Understanding the Structure of Large, Diverse Collections of Shapes

Vladimir Georgievich Kim

A Dissertation

Presented to the Faculty

of Princeton University

in Candidacy for the Degree

of Doctor of Philosophy

Recommended for Acceptance

by the Department of

Computer Science

Adviser: Thomas A. Funkhouser

June 2013

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
JUN 2013		**00-00-2013 to 00-00-2013**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Understanding the Structure of Large, Diverse Collections of Shapes	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Princeton University,Princeton,NJ,08544**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Due to recent developments in modeling software and advances in acquisition techniques for 3D geometry, large numbers of shapes have been digitized. Existing datasets include millions of real-world objects, cultural heritage artifacts, scientific and engineering models all of which capture the world around us at nano- to planetary scales. As large repositories of 3D shape collections continue to grow, understanding the data, especially encoding the inter-model similarity and their variations, is of the utmost importance. In this dissertation we address the challenge of deriving structure from a large, unorganized and diverse collection of 3D polygonal models. By structure we refer to how objects correspond to each other, how they are segmented into semantic parts, and how the parts deform and change across the models. While previous work has generally dealt with small and relatively homogeneous datasets, in this dissertation we concentrate on diverse and large collections. Our contribution is three-fold. First, we present an algorithm for establishing correspondences between pairs of shapes related by a non-uniform deformation. Second, we develop a robust and efficient algorithm for computing per-point similarities between all shapes in a collection of 3D models using only a small subset of all pairwise alignments. And third we describe an algorithm for finding structure in an unorganized, unlabeled collection of diverse 3D shapes, which is achieved by jointly optimizing for point-to-point correspondences part segmentations and an explicit model of part deformations. These algorithms enable finding correspondences in large diverse datasets where models are related by non-uniform deformations and model parts have different multiplicity and geometry. These methods also make it possible to segment large collections into consistent sets of parts and to represent most prominent geometric variations in the entire collection.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**141**	

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std Z39-18

Abstract

Due to recent developments in modeling software and advances in acquisition techniques for 3D geometry, large numbers of shapes have been digitized. Existing datasets include millions of real-world objects, cultural heritage artifacts, scientific and engineering models, all of which capture the world around us at nano- to planetary scales. As large repositories of 3D shape collections continue to grow, understanding the data, especially encoding the inter-model similarity and their variations, is of the utmost importance.

In this dissertation we address the challenge of deriving *structure* from a large, unorganized, and diverse collection of 3D polygonal models. By structure we refer to how objects *correspond* to each other, how they are *segmented* into semantic parts, and how the parts *deform* and *change* across the models. While previous work has generally dealt with small and relatively homogeneous datasets, in this dissertation we concentrate on diverse and large collections.

Our contribution is three-fold. First, we present an algorithm for establishing correspondences between pairs of shapes related by a non-uniform deformation. Second, we develop a robust and efficient algorithm for computing per-point similarities between all shapes in a collection of 3D models using only a small subset of all pairwise alignments. And third, we describe an algorithm for finding structure in an unorganized, unlabeled collection of diverse 3D shapes, which is achieved by jointly optimizing for point-to-point correspondences, part segmentations and an explicit model of part deformations.

These algorithms enable finding correspondences in large diverse datasets where models are related by non-uniform deformations and model parts have different multiplicity and geometry. These methods also make it possible to segment large collections into consistent sets of parts and to represent most prominent geometric variations in the entire collection.

Acknowledgements

First of all, I would like to thank my advisor, Thomas Funkhouser, for his invaluable advice, support and mentorship throughout my graduate studies. In these five years, he set a life-long example for me demonstrating what it means to be a scientist and a teacher.

I am also grateful to my mentors, Wilmot Li, Niloy J. Mitra, Yaron Lipman, Szymon Rusinkiewicz, and Adam Finkelstein, who shared their knowledge, expertise, and time and were always happy to discuss different ideas with me. This dissertation would not be possible without considerable contribution from my collaborators: Stephen DiVerdi, Tianqiang Liu, Siddhartha Chaudhuri, Xiaobai Chen, and Aleksey Golovinskiy. And, of course, my life at Princeton would not be nearly as fun without other graduate students at Princeton Graphics Group.

My collaborators and I would like to thank everyone who helped us in comparisons and evaluations by distributing their code and data. We acknowledge Daniela Giorgi and AIM@SHAPE for the SHREC 2007 Watertight Models, Drago Arguelov and Stanford University for the SCAPE data set, Project TOSCA for the Non-rigid World models, and Yunhai Wang for COSEG dataset. We thank Maks Ovsjanikov, Michael Bronstein, Qi-Xing Huang, and Andy Hguyen for distributing their code, as well as thank Hao Zhang, Oliver van Kaick, Michael Wand, and Art Tevs for running their codes on our test cases.

My graduate studies and this work were supported by Princeton Fellowship, NSERC PGS-M and NSERC PGS-D awards, Siebel Scholarship program, Adobe Inc. research internships, Intel (ISTC-VC), Google, and the following grants: NSF (IIS-0612231, CNS-0831374, CCF- 0702672, CCF-0937139, and CNS-0831374) and AFOSR (1096101).

To my grandmother, Zoya,

my grandfather, Nickolai,

my father, George,

my mother, Irina,

and my wife, Kelima.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

With the increasing number and diversity of 3D polygonal models in online repositories, there is a growing need for automated algorithms that can derive structural and semantic relationships from large model collections. For example, the SketchUp Warehouse contains millions of 3D models in many different classes and, thus, should be a valuable resource for data-driven solutions to common geometry processing problems, including surface reconstruction [73], model completion [112], model-based object recognition [111, 93], and shape synthesis [61]. In addition, analyzing the diversity of shapes in this database could yield insights into the geometric variations of real-world objects. Unfortunately, most Web-based repositories, including the SktechUp Warehouse, do not contain the necessary structural and semantic information required to support such applications. They have little semantic tagging, no consistent part decompositions, and no information about how surfaces on different objects relate to one another. As a result, it is difficult to get an overarching view of what types of models are in the repository, how models correspond to and differ from each other, and whether they contain the necessary information for a given application.

We describe several analysis tools developed for deriving structure from large, unorganized, diverse collections of 3D polygonal models (e.g., shapes within the same general object class, like chairs). More specifically, we provide automatic tools to compute how objects correspond to each other, how they can be segmented into semantic parts, and how parts or arbitrary regions vary across models. Our analysis results can then be directly used for many data-driven geometric acquisition, analysis, processing, and modeling tasks.

In this work we push the limits of existing efforts on analysis of 3D collections in two directions: we provide tools that handle a higher diversity of models, and that are suitable to large collections.

First, we describe a method for finding correspondences between pairs of shapes related by a non-uniform deformation, which is common in diverse collections [71]. We propose a fully automatic pipeline for creating an intrinsic map between two non-isometric, genus zero surfaces. Our approach is based on the observation that efficient methods exist to search for nearly isometric maps, but no single solution found with these methods provides universal low-distortion for pairs of surfaces differing by large deformations. To address this problem, we suggest using a weighted combination of these maps to produce a blended map. This approach allows for algorithms that can leverage efficient search procedures while still providing the flexibility to handle large deformations. During experiments with these methods, we find that our algorithm produces blended maps that align semantic features better than alternative approaches across a variety of data sets.

Second, leveraging our (or any other) pairwise mapping technique, we propose an automatic analysis method for computing similarity relationships between points on 3D shapes across a collection [73]. Our method can accommodate substantial geometric variations, including non-uniform deformations, and partial similarity (e.g. due to different part multiplicities or styles). We encode the inherent ambiguity in similarity relationships using fuzzy point correspondences and propose an efficient computational framework that esti-

mates fuzzy correspondences using only a few pairwise model alignments. We evaluate our analysis method on a range of correspondence benchmarks and report substantial improvements in both speed and accuracy over existing alternatives.

Finally, we tackle the problem of deriving structure from an unorganized, unlabeled collection of diverse 3D shapes [69]. We propose an automatic algorithm that starts with an initial template model and then jointly optimizes for point-to-point surface correspondence, part segmentation, and a compact deformation model to best explain the input collection of shapes. As output, the algorithm produces a set of probabilistic part-based templates that groups the original models into clusters according to their styles and variations. We evaluate our algorithm on several standard datasets and demonstrate its scalability by analyzing large collections, some of which contain thousands of shapes.

Contributions: This dissertation makes the following contributions:

- an idea of combining multiple low-dimensional intrinsic maps to produce a blended map between two non-isometric surfaces,
- an optimization pipeline that produces a globally optimal blending weights for a set of intrinsic maps,
- an idea of using fuzzy correspondences to understand similarity relations across 3D model collections
- an algorithm to compute fuzzy correspondences from sparse and noisy pairwise alignments or intrinsic maps,
- an idea of using joint structural analysis to simultaneously recover segmentation, point-level correspondence, and a probabilistic part-based deformable model for shape collections; and,
- an efficient out-of-core framework to establish structure in diverse collections spanning thousands of models at a scale never demonstrated before.

Chapter 2

Related Work

A significant amount of research has focused on finding structure in data collections such as text documents [12], audio files [107], and natural images [31]. Common analysis tools for these datasets provide high-level abstractions by measuring similarities [54], by classifying the entities [13], by decomposing the entities into parts [21], or by exploring and summarizing interesting similarity relationships [51]. In our work we develop analogous tools for analysis of collections of 3D shapes.

2.1 Analysis of individual shapes

Most of the previous work on structural analysis of 3D geometry concentrates on analyzing individual models.

Feature Point Detection: Many methods have been developed to address the problem of detecting salient feature points in 3D shapes. Existing approaches commonly use extrema of various surface functions, such as average geodesic distances [134, 132, 131], curvature [80], and the difference of Gaussians at multiple scales [22]. Other methods rely on analysis of heat kernel signature [117], scale-space analysis of mean curvature flow [130],

or taking leaf nodes of skeleton extraction [52]. These methods generally identify features that are stable under different geometric perturbations, but do not necessarily correspond to human perception of structurally important regions.

Saliency Estimation: In a related research, Hoffman et al. [53] defined perceptual criteria for part saliency. This work further inspired several automatic methods for the identification of perceptually important regions on 3D shapes. For example, Lee et al. [77] used curvature at different scales to compute the saliency of different regions, and demonstrate applications in mesh simplification and viewpoint selection. Gal and Cohen-Or [39] combined relative region size, curvature, the variance of curvature and the number of curvature changes within the region to find salient regions. Finally, Chen et al. [26] performed a perceptual study to identify the geometric features that are most useful in predicting saliency, and proposed an automatic classifier to extract salient feature points and estimate saliency values on a surface.

Segmentation: Many methods address the problem of decomposing a model into parts using low-level geometric cues, such as convexity of parts, or concavity of boundaries between parts. They optimize for these criteria by using techniques, such as graph cuts [64, 47], spectral clustering [82], hierarchical clustering [40], primitive fitting [8], and the discovery of intrinsic primitives [116]. A thorough survey of these methods can be found in Shamir [109], while Chen et al. [25] examined relative performance of these techniques.

Symmetry Detection: The problem of finding symmetric structures has also received a fair amount of attention in previous years. Existing methods are able to find affine transformations that align symmetric points [88, 89], compute structural regularity [102], and detect hierarchical symmetric relationships [122]. In an intrinsic setting, existing techniques search for global [100, 70] and partial [129] symmetries at multiple scales [128]. A detailed overview of these methods can be found in Mitra et al. [90]. Discovering these symmetric structures enables a variety of interesting editing tools [89][133][72].

Although analysis tools for individual models can extract useful semantic structures, these structures do not represent relationships between different shapes in a large collection, which is the main topic of our work.

2.2 Analysis of 3D collections

Finding relationships across an entire collection of models provides the opportunity to devise high-level understanding of classes of shapes. Previous work has explored applications of this general idea to various problems in computer graphics and vision.

Shape similarity: Defining a geometric similarity for a pair of shapes is a long-standing problem in geometry analysis. Much work has been done on developing concise and discriminative shape descriptors for comparing geometries. They commonly rely on low-level features such as distribution of surface normals [55, 63], histograms of surface distribution with respect to its center of mass [7], histograms of distances between pairs of points [96], 2D projections from different points of view [24], spherical harmonics [67], Zernlike moments [20, 95], symmetry descriptors [66], and others (e.g. see [65, 17], and references therein). Commonly these similarity metrics are employed for classification and retrieval problems.

Classification: Automatically grouping shapes into similar classes is a key problem in organizing collections of 3D models, or analyzing geometries obtained with an acquisition device. A common solution to this problem is to classify a shape based on its nearest neighbors in a descriptor space [114], which requires a small collection of pre-labeled exemplars as an input. In a related effort, Shilane et al. [113] proposed identifying distinctive local features of a shape by employing joint analysis of a 3D collection, and then using these features to improve discrimination between different classes of shapes and to identify salient regions.

