cat point clouds. Note that the head of the left cat is a separated component from the rest of the body, yet we are still able to associate similarity to the neck of the right cat, despite the neck noticeably absent on the left cat.

In Figure 6.11, we compare our method with the skeleton extraction approach of Tagliasac-

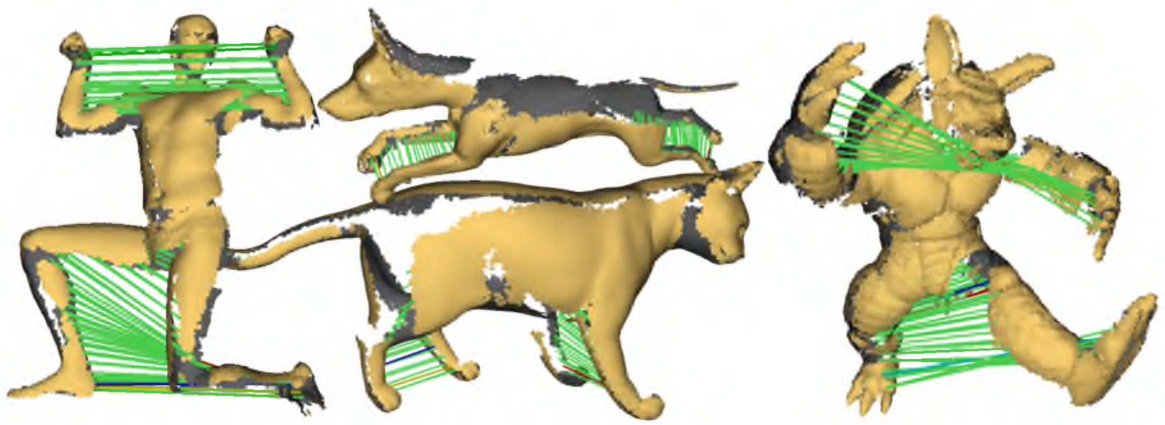


Fig. 6.8. We show detected intrinsic symmetries between medial regions for point clouds containing missing data.

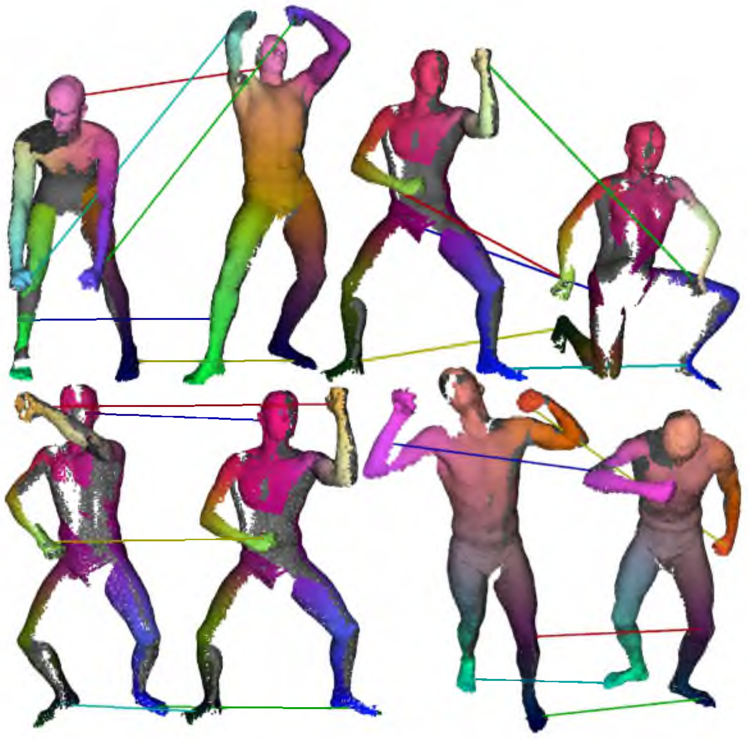


Fig. 6.9. Correspondence results on the SCAPE dataset. On the top left pair, note the absence of data on the left shape's stomach – we are able to infer that the back of the left shape and the stomach of the right shape share a medial region.

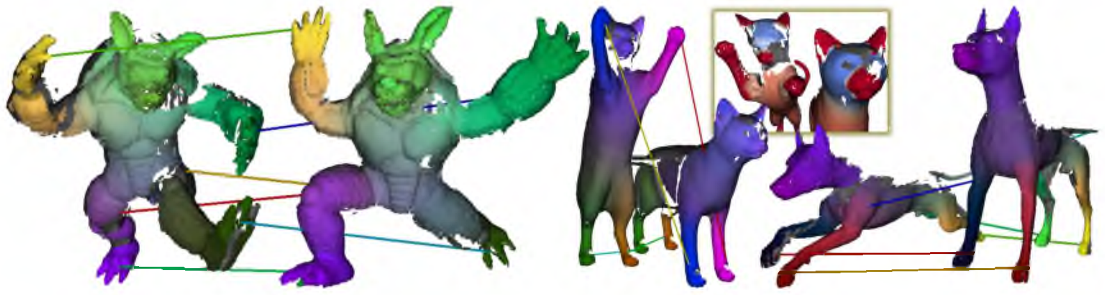


Fig. 6.10. Correspondence results across several different animal categories. In the inset, we depict the differences in the signature $m_t(\mathbf{x}, \mathbf{x})$, color-mapped across the two shapes. Note the signature's insensitivity to the lack of data.

chi et al. [2009] – the skeleton graphs used in Zheng et al. [2010]. The compactness in the representation theoretically makes matching easier, yet as shown in the left shape, a node representing the head is absent, while an additional joint is introduced near the elbow, which makes matching rather ill-posed – it is unclear which to keep and which to prune. Our method incurs no such drawback by instead operating on the entire point cloud. Furthermore, note our method's ability to match medial sheets, shown across the

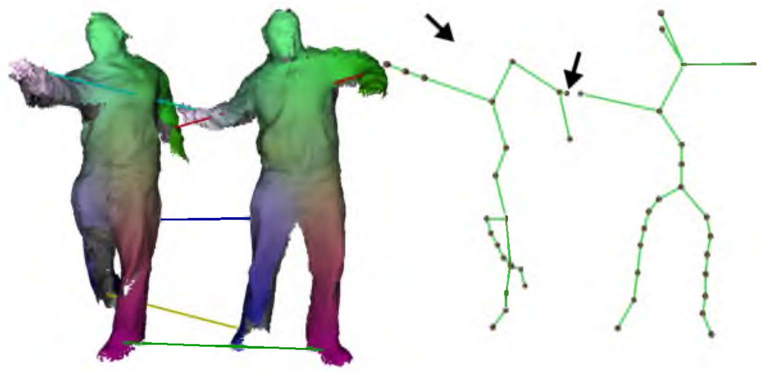


Fig. 6.11. A comparison of our approach with skeletons extracted via Tagliasacchi et al. [2009]. Note the inconsistencies in the graph skeletons, which renders such matching rather ill-posed.

chest, where by construction this is lost in Tagliasacchi et al. [2009].

Last, we have compared our method to the heat kernel matching approach of Ovsjanikov et al. [2010]. As Ovsjanikov et al. [2010] was originally designed for meshes, we instead use our strategy for finding candidate landmark points. We use the color-mapped positions $\mathbf{p}_i$ to visualize their correspondences. See Figure 6.12 for the results. As shown, the heat kernel matching approach faces difficulty in finding landmark points for the SCAPE point clouds; hence, the extrapolated correspondences are rather inaccurate. Our approach remains tolerant to the missing data. For the MVPS data, we see that Ovsjanikov et al. [2010] is able to properly find landmark correspondences for the "Abhijeet" sequence, yet the extrapolated correspondences are still imperfect. Our approach is able to properly match the right shoulder and head.

6.4 Discussion and Limitations

Although our method can handle a large range of incomplete shapes, it can perform poorly when the medial axis prior is not adequately satisfied. In particular, when the missing data are large relative to the size of the medial balls, then we may infer two separate connected components. In such situations, it is difficult to construct a shape prior which this resembles; hence, stronger priors such as templates may be necessary.

Our method is fairly robust to deformations resulting in small changes to volume, but significant volume change can be problematic. For instance, substantial folding of cloth or fluid motion can result in drastic, nonisometric changes to the medial axis. Constructing a measure which is tolerant to such deformations, as well as incomplete data, is a challenging and important area for future work.

As in most shape matching methods, our approach is vulnerable to symmetric flips. As

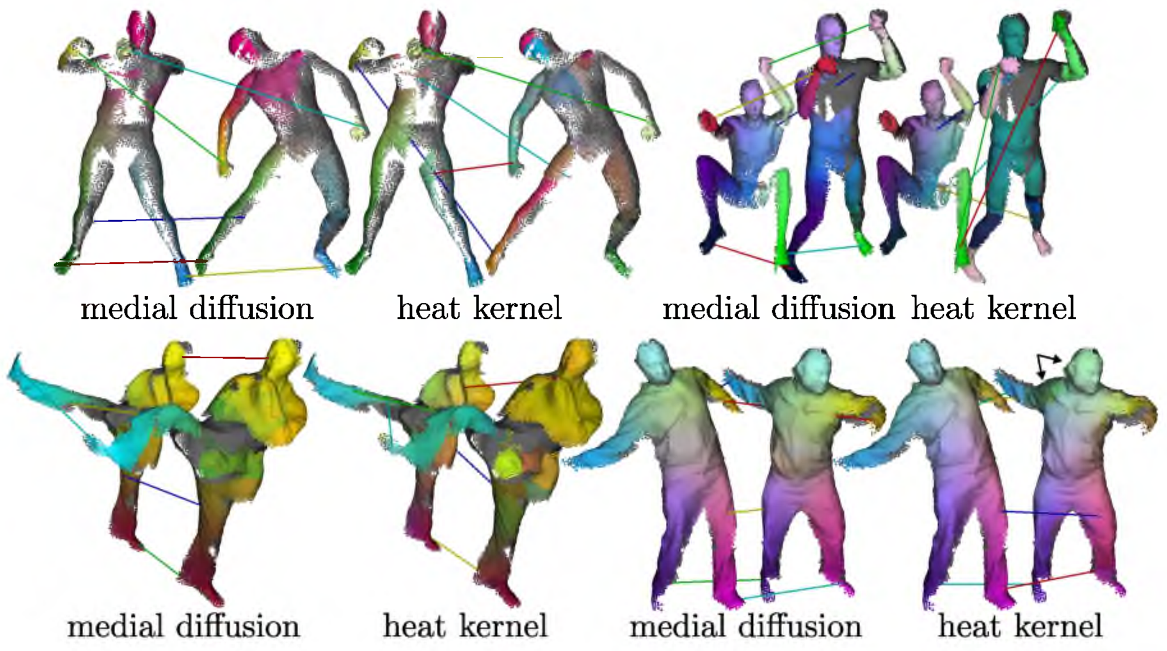


Fig. 6.12. A comparison of our method with Ovsjanikov et al. [2010]. Note how we are able to find landmarks and good-quality extrapolated correspondences, whereas significant differences in the intrinsic structures of the shapes can pose problems for Ovsjanikov et al. [2010].

shown in Ovsjanikov et al. [2011], symmetries in a shape pose a substantial challenge to any shape matching method, where there may exist multiple maps which contain equally low distortion. In our case, we suffer from confusing the symmetries in the medial axis, where Figure 6.13 shows how our method can choose the wrong correspondence due to the inherent bilateral symmetry. However, it should be possible to use either a deformation-driven approach [Zhang et al. 2008], or combining a collection of matches [Kim et al. 2011], in

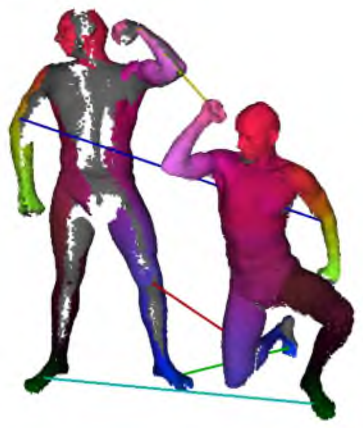


Fig. 6.13. A case where our method gives the wrong correspondence due to intrinsic symmetries between the shapes.

order to resolve this limitation.

6.5 Summary

In this chapter, we have presented a method for matching incomplete shapes undergoing nonrigid motion. Our main contribution is the construction of a diffusion process on the point cloud which measures heat diffusion along the medial axis. As the medial axis is a strong prior for missing data, we have shown how heat diffuses in a nonlocal manner, insensitive to both nonrigid motion and missing data, and how this may be used for matching incomplete shapes.

CHAPTER 7

MULTIRESOLUTION SURFACE REMESHING

In previous chapters, we have considered surface reconstruction, and how shape analysis may benefit the pursuit of geometrically and topologically accurate reconstruction. For certain applications, the quality of the triangles in the reconstructed surface mesh can be just as important as the faithfulness to the original surface. In particular, a hierarchy of good-quality meshes has a large range of applications; for instance, in multigrid approaches for solving PDEs, it is necessary for each mesh in the hierarchy to be of acceptable quality.

In this chapter, we introduce a novel method for *multiresolution remeshing* which is intrinsic to the surface. We propose a top-down, binary, hierarchical surface decomposition to generate well-formed surface patches at every scale. Namely, we utilize the first nontrivial eigenfunction of the Laplace-Beltrami operator to drive the decomposition. This has a natural analogue in the area of graph theory, a process known as *spectral bisection* [Biyikoglu et al. 2007], where a combinatorial or weighted Laplacian is used. The first nontrivial eigenvector used to drive the decomposition is known as the *Fiedler vector*. We adapt this notation to coin our structure the *Fiedler tree*.

By utilizing the Laplace-Beltrami operator instead of the combinatorial Laplacian, we obtain many nice properties: surface patches of uniform area, well-shaped surface patches, mesh-independence, and noise robustness, among others. Moreover, we are able to generate high-quality uniform meshes at multiple scales. Uniform in our case refers to uniform triangle areas and consistently good-quality in the resulting triangles, measured by the triangle radius ratio metric. As our approach is intrinsic to the surface, we completely avoid issues related to operating in the ambient space of the mesh, where existing approaches [Cheng et al. 2007; Yan et al. 2009] face difficulties.

Due to the properties of our construction, we argue that we have a well-defined notion of *scale* on the surface. This provides for a natural means of constructing wavelets on a surface, as scale is notoriously difficult to define on a sampled manifold [Kobbelt et al. 1998;

Guskov et al. 1999]. As an application, we illustrate the construction of a Haar wavelet basis, and from this wavelet basis, a trivial means of producing feature-sensitive meshes.

Figure 7.1 illustrates such flexibility, showing from left to right three different resolutions for the model on the left. Two different representations are presented for each resolution level, illustrating the capability of generating high-quality uniform meshes (top) as well as adaptive meshes capturing surface features (bottom).

Our contributions are summarized as follows:

– *Quality Irregular Multiresolution:* We are able to generate a hierarchy of quality meshes, a task difficult to achieve with respect to current remeshing and simplification schemes.

– *Embedding Independence:* As our decomposition, and corresponding meshes, are completely determined by the Laplace-Beltrami operator, our meshes are intrinsic to the surface.

– *Noise Robustness:* Utilizing the Fiedler vector, we are able to produce quality triangulations even in the presence of high-frequency noise.

– *Multiscale Analysis:* The binary hierarchy permits a multiscale analysis very similar to a Haar wavelet decomposition, making noise and feature identification quite natural.

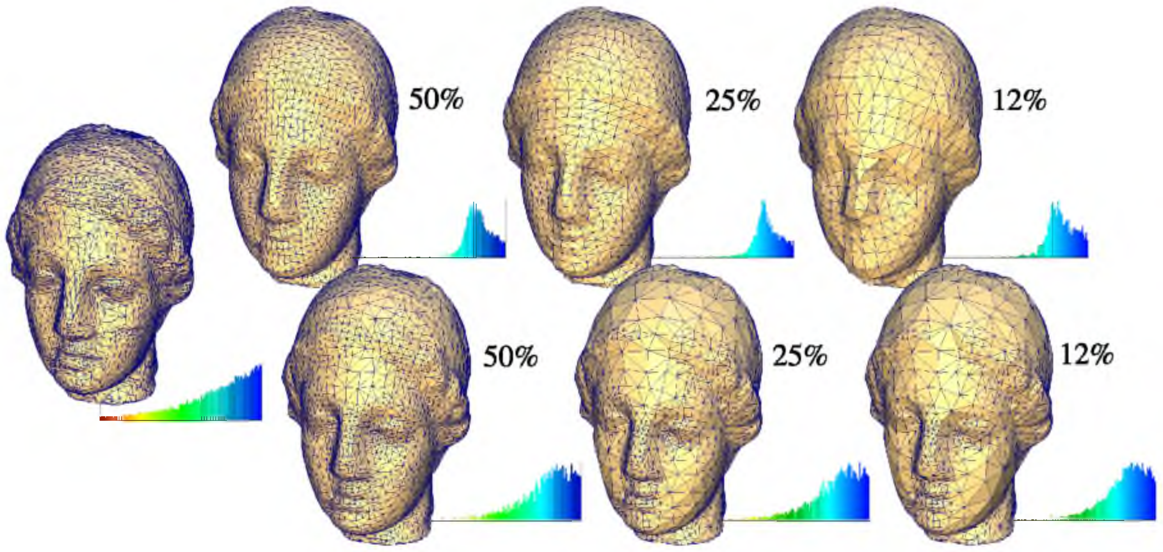


Fig. 7.1. Overview of our Fiedler tree approach. From the original egea model on the left, we are able to generate quality uniform meshes at different scales (top row). Due to the hierarchical nature, feature-sensitive meshes are also easily generated (bottom row)

7.1 Fiedler Binary Tree Decomposition

The proposed framework relies on a binary hierarchical structure to carry out the multiscale decomposition. Once the hierarchical structure is established, a CW complex is constructed from which a triangulation can be built. Details on how to accomplish the tree construction follows in this section, while triangulation is handled in Section 7.2.