Several methods address the problem of classifying shapes in indoor and outdoor environments. These approaches commonly use local shape features to identify and segment objects from the scene, and then use a classifier learned from a training set [45, 74]. Moreover, Nan et al. [93] developed a semi-interactive approach for object recognition in depth scans.

Retrieval: Shape retrieval, which is aimed at finding a particular shape in a collection, has also received substantial research attention in recent years. Many existing search systems (e.g. [119]) rely on user-provided text tags to facilitate navigation in the shape space. In practice, these tags are often unreliable and, more importantly, are not sufficiently descriptive when the search parameters are more subtle than just a shape class. Although it is possible to use an example shape to define a query [114], creating the 3D geometry for each query might be too complicated for typical search operations. Several approaches address this limitation by allowing 2D contours as part of the input [87, 86]. Eitz et al. [28] also demonstrated results with rough sketches, however, their method requires having a collection of annotated sketches of classes of interest.

Exploration: In cases when the exact characteristics of a retrieved object or variety of shapes in a collection are not known there is a need for tools that allow open-ended exploration. Common exploration objectives are to identify what kind of objects are there in the database, how they relate to one another, and how they vary. For example, Giorgi et al. [44] offered a system that allows for the interactive refinement of search results via relevance feedback. More recent approaches focus on helping users to explore and understand the variations within a class, which are often more subtle. Many of these methods are specific to a particular class of shapes, such as human bodies [2, 6]. Ovsjanikov et al. [98] presented a general method that uses a coupled spatial-descriptor space analysis to extract a template-based deformation model that the user can directly manipulate to explore the collection.

Data-driven reconstruction: Reconstructing poorly sampled geometry (e.g. due to limitations of an acquisition device) is one of fundamental problems in geometry processing. Several solutions stemmed from the observation that a collection of shapes can provide strong geometric priors, which in turn can improve reconstruction. For example, Pauly et al. [101] demonstrated a system for completing geometries using similar shapes from a collection. Likewise, Shen et al. [112] used example object parts to reconstruct a 3D model from sparse point samples, and Shao et al. [110] provided an interactive data-driven system to reconstruct 3D scenes from depth scans. For some classes of shapes, such as human faces, specialized methods encode the geometric priors in a class-specific deformable template, which can be further used in reconstruction [10].

Data-driven synthesis: Early efforts concentrated on using a collection to build a generative model for specific classes of shapes, such as human faces [11, 46] or human bodies [6, 2]. Funkhouser et al. [37] introduced the idea that similarities in structure between shapes from the same class can be exploited to synthesize new shapes by reshuffling similar parts or regions. This idea was further developed in a probabilistic generative model for the interactive assembly-based shape synthesis [23], or the automatic synthesis of new collections [61]. Similar analysis tools were developed for collections of scenes. For example, analyzing manually-labeled 3D scene graphs allowed for encoding placement relationships across a variety of objects, such as pieces of furniture in indoor scenes [36, 35].

The shape structure recovered by algorithms described in this thesis can be directly used in all aforementioned applications.

Chapter 3

Correspondence for pairs of 3D shapes with Blended Intrinsic Maps

3.1 Introduction

Finding a map between two surfaces is a fundamental problem in understanding the structural relationship between models. The objective is to find an *intrinsic map* $f : \mathcal{M}_1 \to \mathcal{M}_2$, for a pair of non-isometric meshes $\mathcal{M}_1$ and $\mathcal{M}_2$, such that f is smooth and "low-distortion" everywhere (as isometric as possible). With such a map, it is possible to reason about functional similarities [14], study surface variations [2], and find consistent part decompositions [48].

The general approach to this problem is to search a discrete space of possible maps, selecting the one that minimizes the prescribed distortion measure. With this discrete formulation, the key challenge is to select a space of maps that is both small enough to search efficiently and large enough to contain useful maps between non-isometric surfaces found in real-world problems. One approach is to search an exponentially large space of maps (e.g., all $N!$ sets of correspondences between N sparse feature points), which can include a wide

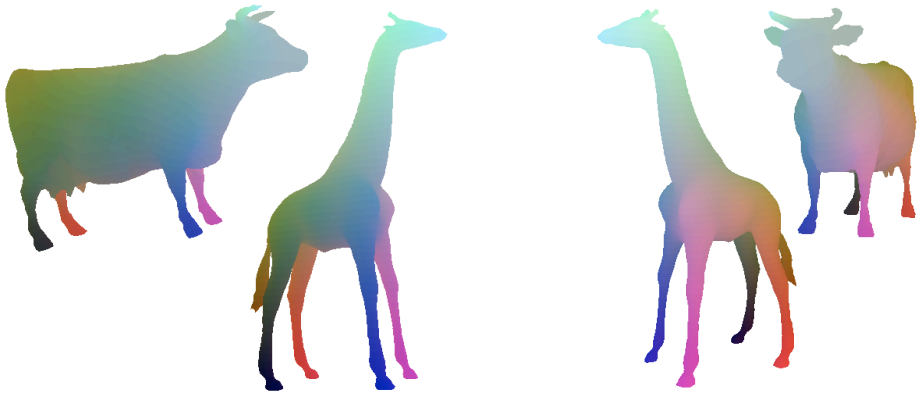

Figure 3.1: *Automatically-extracted map f between cow and giraffe (same map is rendered from two viewpoints). We color each vertex on giraffe's body by it's $\{x, y, z\}$ position. Then every vertex v on the cow's body is mapped to the giraffe by f, and colored the same as $f(v)$*

variety of useful deformations, but requires an NP-Hard search algorithm. An alternative approach is to search a low-dimensional space of intrinsic maps (e.g. using geodesic feature vectors [18], Heat-Kernel maps [99], conformal maps [103], quasi-conformal maps [81] etc.), where polynomial-time search algorithms are available, but where there is hardly any variety of deformations. The problem is that no known space of maps is both polynomial in size and contains the deformations commonly found in real-world surface correspondence problems (e.g., even articulations of people and animals can deviate significantly from conformality or isometry). As such there is no obvious solution to this problem.

Our approach is to search for a continuous blend of multiple low-dimensional maps. By combining maps with weights varying smoothly over the surface, we define a space of maps that includes a large range of deformations, yet can still be searched with polynomial-time algorithms. We consider blends of conformal maps with weights that: 1) are proportional to the area-preservation of the map at every point, and 2) incorporate global similarity relations between different conformal maps. In this way, we favor maps that locally aim to preserve both angles and areas (i.e., near-isometries), but globally are consistent and can achieve extreme deformations.

This method finds a smooth map in polynomial time that empirically aligns semantic features of non-isometric meshes effectively. During experiments with a test set of 334 surface pairs, our blended map is able to align benchmark correspondence points on different meshes within the same object type better than several state-of-the-art methods. For example, a blended map between a cow and a giraffe is shown in Figure 3.1 (a failure case in [80]) – note that the map is nearly-isometric locally, even though it provides a smooth map between significantly different shapes.

Contributions: Blended intrinsic maps make four main research contributions:

- the idea of combining multiple low-dimensional intrinsic maps to produce a blended map,
- an objective function for a weighted collection of maps that favors both the confidence of maps and consistency between pairs of maps,
- a method for estimating the consistency of two maps at a point, and
- an optimization pipeline that produces a globally optimal weight assignment for a set of maps.

3.2 Related Work

Finding correspondences between surfaces is a long standing problem addressed a wide variety of previous methods [121, 16].

3.2.1 Inter-surface mapping

Given a set of correct sparse correspondences (defined by a user or an algorithm), one can use a variety of methods to find a smooth map to interpolate them. A common approach is to map both surfaces to a canonical domain where sparse feature points align and then interpolate the map in that domain [1]. For example, [105] used a base coarse mesh (pro-

vided by a user) as such a domain. In their approach, the surface is cut into triangular patches defined by three geodesic curves, such that each geodesic curve is mapped to a triangle on a coarse base mesh. Further, [108, 76] developed an automatic approach for creating the base domain. These methods, however, were only evaluated with manually labeled sparse correspondences as their input, which are too expensive to be used for finding sparse correspondences in a fully automatic algorithm.

3.2.2 Finding sparse correspondences

Several methods have been proposed for automatically finding a small (sparse) set of feature correspondences, which could then be used to produce an inter-surface map. The most common approach is to first extract a set of feature points and then to explore permutations of them to find the correspondences implying an alignment with minimal deformation error [57, 132]. This approach is effective when local shape descriptors at the feature points are very distinctive, but quickly becomes too expensive when local shapes are different and the measurement of global deformations implied by many points is required for discriminating the optimal solution. Even with pruning based on branch-and-bound [41] or priority-driven search [38], the search space is simply too large to explore efficiently, and so heuristics are employed and/or few feature correspondences are found, which makes it difficult to find a good inter-surface map.

3.2.3 Iterative closest points

Certain approaches find surface correspondences through an iterative procedure that starts with an initial correspondence, and then repeatedly improve it by computing an aligning transformation from the correspondences and then updating the correspondences based on the transformation (e.g., based on mutually closest points). This method is most commonly

used for aligning surfaces related by a rigid transformation [9], but has also been used for moderate non-rigid deformations [2, 19, 78, 101, 118, 42]. Unfortunately, it does guarantee that the final map is smooth or bijective (two points on one surface may map to the same point on another), and it requires a good initial guess to succeed in most cases.

3.2.4 Finding dense correspondences

Other methods directly find correspondences for all points on a surface. For example, the Gromov-Hausdorff distance motivated a purely intrinsic approach by Memoli et al. [85] to measure deviation from isometry between two surfaces. Furthermore, Bronstein et al. [18, 16] developed a Generalized Multidimensional Scaling (GMDS) framework to find the least distortion embedding of one surface onto another, which can be computed more efficiently in a coarse-to-fine manner [106]. These methods use an approximate search procedure to make an initial guess of point correspondences and, thus, may converge to a local minimum.

3.2.5 Surface embedding

Some methods find dense correspondences by embedding surfaces in feature space where similar points have similar coordinates and then produce a dense map based on nearest neighbors in that space. For example, Ovsjanikov et al. [99] showed that a single correspondence can define a Heat Kernel Map (HKM), a high dimensional embedding of a surface invariant under isometry. The disadvantage of these methods is that they are effective only for deformations that are nearly isometric – otherwise features do not align in the embedded space. Ovsjanikov et al. [99] suggested a simple extension of their approach to non-isometric cases: they concatenate features from two heat kernel maps generated by two correspondences into a single feature vector and search for nearest neighbors in that

space. However, it is still not obvious how to best select multiple correspondences, and the resulting map is still not guaranteed to be smooth, or even continuous, when surfaces are not isometric.

3.2.6 Functional Maps

An alternative to searching for point-to-point correspondences is to search for a functional map: a map that puts real-valued functions on two surfaces in correspondence. Ovsjanikov et al. [97] introduced this idea, and proposed eigenfunctions of the Laplace-Beltrami operator as a multi-scale basis for the function space on each shape. Pokrass et al. [104] further demonstrated that a sparse modeling method can generate better results if part-wise decomposition of each shape is provided as an input. However, choosing this as the basis of the function space is only justified for near-isometric shapes, and, thus, these methods perform poorly in non-isometric cases (see Figure 6 in [104]). Recently, Kovnatsky et al. [75] applied coupled quasi-harmonic bases to find functional maps between non-isometric shapes.

Note that these methods were published after the work described in this thesis [71].

3.2.7 Exploring Möbius Transformations

The methods most similar to ours are the ones of [80] and [70]. They both leverage the fact that isometries are a subspace of the conformal maps, which are low-dimensional and can be explored efficiently with Möbius Transformations. They differ in the way they search and combine the maps: [70] uses a RANSAC algorithm to discover the single "best" conformal map that maps a surface onto its reflection, while [80] combines multiple maps with an algorithm that votes for correspondences. The former approach works only for nearly-isometric surfaces (e.g., intrinsic symmetries), while the latter approach may pro-

duce globally inconsistent correspondences (when votes for inconsistent maps combine in the correspondence matrix).

In this thesis we propose a fully automated method that finds a smooth, low-distortion map between significantly non-isometric surfaces in polynomial time.

3.3 Method

3.3.1 Key Idea

Our approach is to search for a map that smoothly blends multiple low-dimensional maps. This approach allows a search procedure to explore a polynomial size space of maps, while providing the flexibility to find smooth maps between surfaces differing by significant deformations.

A motivation for this approach is provided in Figure 3.2. Our goal in this example is to produce a smooth map between the surfaces of a person in two different poses, such that the map is "as isometric as possible." Although the surfaces are nearly isometric, there is no known low-dimensional map that takes one surface onto the other with small distortion everywhere. For example, the top row shows three conformal maps, each of which provides low distortion (blue) for most of the body, but has large distortions (red) on different parts of the arms and head. While none of these conformal maps provide a good solution for the entire body, they can be combined with weights (bottom left) to form a blended map with small distortion almost everywhere (bottom right).