7.1.1 Tree Construction

In order to construct a binary decomposition of the surface mesh, we require a mechanism to recursively split the mesh in two parts. Partitioning a surface into two surface patches amounts to finding a cut along the surface, or equivalently, finding a series of curves which splits the surface into two connected components. We utilize the nodal regions of the Laplace-Beltrami eigenfunctions to make these splits. Namely, we use the first nontrivial eigenfunction of the Laplace-Beltrami operator, which in graph theory circles is commonly referred to as the *Fiedler vector*, when considering the more general Laplacian. Splitting along the zero-set of the Fiedler vector ensures a split of the surface into exactly two connected components from the Courant Nodal Domain theorem [Gebal et al. 2009], hence ensuring a binary decomposition.

To this end, we employ the discrete Laplace-Beltrami operator of Vallet and Levy [2008], utilizing dual barycenter areas. In the computation of the Fiedler vector we also use the method of Vallet and Levy [2008] in performing a spectral shift, in order to ensure a faster convergence in eigenvector computations.

Once we have computed the Fiedler vector on the original surface, we isocontour the zero set, assuming linear interpolation, to split the mesh in two patches. From the two newly created surface patches, we simply recurse this process until a user-defined level of the tree is met. See Figure 7.2 for an illustration.

Note that we exactly isocontour the surface, rather than respect the original connectivity of the surface. By exactly cutting along the zero set, we are not inhibited by highly irregular meshes where portions of the surface may have large triangles, and others may have small triangles. Therefore, we are able to keep the notion of scale on a surface mesh independent.

For numerical robustness, we take care of instances where the Fiedler vector contains values *approximately* zero. If the value of the Fiedler vector in a vertex is very close to zero, then the resulting submesh may contain very poor triangles, for instance skinny triangles, posing numerical instability issues for the eigenvector computations. We assign these vertices a small random value, within a range that will render the submesh numerically stable.

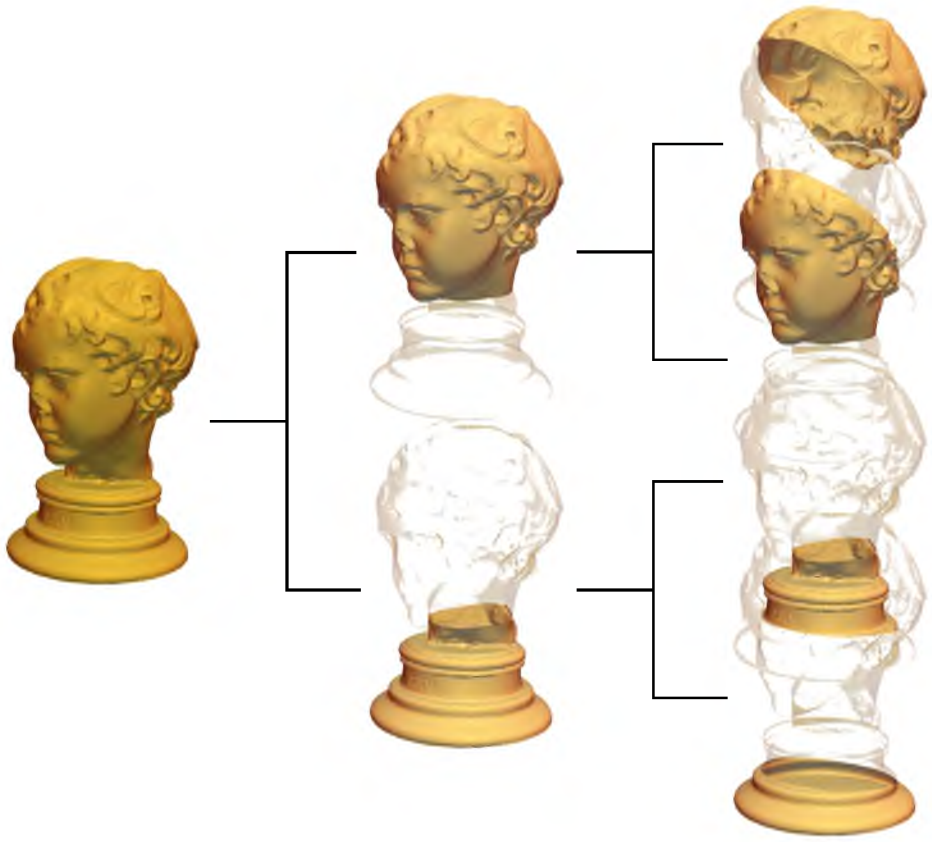


Fig. 7.2. Binary mesh decomposition: each patch (tree node) is recursively split on the Fiedler nodal line.

When splitting the mesh, every triangle along the nodal line is split into three separate triangles, where two triangles will be assigned to one of the submeshes, and the other triangle to the other submesh. This results in a significant amount of triangles being created at finer scales, though we suspect that symbolically cutting triangles similar to Yan et al. [2009] is a viable alternative and would save much memory.

7.1.2 Fiedler Tree Properties

By splitting along the Fiedler vector, we inherit several attractive properties in our decomposition. The Fiedler vector is known to be a good approximation to the normalized min-cut [Shi and Malik 2000] in the segmentation literature. For the decomposition of a surface Ω into $\Omega = \Omega_1 \cup \Omega_2$, we recall the cut energy as:

$$Ncut(\Omega_1, \Omega_2) = \frac{cut(\Omega_1, \Omega_2)}{assoc(\Omega_1, \Omega)} + \frac{cut(\Omega_1, \Omega_2)}{assoc(\Omega_2, \Omega)} \tag{7.1}$$

where *assoc* is a measure of similarity between two domains, and *cut* measures the dissimilarity in the boundary between Ω_1 and Ω_2.

In contrast to segmentation approaches, our measure of similarity is entirely uniform, in that by using the Laplace-Beltrami operator, we are only considering the intrinsic geometry for the purposes of segmentation. Thus, if we denote $\xi = \Omega_1 \cap \Omega_2$ as the nodal set, we in fact have:

$$Ncut(\Omega_1, \Omega_2) = \frac{l(\xi)}{s(\Omega_1)} + \frac{l(\xi)}{s(\Omega_2)} \tag{7.2}$$

where l represents the length of a curve, and s represents surface area. Thus, in our case, the Fiedler vector approximates the minimization of the ratio of nodal set length to surface area [Szlam et al. 2005]. As a result, for every split, we are likely to obtain surface patches which are of similar surface area, while the split itself is of small length, and typically of small Gaussian curvature magnitude on the boundary. We argue that both of these properties give rise to a well-defined notion of *scale* in the decomposition.

Our tree construction is also mesh-independent. That is, for a given surface meshed in two different ways, our construction will produce identical decompositions. Seeing as the Laplace-Beltrami operator is isometry-invariant, this should come as no surprise. Only at very fine scales does the decomposition begin to differ, due to using linear interpolation in making the cuts. Figure 7.3 illustrates the mesh-independent property, showing patches in three different levels of the hierarchy. Notice that patches are practically indentical in the top and botton rows, even though the construction is performed with respect to completely different meshings (the left-most models).

Last, it has been illustrated in previous works [Levy 2006; Gebal et al. 2009] that the Fiedler vector, in some sense, follows the "shape" of the surface. For the purposes of our construction, we find that for tubular and anisotropic surface patches, the zero set of the Fiedler vector consistently aligns with the maximum principal curvature directions. In other words, the cut tends to be along the minimum axis of the surface, and as a consequence, effectively removes the anisotropy of the surface. See Figure 7.4 for an illustration.

7.2 Triangle Mesh Generation

Producing a triangulation from the tree construction involves topological and geometric considerations. We handle both in turn.

7.2.1 Topological Construction

At the end of the tree construction process, we are left with a set of surface patches at all scales. At some scale, each surface patch will become homeomorphic to a topological disk. At this scale, we have in fact constructed a *cell complex*, or CW-complex. For the space of a 2-manifold, a CW-complex consists of a set of 0, 1, and 2-cells, where an n-cell

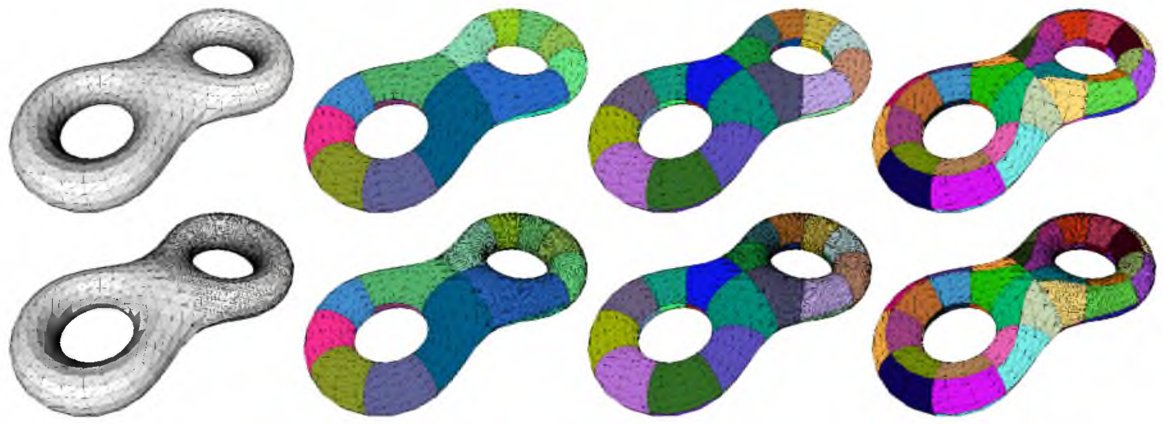


Fig. 7.3. Intrinsic nature of the hierarchical decomposition. Almost identical decompositions are generated from meshes with varying refinement.

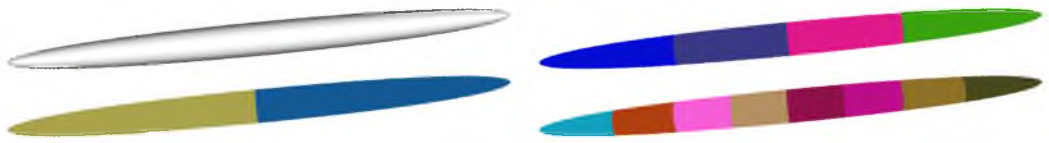


Fig. 7.4. Our decomposition tends to split along the minimum axis, and consequently along maximum principal curvatures, as illustrated for an ellipsoid.

is homeomorphic to an n-ball, and the boundary of an n-cell strictly consists of cells of dimension $m < n$ [Munkres 1993]. In our context, 2-cells are the surface patches, 1-cells are arcs on the boundary of the patches whose ends are the 0-cells, or vertices.

The significance of the CW-complex for our purposes lies in the fact that, under certain circumstances, its dual complex is a valid triangulation. The dual complex of the CW-complex takes every n-cell and maps it to a unique $(2 - n)$-cell, such that every 2-cell becomes a point, every 1-cell becomes an edge, and a 0-cell becomes a facet. Each 0-cell will map to a triangle if and only if the number of 1-cells which intersect to form the 0-cell is exactly three. As our tree construction always cuts every edge the zero set crosses, open zero sets along the surface will always start/end at unique points, and consequently, we are always guaranteed triangle elements.

The only remaining issue is whether or not the dual complex is indeed a valid triangulation. There are three cases where zero set cuts will result in invalid triangulations, which correspond with violations of the *closed ball property* [Edelsbrunner and Shah 1997]:

- The zero set is closed.

- The zero set consists of multiple connected components.

– The zero set starts and ends at the same 1-cell.

The first case results in a dangling edge, the second case results in a degenerate triangle, and the third case results in the creation of duplicate triangles. Hierarchical space partitioning approaches [Schaefer and Warren 2003; Boubekeur et al. 2006] suffer from similar problems; however, since we are partioning the surface directly, we may trivially detect these cases. We find that the first two cases only occur in coarse levels of the tree, as when we approach finer levels, the 2-cells begin to resemble developable, convex, topological disks, for which the zero set is known to be open and of a single component [Melas 1992]. The third case, however, may still occur at any level, although in practice, it is rare to occur at finer levels of the tree. In all examples in this chapter, we have found that the closed ball property is first satisfied at a rather coarse level, and is consistently satisfied at all finer levels.

Care must be taken in the implementation of this hierarchical CW-complex for the purposes of memory efficiency. To this end, we only store the triangles of the finest-scale CW-complex; that is, we label the triangles in the finest level in accordance with patches in that level. Moreover, ids are assigned such that the multiresolution structure is maintained. In other words, if a triangle has a label k in the finest level, then it will be labeled in its father node as $\lfloor \frac{k}{2} \rfloor$, ensuring a consistent hierarchical labeling scheme. Therefore, a patch with id k at level j is labeled as $2k$ or $2k+1$ at level $j+1$ (the same being valid for the triangles representing these patches). Hence, we are always able to process the CW-complex at any scale, strictly from the finest scale.

We next illustrate two mechanisms for generating meshes: multiresolution uniform meshes, and quadric error meshes.

7.2.1.1 Multiresolution Uniform Meshing

Generation of a uniform mesh amounts to reconstruction at a particular depth, or scale, in the tree. Namely, for a prescribed resolution j, we identify the patches corresponding to depth j using the scheme as described above. This effectively corresponds to the CW-complex at scale j. From here, we identify the 0-cells to be the triangles in the dual triangulation, where a dual triangle's vertices are determined by the intersecting three 2-cells. This construction guarantees an oriented simplicial complex decomposition of the surface. Spatial partioning approaches [Rossignac and Borrel 1993; Schaefer and Warren 2003], on the other hand, encounter difficulty in ensuring a decomposition that guarantees a well-defined simplicial complex as output, as issues may occur in clustering points which are close in Euclidean distance yet far apart in geodesic distance.

7.2.1.2 Quadric Error Meshing

Similar to previous approaches [Schaefer and Warren 2003; Boubekeur et al. 2006], we may utilize our spatial decomposition for the purposes of applying quadric error-based decimation [Garland and Heckbert 1997]. The primary difference here is that we have well-defined surface patches, both in terms of shape and uniform area across all scales, whereas spatial partitioning approaches greatly suffer from nonuniformity as a result of axis-aligned spatial decompositions.

We prioritize nodes of the tree starting from the finest level, where the priority is the quadric error metric. When two neighboring nodes have both been removed, we may add their parent to the queue for processing. When adding parent nodes to the queue, we may simply add their child quadric error functions together; however, note that since we have a binary tree structure, it is relatively inexpensive to compute the quadric error function from scratch. In fact, it is $O(|V|log|V|)$ in the number of vertices $|V|$, whereas Garland and Heckbert [1997] rely on edge collapses, and consequently, it would be quadratic in their approach.

Once we have selected the subset of nodes to be retained, we need to generate the dual triangulation. We associate each 2-cell with its scale and id, and then generate a unique id for each (scale, id) pairing. This gives us a consistent CW-complex representative of the quadric error decimation. Generation of the dual triangulation then proceeds in exactly the same manner as above.

See Figure 7.5 for a comparison between our approach and qslim. Note that the results are quite similar; however, the order of complexity of our approach is $\frac{|V|}{2}$, where $|V|$ is the number of vertices, whereas qslim works off of edge collapses; hence, the complexity for a typical mesh with qslim is of the order $3|V|$, which is roughly the number of edges.

7.2.2 Geometric Embedding

In computing a representative vertex for every 2-cell, its center of mass is a logical choice. That is, for every 2-cell, we may take the area-weighted coordinate as the vertex position.

A disadvantage to using the center of mass is that we may miss features on the surface. If feature preservation is desired, we may position vertices according to the quadric error metric, taken with respect to the 2-cell. By doing so, however, our mesh quality suffers. To satisfy both ends, we opt to interpolate between the center of mass and the quadric error vertex, by a user-defined parameter α. This way, the user may choose between high-quality triangulations and feature preservation.