We investigate ways to define blended maps and compute them automatically for pairs of genus zero surfaces. Specifically, given a pair of surface meshes $\mathcal{M}_1$ and $\mathcal{M}_2$, our goal is to find a set of K candidate maps $\{m_i\}_{i=1}^{K} : \mathcal{M}_1 \to \mathcal{M}_2$ associated with smooth blending weights $b_i(p)$ for every point p, such that the blended map $f : \mathcal{M}_1 \to \mathcal{M}_2$ defined as

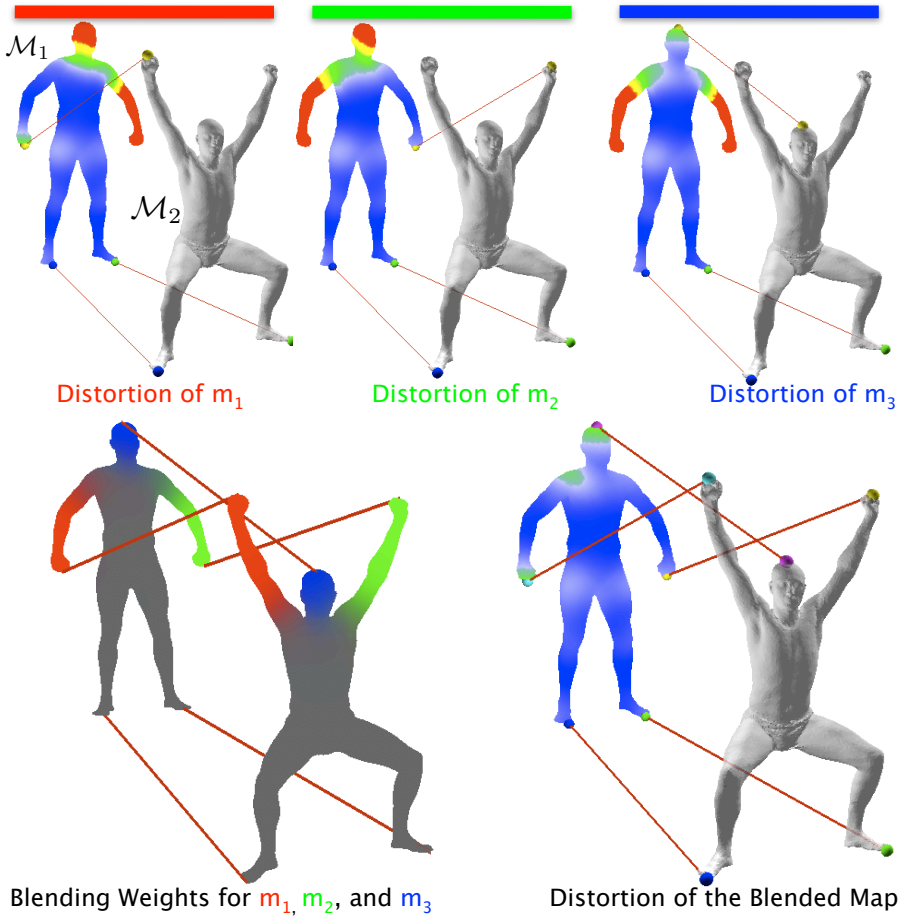

Figure 3.2: *Motivation for our approach. In the top row, three low-dimensional conformal maps $m_1, m_2,$ and m_3 are defined by three correspondences rendered as red lines. Map confidence is rendered in color on a surface, where blue corresponds to 0 area distortion, and red is high distortion. The three low dimensional maps are blended according to weights encoded as RGB colors in the bottom-left image. The resulting blended map in the bottom-right corner has lower local distortion than any of the three maps above.*

follows has low distortion across the entire surface:

$$f(p) = \underset{p' \in \mathcal{M}_2}{\operatorname{argmin}} \sum_{i=1}^{K} b_i(p) d_{\mathcal{M}_2}\left(p', m_i(p)\right)^2, \tag{3.1}$$

where $d_{\mathcal{M}_2}(\cdot,\cdot)$ denotes the geodesic distance on surface $\mathcal{M}_2$.

Intuitively, this definition maps every point p to the weighted geodesic centroid of its images $\{m_i(p)\}$ for all maps $\{m_i\}_{i=1}^K$. We use this definition because it guarantees smoothness of the blended map if the candidate maps $\{m_i(p)\}$ and blending weights $\{b_i(p)\}$ are

both smooth, and because we believe that it contains most deformations commonly found in real-world surface correspondence problems. For instance, in the example shown in Figure 3.2, weights are chosen such that only one of the three candidate maps influences each of the arms and the head strongly, but they blend together smoothly to form a map with no discontinuities and low overall distortion on the body.

Given this definition, the key research challenge is to provide automatic methods to generate a set of candidate maps with associated blending weights. Intuitively, the ideal solution should guarantee smoothness of the blending weights $b_i(p)$, assign large weights at p only to maps m_i that induce small distortions at $m_i(p)$, and assign non-zero weights at p only for sets of maps that are consistent with one another (i.e., if $b_i(p) > 0$ and $b_j(p) > 0$, then $d_{\mathcal{M}_2}(m_i(p), m_j(p))$ should be small). If these constraints are satisfied, then the resulting map will be smooth and have low distortion everywhere.

In this paper, we focus on algorithms to address these challenges for blending *conformal* maps. Conformal maps are low-dimensional and thus efficient to search; they preserve angles and thus avoid distortions with shear; they contain isometries as a special case and thus they are a common type of map for non-rigid deformations; and, finally, they simplify the problem of finding blending weights in our formulation, because it is possible to estimate analytically the distortion of a conformal map at given point (e.g., how well area is preserved in the point's neighborhood). Thus, we can partition the computation of blending weights into two factors:

$$b_i(p) = c_i(p) \cdot w_i(p),$$

where $c_i(p)$ measures the "confidence" of the conformal map m_i at point p based on an estimate of its area-preservation at p, and $w_i(p)$ are the "consistency" weights that indicate to what extent a map should be used for blending. The key observation is that the first factor, $c_i(p)$, already provides a smoothly varying estimate for the distortion of each map m_i at p (as shown in the top row of Figure 3.2), and thus captures the spatially varying

aspects of $b_i(p)$. The second factor, $w_i(p)$, can then be treated as a constant across the surface, which greatly simplifies computation of optimal blending weights.

The following four sections describe our algorithm to compute blended maps between two surfaces automatically. Given two input surfaces, we first generate a set of candidate conformal maps (Section 3.3.2). Then, we estimate the confidence $c_i(p)$ for each map at every point p (Section 3.3.3). We next compute consistency weights w_i for every conformal map m_i by optimizing an objective function that favors non-zero weights only for sets of maps that are both high confidence and consistent with one another (Section 3.3.4). Finally, we produce a final blend with these weights using Equation (3.1) (Section 3.3.5). The methods employed in Sections 3.3.2, 3.3.3, and 3.3.5 are straight-forward – detailed descriptions are included mainly for the sake of completeness and reproducibility. Figure 3.3 provides an illustrative example for our pipeline. Our main algorithmic contribution is in Section 3.3.4, which describes a method for finding optimal consistency weights to be used for blending in our formulation.

3.3.2 Generating Maps: $\{m_i\}_{i=1}^{K}$

Our first step is to generate conformal maps that will form a candidate set for blending. Our goal is to provide a small set of maps such that at least one map achieves small distortion at every important feature point, and such that areas mapped with low distortion by different maps overlap significantly so that they can be blended without distortion.

To generate such a candidate set, we follow the procedure used in [80, 70]. We first compute a small collection of feature points, $\mathcal{P}_1 \subset \mathcal{M}_1$ and $\mathcal{P}_2 \subset \mathcal{M}_2$, on both surfaces. Then, we generate candidate conformal maps by enumerating triplets of three feature point correspondences, where each pair of triplets uniquely defines a Möbius transformation that maps one surface onto the other conformally while interpolating the feature point correspondences.

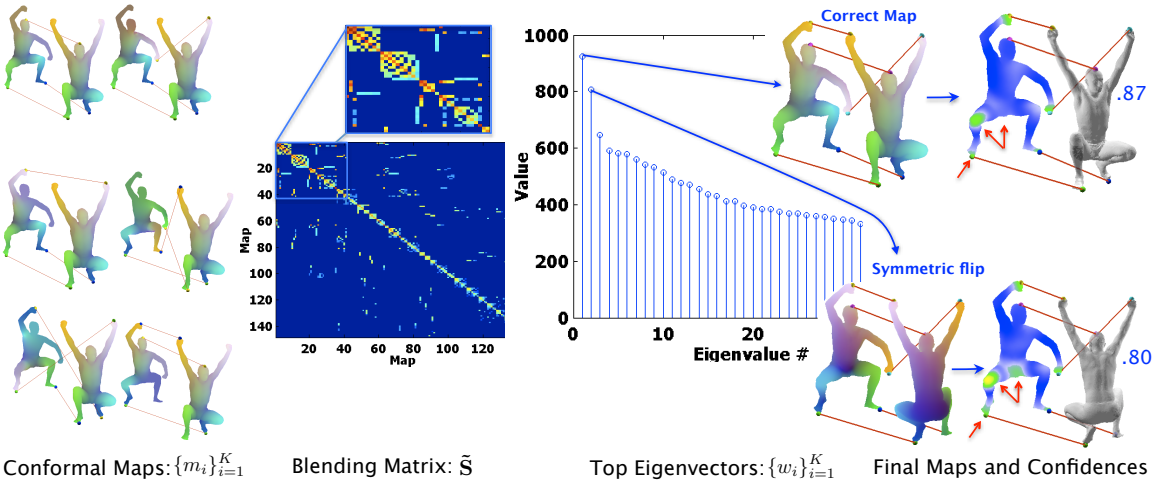

Figure 3.3: *This figure depicts the blending matrix $\tilde{\mathbf{S}}_{i,j}$ rearranged by consistent blocks of decreasing size. Each row (and column) corresponds to a conformal map. Several conformal maps are illustrated on the left. Note that $\tilde{\mathbf{S}}_{i,j}$ is sparse and contains approximately two large blocks that correspond to the correct near-isometric map and its symmetric flip. We also show the spectrum of this matrix. Note the spectral gap separating the two eigenvectors corresponding to the near-isometries from the rest of the spectrum. On the right we show the two blended maps corresponding to these top two eigenvectors, together with their confidence functions and its integral (single number printed in blue). Note the red arrows that indicate small regions that allow us to distinguish between the correct map and the symmetric flip. For instance, mapping front of a human to the back results in more area distortion at feet (due to heels), at knees, and at buttocks.*

Generating feature points: our first task is to generate sets of feature points $\mathcal{P}_1$ and $\mathcal{P}_2$ on $\mathcal{M}_1$ and $\mathcal{M}_1$. Our goal is to produce a small number of feature points with a large fraction of semantic correspondences. Although many methods are possible, we currently extract points at maxima of the Average Geodesic Distance function $AGD_{\mathcal{M}_\ell}(p) = \int_{\mathcal{M}_\ell} d_g(p, p')dA(p')$, where dA denotes the area element on the surface $\mathcal{M}_\ell$, $\ell = 1, 2$. This method provides corresponding feature sets particularly well for articulated figures (e.g., tips of extremities and top of head), and thus a small number of features is usually required to achieve multiple semantic correspondences spread throughout the surfaces. For most examples presented in this paper $|\mathcal{P}_\ell| \leq 10$.

Generating conformal maps: our next task is to generate a candidate set of conformal maps $\{m_i\}_{i=1}^K$. Following [80], we first conformally map the two surfaces to the extended

complex plane using mid-edge uniformization [103]. We then generate a set of conformal maps by enumerating all possible combinations of three correspondences between feature points in $\mathcal{P}_1$ and $\mathcal{P}_2$ (*generating correspondences*) and construct a conformal map m_i for each one by computing the Möbius transformation that interpolates all three corresponding points. This procedure produces $K = \binom{|\mathcal{P}_1|}{3} \cdot \binom{|\mathcal{P}_2|}{3} \cdot 6$ distinct conformal maps that form the set $\{m_i\}_{i=1}^{K}$ that will be candidates for blending in the following steps. Please refer to [80, 70] for details.

3.3.3 Defining Confidence Weights: $\{c_i(p)\}_{i=1}^{K}$

Our second step is to compute a confidence value $c_i(p)$ that estimates how much distortion is induced by each map m_i at every point p. Though many formulations are possible to measure distortion at a point, in this work we aim to estimate deviations from isometry.

Since conformal maps preserve angles, we can estimate deviation from isometry simply by measuring the scale factor induced by the map at every point p (isometries are conformal maps that preserve scales). To do so, we define

$$c_i(p) = 2 \Big/ \left[\frac{area(N_p)}{area(m_i(N_p))} + \frac{area(m_i(N_p))}{area(N_p)} \right], \tag{3.2}$$

where $area(N_p)$ is the area of a neighborhood, N_p, around point p on $\mathcal{M}_1$ and $area(m_i(N_p))$ is the area of its image, $m_i(N_p)$, on $\mathcal{M}_2$.