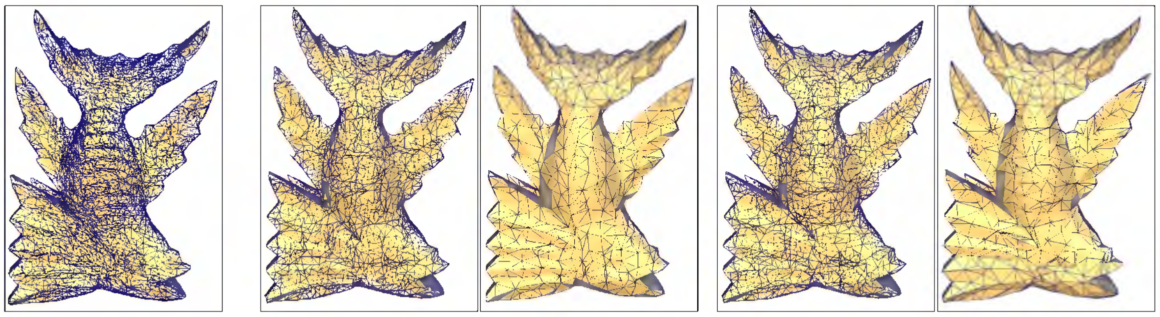


Fig. 7.5. QSlim decimation (middle), compared to our quadric error meshing approach (right). Eigenvector computation time: 14s. Qslim timing for 4K and 1K vertex decimation, respectively: 44ms and 54ms. Our timing for 4K and 1K vertex decimation, respectively: 31ms and 57ms.

7.2.3 Triangulation Properties

If we are to use the center of mass for vertex positions, then our construction is able to produce high-quality triangulations. This is a consequence of the tree construction properties discussed in Section 7.1.2. The fact that the nodal curves tend to follow the maximum principal curvature directions results in edges in the dual triangulation following the minimum principal curvature directions. This also accounts for the "quad-like" structure in our meshes, and consequently, our triangles are slightly anistropic in the principal directions of the curvature tensor. As well, the property of surface patches being of almost uniform area for each level results in triangles containing very similar areas in the dual triangulation. See Figure 7.6 for an example illustrating these properties across several scales.

Simultaneously satisfying small-length nodal curves and equi-areal surface patches is rather difficult, and occasionally, the Fiedler vector will favor one over the other. In the former case, this will result in nonuniform surface areas, and hence, the dual triangulation will have triangles of varying areas. In most cases, however, we have noticed this to be desirable; for instance, the legs of the horse in Figure 7.6 should be meshed denser than the stomach. In the latter case, nodal curves may result in surface patches being nonconvex, in which case, skinny triangles and high-valence vertices are produced. In practice, we have observed that this rarely occurs.

The property of mesh independent tree constructions in fact translates to near identical triangulations. See Figure 7.7 for an example. Note that there are subtle differences in the meshes, as neighboring 2-cells may differ, corresponding to a difference of an edge flip in the triangulations.

Last, we note that our meshes are very robust to geometric noise. As pointed out in

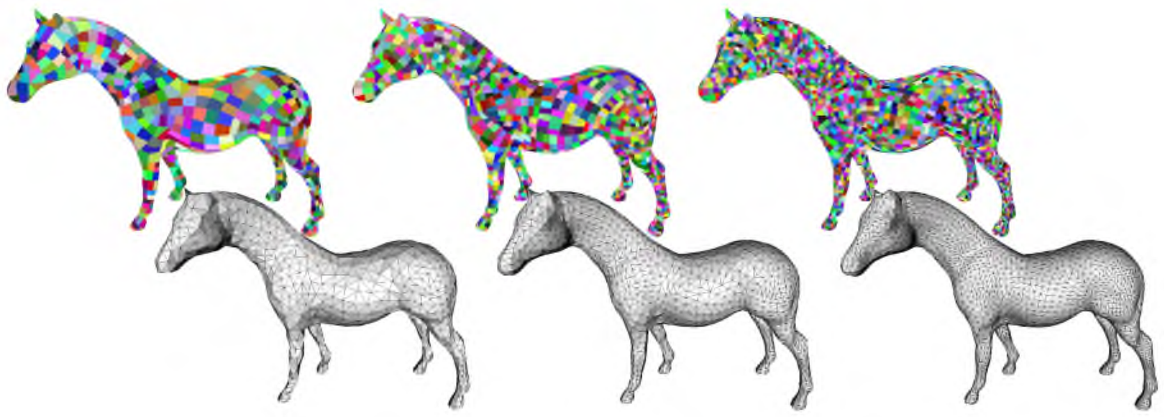


Fig. 7.6. The CW-complex and corresponding triangulations, for different scales. Note the consistency in the quality of the decomposition, as we go to finer scales.

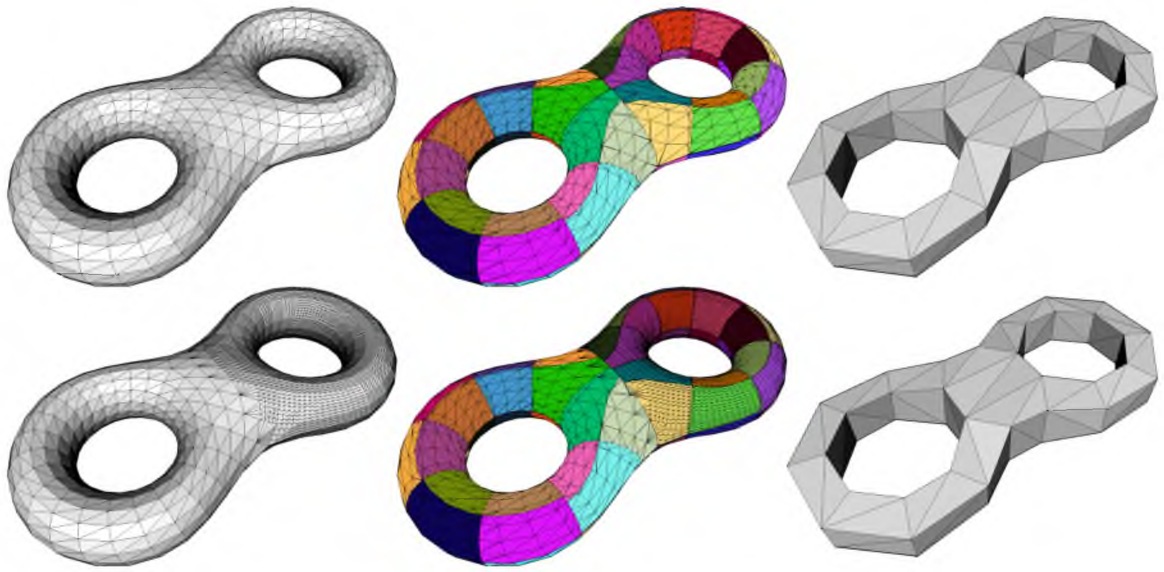


Fig. 7.7. Mesh generation for the eight model from two different meshings of the same surface.

previous work [Rustamov 2007], the low-frequency eigenfunctions of the Laplace-Beltrami operator are robust to even topological noise, in addition to geometric noise. The Fiedler vector being the lowest frequency nontrivial eigenfunction, it is most robust. This is a property inherited throughout our hierarchy, as Figure 7.8 illustrates. The noise in this example is generated by perturbing the per-vertex normals, and displacing the vertices a small amount along this perturbation. We are additionally able to produce high-quality triangles in the presence of noise, as our triangle radius ratio histograms demonstrate.

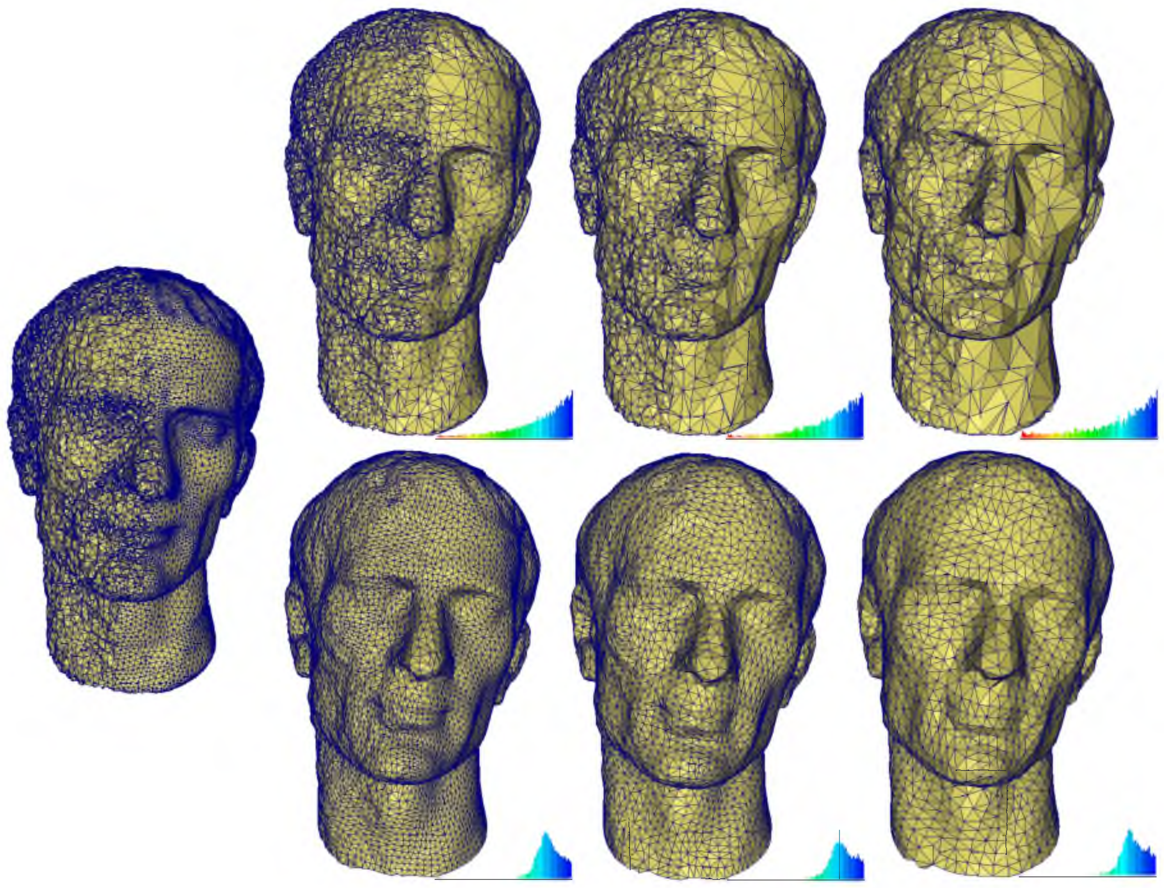


Fig. 7.8. Multiscale representation of half-noise Julius model (left most). Our approach (bottom row) can robustly smooth noise out while still producing good-quality meshes in every level of the hierarchy. The noise remains prevalent when QEM is used as a simplification mechanism, thus preventing the generation of good meshes. Histograms on the bottom right of each model show the triangle radius ratio quality for each model.

7.3 Fiedler Multiscale Analysis

Multiscale analysis usually relies on recursively decomposing a given signal into low-frequency and high-frequency components. Although different approaches can be used to compute low-frequency and high-frequency components of a signal in each resolution, such as expansion in a set of basis functions or prediction/updating schemes [Jansen and Oonincx 2005], all multiscale methods demand a splitting mechanism (also called up-sampling) in order to identify the subset of data that will be "shifted" to the next coarser level. Efficient splitting schemes are particularly difficult to be defined on unstructured data, as a biased choice might introduce artifacts in the multiscale decomposition. Our hierarchical scheme, however, provides for an intuitive notion of *scale*, and hence is an attractive starting point for many multiresolution methods. We illustrate such functionality by implementing a Haar-like multiscale analysis using our decomposition as a splitting mechanism.

Let γ_k^j be a surface patch with index k at scale j of the tree. Denoting by γ_{2k}^{j+1} and γ_{2k+1}^{j+1} the children nodes of γ_k^j, we can compute scaling and detail coefficients c_k^j, d_k^j in γ_k^j by simple averaging and differencing from scaling coefficients c_{2k}^{j+1} and c_{2k+1}^{j+1} in γ_{2k}^{j+1} and γ_{2k+1}^{j+1}. More specifically, scaling and detail coefficients in level j can be computed as [Jansen and Ooninckx 2005]:

$$c_k^j = \frac{|\gamma_{2k}^{j+1}|}{|\gamma_k^j|} c_{2k}^{j+1} + \frac{|\gamma_{2k+1}^{j+1}|}{|\gamma_k^j|} c_{2k+1}^{j+1} \tag{7.3}$$

$$d_k^j = c_{2k}^{j+1} - c_{2k+1}^{j+1} \tag{7.4}$$

where $|\gamma_k^j|$ is the area of the surface patch k at scale j. At the finest scale J, we take the c_k^J to be the area-weighted average of the function values on that surface patch (assuming the function is constant in each patch of the finest level). Similarly, an inverse transform may be applied as follows:

$$c_{2k}^{j+1} = c_k^j + \frac{|\gamma_{2k+1}^{j+1}|}{|\gamma_k^j|} d_k^j \tag{7.5}$$

$$c_{2k+1}^{j+1} = c_k^j - \frac{|\gamma_{2k}^{j+1}|}{|\gamma_k^j|} d_k^j \tag{7.6}$$

The capability of computing scaling and detail coefficients complements the binary hierarchical decomposition with a natural mechanism to detect features and surface details. In fact, we may utilize the Haar wavelet decomposition for the purposes of detecting multiscale features in the mesh. To this end, we analyze the variation in per-vertex normals. If we denote the components of normal vectors as functions n_x, n_y, n_z over the surface, we may run our Haar decomposition, as described in Equation 7.4, to obtain wavelet (detail) coefficients dx, dy, and dz for each coordinate function, respectively. By setting $\mathbf{d}_k^j = (dx_k^j, dy_k^j, dz_k^j)$ as a vector in every node k at scale j, we can take $\| \mathbf{d}_k^j \|$ as a feature measure at node k (and level j) of the tree. An example of such a Haar-like decomposition can be seen in Figure 7.9, where the warmer colors in the bottom models represent high values of detail coefficients. Notice that by going from right to left, more details are added in the model, characterizing the typical behavior of a multiscale scheme.

Scaling and detail coefficients may also be exploited for the purposes of feature detection and vertex positioning during the multiresolution process. In fact, we have exploited the Haar-like multiscale analysis for feature-sensitive meshing. The feature measure described above may be easily leveraged to produce adaptive meshes; that is, meshes where the sampling density is a function of the features of the mesh. This is achieved by culling nodes (2-cells) from the tree in a greedy manner prioritized by $\| \mathbf{d}_k^j \|$.

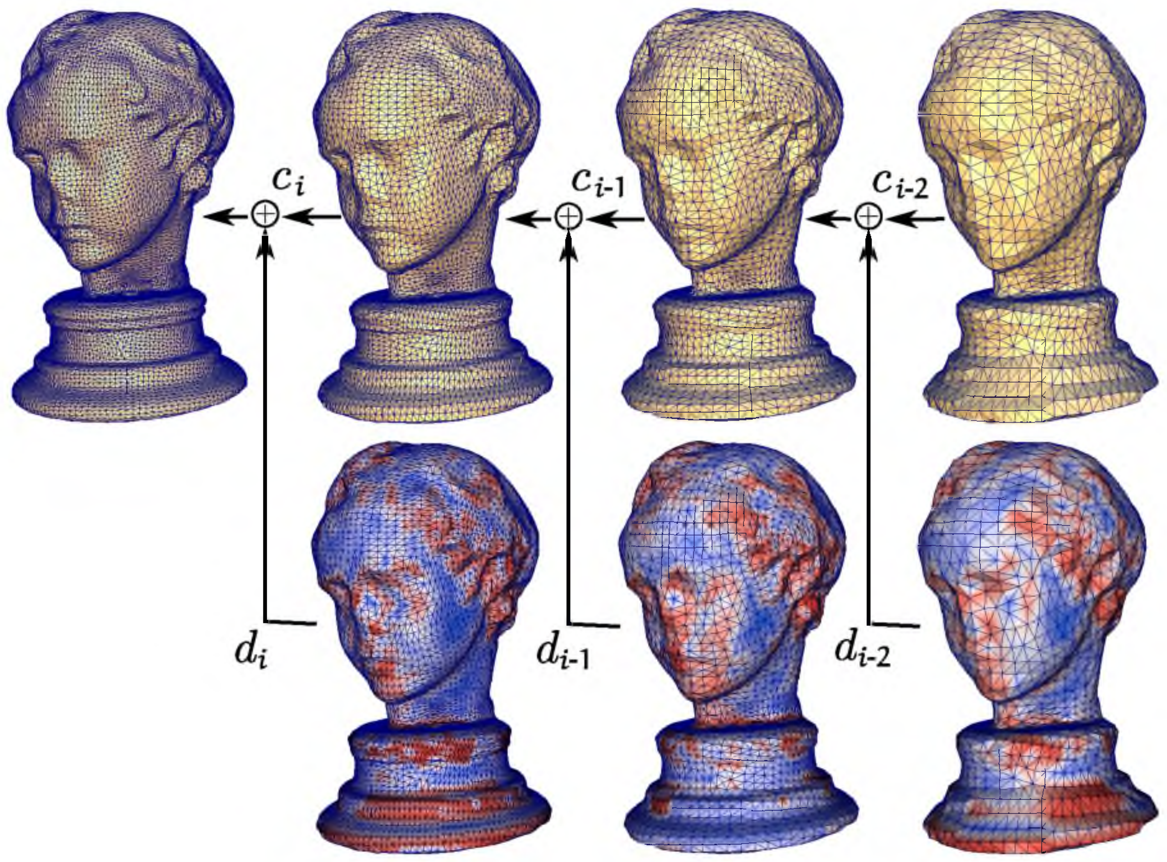


Fig. 7.9. Illustration of the multiscale decomposition of the normals. From right to left, we are adding more details to the model, until we get the original surface back.

Similar to the quadric error meshing, we first place all leaf nodes in the tree in a priority queue. A tree node is added to the queue only if its children have been removed. Additionally, in order to maintain nice triangulations and prevent high valence vertices, we do not allow the merging of two nodes n_{2k}^{j+1}, n_{2k+1}^{j+1} into n_k^j if a child of the neighbor node of n_k^j still exists. Once all nodes have been removed, the triangulation is generated in the exact same manner as Section 7.2.1. This adaptive mechanism was used to generate the bottom models in Figure 7.1.

In Section 7.2.2, we demonstrated a means of computing the center of mass over every surface patch. This is unfortunately of complexity $O(|V|log|V|)$ to compute. However, we may make the computation linear by noting that the projection of the coordinate functions onto the Haar basis exactly corresponds to the center of masses at different scales. That is, the scaling coefficients of the coordinate functions at a particular scale correspond to the center of masses computed at that scale. Only the finest scale integration needs to be computed.

7.4 Experimental Results

In this section, we present the results of applying the described methodology for the purposes of generating multiresolution uniform meshes and feature-sensitive meshes. All the models presented in the following applications were generated on a MacBook with a dual-core processor of 2 GHz and 2 GB of memory.

While minimum angle in a triangle is a common quality measure in the remeshing literature, we find that our meshes are slightly anisotropic in the curvature tensor; see Section 7.2.3 for a discussion on this matter. Hence, minimum angle is not a fair measure of quality for our meshes. For this reason, we measure mesh quality by the incircle to circumcircle ratio, commonly referred to as the radius ratio.

Figure 7.10 demonstrates our results for a variety of surface meshes, uniform and adaptive meshing alike. The rocker arm mesh demonstrates our method's robustness to meshes with highly irregular geometry and connectivity, where discrete variational methods face problems [Valette et al. 2008].

Figure 7.11 shows our multiresolution scheme applied to the fertility model, decimated to 16K and 8K vertices from 240K vertices. Note the drastic improvement in mesh quality (top part), and our method's resilience to the input triangulation. The mesh independence of our construction ensures a high-quality triangulation, regardless of how the input surface is meshed.

Table 7.1 shows quality statistics for these meshes, under the radius ratio measure. We note that for the uniform meshes, and the other uniform meshing results shown throughout, we obtain very consistent histograms, *independent* of the particular mesh, in a similar manner to Schreiner et al. [2006]. Note that our approach often results in more than 99% of good-quality triangles, where the notion of a good-quality triangle is such that its radius ratio is greater than 0.5 [Schreiner et al. 2006].

Table 7.2 shows quality statistics in terms of the minimum angle in a triangle. Since our method tends to produce anisotropic triangles, most triangles contain an angle of 90°, where for such an angle, its adjacent edges are typically aligned with the principal curvatures. As a result, the best we can expect the minimum angle to be in such a triangle is at most 45°. As shown in the table, we typically achieve an average minimum angle of around 40°, indicative that we remain close to the best possible minimum angle. Moreover, even despite the anisotropy, most of the triangles contain a minimum angle above 30°, while for most meshes, the worst minimum angle is bound below reasonably well.

Table 7.3 shows the computational time involved in the Fiedler vector computation. Times refer to the total time; that is, the 8 seconds shown in the column of the rocker arm

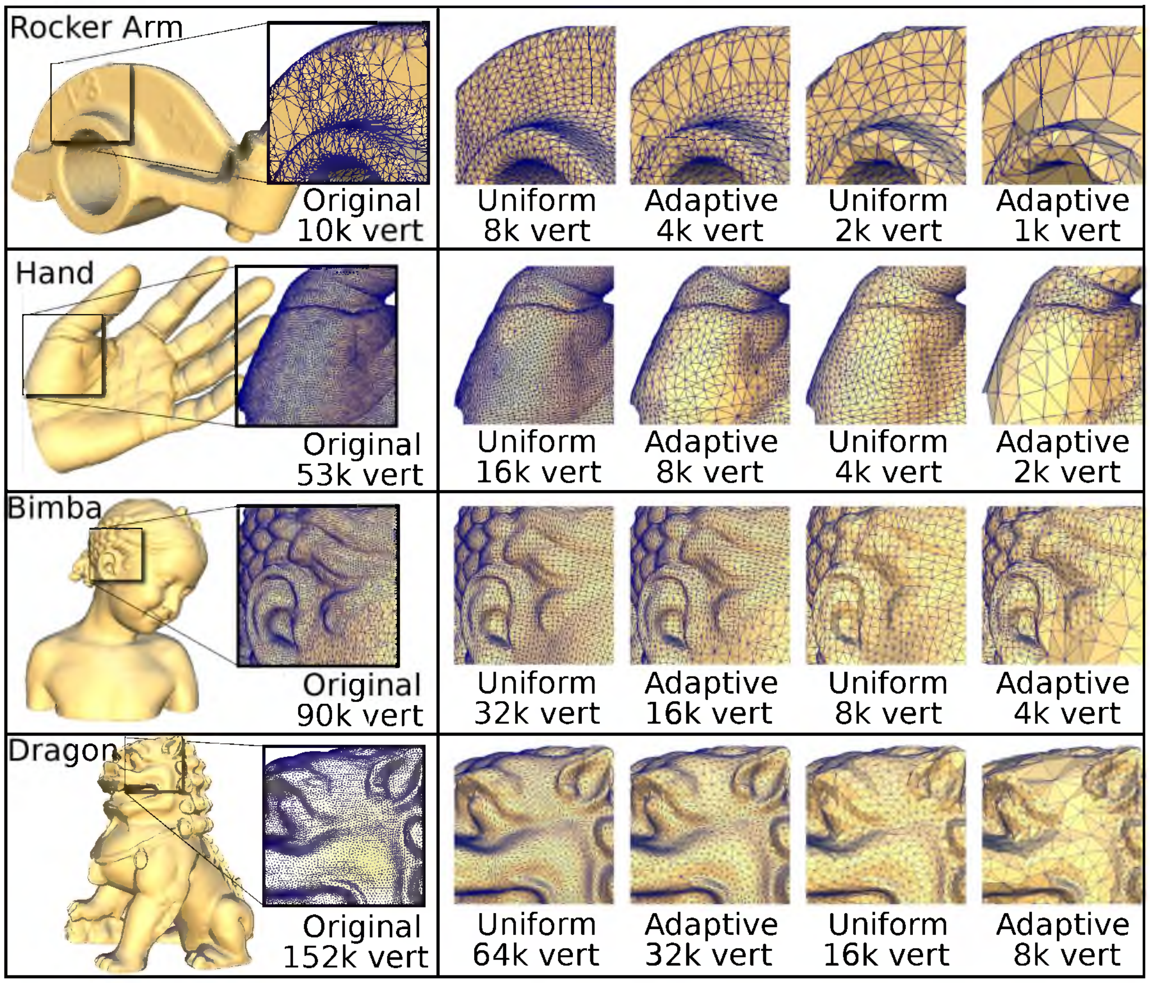


Fig. 7.10. Uniform and adaptive meshing results for a variety of surface meshes.

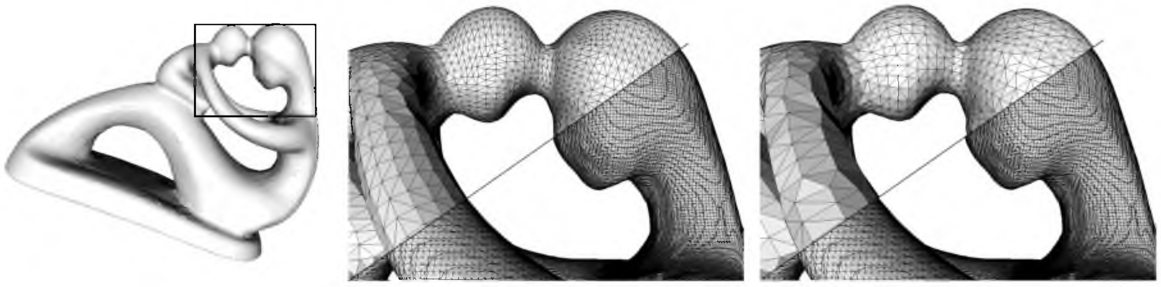
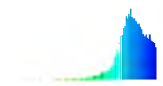
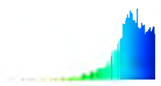
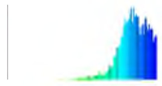
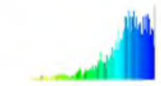
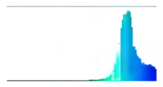
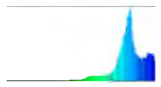
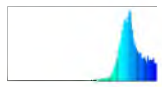
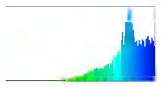
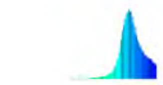
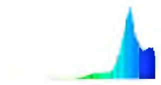
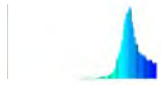
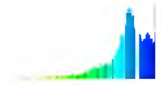
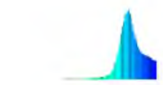
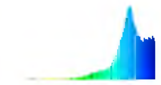
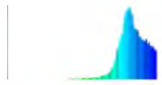
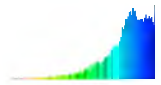
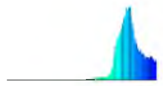
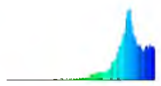
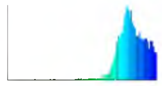
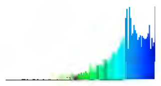


Fig. 7.11. Fertility model, 240K vertices, uniformly decimated to 16K vertices, 8K vertices. Our Fiedler approach is shown in the top-half image.

Table 7.1. Radius ratio of multiresolution models presented in Figure 7.10. Numbers in each entry correspond to average-quality, worst case, and percentage of triangles within the interval $[0.5, 1.0]$, which comprise the good-quality triangles.

Rocker	Un 8K	Ad 4K	Un 2K	Ad 1K
Av Wt [%]	0.84 0.06 99.0%	0.82 0.02 98%	0.84 0.15 99.3%	0.80 0.18 94.4%
				
Hand	Un 32K	Ad 16K	Un 8K	Ad 4K
Av Wt [%]	0.82 0.13 99.7%	0.82 0.15 98.8%	0.83 0.25 99.8%	0.81 0.01 96.2%
				
Bimba	Un 64K	Ad 32K	Un 16K	Ad 8K
Av Wt [%]	0.83 0.02 99.5%	0.83 0.01 98.5%	0.83 0.16 99.7%	0.80 0.05 95.2%
				
Dragon	Un 64K	Ad 32K	Un 16K	Ad 8K
Av Wt [%]	0.83 0.01 99.7%	0.83 0.01 98.5%	0.84 0.15 99.7%	0.81 0.04 95.3%
				
Fertility	Un 16K	Ad 8K	Un 4K	Ad 2K
Av Wt [%]	0.82 0.10 99.8%	0.82 0.10 98.2%	0.83 0.17 99.6%	0.80 0.11 95.1%
				

model is the time to carry out the eigen decomposion in the $2^{13} - 1 = 8,191$ nodes (the Fiedler vector is not computed in the tree leaves).

Figure 7.8 demonstrates qslim's inherent limitation in mistaking noise as features. Space decomposition-based methods tend to be more robust to noise, so we have compared our approach to that of the VS-tree [Boubekeur et al. 2006] in Figure 7.12. Although the VS-tree has the capability to construct a decomposition on the surface at a fine-enough level, utilizing a height field indicator in the presence of high-frequency noise results in unreliable analysis. The Fiedler tree, however, remains invariant to this high-frequency noise, sufficiently smoothing the mesh. We note that the VS-tree and qslim have the advantage of being computationally efficient, whereas our method is significantly more time consuming. However, our comparisons illustrate flaws in these approaches, resulting from the lack of a proper analysis of the surface at multiple scales, which is precisely what our method excels at.

Last, we have compared the quality of our meshes to that of a state of the art remeshing algorithm, delpsc [Cheng et al. 2007]. See Figure 7.13 for a comparison of the egea model,

Table 7.2. Minimum angle of multiresolution models presented in Figure 7.10. Numbers in each entry correspond to average minimum angle, worst minimum angle, and percentage of triangles whose minimum angle is greater than 30°.

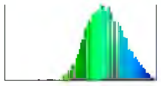
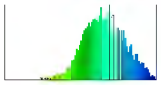
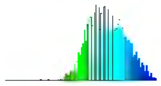
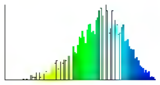
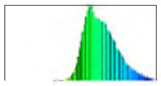
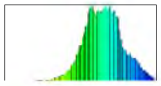
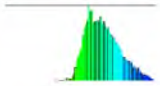
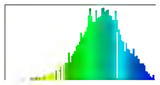
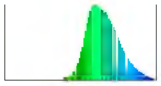
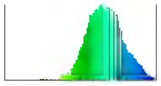
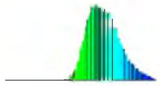
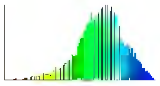
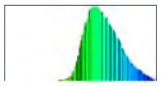
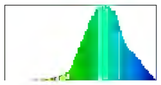
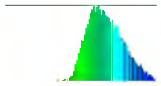
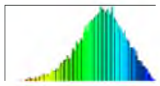
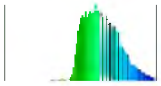
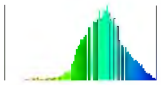
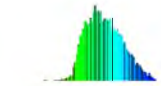
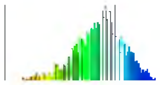
Rocker	Un 8K 39.82° 9.19° 92.0% 	Ad 4K 39.41° 5.29° 88.3% 	Un 2K 40.4° 13.82° 93.5% 	Ad 1K 37.79° 7.41° 80.0% 
Hand	Un 32K 38.2° 12.64° 93.7% 	Ad 16K 39.19° 11.67° 90.4% 	Un 8K 38.88° 18.12° 94.0% 	Ad 4K 38.37° 2.73° 83.7% 
Bimba	Un 64K 39.09° 5.76° 94.6% 	Ad 32K 39.59° 4.1° 91.7% 	Un 16K 39.24° 12.71° 94.8% 	Ad 8K 38.14° 3.66° 83.2% 
Dragon	Un 64K 38.96° 2.59° 93.7% 	Ad 32K 39.65° 2.15° 90.0% 	Un 16K 39.76° 11.29° 93.9% 	Ad 8K 38.37° 3.6° 81.9% 
Fertility	Un 16K 38.52° 9.81° 90.7% 	Ad 8K 39.11° 4.91° 88.5% 	Un 4K 39.54° 15.89° 92.2% 	Ad 2K 37.7° 7.32° 80.0% 

Table 7.3. Computational times to compute the Fiedler vector during the tree construction.