For computational efficiency, we calculate $c_i(p)$ only at a set $\mathcal{P}_{\text{even}}$ of 256 approximately evenly distributed points. The set is produced by starting with a random vertex and then iteratively adding vertices that are farthest from the set $\mathcal{P}_{\text{even}}$ until $|\mathcal{P}_{\text{even}}| = 256$ [29]. Confidence weights $c_i(p)$ for all other vertices are calculated using smooth interpolation with Gaussian weights.

These estimates of $c_i(p)$ based on area preservation are quick to compute, vary smoothly across the surface, and correlate well with low-distortion in a map, and thus they provide the desired properties of the spatially varying factor in our blending weights.

3.3.4 Finding Consistency Weights: $\{w_i\}_{i=1}^K$

The next step is to compute a set of consistency weights $\{w_i\}_{i=1}^K$ for all conformal maps $\{m_i\}_{i=1}^K$. Ideally, this set will have weights with zero values for conformal maps that induce high distortion (e.g., the generating triplet of feature correspondences contains incorrect matches) and non-zero weights only for conformal maps that are consistent with one another.

Objective Function: Following this intuition, we define the consistency weights $\vec{w} := \{w_i\}_{i=1}^K$ as the minimizer of an objective function, $E(\vec{w})$:

$$\begin{aligned} E_{\mathcal{M}_1}(\vec{w}) \quad &= \sum_{i=1}^K \sum_{j=1}^K w_i w_j \int_{p \in \mathcal{M}_1} S_{i,j}(p) c_i(p) c_j(p) dA(p) \\ &\text{subject to} \quad \sum_{i=1}^K w_i^2 = 1, \end{aligned} \tag{3.3}$$

where confidence values $c_i(p)$ are defined as in the previous section, and pairwise map consistency values $S_{i,j}(p) : \mathcal{M}_1 \rightarrow \mathbb{R}$ provide an estimate of how consistent two maps are at a point p. We constrain the L_2 norm of weights to be 1 since we want to favor global maps that include multiple similar maps.

Roughly speaking, for every choice of weights $\vec{w}$ giving non zero weights to some subset of maps, the functional measures how pairwise consistent is this set and how well each individual in this set preserves area. The weights achieving the minimum of this objective function will signal out the correct set of maps to be used in the blending.

Map Consistency: The most important term in the objective function is the similarity measure $S_{i,j}(p)$ for a pair of maps m_i, m_j. Intuitively, $S_{i,j}(p)$ should be high if maps m_i

and m_j are similar at point p. To model this intuition, we define the consistency for a pair of maps m_i and m_j at a point p to be inversely related to the geodesic distance between images of the point p under the two maps, $m_i(p)$ and $m_j(p)$:

$$S_{i,j}(p) = \exp\left(-\frac{d_{\mathcal{M}_2}\left(m_i(p), m_j(p)\right)}{\sigma^2}\right) \tag{3.4}$$

Note that $0 \leq S_{i,j}(p) \leq 1$ at any point p, $S_{i,j}(p) = 1$ iff $m_i(p) = m_j(p)$, and σ is a controllable parameter that controls how close images of a mapped point should be in order for a pair of maps to be considered similar (we use $\sigma = 0.5$ for all results in this paper).

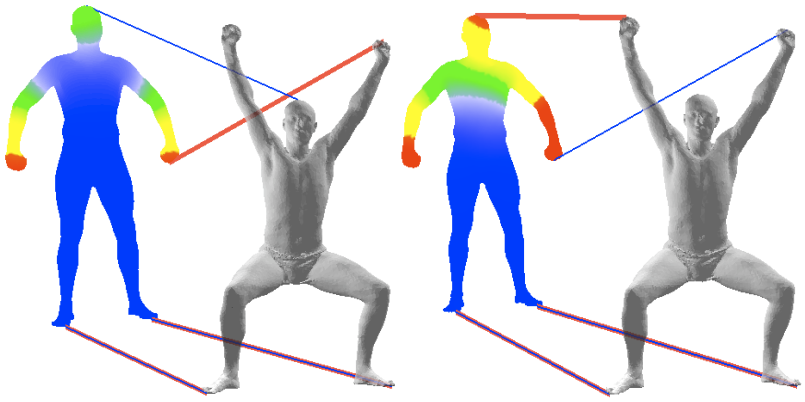

Figure 3.4: *These images show the similarity function $S_{i,j}(p)$ for a pair of conformal maps m_i, m_j (values range from 0 (red) to 1 (blue)). The generating correspondences are depicted by red lines for map m_i and by blue lines for map m_j.*

Since calculating $S_{i,j}$ is at the core of our objective function we need to make this computation as efficient as possible. However, for high-resolution meshes, calculating geodesic distances between any arbitrary pair of vertices can be expensive. Instead of calculating Equation (3.4) directly, we replace it with:

$$S_{i,j}(p) = \exp\left(-\frac{d_{\mathcal{M}_1}\left(p, m_j^{-1}(m_i(p))\right)}{\sigma^2}\right) \tag{3.5}$$

The reason is that we can then evaluate the consistency function $S_{i,j}$ only for a subset of points $\mathcal{P}_{\text{even}}$ and therefore geodesic distances from point p to every other vertex on $\mathcal{M}_1$ can be precomputed and stored. For example, Figure 3.4 shows similarity values for two pairs of maps.

Optimizing for map consistency weights: In this step, we optimize Equation (3.3) for the consistency weights: $\{w_i\}$. To do this, we define a blending matrix

$$\tilde{\mathbf{S}}_{i,j} = \int_{\mathcal{M}_1} c_i(p)c_j(p)S_{i,j}(p)dA(p).$$

with which Equation (3.3) can be written as

$$E_{\mathcal{M}_1}(\vec{w}) = \vec{w}^T\tilde{\mathbf{S}}\vec{w}\ ,\ \|\vec{w}\|_2 = 1. \tag{3.6}$$

Since $\tilde{\mathbf{S}}$ is symmetric, the top eigenvector is the optimal maximal solution (maximizing the Rayleigh quotient $E_{\mathcal{M}_1}$). The Perron-Frobenius theorem assures us that all the entries of this optimal $\vec{w}$ are of constant sign and therefore can be chosen to be positive. Thus, a simple optimization to achieve the consistency weights is to find the top eigenvector $\vec{w}$ of §, define the blending weights $b_i(p) = c_i(p)w_i$, and construct the blended map f as described in Equation (3.1).

However, there are two issues: 1) the matrix § for many feature points is large (remember we have $\binom{|\mathcal{P}_1|}{3} \cdot \binom{|\mathcal{P}_2|}{3} \cdot 6$ distinct conformal maps), and 2) in a presence of intrinsic symmetries (or near-intrinsic-symmetries) there is more than one "correct" (i.e., near-isometry) map between the surfaces. We address these issues as follows.

Computing the Blending Matrix: Filling in the matrix is the most computationally involved step of our approach. For example, given N feature points on both surfaces, one can construct $\binom{N}{3} \cdot \binom{N}{3} \cdot 6 = O(N^6)$ distinct conformal maps, thus filling the matrix $\tilde{\mathbf{S}}$ requires $O(N^{12}) \cdot |\mathcal{P}_{\text{even}}|$ operations. Fortunately, this matrix is extremely sparse (see Figure

4.2), and the sparsity can be exploited with a few simple observations. Practically, highly consistent values between conformal maps are possible mainly when they share consistent subsets of generating correspondences. Thus, we only calculate $S_{i,j}$ for pairs of maps that share two (out of a possible three) generating correspondences and do not have any "conflicting correspondences" (i.e., when a feature point on one surface is in correspondence with two different feature points on the other surface), and set $S_{i,j}$ for others to zero. Furthermore, we restrict the maximal number of conformal maps to $10,000$. If more maps generated we randomly remove maps until we are left with $10,000$.

To further speed the computation, we approximate $S_{i,j}$ using a uniform point sampling. Specifically, for each pair of maps, we compute $\tilde{\mathbf{S}}_{i,j}$ by summing over a discrete set of 256 evenly distributed points ($\mathcal{P}_{\text{even}}$):

$$\tilde{\mathbf{S}}_{i,j} \approx \sum_{p \in \mathcal{P}_{\text{even}}} c_i(p) c_j(p) S_{i,j}(p) A_i,$$

where A_i are the constant units of area $\frac{area(\mathcal{M}_1)}{256} = \frac{1}{256}$.

Processing Eigenvectors: If either of the two surfaces has an intrinsic near-symmetry, there may be more than one near-isometry between the surfaces. Hence, there will be more than one group of Möbius transformations such that the corresponding consistency weights $\{w_i\}$ yield a high energy value in Equation (3.3). In case one of the groups of Möbius transformations is (even slightly) better than the rest (in the sense that its consistency vector produce higher energy), the eigenspaces of $\tilde{\mathbf{S}}$ will naturally separate this better map from the other candidate maps. For example, Figure 4.2 shows two humans where there are two possible near isometric solutions corresponding to the two top eigenvectors. In this case the better map was characterized by higher energy level (eigenvalue).

Nevertheless, sometimes the different near isometries have very close energy level and the corresponding eigenvectors are blended. In this case any top eigenvector can contain

a linear combination of good weight vectors $\vec{w}$ that are originated from different near-isometries of the two surfaces. To avoid blending inconsistent maps we follow the next steps to extract sets of candidate consistent weight vectors $\vec{w}^1, \vec{w}^2, ..., \vec{w}^n$ and analyze them to select the best one.

First we recognize the top eigenvectors by taking all eigenvectors with eigenvalues separated by the spectral gap to the rest of the spectrum of $\tilde{\mathbf{S}}$, see Figure 4.2. Practically, we take eigenvectors $\vec{w}$ that correspond to eigenvalues within 75% of the top eigenvalue.

Second, we construct the weights $\vec{w}^1, \vec{w}^2, ..., \vec{w}^n$ by separating the different conformal maps with high values in these eigenvectors to different groups $G^1, G^2, ..., G^n$ (clusters) as follows. We start by seeding the first group G^1 to contain the conformal map that corresponds to the top entry (measured in absolute value of the top eigenvector). Then, we traverse the rest of the conformal maps corresponding to high entries (top 25% of that eigenvector). For each conformal map, we check whether its generating correspondences are consistent with the maps chosen already in G^1 (i.e., has no conflicting correspondences). If so, it is not conflicting, and we add it to G^1. Otherwise, we start a new group G^2 seeded with this map. We continue in this fashion until all the eigenvectors belonging to top eigenvalues are processed. We then threshold the weights to $\{0, 1\}$ to enforce the expected block structure of the group in the matrix and to eliminate maps with nearly zero weight from further processing, yielding a set of groups $G_1, .., G_n$ with corresponding binary weights $\vec{w}^1, .., \vec{w}^n$.

The last step is to choose among the different candidate weights $\vec{w}^1, ..., \vec{w}^n$ the best one. The above procedure generates weights corresponding to clusters of consistent maps, all of which provide an approximately optimal solution to the objective function defined in Equation (3.3). To select the best among them, we construct from each candidate vector the final blended map $\vec{w}^j \rightarrow f^j$, and pick the blended map that is most confident overall –

i.e., globally preserves area best over the whole surface:

$$f = \underset{\{f_j\}_{j=1}^{n}}{\operatorname{argmin}} \int_{\mathcal{M}_1} c_{f_j}(p) dA(p).$$

For example, the mapping between two human body surfaces would usually generate two weight vectors $\vec{w}^1, \vec{w}^2$: one corresponds to the correct map, and one to an intrinsic rotation by 180° under which the front of a human goes to the back, left side maps to right side, etc... Note that both these assignments are globally consistent, and each of the maps have similar confidence to their symmetric counterparts. However, the blended map corresponding to the intrinsic rotation (flip) usually introduces more area distortion for some parts of a body like feet or knees, see Figure 4.2. This allows us to distinguish between good and flipped solution.

3.3.5 The Blended Map

Now we can use low-dimensional maps m_i defined in Section 3.3.2 with the importance weights w_i obtained in Section 3.3.4 to construct the blending map defined in Equation (3.1). Note that as long as our confidence $c_i(p)$ changes smoothly over the surface the resulting blending map will also be smooth.

Figure 3.5 shows how $f(p)$ is found at a point p by blending $m_i(p)$ with the calculated weights. For efficiency sake, we find it useful to approximate the geodesic centroid by projecting the weighted Euclidean centroid $\tilde{f}(p) = \sum_i b_i(p) m_i(p) \Big/ \sum_i b_i(p)$ onto the closest point on the surface $\mathcal{M}_2$, an alternative that trades efficiency for accuracy. This is a favorable trade-off because weights used for blending tend to be non-negligible only for a small number of points concentrated very close to one another, in practice, in which case centroids based on Euclidean distances provide a good approximation.