Model	Rocker	Hand	Bimba	Dragon	Fertility
Size	10K vert.	53K vert.	90K vert.	152K vert.	240K vert.
# Levels	13	15	16	16	14
Eigen Calc.	8s	49s	1m40s	2m48s	4m44s

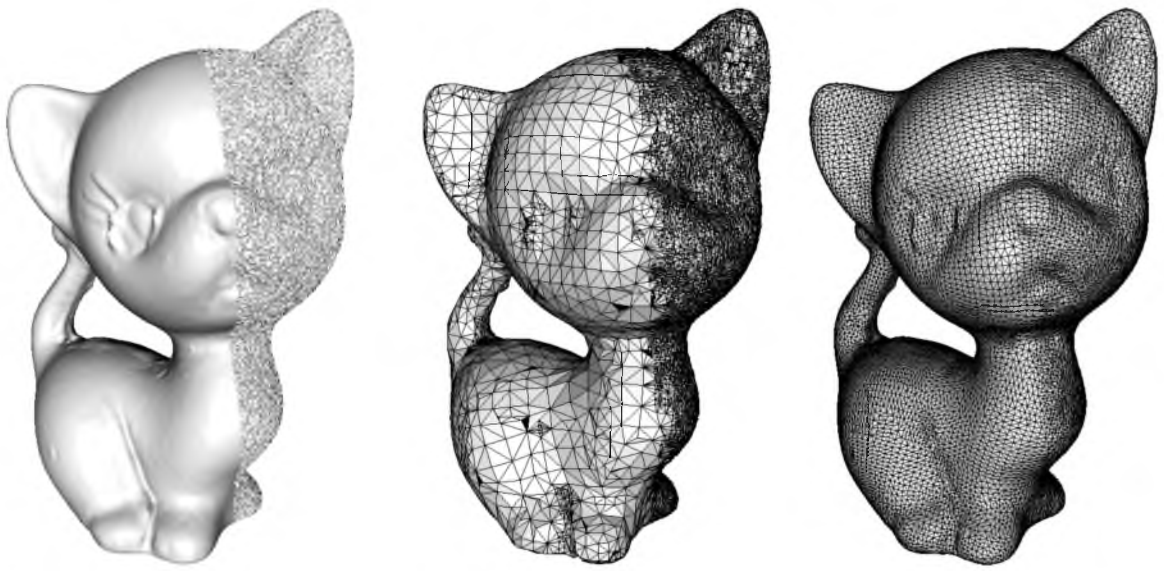


Fig. 7.12. Comparison of VS-tree [Boubekeur et al. 2006] (left) to our approach (right), for simplification of a noisy surface. The original surface (135K vertices) is decimated to 21K vertices for both approaches. Timing for VS-tree: 70ms, timing for our method: 2m30s for eigenvector computations, and 900ms to generate the mesh

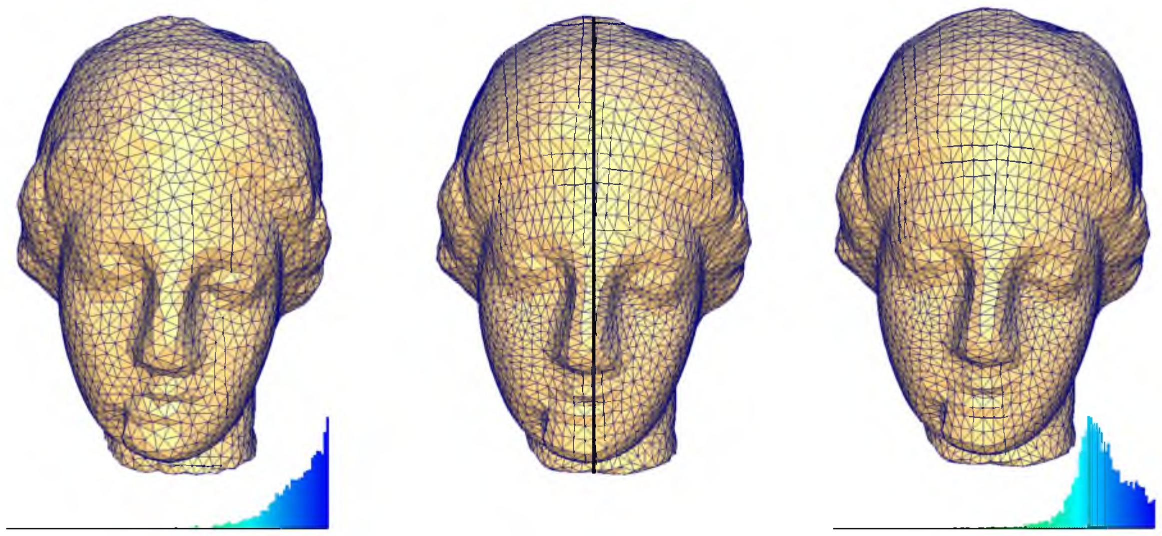


Fig. 7.13. Comparison of delpsc [Cheng et al. 2007] on the left, to our method on the right, with corresponding triangle radius ratio histograms. Timing for delpsc: 7.4s, timing for our method: 7.5s for eigenvector computations, and 109ms to generate the mesh

remeshed to approximately 4K vertices. Our results are competitive in terms of triangle radius ratio, albeit not quite as good; however, we are able to construct a multiresolution hierarchy of quality meshes, whereas delpsc operates with respect to a target number of vertices.

7.5 Discussion and Limitations

Examples and comparisons presented in Sections 7.1–7.4 support that our multiresolution scheme gathers a set of properties not present in any other approach devoted to represent meshes in multiresolution. Table 7.4 exemplifies this fact, in comparing our approach to the various methodologies. As can be observed (the symbol ✓ means a property is present), only the Fiedler tree endows the intrinsic properties of mesh independence, noise robustness, mesh quality, feature detection, and multiresolution. The symbol ● indicates a property is not intrinsic, but can be somehow approximated through tuned implementation.

Table 7.4 also suggests that the proposed Fiedler tree represents a methodology that stands between hierarchical space decomposition and remeshing approaches. Our approach shares the conceptual simplicity of space decomposition techniques, as we are merely performing a top-down hierarchical partitioning of the surface, instead of the volume in which the surface resides. We are able to produce meshes which are of competitive quality to that of remeshing schemes, yet at the same time, our approach is much simpler in comparison to most remeshing schemes.

Another interesting aspect of our approach is the ability to analyze features at multiple scales. The intrinsic hierarchical structure provided by the Fiedler tree makes multiscale analysis quite natural. In fact, the Haar-like implementation described in Section 7.3 is

Table 7.4. Comparison of our approach to other methodologies. The symbol ✓ means the property is present while the symbol ● indicates the property can be somehow incorporated.

Method / Property	Mesh Independence	Noise Robustness	Mesh Quality	Feature Detection	Multi-resolution	Comput. Efficiency
Decimation Methods				✓	✓	●
Space/Tree Partition		●		●	✓	✓
Fiedler Decomposition	✓	✓	✓	✓	✓	
Remeshing Methods			✓	✓		

only the simplest mechanism in carrying out multiscale feature analysis. We believe that more sophisticated and accurate schemes can be derived on top of our decomposition.

Our binary hierarchical mesh decomposition is only one way of decomposing a mesh, and many hierarchical segmentation methods, including spectral methods, exist in the literature [Liu and Zhang 2007; De Goes et al. 2008; Reuter et al. 2009; Reuter 2010]. However, recall that the advantage of utilizing the Fiedler vector is in generating patches which have small boundary length, and consistent surface areas. As segmentation methods assume some notion of part saliency, they are unlikely to satisfy these properties, especially in the absence of saliency, which is common at finer depths in the decomposition. We note that a possible extension to our decomposition is choosing a different eigenfunction which still splits the mesh into two connected components, while satisfying other properties such as reflectional symmetries [Ovsjanikov et al. 2008; Vallet and Lvy 2009]. This could lead to a method for intrinsically symmetric remeshing, and we leave this for future work.

The main limitation of our approach is the computational burden, including processing time and memory consumption. Althouth Table 7.3 shows our technique could be applied to process fairly big meshes on a conventional laptop, massive meshes would demand out-of-core eigenvector computation methods, especially in the first levels of the hierarchy, increasing computational times considerably. Moreover, by cutting exactly along the surface, we are encumbered by an increasing number of triangles being produced at every scale. This hinders the performance and memory efficiency of our method.

7.6 Summary

In this chapter, we have presented a novel method for multiresolution remeshing by utilizing spectral surface processing. In particular, we demonstrate the utility of the Fiedler vector for generating a balanced hierarchy of well-formed surface patches which are intrinsic to the surface. We have demonstrated applications to quality uniform mesh generation, adaptive mesh generation in the spirit of Haar wavelets, and the inherent robustness to noise.

CHAPTER 8

CONCLUSIONS

In this thesis, we have explored the use of shape analysis for imperfect, defect-laden data. In order to first gain insight into the types of imperfections which should be considered for point clouds, we established a benchmark for surface reconstruction, providing us with a means of quantitatively comparing surface reconstruction algorithms and exploring the impact of various data imperfections.

We then considered how existing shape analysis methods can benefit several tasks, namely normal orientation and surface remeshing. We demonstrated how to use a set of harmonic functions defined on the point cloud to orient normals, and how the Fiedler vector can be used to produce a hierarchy of good-quality surface meshes. Both of these methods use the Laplace-Beltrami operator of a surface: the former considers functions which lie in the kernel of the Laplacian, while the latter considers the first nontrivial eigenfunction of the Laplacian. As these functions have been shown to be robust to noise and nonuniform sampling in different areas, we leveraged these properties for improving normal orientation and multiresolution remeshing.

Last, we developed new shape analysis methods in order to process incomplete point clouds. In particular, we used the medial axis as a prior in order to construct new distances, and illustrated how these distances benefit segmentation and reconstruction. We then extended this to the construction of a diffusion process implicitly defined on the medial axis, and how this may be used to compute correspondences and intrinsic symmetries. These approaches demonstrate a unique blend of extrinsic and intrinsic shape analysis: they are extrinsic in that we are considering the volume of the shape via the medial axis, while also being intrinsic to the medial representation in that we employ diffusion distances and heat diffusion.

8.1 Future Work

The work established in this thesis should provide for many avenues of future work. In particular, the processing of incomplete point clouds still remains a challenging task, and the medial kernel provides but one way of handling this type of data.

8.1.1 New Kernels

The medial kernel is driven by the medial axis prior for incomplete point clouds. We envision a *family* of like-minded kernels to be developed, each of which exploiting different structures of missing data. The symmetry-factored kernel of Lipman et al. [2010] is another existing example: symmetry is a redundant cue in shape analysis, in that incomplete data may still exhibit partial symmetries.

Structure repetition [Li et al. 2011] is a useful prior in missing data, where the challenge in developing such a kernel lies in constructing a smooth measure which represents the association of repeating elements, and relationship between elements. If photometric and reflectance information is available, then an analogous structure repetition kernel could easily be developed for this type of information. Visibility priors are well-known to be robust to missing data [Curless and Levoy 1996; Tagliasacchi et al. 2011], and so the construction of a visibility kernel would be a natural extension.

8.1.2 Multiple Kernel Processing

With all of these kernels, and the various parameters each one will inevitably contain, a principled method of *combining* these kernels is desirable, and which is ideally independent of the type of task to perform. We think the multiple kernel learning literature [Bach et al. 2004] should prove extremely useful.

The goal in multiple kernel learning is to find the optimal linear combination of a set of kernels for a given task, typically classification. In some sense, our combination of the medial kernel with the standard Gaussian kernel in our shape matching approach represents a rather crude approach to this. One possible future direction is to consider the optimal linear combination of kernels for shape matching, based on a training set of incomplete point clouds in correspondence, resulting in *multiple kernel shape matching*. For instance, we may consider the aforementioned class of kernels over a set of different parameters, and a class of shapes containing missing data, and consequently learn the important weights. This may result in certain shape classes where the medial kernel is assigned a low weight while the visibility and symmetry priors are assigned more larger weights, and hence higher importance.

8.1.3 Towards Minimal Acquisition

A motivating application for developing methods to process incomplete point clouds is minimal geometry acquisition. Despite the numerous advances in 3D geometry acquisition, it still remains quite challenging to fully acquire a surface. For instance, while multiview stereo methods can produce rather complete geometry, they are rather cumbersome to setup and calibrate. On the other hand, while structure-from-motion methods are convenient for passively obtaining geometry from image collections, they often produce extremely sparse data.

Methods which can strictly operate on incomplete point clouds are hence attractive in these scenarios. In this thesis, we have developed a number of ways of processing incomplete point clouds, ranging from segmentation to finding correspondences, but we can see other methods developed strictly for incomplete data, such as shape deformation, skeletonization, physically-based simulation, and shape modeling. In particular, we have used our distances derived from the medial kernel for the purposes of skinning incomplete point clouds, and have found preliminary results for animation to be promising. Being able to process incomplete, defect-laden data in this manner relieves the burden from the acquirer, resulting in simpler and cost-effective ways of capturing and processing the real world.

APPENDIX A

RECONSTRUCTION BENCHMARK DETAILS

Here, we provide additional details regarding the surface reconstruction benchmark.

A.1 Polygonal Weight Functions

In this section, we detail the closed-form solution for Equation 3.2, used in the formation of our implicit functions. The basic idea is to cast the integral into the local coordinate system of the triangle, and perform integration in terms of polar coordinates, analogous to the construction of Green coordinates [Lipman and Levin 2010].

For a given evaluation point $\mathbf{x}$ and triangle t composed of the vertices $\mathbf{p}_1$, $\mathbf{p}_2$, and $\mathbf{p}_3$, and normal $\mathbf{n}$, we project $\mathbf{x}$ onto the plane of t:

$$\tilde{\mathbf{x}} = \mathbf{x} + \langle \mathbf{p}_1 - \mathbf{x}, \mathbf{n} \rangle \mathbf{n} \tag{A.1}$$

Now, for a given $\mathbf{p} \in t$, $|\mathbf{x} - \mathbf{p}|^2 + \epsilon^2 = |\tilde{\mathbf{x}} - \mathbf{p}|^2 + |\mathbf{x} - \tilde{\mathbf{x}}|^2 + \epsilon^2 = |\tilde{\mathbf{x}} - \mathbf{p}|^2 + \lambda_1$, where $\lambda_1 = |\mathbf{x} - \tilde{\mathbf{x}}|^2 + \epsilon^2$ and is constant throughout the integration. We can now rewrite the integral as:

$$\int_{\mathbf{p} \in t} w(\mathbf{x}, \mathbf{p}) \mathrm{d}\mathbf{p} = \sum_{t_i} sgn(t_i) \int_{\mathbf{p} \in t_i} \frac{\mathrm{d}\mathbf{p}}{(|\tilde{\mathbf{x}} - \mathbf{p}|^2 + \lambda_1)^2} \tag{A.2}$$

where t is broken up into t_1, t_2, t_3, formed from the triangles composed of $\tilde{\mathbf{x}}$ and $\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3$, and sgn represents the orientation of the triangle: positive if oriented properly, and negative otherwise. See the left image of Figure A.1 for an illustration of this decomposition.

Without loss of generality, we consider a single triangle t_1. We now convert this integral into polar coordinates:

$$\begin{aligned}\int_{\mathbf{p} \in t_1} \frac{\mathrm{d}\mathbf{p}}{(|\tilde{\mathbf{x}} - \mathbf{p}|^2 + \lambda_1)^2} &= \int_{\theta=0}^{\theta=\beta} \int_{r=0}^{R(\theta)} \frac{r \, \mathrm{d}r \, \mathrm{d}\theta}{(r^2 + \lambda_1)^2} \\ &= -\frac{1}{2} \int_0^{\beta} \frac{\mathrm{d}\theta}{R(\theta)^2 + \lambda_1} + \frac{\beta}{2\lambda_1}\end{aligned}$$

The integration is centered with respect to $\tilde{\mathbf{x}}$, where β is the angle in t_1 opposite $\tilde{\mathbf{x}}$, and $R(\theta)$ is the length parameterized by θ. See the middle image of Figure A.1.

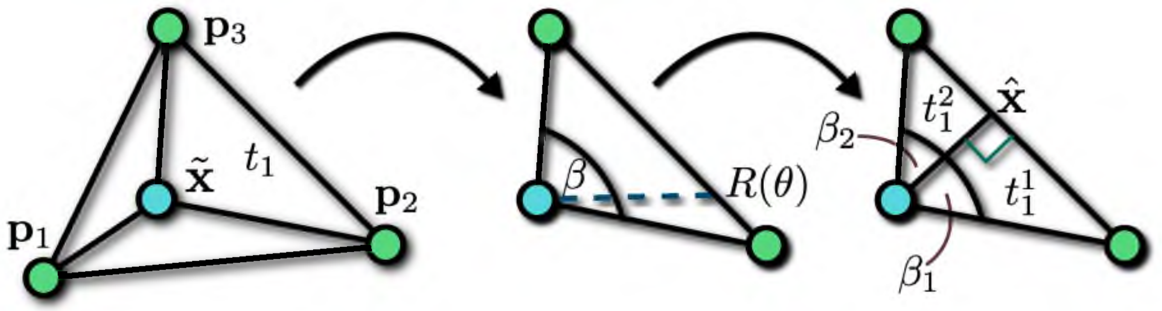


Fig. A.1. We illustrate the decomposition of the integration of polygonal weight functions. We first decompose integration into three separate triangles (left), for such a single triangle perform integration in polar coordinates (middle), followed by breaking up the integration into simpler components through orthogonal projection onto the opposing edge (right).