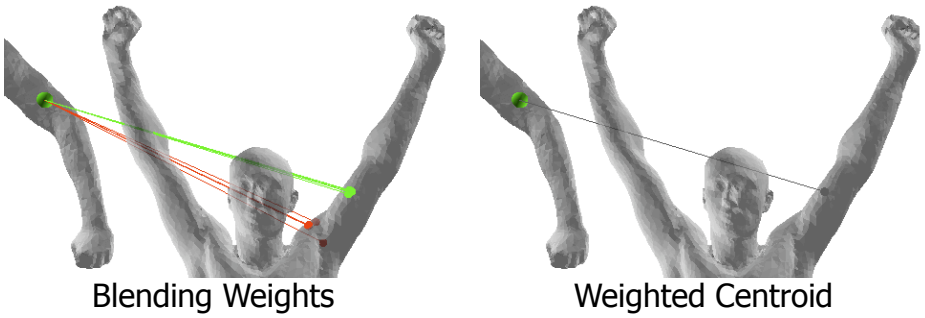

Figure 3.5: *Correspondences due to conformal maps with non-zero blending weight are depicted on the left image, where the color is set according to confidence* $c_i(p)$ *ranging from red* $= 0$ *to green* $= 1$*. The resulting approximate geodesic centroid is on the right image.*

3.4 Results

We test our approach on a benchmark constructed from TOSCA, SCAPE and Watertight data sets. We quantitatively analyze performance of our algorithm in various settings and compare results to several state of art methods for finding inter-surface maps and correspondences.

3.4.1 Data Sets

We selected three data sets that have a large variety of objects with ground-truth correspondences:

SCAPE: 71 meshes representing a human body in different poses [5]. All the meshes were fit to scanner data with a common template, and thus they share the same mesh topology, providing a ground truth map for every vertex for any pair of surfaces (corresponding colors in Figure 3.6a).

TOSCA: 80 meshes representing people and animals in a variety of poses [16]. The meshes appear in 8 groups with common topology, providing a per vertex ground truth map for any pair within a class (corresponding colors in Figure 3.6b).

Watertight: 400 meshes arranged evenly in 20 object categories, many of which are articulated figures (humans, octopus, four-legged animals, ants, etc.). The meshes were originally created for the SHREC 2007 Watertight Shape Retrieval Contest [43]. We selected 11 classes for our experiments that have well defined correspondences and genus zero (Human, Glasses, Airplane, Ant, Teddy, Hand, Plier, Fish, Bird, Armadillo, Four-legged Animal). In addition, we excluded two human models with non-zero genus.

In cases where no ground truth map was provided with a data set (e.g., Watertight), we established a sparse set of "ground truth" correspondences manually. Specifically, we recruited a volunteer to use an interactive program to select 10-35 semantically meaningful feature points in a manner that is consistent across all meshes within the same object class. For example, our volunteer selected 35 feature points for each human and 20 feature points for each four-legged animal (Figure 3.6c). These feature points form the basis for establishing symmetric correspondences and for evaluating maps between surfaces in the same object class.

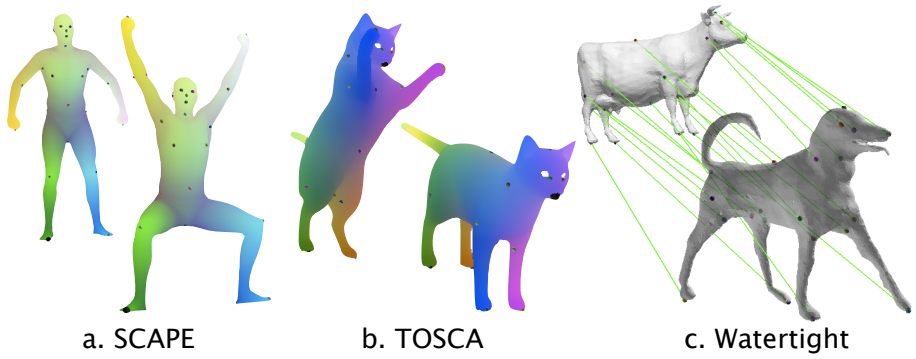

Figure 3.6: *Ground truth examples. SCAPE and TOSCA models are colored according to ground truth per-vertex correspondences.*

3.4.2 Evaluation Methods

To evaluate the accuracy of a predicted map, $f : \mathcal{M}_1 \rightarrow \mathcal{M}_2$ with respect to a "ground truth" map, $f_{\text{true}} : \mathcal{M}_1 \rightarrow \mathcal{M}_2$, we compute for every point, p, on $\mathcal{M}_1$ in the ground

truth correspondence the geodesic distance, $d_{\mathcal{M}_2}(f(p), f_{\text{true}}(p))$, between its image in the predicted map, $f(p)$, and its true correspondence, $f_{\text{true}}(p)$.

We aggregate these geodesic distances into an error measure:

$$Err(f, f_{\text{true}}) = \sum_{p \in \mathcal{M}_1} d_{\mathcal{M}_2}(f(p), f_{\text{true}}(p))$$

where $d_{\mathcal{M}_2}(f(p), f_{\text{true}}(p))$ is normalized by $\sqrt{Area(\mathcal{M}_2)}$, as all distances are throughout this paper.

We also generate plots to examine the distributions of errors, where the x-axis represents a varying geodesic distance threshold, D, and the y-axis shows the average percentage of points for which $d_{\mathcal{M}_2}(f(p), f_{\text{true}}(p)) < D$ (Figure 3.7 provides a scale bar for D).

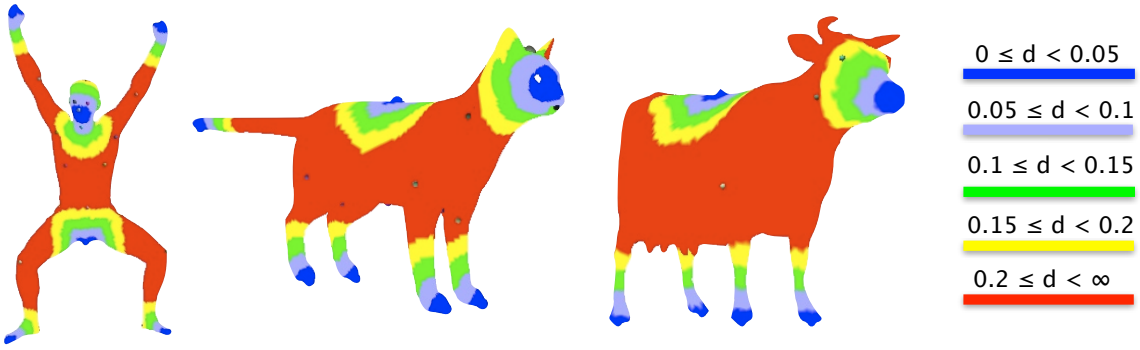

Figure 3.7: *Reference for normalized geodesic distances on surfaces (measured to the nearest seed point). Colors are labeled by distances as shown in the legend on the right hand side.*

To separate errors due to poor alignments from ones due to symmetric flips, we produce two such plots. The first is as already described. The second is similar, but factors out the effects of confusion in the predicted map due to global intrinsic reflective symmetries (e.g., bilateral symmetries that map the left side of a human to the right side). It plots the fraction of predicted correspondence points that either are closer than the geodesic distance threshold D to the correct correspondence point *OR* are closer than the threshold to the symmetric image of the correct correspondence point – i.e., it provides no penalty

for predicting a map that is a symmetric flip with respect to the correct one. These plots favor methods that do not preserve orientations in predicted maps (other methods, not ours, as conformal maps are orientation-preserving).

3.4.3 Comparison to Other Methods

We compare our work to several state of the art methods for finding inter-surface correspondences [1]:

- **Blended Map** - the method proposed in this paper
- **Best Conformal** - the least-distortive conformal map roughly describes what is the best performance achieved by a single conformal map without blending.
- **Möbius Voting*** - the method proposed by [80] explores millions of conformal maps generated by random triplets on a surface, and votes for correspondences generated by area-preserving maps. The output of this method is 50-100 coarse correspondences.
- **Heat Kernel Matching (HKM) with 1 correspondence** - This method is based on matching features in a space of a heat kernel for a given source point as described in [99]. A full map is constructed from a single correspondence, which is obtained by searching a correspondence that gives the most similar heat kernel maps. We use code provided by authors for this experiment.
- **HKM with 2 correspondences** - in a non-isometric case the previous method might obtain better results by using a second correspondence. The matching is then performed in the augmented feature space of two heat kernel maps. With minimal changes to the original author's code we follow a procedure outlined in [99] ex-

[1] comparison to Functional Maps on near-isometric benchmarks can be found in [97] and [104], and comparison to a method that leverages bilateral reflective symmetry of surfaces can be found in [83]

haustively searching for the second correspondence that minimizes the geodesic distortion.

- **GMDS*** - we use the method of [18] for surface matching. Authors hierarchically find correspondences between points by searching for assignment that best preserve geodesic features. We use authors implementation of this method with default parameter settings to find 50 coarse correspondences.

- **Deformation Driven*** - the method of [132] is based on searching for correspondences while minimizing the induced deformation error. It usually produces 5-10 correspondences for a pair of meshes. Unfortunately, it was not possible to execute this method for a large set of examples so we asked authors to run it on a small set of representative examples from our benchmark.

Note that some of these methods (marked with a "*" in the list above) only produce a sparse set of point correspondences, rather than a full surface map as required for comparison with our evaluation metrics. In those cases, we produce a full map from surface $\mathcal{M}_1$ to $\mathcal{M}_2$ by interpolating the sparse correspondence to using a method based on GMDS – we compute for each vertex on $\mathcal{M}_1$ the geodesic distances to all sparse correspondence points on $\mathcal{M}_1$, and then establish a correspondence to the vertex on $\mathcal{M}_2$ with the most similar distances to sparse correspondence points on $\mathcal{M}_2$ [16]. This method was chosen because it is simple to implement and because the accuracy is sufficient when a large number of sparse correspondences is provided, as is the case for all methods considered in this study.

Note also that the code available for some of these other methods crashed on meshes in our test data sets (8 in all). To keep comparisons fair, we eliminated those meshes from our evaluation in all experiments.

Near isometric pairs in TOSCA: In our first experiment, we studied how methods perform for nearly isometric pairs of surfaces from the TOSCA data set. Specifically, for each model, we picked a random model within the same class, and then computed a map from

one to the other use each algorithm. Results of this experiment are shown in Figure 3.8, where each curve depicts the percentage of correspondences with error below some normalized geodesic distance for a different method. Please note that our method (blue curve) detects over 75% correct correspondences for a small threshold of 0.05 and converges to finding almost all correct correspondences within geodesic error 0.2. We visually examine maps produced with our method and observe several small misalignments in some faces and limbs of animals mostly due to badly selected feature points.

In the bottom image of Figure 3.8, we show the fraction of correspondences within the distance threshold, if we also allow maps that invert surface orientation. This plot reveals that methods based on geodesic distances: GMDS and HKM (green, magenta and black curves) are commonly confused by bilateral reflective symmetry and intrinsic 180° rotation present in humans and animals. Note that although methods based on conformal geometry cannot produce a reflected solution, they still can intrinsically rotate a surface by 180° mapping front of a human to the back. In this case, errors also become smaller, since distances between front and back of a limb are smaller than distances between left and right limbs. Still the ranking of methods is largely the same.

Near isometric pairs in SCAPE: The next experiment compared maps found between nearly isometric models in the SCAPE data set. For each SCAPE model, we picked another one at random, totaling to 71 pairs, and computed the mapping. Results are shown in Figure 3.9. These meshes are less smooth than TOSCA meshes, which explains the decrease in performance for algorithms based on conformal geometry, since mid-edge uniformization suffers from non-delaunay triangles, but does not affect GMDS (green) and heat kernel methods (magenta and black). The main source of error in this data set is confusion due to intrinsic symmetry in humans. Note that due to the aforementioned problem with individual conformal maps the resulting blended map in some cases does not have enough accuracy to distinguish between front and back of a human. Also note that al-

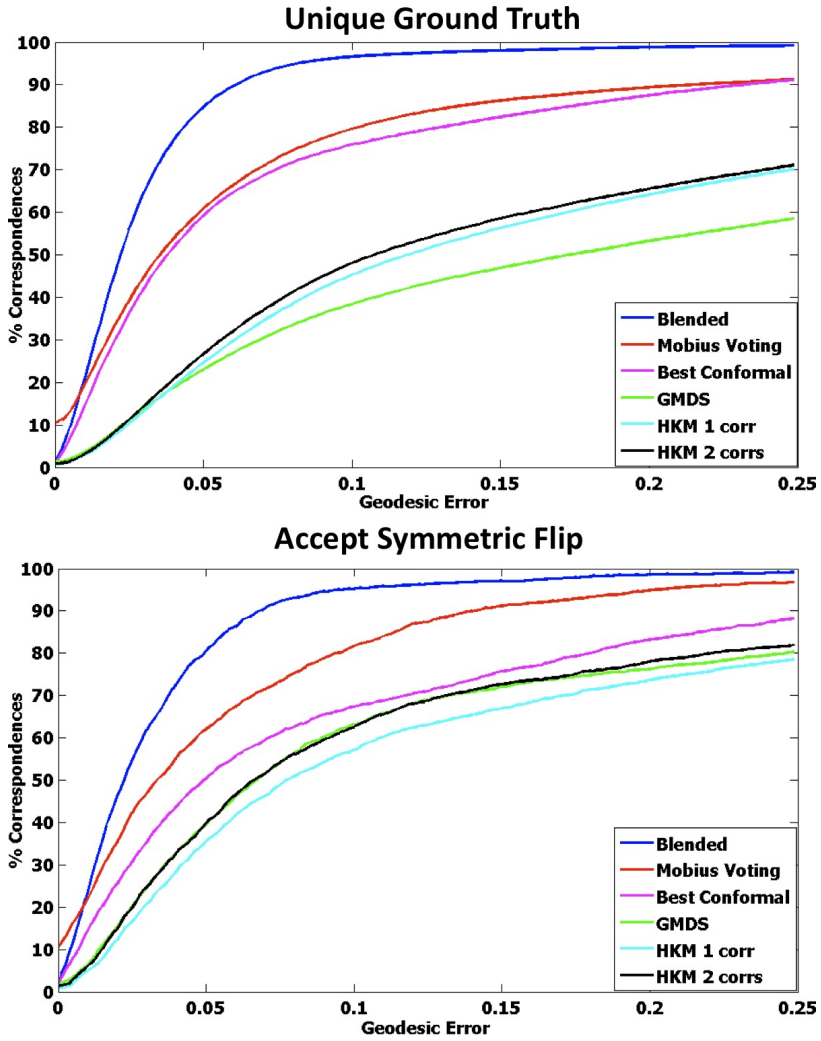

Figure 3.8: ***TOSCA:*** *performance of various methods on nearly-isometric human and animal models. We depict a geodesic distance on the x-axis, and a percentage of correspondences within the prescribed distance of the ground truth on y-axis.*

though Möbius Voting (red) always provides better coverage than the best conformal map it suffers from picking inconsistent correspondences, which explains similar performance of these two methods in this experiment.

Non-isometric humans: In a realistic settings, it is desirable to obtain correspondences between surfaces with different resolutions, tessellations, and even semantic variations. In this experiment, we used all 71 human models from SCAPE data set, 43 from TOSCA (including gorilla), and 18 from SHREC (excluding two humans with non-zero genus), totaling 132 meshes. We used each method to find a map from each model to another selected at

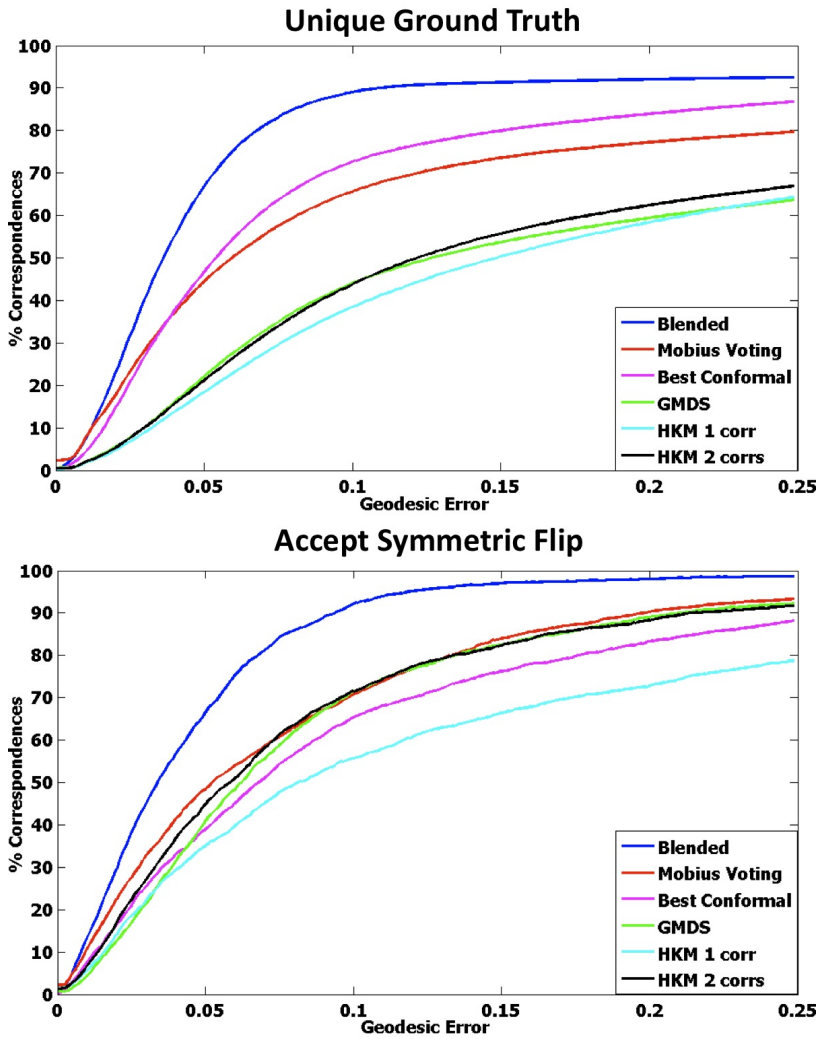

Figure 3.9: ***SCAPE****: humans. Most of our errors on this data set are due to confusion by* 180° *rotations that map the front of a body to the back.*

random from this set (excluding near-isometric examples from previous experiments) and plotted results in Figure 3.10. As in the experiment with SCAPE models, results mainly suffer from symmetric flips. Note that while our method performed just as well as in the isometric case, Möbius Voting is suffering from inconsistent correspondences, because for large deformations locally accumulated votes have more noise. Observe also that if one is allowed to invert surface orientation, HKM with 2 correspondences and GMDS (black and green curves) outperform Möbius Voting because they produce more consistent results.

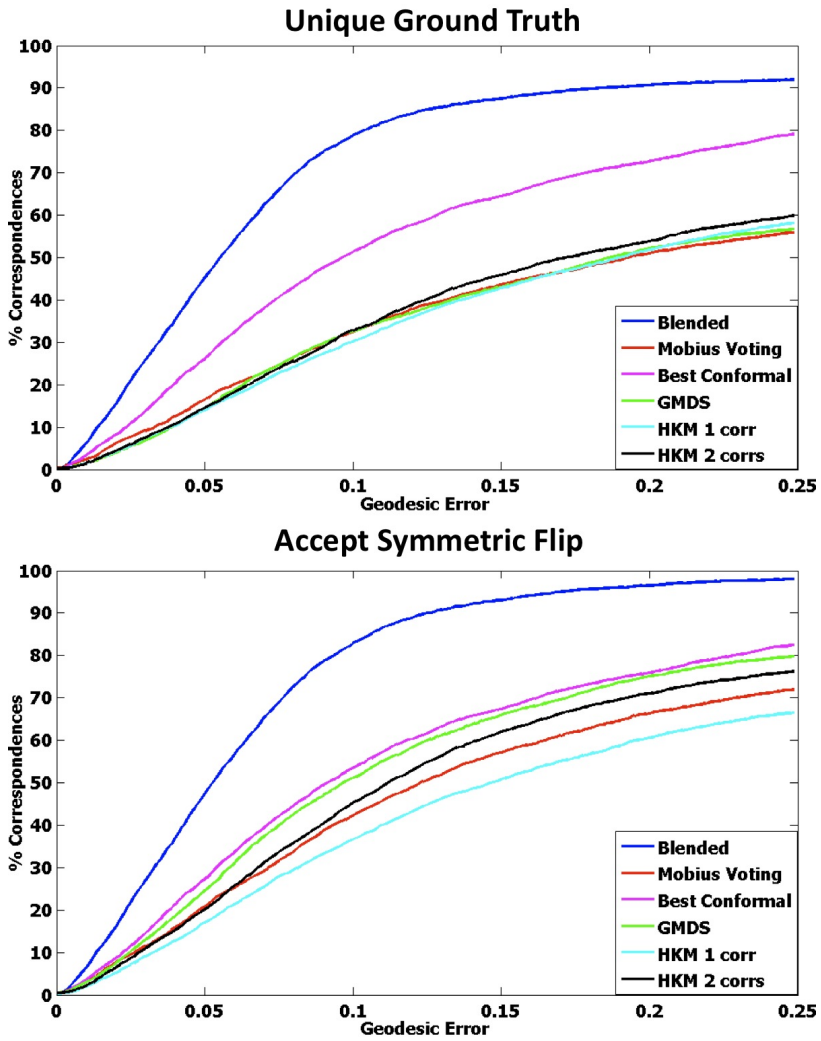

Figure 3.10: ***Humans:*** *performance of various methods on non-isometric pairs of humans. Our method outperforms other approaches because it extracts two consistent solutions (due to intrinsic symmetry) and then uses local cues (integrated confidence) to choose the best map.*

Non-isometric animals: In another comparison, we found a map from every animal model in TOSCA and Watertight data sets to a random animal model (excluding near-isometric pairs). Thus, we used 31 TOSCA models and 20 Watertight models to produce maps between 51 pairs using each inter-surface mapping method. Note in Figure 3.11, that Möbius Voting performed significantly better than for non-isometric humans since there is no intrinsic near-symmetry in animals (the left-to-right symmetry requires inverting orientation of the surface, which is impossible with conformal maps). However, our method still out-

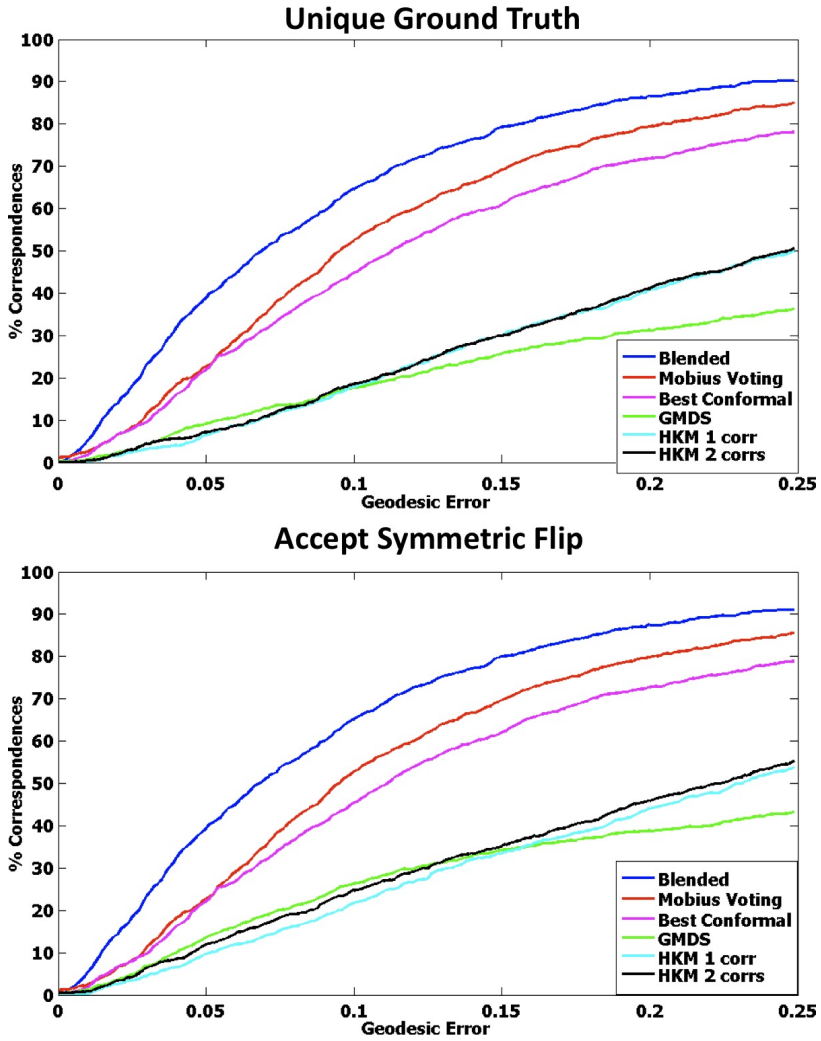

Figure 3.11: ***Animals:*** *performance of various methods on non-isometric pairs from TOSCA and SHREC Watertight'07 data sets.*

performs Möbius Voting on average, while providing a consistent map (avoiding outlier correspondences, which can appear with Möbius Voting). Since limbs and outliers usually cover only a small part of the surface area, these errors do not contribute largely to the fraction of correspondences. However the difference becomes more obvious if we look at maximal per mesh errors, whose average is presented in Table 3.1. Our method performs more uniformly across different experiments and produces smaller maximal per map errors.