In order to have a clean parameterization of the length $R(\theta)$, we break up the integral into two parts by considering the orthogonal projection of the point $\tilde{\mathbf{x}}$ onto its opposing edge, $\hat{\mathbf{x}}$, and breaking t_1 into: $t_1^1 =< \tilde{\mathbf{x}}, \mathbf{p}_2, \hat{\mathbf{x}} >$ and $t_1^2 =< \tilde{\mathbf{x}}, \hat{\mathbf{x}}, \mathbf{p}_3 >$. Without loss of generality, we consider t_1^1, and we obtain: $R(\theta) = \frac{|\tilde{\mathbf{x}}-\hat{\mathbf{x}}|}{\cos(\theta)}$; see the right image of Figure A.1. Hence, the integral becomes:

$$\int_0^{\beta_1} \frac{\mathrm{d}\theta}{R^2(\theta)+\lambda_1} = sgn(t_1^1) \int_0^{\beta_1} \frac{\cos^2(\theta)}{|\tilde{\mathbf{x}}-\hat{\mathbf{x}}|^2+\lambda_1\cos^2(\theta)}$$
$$= sgn(t_1^1)\left(\frac{\beta_1}{\lambda_1} - \frac{|\tilde{\mathbf{x}}-\hat{\mathbf{x}}|^2}{\lambda_1}\int_0^{\beta_1} \frac{\mathrm{d}\theta}{|\tilde{\mathbf{x}}-\hat{\mathbf{x}}|^2+\lambda_1\cos^2(\theta)}\right)$$

where $sgn(t_1^1)$ is the sign of the orientation of the triangle, which may be negative if $\hat{\mathbf{x}}$ projects outside of t_1. Applying the double angle formula to the above integral we obtain:

$$= \int_0^{\beta_1} \frac{\mathrm{d}\theta}{(|\tilde{\mathbf{x}}-\hat{\mathbf{x}}|^2+\frac{\lambda_1}{2})+\frac{\lambda_1}{2}\cos(2\theta)}$$

Setting $b = |\tilde{\mathbf{x}}-\hat{\mathbf{x}}|^2+\frac{\lambda_1}{2}$ and $c = \frac{\lambda_1}{2}$, we may apply the relevant antiderivative [Abramowitz and Stegun 1964] to obtain:

$$\int \frac{\mathrm{d}\theta}{b+c\cos(2\theta)} = \frac{1}{\sqrt{b^2-c^2}}\tan^{-1}\left\{\sqrt{\frac{b-c}{b+c}}\tan\theta\right\}+C$$

A.2 Description of Synthetic Scanner

Here, we provide additional details on our synthetic scanner, as described in Section 3.2.1. To clarify the following discussion, we note that for each shape in our benchmark, we have set its maximum dimension to be 70mm. Hence, any scanning parameter based on distance is defined with respect to the bound of 70mm. Additionally, we place an upper bound on the radiance to be 1.

Our synthetic scanner is controlled by the following parameters:

- **Image resolution**. The image resolution, in conjunction with the number of scans used, effectively defines the resolution of the point cloud.
- **Baseline distance**. A small baseline distance magnifies depth errors in triangulation, while a large baseline results in greater occlusion. We have fixed our baseline to be with respect to the x-axis of the camera, though this may easily be adjusted to the y-axis by changing the laser sweep direction. We found that baseline distances ranging from 10mm to 150mm provide good variety in triangulation accuracy and occlusions.
- **Stripe frustum field of view**. The thickness of the laser stripe has an impact on peak detection, in appropriately fitting a Gaussian. By default, we set the field of view such that the number of pixels visible within a distance of 50mm from the camera is roughly 10, which is a function of the image resolution.
- **Stripe resolution**. The number of laser stripes to project impacts the resolution of the depth. By default, we set this to be the x resolution of the camera, in order to obtain sufficient coverage. Setting the stripe resolution to be lower than the x resolution may result in some points not being affected by the laser stripes. By assigning a sufficiently large stripe frustum field of view, one may be able to obtain sufficient coverage.
- **Noise magnitude**. The magnitude of the noise corrupts the laser projection, making peak detection imprecise. Typical noise magnitudes we have used range from 0, or no noise, to 0.6, which can greatly corrupt the radiance signal.
- **Radiance smoothing bandwidth**. Smoothing the radiance image reduces noise, though at the potential cost of sacrificing the expected Gaussian laser profile. The bandwidth to use is largely dependent on the stripe frustum field of view and noise level. For instance, a thick laser under large noise magnitude will require a fairly large bandwidth to sufficiently smooth out the noise. We note that smoothing, in conjunction with additive noise, may result in a radiance signal with smaller peak magnitudes, which can impact the peak magnitude threshold.
- **Peak magnitude threshold**. For large thresholds, this will reject parts of the surface whose radiance signal is determined weak by a pixel's corresponding Gaussian fit. This is a major cause of missing data. For a laser containing little or no noise,

typical thresholds range from 0.8, which will result in only highly confident range data, to 0.1, which will result in the rejection of few points. Under noise and radiance smoothing, the peak threshold must be adjusted to account for an expected reduction in peak magnitude.

- **Variance threshold**. Range at depth discontinuities are likely to be rejected under this threshold. We set the variance with respect to the width of the laser, where by default we only reject range whose variance in the Gaussian fit is larger than twice that of the laser width. Similar to the peak magnitude threshold, the variance threshold is sensitive to the noise magnitude and smoothing bandwidth.

We note that in our experiments, although we have generated quite a large number of point clouds, we have hardly explored the full parameter space of our scanner. By publicly releasing our synthetic scanner software, surface reconstruction researchers and practitioners will be able to replicate specific scanning conditions which they are interested in operating on.

APPENDIX B

IMPLEMENTATION DETAILS

The algorithms developed in this thesis were designed with simplicity in mind, such that the core implementation of each algorithm is closely tied with the novelty and contribution of the method. This results in a rather straightforward implementation for each method, so that our algorithms are easily reproducible. In this appendix, we provide more detailed descriptions and pseudocode for an important subset of these methods. In conjunction with standard numerical linear algebra libraries, this should serve as a set of recipes for one to rapidly implement the methods described in this thesis.

B.1 Medial Kernel

We describe the construction of the medial kernel, and in particular distances, in this section. The other applications of our approach, namely segmentation and reconstruction, follow as straightforward extensions of the kernel and distance construction. The construction of the medial kernel can be broken up into two main components: generation of the candidate ball, and deviation of the candidate ball from a medial ball.

The **candidateball** routine of Algorithm 1 takes a pair of points, along with their corresponding normals, and returns a ball's center and radius. We first perform ray-intersections against the bisecting plane, where rays are formed for each point along the corresponding normal. If one of these rays fails to intersect the bisecting plane, then we do not form a candidate ball for these point pairs. We then take the corresponding bisecting points $\mathbf{b}_i$, $\mathbf{b}_j$, and the center of these points $\mathbf{b}_k$. Note that all lie on the bisecting plane – we consequently take the candidate ball as the one with smallest radius.

The **emptiness** routine of Algorithm 2 takes a candidate ball and efficiently computes the emptiness term γ. Here, we assume that a kd-tree has been built on the point cloud, such that each node contains the precomputated expansion terms for all points inside of the node, as described in Section 5.2.2. As we traverse the tree and construct the emptiness measure, if the current measure is determined to exceed a maximum dissimilarity, resulting

Algorithm 1 Candidate Ball Generation: **candidateball**($\mathbf{p}_i, \mathbf{p}_j, \mathbf{n}_i, \mathbf{n}_j$)

```
$\mathbf{b} = \frac{\mathbf{p}_i+\mathbf{p}_j}{2}$ ; $\mathbf{n}_b = \frac{\mathbf{p}_j-\mathbf{p}_i}{\|\mathbf{p}_j-\mathbf{p}_i\|}$
$\mathbf{b}_i = \mathbf{p}_i + \frac{\langle \mathbf{b}-\mathbf{p}_i,\mathbf{n}_b \rangle}{\langle \mathbf{n}_i,\mathbf{n}_b \rangle}\mathbf{n}_i$ ; $\mathbf{y}_j = \mathbf{b}_j + \frac{\langle \mathbf{b}-\mathbf{p}_j,\mathbf{n}_b \rangle}{\langle \mathbf{n}_j,\mathbf{n}_b \rangle}\mathbf{n}_j$ ; $\mathbf{b}_k = \frac{\mathbf{p}_i+\mathbf{p}_j}{2}$
if $\mathbf{b}_i = \infty$ or $\mathbf{b}_j = \infty$ then
  return no intersection
end if
$B = \{\mathbf{b}_i, \mathbf{b}_j, \mathbf{b}_k\}$
$\mathbf{c}_{ij} = \underset{\mathbf{c}}{\text{argmin}}\{\|\mathbf{c} - \mathbf{p}_i\| | \mathbf{c} \in B\}$
$r_{ij} = \|\mathbf{c}_{ij} - \mathbf{p}_i\|$
return $(\mathbf{c}_{ij}, r_{ij})$
```

in the medial kernel to be numerically zero, then we terminate the traversal early. This results in a substantial speed-up, in that we can quickly discard pairs of points which are clearly dissimilar.

The medial kernel ϕ easily follows; see Algorithm 3: first, we construct a candidate ball for the points, measure the tangential dissimilarity τ and emptiness γ, and convert this into a similarity measure. In practice, if the medial kernel is sufficiently small for a pair of points $(\mathbf{p}_i, \mathbf{p}_j)$, then we do not store the value. In doing so, we obtain a fairly sparse number of entries whose medial kernel is nonzero. Note that this is largely a function of the medial axis – clean spherical parts, for instance, will result in many nonzero entries.

Our distances are a straightforward adaptation of diffusion distances applied to ϕ, as shown in Algorithm 4. In order to properly apply diffusion distances, we must make the matrix M row-stochastic, so that it encodes a random walk. We use the normalization of Coifman and Lafon [2006], where we divide every entry of each row by the row's summation.

B.2 Medial Laplacian

We next describe the construction of the Medial Laplacian, used for the purposes of shape matching in Section 6.1.3. We have omitted pseudocode for the matching process, as it is a straightforward adaptation of Ovsjanikov et al. [2010]. Aside from the medial kernel, as just described, there are two main components to the Medial Laplacian: area estimation and combining kernels.

Estimating the area at a given point amounts to finding other points which belong to a similar medial region, and taking the convex hull of these points, see Algorithm 5. Once the convex hull is constructed, we then take the area as one-third of the area of all incident triangles to the point. If the query point does not lie on the convex hull, then we project

Algorithm 2 Efficiently Computing Emptiness: **emptiness**($\mathbf{c}, r$)

```
$\gamma = 0$
L $\leftarrow$ root node of kd-tree containing precomputed emptiness expansions
**while** $|L| > 0$ **do**
  node $\leftarrow$ dequeue(L)
  **if** node.is_leaf **then**
    $\gamma = \gamma+$node.precomputed_expansion($\mathbf{c}, r$)
    **if** $\gamma$ exceeds maximum dissimilarity **then**
      **return** $\gamma$
    **end if**
  **end if**
  **if** $\neg$ node.intersects_sphere($\mathbf{c}, r$) **then**
    **continue**
  **end if**
  **if** node.contained_in_sphere($\mathbf{c}, r$) **then**
    $\gamma = \gamma+$node.precomputed_expansion($\mathbf{c}, r$)
    **if** $\gamma$ exceeds maximum dissimilarity **then**
      **return** $\gamma$
    **end if**
    **continue**
  **end if**
  left_node $\leftarrow$ node.left_child    ;   right_node $\leftarrow$ node.right_child
  $\mathbf{l}$ =left_node.center    ;   $\mathbf{r}$ =right_node.center
  **if** $\|\mathbf{l} - \mathbf{c}\| < \|\mathbf{r} - \mathbf{c}\|$ **then**
    L.enqueue(left_node)    ;   L.enqueue(right_node)
  **else**
    L.enqueue(right_node)    ;   L.enqueue(left_node)
  **end if**
**end while**
```

the neighbor points onto the estimated medial ball of the query point, and use the convex hull of this neighborhood instead.

The construction of the Medial Laplacian is a combination of two different kernels: a local Gaussian α, and the medial kernel ϕ. As both of these kernels can have widely varying densities, they must be normalized prior to being combined – this is the purpose of the d_α and d_ϕ arrays. Once constructed, then the weight entries of the Medial Laplacian are simply a linear combination of the two kernels, normalized by their respective densities; see Algorithm 6. Note that although the above construction is quadratic in the number of points, as both ϕ and α are typically sparse, a more efficient (albeit more notationally cumbersome) implementation is to store each entry of ϕ and α in a hash, indexed by the particular pair of points.

Algorithm 3 Medial Kernel Computation: $\phi(\mathbf{p}_i, \mathbf{p}_j, \mathbf{n}_i, \mathbf{n}_j)$

```
$(\mathbf{c}_{ij}, r_{ij}) = \mathbf{candidateball}(\mathbf{p}_i, \mathbf{p}_j, \mathbf{n}_i, \mathbf{n}_j)$
if no intersection then
    return $10^{-10}$ // arbitrarily small value indicating no similarity
end if
$\mathbf{s}_i = \frac{\mathbf{p}_i - \mathbf{c}_{ij}}{\|\mathbf{p}_i - \mathbf{c}_{ij}\|}$ ; $\mathbf{s}_j = \frac{\mathbf{p}_j - \mathbf{c}_{ij}}{\|\mathbf{p}_j - \mathbf{c}_{ij}\|}$
$\gamma = \mathbf{emptiness}(\mathbf{c}_{ij}, r_{ij})$
$\tau = \|\mathbf{n}_i - \mathbf{s}_i\| + \|\mathbf{n}_j - \mathbf{s}_j\|$
return $e^{-\left(\frac{\gamma}{\sigma_e}\right)^2 - \left(\frac{\tau}{\sigma_t}\right)^2}$
```

Algorithm 4 Distances: $d_t(i, j)$

```
$M : M_{ij} = \phi(\mathbf{p}_i, \mathbf{p}_j, \mathbf{n}_i, \mathbf{n}_j)$
$D = \text{diag}(M\mathbf{1})$ ; $\hat{M} = D^{-1}M$ ; $\hat{M}\Psi_k = \lambda_k \Psi_k$
$\Phi_t(\mathbf{p}_i) = \{\lambda_1^t \Psi_1(\mathbf{p}_i), \lambda_2^t \Psi_2(\mathbf{p}_i), \lambda_3^t \Psi_3(\mathbf{p}_i), ...\}$
$\Phi_t(\mathbf{p}_j) = \{\lambda_1^t \Psi_1(\mathbf{p}_j), \lambda_2^t \Psi_2(\mathbf{p}_j), \lambda_3^t \Psi_3(\mathbf{p}_j), ...\}$
return $\|\Phi_t(\mathbf{p}_i) - \Phi_t(\mathbf{p}_j)\|$
```

B.3 Fiedler Tree

Last, we describe the construction of the Fiedler Tree. In particular, we show how to construct the hierarchy of surfaces via spectral bisection, followed by how to generate a mesh at any level in the hierarchy. The multiresolution analysis, for instance adaptive mesh generation, follows as a straightforward adaptation.

The Fiedler tree is constructed by performing spectral bisection with respect to the Fiedler vector of the Laplace-Beltrami operator; see Algorithm 7. Starting from the input mesh, we compute the Fiedler vector, and then construct two child meshes separated by the Fiedler vector. Triangles in which all vertices are negative or positive are assigned to the negative child mesh ($2i$) and the positive child mesh ($2i + 1$), respectively. When the zero set cuts through a triangle, we use linear interpolation (zero_intersect) to split the triangle into one triangle and one quad – though in practice, we triangulate the quad so that we can still easily construct the Laplace-Beltrami operator and perform linear interpolation.

This process is recursively done until a user-specificed maximum depth, D, is specified, giving us a collection of surface meshes at every depth. This is a slight departure from the implementation described in Section 7.2.1, where the pseudocode presented is less memory-efficient. However, it is a faster construction, as it is not necessary to construct a level of the hierarchy from scratch, as described in Section 7.2.1.