Small Set: We also selected a smaller set of 20 representative examples for those authors who could not run their method on the whole benchmark. We selected five examples from

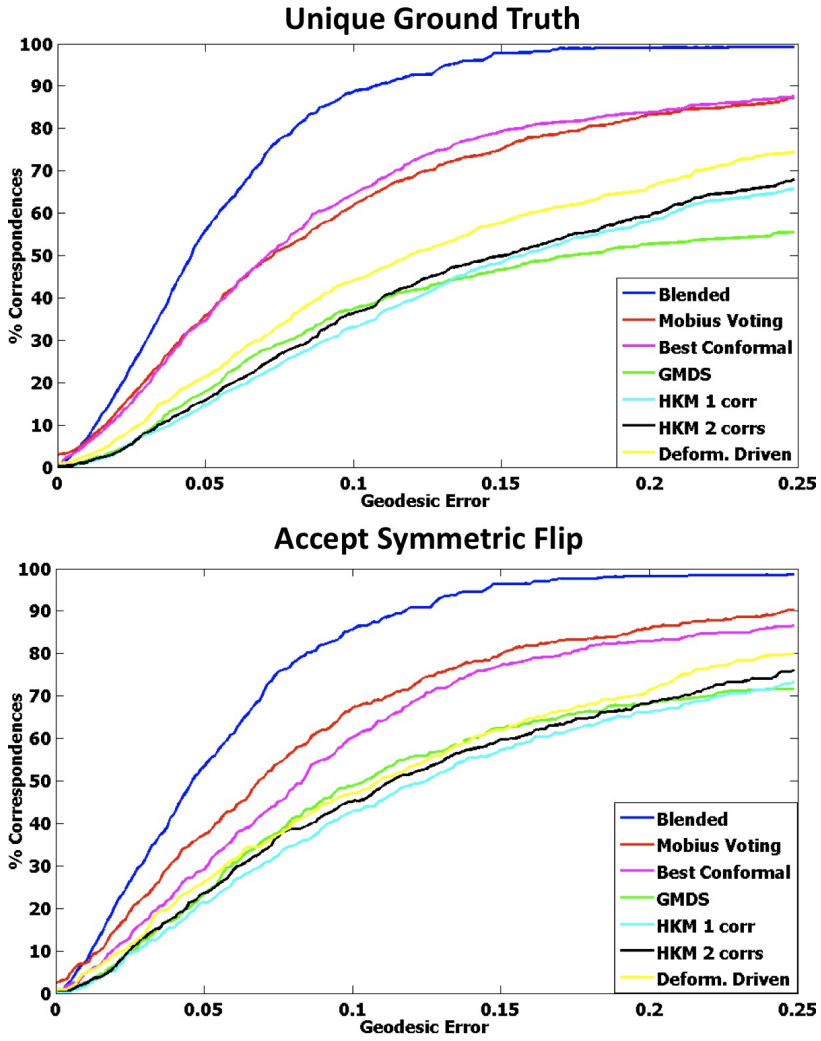

Figure 3.12: ***Small Set:*** *performance of various algorithms on a small subset augmented from isometric and non-isometric experiments.*

each experiment (SCAPE, TOSCA, Animals, Humans) and compared all methods on those examples as illustrated in Figure 3.12. Note that this is the only experiment that includes Deformation-Driven Correspondence Search method of [132]. This method explores a large space of permutations of correspondences, and thus find sub-optimal solution in many cases. Note that it also extracts only a few correspondences (5-10), thus performance of the method partially suffers from the interpolation technique that we use, and using a more sophisticated method for interpolation could potentially improve results.

	TOSCA	SCAPE	Humans	Animals
Blended	0.31	0.25	0.32	0.33
Möbius Voting	0.45	0.64	0.97	0.57
Best Conformal	0.54	0.60	0.69	0.56
GMDS	0.97	0.86	1.11	1.02
HKM 1 corr	1.27	1.29	1.39	1.08
HKM 2 corrs	1.21	1.07	1.33	1.06

Table 3.1: *Table of averaged maximal per map geodesic errors. Note that our method has smaller average maximal error compared to all other methods. Even in the experiment with non-isometric animals, where Möbius Voting gives a comparable performance in terms of fraction of accepted correspondences, we give substantially smaller maximal per mesh error: first, because we better map limbs, and second, because we generate consistent map that does not have outlier correspondences.*

Visualization: We also present visualizations of maps produced by each method for two examples in Figure 3.13. In the top row we see a map between two humans. A common problem a single conformal map is that it collapses one of extremities to a small area, as seen in areas (A) and (E). Image area (B) reveals an inconsistent correspondence assigned by Möbius Voting on a shoulder. Areas (C) and (C') show that adding a second correspondence improves results for some examples, in this case a left hand was correctly mapped with two heat kernel maps. Note however, that maps produced by searching for nearest neighbors in some space usually do not produce continuous maps (HKM and GMDS (D) methods). For stronger deformations, for example mapping a cat to a dog, it is challenging to even find a general structure of a map. For example, in cases labeled (F), (G), (H) coarse correspondences map at least some limbs (or a whole body) incorrectly. Note that for such strong deformations a single heat kernel map is not sufficient to map the whole body, augmenting a second correspondence improves the result for tail, but other limbs are still mapped incorrectly (G), (G').

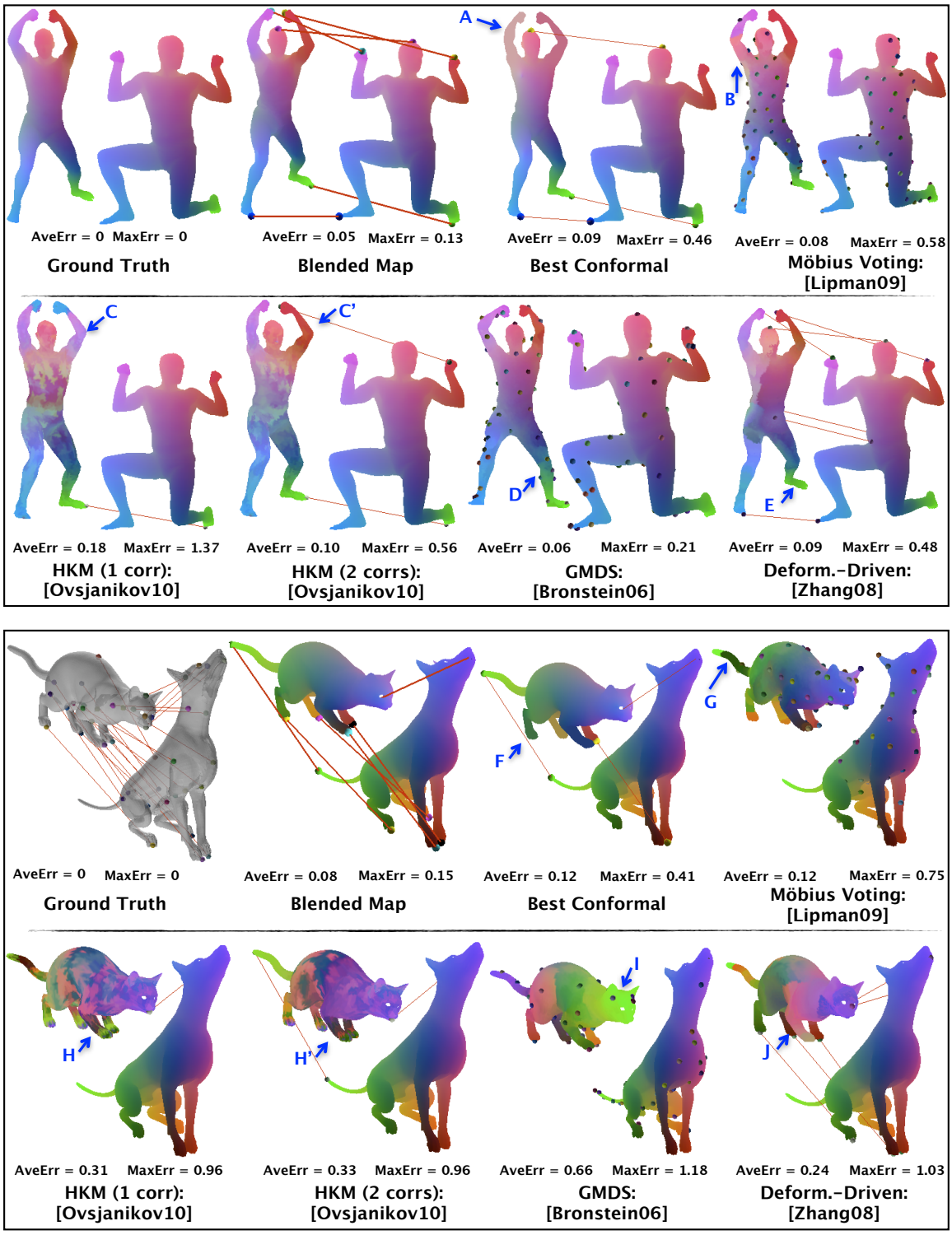

Figure 3.13: *This figure shows a comparison of our method (Blended Map) to others.*

3.4.4 Per-class performance

Finally, we investigated performance of our algorithm for a variety of object classes in the SHREC Watertight 2007 data set, where models vary in semantic content, resolution, and tessellation. As in other experiments, we mapped every model in each considered class to a random model within the same class, yielding 218 blended maps. In Figure 3.14, we observe that our method gives the best performance for articulated figures, but it also successfully handles many cases from other classes (such as planes, fish or birds). Some examples, can be found in Figure 3.15. The first three rows of this plot include success cases for our method, and the bottom row depicts four typical failure cases. Note that our method successfully handles highly non-isometric pairs. Observe that our method does not rely on local geometry cues, and thus succeeds in a map even if local geometry of surfaces is very different, for example, there is no local resemblance of a shape of giraffe and a shape of a cat, or shape of a dog and a cow. The fact that conformal maps do not preserve geodesic distances can also serve as an advantage in case of partial scaling. For example, in the rightmost armadillo example one of the models is missing part of a hand, but associating a full hand with a stump does not propagate any errors to the rest of the shape. You can observe the most common failures of our methods in the bottom row. We believe the most common problem is misalignment in feature points. For example, fish and dolphin in the first image have different orientation of a tail, thus conformal maps generated by feature points on a tail twist map on the body. Another common issue is symmetric flip as depicted for ant and airplane examples. Note that the the only hint for our method on deciding which is the correct solution is a small asymmetry in location of ant's limbs and orientation of plane's tail. This in many cases is too subtle to be captured by our integrated confidence value. Note also that for creatures with more than 7 feature points we prune the blending matrix to bound processing time. This introduces an undesired bias, which can (as in the example with the ant) result in incorrect eigenvectors; for example, two limbs of the ant are twisted. Finally, individual conformal maps can also be low quality if objects

have long, thin parts. Specifically, map for glasses introduces a lot of distortion of tips of a frame which is not distributed uniformly.

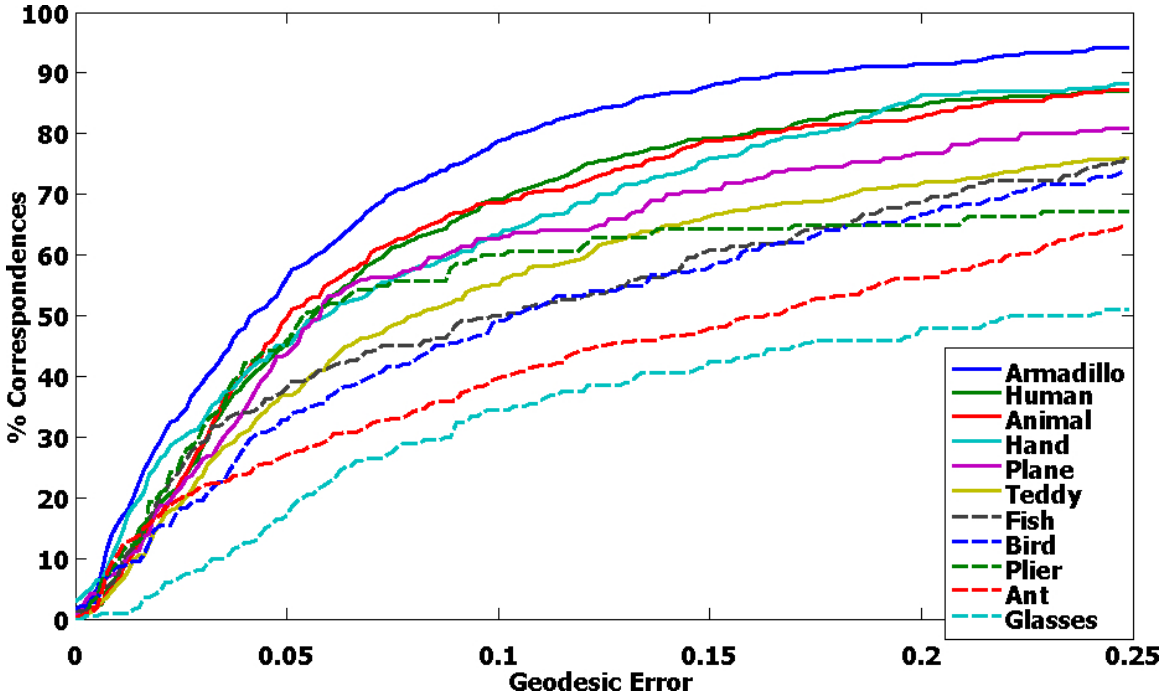

Figure 3.14: ***Per class*** *performance of our method on models from SHREC Watertight'07 data sets. Our method currently performs best for articulated figures with small numbers of feature points.*