Once the Fiedler tree has been constructed, we may take an arbitrary depth $d \leq D$ and generate its dual triangulation from the CW complex; see Algorithm 8. The CW complex

Algorithm 5 Area Estimation: **area_estimation**(P, i)

```
for $k = 1$ to $N$ do
  $\hat{\mathbf{p}}_k = \frac{\sum_j \mathbf{c}_{kj}\phi(\mathbf{p}_k,\mathbf{p}_j)}{\sum_j \phi(\mathbf{p}_k,\mathbf{p}_j)}$
end for
$r_i = \frac{\sum_j r_{ij}\phi(\mathbf{p}_i,\mathbf{p}_j)}{\sum_j \phi(\mathbf{p}_i,\mathbf{p}_j)}$
$B_i = \{\mathbf{p}_j \in P \mid |\hat{\mathbf{p}}_i - \hat{\mathbf{p}}_j| < \epsilon\}$ // $\epsilon$ is 1.5 times average sampling density
$C$ =convex_hull($B_i$)
area = 0
if $\mathbf{p}_i \in C$ then
  for $t \in C$ and $\mathbf{p}_i \in t$ do
    area=area+$\frac{1}{3}|t|$
  end for
else
  $\bar{B}_i = \{$sphere_projection$(\mathbf{p}_i, \hat{\mathbf{p}}_i, r_i) \mid |\hat{\mathbf{p}}_i - \hat{\mathbf{p}}_j| < \epsilon\}$
  $\bar{C}$ =convex_hull($\bar{B}_i$)
  for $t \in \bar{C}$ and $\mathbf{p}_i \in t$ do
    area=area+$\frac{1}{3}|t|$
  end for
end if
return area
```

is composed of a collection of surface patches, noted as M_i^d, where we first generate a triangulated mesh composed of all M_i^d. The centroid method is simply the area-weighted center of a surface patch. Then, for each vertex in M^d, if it is incident to three unique cells, we generate a triangle. Note that the number of unique incident cells will only range from one – three, due to how we construct the Fiedler tree; hence, we are guaranteed a valid triangulation so long as the closed ball property is satisfied, as discussed in Section 7.2.1.

Algorithm 6 Medial Laplacian Construction: **medial_laplacian**(P, η)

```
for $i = 1$ to $n$ do
    $a_i$ =area_estimation($i$)
    for $j = 1$ to $i$ do
        $\alpha_{ij} = e^{-\left(\frac{\|\mathbf{p}_i - \mathbf{p}_j\|}{\sigma_e}\right)^2}$
        $\phi_{ij} = \phi(\mathbf{p}_i, \mathbf{p}_j, \mathbf{n}_i, \mathbf{n}_j)$
    end for
end for
init($d_\alpha$)    ;   init($d_\phi$) // arrays containing local densities, initially all entries are 0
for $i = 1$ to $n$ do
    for $j = 1$ to $i$ do
        $d_\alpha[i] = d_\alpha[i] + \alpha_{ij} a_i$   ;   $d_\alpha[j] = a_\alpha[j] + \alpha_{ij} a_j$
        $d_\phi[i] = d_\phi[i] + \phi_{ij} a_i$   ;   $d_\phi[j] = a_\phi[j] + \phi_{ij} a_j$
    end for
end for
for $i = 1$ to $n$ do
    $D(i, i) = a_i$
    for $j = 1$ to $i$ do
        $W(i, j) = (1 - \eta) \frac{\phi_{ij}}{d_\phi[i] d_\phi[j]} + \eta \frac{\alpha_{ij}}{d_\alpha[i] d_\alpha[j]}$
    end for
end for
return $WD$
```

Algorithm 7 Fiedler Tree: **fiedlersplit**(M, d, i, D)

```
if $d = D$ then
  return
end if
Form Laplace-Beltrami operator $\Delta_M$
$\Delta_M f_2 = \lambda_2 f_2$ // $f_2$ is the Fiedler vector
$M_{2i}^{d+1} = \emptyset$ ; $M_{2i+1}^{d+1} = \emptyset$ // child meshes initially empty
for $t = \{v_1, v_2, v_3\} \in M$ do
  if $f_2(v_1) < 0$ and $f_2(v_2) < 0$ and $f_2(v_3) < 0$ then
    $M_{2i}^{d+1}$.add_tri($t$) ; continue
  else if $f_2(v_1) > 0$ and $f_2(v_2) > 0$ and $f_2(v_3) > 0$ then
    $M_{2i+1}^{d+1}$.add_tri($t$) ; continue
  end if
  if $f_2(v_1) < 0$ then
    if $f_2(v_2) < 0$ then
      $M_{2i}^{d+1}$.add_quad($v_1$,zero_intersect($f_2, v_1, v_3$),zero_intersect($f_2, v_2, v_3$), $v_2$)
      $M_{2i+1}^{d+1}$.add_tri($v_3$,zero_intersect($f_2, v_2, v_3$),zero_intersect($f_2, v_1, v_3$))
    else
      $M_{2i}^{d+1}$.add_tri($v_1$,zero_intersect($f_2, v_1, v_2$),zero_intersect($f_2, v_1, v_3$))
      $M_{2i+1}^{d+1}$.add_quad($v_2, v_3$,zero_intersect($f_2, v_1, v_3$),zero_intersect($f_2, v_1, v_2$))
    end if
  else
    if $f_2(v_2) > 0$ then
      $M_{2i+1}^{d+1}$.add_quad($v_1$,zero_intersect($f_2, v_1, v_3$),zero_intersect($f_2, v_2, v_3$), $v_2$)
      $M_{2i}^{d+1}$.add_tri($v_3$,zero_intersect($f_2, v_2, v_3$),zero_intersect($f_2, v_1, v_3$))
    else
      $M_{2i+1}^{d+1}$.add_tri($v_1$,zero_intersect($f_2, v_1, v_2$),zero_intersect($f_2, v_1, v_3$))
      $M_{2i}^{d+1}$.add_quad($v_2, v_3$,zero_intersect($f_2, v_1, v_3$),zero_intersect($f_2, v_1, v_2$))
    end if
  end if
end for
fiedlersplit($M_{2i}^{d+1}, d + 1, 2i, D$) ; fiedlersplit($M_{2i+1}^{d+1}, d + 1, 2i + 1, D$)
```

Algorithm 8 Dual Triangulation: **triangulate**(d)

```
$M^d = \emptyset$    ;   $\mathbf{C} = []$
for $i = 0$ to $2^d - 1$ do
   for $t \in M_i^d$ do
      $t$.id$= i$
   end for
   $M^d = M^d \cup M_i^d$
   $\mathbf{C}[i]$ =centroid($M^d$)
end for
$U = \emptyset$
for $\mathbf{v} \in M^d$ do
   dual_ids $= \emptyset$
   for $t \in M^d$ incident to $\mathbf{v}$ do
      dual_ids = dual_ids $\cup$ $t$.id
   end for
   if |dual_ids| $\neq 3$ then
      continue
   end if
   $U$.add_tri($\mathbf{C}$[dual_ids[0]], $\mathbf{C}$[dual_ids[1]], $\mathbf{C}$[dual_ids[2]])
end for
return $U$
```

REFERENCES

Abbasinejad, F., Kil, Y., Sharf, A., and Amenta, N. 2009. Rotating scans for systematic error removal. *Computer Graphics Forum 28,* 5, 1319–1326.

Abramowitz, M. and Stegun, I. 1964. *Handbook of mathematical functions with formulas, graphs, and mathematical tables.* Vol. 55. Dover publications.

Adamson, A. and Alexa, M. 2003. Approximating and intersecting surfaces from points. In *Symposium on Geometry Processing.* Eurographics Association, 230–239.

Alexa, M., Behr, J., Cohen-Or, D., Fleishman, S., Levin, D., and Silva, C. 2003. Computing and rendering point set surfaces. *IEEE Transactions on Visualization and Computer Graphics 9,* 1, 3–15.

Alliez, P., Cohen-Steiner, D., Tong, Y., and Desbrun, M. 2007. Voronoi-based variational reconstruction of unoriented point sets. In *Symposium on Geometry Processing.* Eurographics Association, 39–48.

Alliez, P., de Verdire, E., Devillers, O., and Isenburg, M. 2003. Isotropic surface remeshing. In *Shape Modeling International.* IEEE, 49–58.

Amenta, N. and Bern, M. 1999. Surface reconstruction by Voronoi filtering. *Discrete and Computational Geometry 22,* 4, 481–504.

Amenta, N., Choi, S., Dey, T. K., and Leekha, N. 2002. A simple algorithm for homeomorphic surface reconstruction. *Int. J. Comput. Geometry Appl. 12,* 1-2, 125–141.

Amenta, N., Choi, S., and Kolluri, R. 2001. The power crust. In *Symposium on Solid Modeling and Applications.* ACM, 249–266.

Anguelov, D., Srinivasan, P., Pang, H., Koller, D., Thrun, S., and Davis, J. 2005. The correlated correspondence algorithm for unsupervised registration of nonrigid surfaces. *Advances in neural information processing systems 17*, 33–40.

Au, O., Fu, H., Tai, C., and Cohen-Or, D. 2007. Handle-aware isolines for scalable shape editing. *ACM Transactions on Graphics 26,* 3, 83:1–83:10.

Au, O., Tai, C., Chu, H., Cohen-Or, D., and Lee, T. 2008. Skeleton extraction by mesh contraction. *ACM Transactions on Graphics 27,* 3, 44:1–44:10.

Bach, F., Lanckriet, G., and Jordan, M. 2004. Multiple kernel learning, conic duality, and the smo algorithm. In *International Conference on Machine Learning.* ACM, 6.

Baribeau, R. and Rioux, M. 1991. Influence of speckle on laser range finders. *Applied Optics 30,* 20, 2873–2878.

Belkin, M. and Niyogi, P. 2005. Towards a theoretical foundation for laplacian-based manifold methods. In *Conference on Learning Theory.* Springer, 486–500.

Belkin, M., Sun, J., and Wang, Y. 2008. Discrete laplace operator on meshed surfaces. In *Symposium on Computational Geometry.* ACM, 278–287.

Belkin, M., Sun, J., and Wang, Y. 2009. Constructing laplace operator from point clouds in r d. In *Proceedings of the twentieth Annual ACM-SIAM Symposium on Discrete Algorithms.* Society for Industrial and Applied Mathematics, 1031–1040.

Ben-Chen, M., Butscher, A., Solomon, J., and Guibas, L. 2010. On discrete killing vector fields and patterns on surfaces. *Computer Graphics Forum 29,* 5, 1701–1711.

Berger, M., Gustavo Nonato, L., Pascucci, V., and Silva, C. 2010. Fiedler trees for multiscale surface analysis. *Computers & Graphics 34,* 3, 272–281.

Berger, M., Levine, J., Nonato, L., Taubin, G., and Silva, C. 2013. A benchmark for surface reconstruction. *ACM Transactions on Graphics (to appear).*

Berger, M. and Silva, C. 2012a. Medial kernels. *Computer Graphics Forum 31,* 2, 795–804.

Berger, M. and Silva, C. 2012b. Nonrigid matching of undersampled shapes via medial diffusion. *Computer Graphics Forum 31,* 5, 1587–1596.

Biyikoglu, T., Leydold, J., and Stadler, P. 2007. *Laplacian eigenvectors of graphs: Perron-Frobenius and Faber-Krahn type theorems.* Springer.

Boissonnat, J. and Oudot, S. 2005. Provably good sampling and meshing of surfaces. *Graphical Models 67,* 5, 405–451.

Boubekeur, T., Heidrich, W., Granier, X., and Schlick, C. 2006. Volume-surface tree. *Computer Graphics Forum 25*, 399–406.

Bouguet, J. 2010. Camera calibration toolbox for matlab. Available at: `http://www.vision.caltech.edu/bouguetj/calib_doc`.

Bowers, J., Wang, R., Wei, L.-Y., and Maletz, D. 2010. Parallel poisson disk sampling with spectrum analysis on surfaces. *ACM Transactions on Graphics 29,* 6, 166:1–166:10.

Bronstein, A., Bronstein, M., Bronstein, M., and Kimmel, R. 2008. *Numerical geometry of non-rigid shapes.* Springer-Verlag New York Inc.

Bronstein, A., Bronstein, M., and Kimmel, R. 2006. Generalized multidimensional scaling: a framework for isometry-invariant partial surface matching. *Proceedings of the National Academy of Sciences of the United States of America 103,* 5, 1168–1172.

Brown, B. and Rusinkiewicz, S. 2007. Global non-rigid alignment of 3-D scans. *ACM Transactions on Graphics 26,* 3, 21:1–21:10.

Cao, J., Tagliasacchi, A., Olson, M., Zhang, H., and Su, Z. 2010. Point Cloud Skeletons via Laplacian Based Contraction. In *Shape Modeling International.* IEEE, 187–197.

Carr, J., Beatson, R., Cherrie, J., Mitchell, T., Fright, W., McCallum, B., and Evans, T. 2001. Reconstruction and representation of 3d objects with radial basis functions. In *SIGGRAPH*. ACM, 67–76.

Chang, M. and Kimia, B. 2008. Regularizing 3d medial axis using medial scaffold transforms. In *CVPR*. IEEE, 1–8.

Chang, W. and Zwicker, M. 2008. Automatic registration for articulated shapes. *Computer Graphics Forum 27,* 5, 1459–1468.

Chang, W. and Zwicker, M. 2009. Range scan registration using reduced deformable models. *Computer Graphics Forum 28,* 2, 447–456.

Chazal, F. and Lieutier, A. 2005. The λ-medial axis. *Graphical Models 67,* 4, 304–331.

Chen, Y., Chen, B., Lai, S., and Nishita, T. 2010. Binary orientation trees for volume and surface reconstruction from unoriented point clouds. *Computer Graphics Forum 29,* 7, 2011–2019.

Chen, Y., Lee, T., Chen, B., and Lai, S. 2011. Bipartite polar classification for surface reconstruction. *Computer Graphics Forum 30,* 7, 2003–2010.

Cheng, S., Dey, T., and Levine, J. 2007. A practical Delaunay meshing algorithm for a large class of domains. In *International Meshing Roundtable*. Springer, 477–494.

Cignoni, P., Montani, C., and Scopigno, R. 1998. A comparison of mesh simplification algorithms. *Computers & Graphics 22,* 1, 37–54.

Coifman, R. and Lafon, S. 2006. Diffusion maps. *Applied and Computational Harmonic Analysis 21,* 1, 5–30.

Curless, B. and Levoy, M. 1995. Better optical triangulation through spacetime analysis. In *ICCV*. IEEE, 987–994.

Curless, B. and Levoy, M. 1996. A volumetric method for building complex models from range images. In *SIGGRAPH*. ACM, 303–312.

De Goes, F., Goldenstein, S., and Velho, L. 2008. A hierarchical segmentation of articulated bodies. *Computer Graphics Forum 27,* 5, 1349–1356.

Desbrun, M., Meyer, M., and Alliez, P. 2002. Intrinsic parameterizations of surface meshes. *Computer Graphics Forum 21,* 3, 209–218.

Dey, T., Li, K., Luo, C., Ranjan, P., Safa, I., and Wang, Y. 2010. Persistent heat signature for pose-oblivious matching of incomplete models. *Computer Graphics Forum 29,* 5, 1545–1554.

Dey, T., Ranjan, P., and Wang, Y. 2010. Convergence, stability, and discrete approximation of laplace spectra. In *Proceedings of the twenty-first Annual ACM-SIAM Symposium on Discrete Algorithms*. 650–663.

Dey, T. and Sun, J. 2006a. Defining and computing curve-skeletons with medial geodesic function. In *Symposium on Geometry Processing*. Eurographics Association, 143–152.

Dey, T. and Sun, J. 2006b. Normal and feature approximations from noisy point clouds. *FSTTCS 2006: Foundations of Software Technology and Theoretical Computer Science*, 21–32.

Dey, T. and Zhao, W. 2004. Approximate medial axis as a voronoi subcomplex. *Computer-Aided Design 36,* 2, 195–202.

Edelsbrunner, H., Harer, J., Natarajan, V., and Pascucci, V. 2004. Local and global comparison of continuous functions. *IEEE Visualization*, 275–280.

Edelsbrunner, H. and Shah, N. 1997. Triangulating Topological Spaces. *International Journal of Computational Geometry & Applications 7,* 4, 365–378.

Federer, H. 1959. Curvature measures. *Transactions of the American Mathematical Society 93,* 3, 418–491.

Garland, M. and Heckbert, P. S. 1997. Surface simplification using quadric error metrics. In *SIGGRAPH.* ACM, 209–216.

Gebal, K., Bærentzen, J., Aanæs, H., and Larsen, R. 2009. Shape Analysis Using the Auto Diffusion Function. *Computer Graphics Forum 28,* 5, 1405–1413.

Goldberg, M. J. and Kim, S. 2010. Some remarks on diffusion distances. *Journal of Applied Mathematics 2010*, 17.

Guennebaud, G. and Gross, M. 2007. Algebraic point set surfaces. *ACM Transaction on Graphics 26,* 3, 23:1–23:10.

Guskov, I., Sweldens, W., and Schröder, P. 1999. Multiresolution signal processing for meshes. In *SIGGRAPH.* ACM, 325–334.

Hilaga, M., Shinagawa, Y., Kohmura, T., and Kunii, T. 2001. Topology matching for fully automatic similarity estimation of 3d shapes. In *SIGGRAPH.* ACM, 203–212.

Hildebrandt, K., Polthier, K., and Wardetzky, M. 2006. On the convergence of metric and geometric properties of polyhedral surfaces. *Geometriae Dedicata 123,* 1, 89–112.

Hoppe, H. 1996. Progressive meshes. In *SIGGRAPH.* ACM, 99–108.

Hoppe, H., DeRose, T., Duchamp, T., McDonald, J., and Stuetzle, W. 1992. Surface reconstruction from unorganized points. In *SIGGRAPH.* ACM, 71–78.

Huang, H., Li, D., Zhang, H., Ascher, U., and Cohen-Or, D. 2009. Consolidation of unorganized point clouds for surface reconstruction. *ACM Transactions on Graphics 28,* 5, 176:1–176:7.

Huang, Q., Adams, B., Wicke, M., and Guibas, L. 2008. Non-rigid registration under isometric deformations. *Computer Graphics Forum 27,* 5, 1449–1457.

Jansen, M. and Ooníncx, P. 2005. *Second generation wavelets and applications.* Springer.

Katz, S., Tal, A., and Basri, R. 2007. Direct visibility of point sets. *ACM Transactions on Graphics 26,* 3, 24:1–24:11.

Kazhdan, M. 2005. Reconstruction of solid models from oriented point sets. In *Symposium on Geometry Processing.* Eurographics Association, 73–82.

Kazhdan, M., Bolitho, M., and Hoppe, H. 2006. Poisson surface reconstruction. In *Symposium on Geometry processing.* Eurographics Association, 61–70.

Kim, V., Lipman, Y., and Funkhouser, T. 2011. Blended intrinsic maps. *ACM Transactions on Graphics 30,* 4, 79:1–79:12.

Kobbelt, L., Campagna, S., Vorsatz, J., and Seidel, H.-P. 1998. Interactive multiresolution modeling on arbitrary meshes. In *SIGGRAPH.* ACM, 105–114.

Kolluri, R. 2005. Provably good moving least squares. In *Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms.* SODA '05. Society for Industrial and Applied Mathematics, 1008–1017.

König, S. and Gumhold, S. 2009. Consistent propagation of normal orientations in point clouds. In *VMV.* 83–92.

Levy, B. 2006. Laplace-Beltrami Eigenfunctions Towards an Algorithm That Understands Geometry. In *Shape Modeling and Applications.* IEEE, 13–20.

Li, G., Liu, L., Zheng, H., and Mitra, N. J. 2010. Analysis, reconstruction and manipulation using arterial snakes. *ACM Transactions on Graphics 29,* 6, 152:1–152:10.

Li, H., Adams, B., Guibas, L., and Pauly, M. 2009. Robust single-view geometry and motion reconstruction. *ACM Transactions on Graphics 28,* 5, 175:1–175:10.

Li, H., Luo, L., Vlasic, D., Peers, P., Popović, J., Pauly, M., and Rusinkiewicz, S. 2012. Temporally coherent completion of dynamic shapes. *ACM Transactions on Graphics 31,* 1, 2:1–2:11.

Li, H., Sumner, R., and Pauly, M. 2008. Global correspondence optimization for non-rigid registration of depth scans. *Computer Graphics Forum 27,* 5, 1421–1430.

Li, Y., Wu, X., Chrysathou, Y., Sharf, A., Cohen-Or, D., and Mitra, N. J. 2011. Globfit: consistently fitting primitives by discovering global relations. *ACM Transactions on Graphics 30,* 4, 52:1–52:12.

Lipman, Y., Chen, X., Daubechies, I., and Funkhouser, T. 2010. Symmetry factored embedding and distance. *ACM Transactions on Graphics 29,* 4, 103:1–103:12.

Lipman, Y. and Funkhouser, T. A. 2009. Möbius voting for surface correspondence. *ACM Transactions on Graphics 28,* 3, 72:1–72:12.

Lipman, Y. and Levin, D. 2010. Derivation and Analysis of Green Coordinates. *Computational Methods and Function Theory 10,* 1, 167–188.

Liu, R. and Zhang, H. 2007. Mesh segmentation via spectral embedding and contour analysis. *Computer Graphics Forum 26,* 3, 385–394.

Liu, R., Zhang, H., Shamir, A., and Cohen-Or, D. 2009. A part-aware surface metric for shape analysis. *Computer Graphics Forum 28,* 2, 397–406.

Liu, S. and Wang, C. 2010. Orienting unorganized points for surface reconstruction. *Computers & Graphics 34,* 3, 209–218.

LUO, C., SAFA, I., AND WANG, Y. 2009. Approximating gradients for meshes and point clouds via diffusion metric. *Computer Graphics Forum 28,* 5, 1497–1508.

LUO, C., SUN, J., AND WANG, Y. 2009. Integral estimation from point cloud in d-dimensional space: A geometric view. In *Symposium on Computational Geometry.* ACM, 116–124.

MANSON, J., PETROVA, G., AND SCHAEFER, S. 2008. Streaming surface reconstruction using wavelets. In *Computer Graphics Forum.* Vol. 27. Blackwell Publishing Ltd, 1411–1420.

MELAS, A. 1992. On the nodal line of the second eigenfunction of the Laplacian in R2. *Journal of Differential Geometry 35,* 1, 255–263.

MEYER, M., DESBRUN, M., SCHRÖDER, P., AND BARR, A. 2002. Discrete differential-geometry operators for triangulated 2-manifolds. *Visualization and mathematics 3,* 7, 34–57.

MEYER, M., KIRBY, R., AND WHITAKER, R. 2007. Topology, accuracy, and quality of isosurface meshes using dynamic particles. *IEEE Transactions on Visualization and Computer Graphics 13,* 6, 1704–1711.

MIKLOS, B., GIESEN, J., AND PAULY, M. 2010. Discrete scale axis representations for 3d geometry. *ACM Transactions on Graphics 29,* 4, 101:1–101:10.

MUNKRES, J. 1993. *Elements of algebraic topology.* Perseus Books.

NAGAI, Y., OHTAKE, Y., AND SUZUKI, H. 2009. Smoothing of Partition of Unity Implicit Surfaces for Noise Robust Surface Reconstruction. In *Computer Graphics Forum.* Vol. 28. Blackwell Publishing Ltd, 1339–1348.

NAN, L., SHARF, A., ZHANG, H., COHEN-OR, D., AND CHEN, B. 2010. SmartBoxes for interactive urban reconstruction. *ACM Transactions on Graphics 29,* 4, 93:1–93:10.

NEXTENGINE. 2011. Nextengine 3d laser scanner. `http://www.nextengine.com`.

OHTAKE, Y., BELYAEV, A., ALEXA, M., TURK, G., AND SEIDEL, H. 2003. Multi-level partition of unity implicits. *ACM Transactions on Graphics 22,* 3, 463–470.

OHTAKE, Y., BELYAEV, A., AND SEIDEL, H. 2005a. An integrating approach to meshing scattered point data. In *Symposium on Solid and Physical Modeling.* ACM, 61–69.

OHTAKE, Y., BELYAEV, A. G., AND SEIDEL, H.-P. 2005b. 3d scattered data interpolation and approximation with multilevel compactly supported rbfs. *Graphical Models 67,* 3, 150–165.

OVSJANIKOV, M., HUANG, Q., AND GUIBAS, L. 2011. A condition number for non-rigid shape matching. *Computer Graphics Forum 30,* 5, 1503–1512.

OVSJANIKOV, M., MÉRIGOT, Q., MÉMOLI, F., AND GUIBAS, L. J. 2010. One point isometric matching with the heat kernel. *Computer Graphics Forum 29,* 5, 1555–1564.

OVSJANIKOV, M., SUN, J., AND GUIBAS, L. 2008. Global intrinsic symmetries of shapes. *Computer Graphics Forum 27,* 5, 1341–1348.

Öztireli, A. C., Alexa, M., and Gross, M. 2010. Spectral sampling of manifolds. *ACM Transactions on Graphics 29,* 6, 168:1–168:8.

Pauly, M., Gross, M., and Kobbelt, L. 2008. Efficient simplification of point-sampled surfaces. In *IEEE Visualization*. 163–170.

Popa, T., South-Dickinson, I., Bradley, D., Sheffer, A., and Heidrich, W. 2010. Globally consistent space-time reconstruction. *Computer Graphics Forum 29,* 5, 1633–1642.

Reuter, M. 2010. Hierarchical shape segmentation and registration via topological features of laplace-beltrami eigenfunctions. *International Journal of Computer Vision 89,* 2, 287–308.

Reuter, M., Biasotti, S., Giorgi, D., Patanè, G., and Spagnuolo, M. 2009. Discrete laplace-beltrami operators for shape analysis and segmentation. *Computers & Graphics 33*, 381–390.

Rossignac, J. and Borrel, P. 1993. Multi-resolution 3d approximations for rendering complex scenes. In *Geometric Modeling and Computer Graphics*. Springer-Verlag, 455–465.

Rustamov, R. 2007. Laplace-Beltrami eigenfunctions for deformation invariant shape representation. In *Symposium on Geometry Processing*. Eurographics Association, 225–233.

Schaefer, S. and Warren, J. 2003. Adaptive vertex clustering using octrees. In *SIAM Geometric Design and Computing*. 491–500.

Schreiner, J., Scheidegger, C., Fleishman, S., and Silva, C. 2006. Direct (re) meshing for efficient surface processing. *Computer Graphics Forum 25,* 3, 527–536.

Seitz, S., Curless, B., Diebel, J., Scharstein, D., and Szeliski, R. 2006. A comparison and evaluation of multi-view stereo reconstruction algorithms. In *CVPR*. IEEE, 519–528.

Seversky, L., Berger, M., and Yin, L. 2011. Harmonic point cloud orientation. *Computers & Graphics 35,* 3, 492–499.

Shalom, S., Shamir, A., Zhang, H., and Cohen-Or, D. 2010. Cone carving for surface reconstruction. *ACM Transactions on Graphics 29,* 6, 150:1–150:10.

Shapeways. 2011. Shapeways. `http://www.shapeways.com`.

Shapira, L., Shamir, A., and Cohen-Or, D. 2008. Consistent mesh partitioning and skeletonisation using the shape diameter function. *The Visual Computer 24,* 4, 249–259.

Sharf, A., Alcantara, D., Lewiner, T., Greif, C., Sheffer, A., Amenta, N., and Cohen-Or, D. 2008. Space-time surface reconstruction using incompressible flow. *ACM Transactions on Graphics 27,* 5, 110:1–110:10.

Sharf, A., Lewiner, T., Shamir, A., Kobbelt, L., and Cohen-Or, D. 2006. Competing fronts for coarse–to–fine surface reconstruction. *Computer Graphics Forum 25,* 3, 389–398.

Shen, C., O'Brien, J., and Shewchuk, J. 2004. Interpolating and approximating implicit surfaces from polygon soup. *ACM Transactions on Graphics 23,* 3, 896–904.

Shi, J. and Malik, J. 2000. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence 22,* 8, 888–905.

Solomon, J., Ben-Chen, M., Butscher, A., and Guibas, L. 2011. Discovery of intrinsic primitives on triangle meshes. *Computer Graphics Forum 30,* 2, 365–374.

Sorkine, O., Cohen-Or, D., Irony, D., and Toledo, S. 2005. Geometry-aware bases for shape approximation. *IEEE Transactions on Visualization and Computer Graphics 11,* 2, 171–180.

Sud, A., Foskey, M., and Manocha, D. 2005. Homotopy-preserving medial axis simplification. In *Symposium on Solid and Physical Modeling.* ACM, 39–50.

Sun, J., Ovsjanikov, M., and Guibas, L. 2009. A concise and provably informative multi-scale signature based on heat diffusion. *Computer Graphics Forum 28,* 5, 1383–1392.

Sun, X., Rosin, P., Martin, R., and Langbein, F. 2009. Noise analysis and synthesis for 3D laser depth scanners. *Graphical Models 71,* 2, 34–48.

Surazhsky, V. and Gotsman, C. 2003. Explicit surface remeshing. In *Symposium on Geometry Processing.* Eurographics Association, 20–30.

Süssmuth, J., Winter, M., and Greiner, G. 2008. Reconstructing animated meshes from time-varying point clouds. *Computer Graphics Forum 27,* 5, 1469–1476.

Szlam, A., Maggioni, M., Coifman, R., and Bremer Jr, J. 2005. Diffusion-driven multiscale analysis on manifolds and graphs: Top-down and bottom-up constructions. In *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series.* Vol. 5914. 445–455.

Tagliasacchi, A., Olson, M., Zhang, H., Hamarneh, G., and Cohen-Or, D. 2011. Vase: Volume-aware surface evolution for surface reconstruction from incomplete point clouds. *Computer Graphics Forum 30,* 5, 1563–1571.

Tagliasacchi, A., Zhang, H., and Cohen-Or, D. 2009. Curve skeleton extraction from incomplete point cloud. *ACM Transactions on Graphics 28,* 3, 71:1–71:9.

Tam, R. and Heidrich, W. 2003. Shape simplification based on the medial axis transform. In *IEEE Visualization.* 481–488.

ter Haar, F., Cignoni, P., Min, P., and Veltkamp, R. 2005. A Comparison of Systems and Tools for 3D Scanning. In *3D Digital Imaging and Modeling: Applications of Heritage, Industry, Medicine and Land, Workshop Italy-Canada.*

Tevs, A., Berner, A., Wand, M., Ihrke, I., Bokeloh, M., Kerber, J., and Seidel, H. 2012. Animation cartography - intrinsic reconstruction of shape and motion. *ACM Transactions on Graphics 31,* 2, 12:1–12:15.

Tevs, A., Berner, A., Wand, M., Ihrke, I., and Seidel, H.-P. 2011. Intrinsic shape matching by planned landmark sampling. *Computer Graphics Forum 30,* 2, 543–552.

TEVS, A., BOKELOH, M., WAND, M., SCHILLING, A., AND SEIDEL, H. 2009. Isometric registration of ambiguous and partial data. In *CVPR*. IEEE, 1185–1192.

VALETTE, S., CHASSERY, J., AND PROST, R. 2008. Generic remeshing of 3d triangular meshes with metric-dependent discrete voronoi diagrams. *IEEE Transactions on Visualization and Computer Graphics 14,* 2, 369–381.

VALLET, B. AND LEVY, B. 2008. Spectral geometry processing with manifold harmonics,. *Computer Graphics Forum 22,* 2, 251–260.

VALLET, B. AND LVY, B. 2009. What you seam is what you get. Tech. rep., INRIA - ALICE Project Team.

VAN KAICK, O., ZHANG, H., HAMARNEH, G., AND COHEN-OR, D. 2011. A survey on shape correspondence. *Computer Graphics Forum 30,* 6, 1681–1707.

VLASIC, D., PEERS, P., BARAN, I., DEBEVEC, P., POPOVIĆ, J., RUSINKIEWICZ, S., AND MATUSIK, W. 2009. Dynamic shape capture using multi-view photometric stereo. *ACM Transactions on Graphics 28,* 5, 174:1–174:11.

WAND, M., ADAMS, B., OVSJANIKOV, M., BERNER, A., BOKELOH, M., JENKE, P., GUIBAS, L., SEIDEL, H., AND SCHILLING, A. 2009. Efficient reconstruction of nonrigid shape and motion from real-time 3d scanner data. *ACM Transactions on Graphics 28,* 2, 15:1–15:15.

XIE, H., WANG, J., HUA, J., QIN, H., AND KAUFMAN, A. 2003. Piecewise c1 continuous surface reconstruction of noisy point clouds via local implicit quadric regression. In *IEEE Visualization*. 91–98.

XU, K., ZHANG, H., COHEN-OR, D., AND XIONG, Y. 2009. Dynamic harmonic fields for surface processing. *Computers & Graphics 33,* 3, 391–398.

YAN, D., LÉVY, B., LIU, Y., SUN, F., AND WANG, W. 2009. Isotropic remeshing with fast and exact computation of restricted Voronoi diagram. *Computer Graphics Forum 28,* 5, 1445–1454.

ZELNIK-MANOR, L. AND PERONA, P. 2004. Self-tuning spectral clustering. *Advances in neural information processing systems 17*, 1601–1608.

ZHANG, H., SHEFFER, A., COHEN-OR, D., ZHOU, Q., VAN KAICK, O., AND TAGLIASACCHI, A. 2008. Deformation-driven shape correspondence. *Computer Graphics Forum 27,* 5, 1431–1439.

ZHANG, J., SIDDIQI, K., MACRINI, D., SHOKOUFANDEH, A., AND DICKINSON, S. 2005. Retrieving articulated 3-d models using medial surfaces and their graph spectra. In *Energy minimization methods in computer vision and pattern recognition*. Springer, 285–300.

ZHENG, Q., SHARF, A., TAGLIASACCHI, A., CHEN, B., ZHANG, H., SHEFFER, A., AND COHEN-OR, D. 2010. Consensus skeleton for non-rigid space-time registration. *Computer Graphics Forum 29,* 2, 635–644.

ZHENG, Y. AND TAI, C. 2010. Mesh decomposition with cross-boundary brushes. *Computer Graphics Forum 29,* 2, 527–535.

Zhou, D. and Burges, C. 2007. Spectral clustering and transductive learning with multiple views. In *International Conference on Machine Learning*. ACM, 1159–1166.