3.4.5 Timing

We assume that our method is used in a non-interactive manner for a large collection of surfaces that require pairwise maps. Thus, we want our method to find the best maps as quickly as possible with no user intervention. The running time of our method depends on two factors: the number of vertices in the mesh $|\mathcal{M}|$ and the number of extracted feature points $|\mathcal{P}|$. In a pre-processing stage, we find feature points and precompute geodesic distances for them which requires about 30s for a Scape model with 12500 vertices, 60s for a TOSCA model with 27894 vertices (cat), and 170s for a TOSCA model with 52565 vertices (David) on 2.2GHz Opteron 275 processor. The remaining part of the algorithm mainly depends on the number of selected feature points $|\mathcal{P}|$. For a pair of SCAPE models

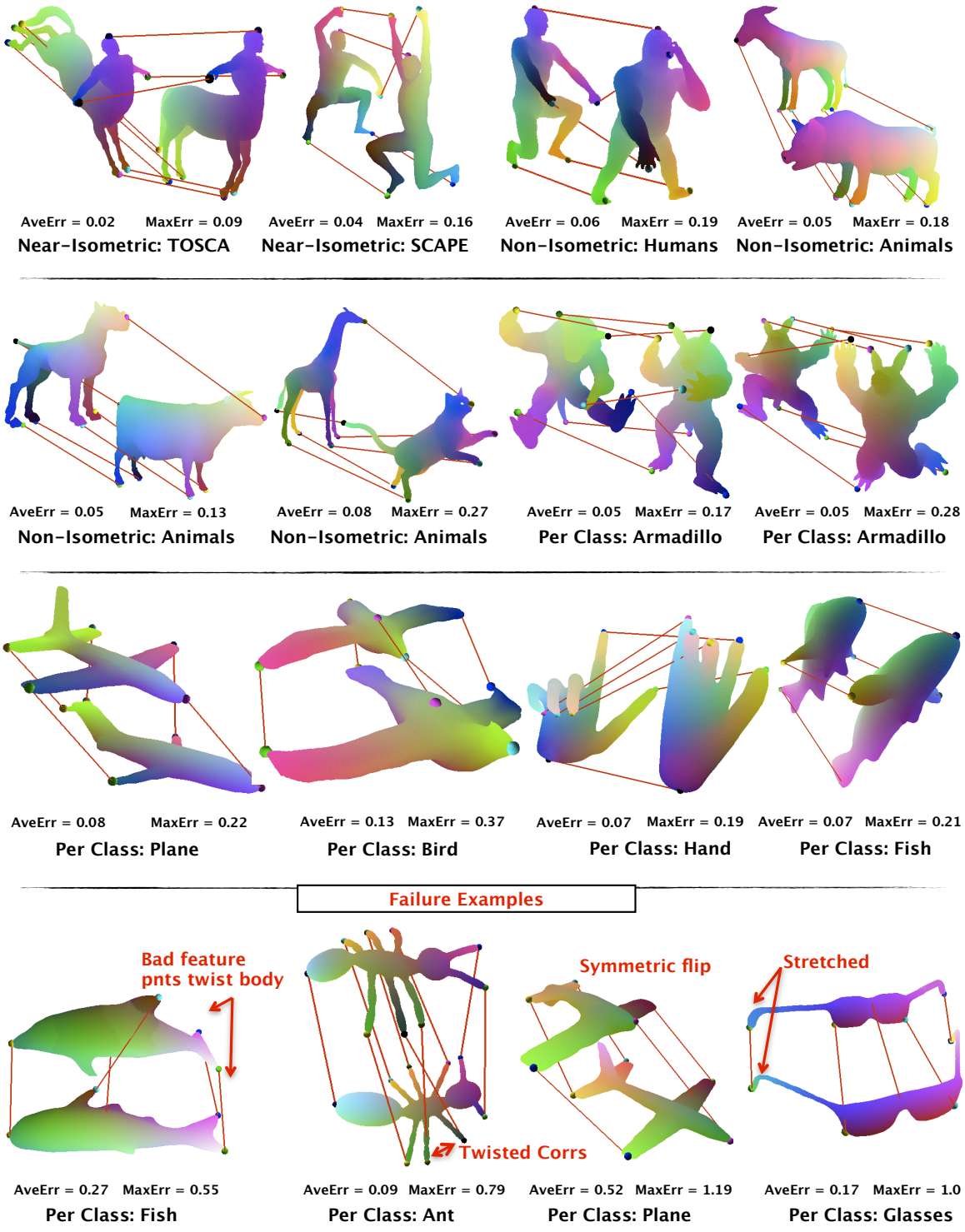

Figure 3.15: *Performance of our method on various classes.*

usually $|\mathcal{P}_{\mathcal{M}_1}| = |\mathcal{P}_{\mathcal{M}_2}| = 5$, and our algorithm takes 50s, of which 24s spent on calculating per map confidences, 22s spent on filling the weight matrix, and 4s is spent on processing

eigenvectors and finding correspondences for every vertex on a surface. For a pair of cats from TOSCA data set with $|\mathcal{P}_{\mathcal{M}_1}| = |\mathcal{P}_{\mathcal{M}_2}| = 6$, confidence calculations take 86s, and the matrix is filled in 212s. In the hardest cases (e.g., ants, centaur), we randomly remove maps until we are left with 10,000, thus in the slowest case confidences are calculated in 364s, and the matrix is filled in 1411s. A more intelligent pruning of the blending matrix is required to obtain better results for more complex shapes with larger number of relevant feature points.

Chapter 4

Fuzzy Correspondences for analysis of collections of 3D shapes

4.1 Introduction

Computing per-point correspondences across all shapes in a collection is a fundamental tool that enables reasoning about functional and structural similarities between regions on different shapes. Finding these relationships provides the opportunity to study geometric variations [2] and transfer properties [105] between shapes from the same class. There are two big challenges faced by traditional pairwise mapping methods. First, aligning each pair independently is inefficient and does not leverage transitivity of correspondences in the whole dataset. Second, if a collection includes diverse shapes, the task of establishing point-to-point map becomes ambiguous. For example, how does the chair leg on the left of Figure 4.1 correspond to the chair base of the model on the right?

We propose an algorithm that produces correspondences for the whole collection using just a subset of pairwise maps. We further address the issue of ambiguity with *fuzzy correspon-*

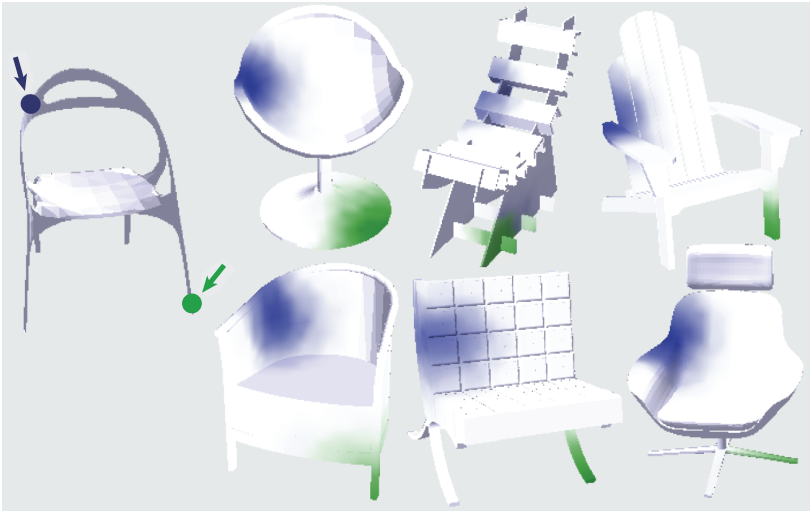

Figure 4.1: ***Fuzzy correspondence values for two points.*** *The blue and green regions of the chairs on the right have the largest fuzzy correspondence to the two selected points on the chair on the left.*

dences[1]. Specifically, given a collection of N shapes $S := \{S_1, S_2, \ldots, S_N\}$, we use fuzzy correspondences as a function $f(p_i, p_j) : S \times S \rightarrow \mathbb{R}$ to denote a continuous similarity measure between points $p_i \in S_m$ and $p_j \in S_n$.

To estimate fuzzy correspondences, $f(p_i, p_j)$, we utilize geometric matching methods that align pairs of shapes. Although these methods are computationally expensive and often produce noisy alignments, we observe that for collections of shapes from the same class, a correspondence matrix that stores high values for corresponding pairs of points is (i) sparse, (ii) low-rank, and (iii) its rank does not depend on the number of models. We propose a method based on diffusion maps [92] to reconstruct f from sparse and noisy samples (i.e., pairwise alignments) and an iterative procedure to refine f by adaptively sampling according to the current estimate.

We test the accuracy of our estimate of f using the correspondence benchmark for intrinsically-similar shapes [71], and a more diverse correspondence benchmark using data

[1]We use the word "fuzzy" as previously used in non-rigid surface matching [27] rather than fuzzy set theory.

obtained from the 3D Warehouse [49]. Our method successfully utilizes the collection to improve alignments of shapes in comparison to previous methods.

Finally, we demonstrate that fuzzy correspondences enable an interactive exploration of the interesting structural relationships in a collection. More specifically, we present a new exploration interface for 3D model collections that allows users to directly specify regions of interest (ROI) on example shapes, and visualize geometric similarities and variations in these regions.

Contributions: In summary, we

- introduce an approach for using fuzzy correspondences to understand similarity relations across 3D model collections,
- propose a robust and efficient algorithm to compute fuzzy correspondences from sparse and noisy pairwise alignments,
- evaluate our algorithm on correspondence benchmarks and report substantial improvement over existing alternatives, and
- present an interactive tool for exploring structural relationships in a collection.

4.2 Related Work

Establishing similarities between 3D shapes in collections has received a large amount of attention in recent years.

4.2.1 Map optimization

Nguyen et al. [94] proposed an algorithm to improve point-to-point mappings given all pairwise maps. Their method is based on the assumption that all cycles of consistent maps must return to identity. They start with $O(N^2)$ pairwise maps between surfaces and then perform an optimization that iteratively improves the consistency of 3-cycles, thus leverag-

ing information from the whole model collection. The method, however, has four important limitations: (i) it requires $O(N^2)$ pairwise alignments, (ii) it computes point-to-point correspondences and thus is not applicable to heterogeneous quality models, (iii) it propagates information only across 3-cycles and thus converges slowly, and (iv) it only aligns pairs of models by concatenating full maps, which limits the applicability of this method to heterogeneous datasets where most pairs of models might not have a bijective map between them. We extend their optimization strategy to support fuzzy correspondences, which work with fewer pairwise alignments and more diverse datasets.

Following our work on fuzzy correspondences described in this thesis [68], Huang et al. [58] devised a method for optimizing point-to-point correspondences using a sparse graph of pairwise alignments. Their method assumes that all final optimized maps can be represented by composing initial maps connected in a hub-and-spoke network. Their algorithm jointly optimizes for a small set of base shapes (hubs), such that the optimized maps have consistent cycles (i.e. all cycles approximate the identity map), and map neighboring points to neighboring points. Similar to our approach, they produce the optimized correspondences by searching in a space of diffused initial pairwise maps.

4.2.2 Diffusion maps

Introduced by Nadler et al. [92], diffusion maps provide a probabilistic interpretation of spectral clustering and dimension reduction algorithms, and have previously been used for analyzing image collections [51], establishing symmetric correspondences [79], and clustering similar segments for consistent segmentation [115]. We use diffusion maps to compute fuzzy correspondences. Our approach has some similarities to that of Lipman et al. [79] and Sidi et al. [115], since we use the manifold induced by spectral embedding to estimate correspondences in shapes. In contrast to Lipman et al., our method works on

model collections. In contrast to Sidi et al., we focus on point correspondences (rather than parts).

4.3 Method

We cast the problem of computing fuzzy correspondences as a sampling problem, where the goal is to reconstruct the fuzzy correspondence function $f(p_i, p_j)$. We represent each shape by K discrete points, and thus a discrete representation of f for a database of N shapes is an $NK \times NK$ matrix. To sample entries in f, we use automatic pairwise matching techniques that predict point correspondences based on geometric alignments. While these matching algorithms are often effective at finding semantic correspondences between pairs of similar shapes, they have two primary limitations: (i) geometric shape matching is slow and (ii) matching based on geometry alone can result in semantically incorrect alignments for pairs that differ significantly in geometry or topology, differ by extreme non-homogeneous deformations, and/or have missing or extraneous parts with non-uniform proportions. Thus, the challenge is to reconstruct f with the fewest possible and most robust samples.

Our approach is based on diffusion. First, let us denote an approximate correspondence matrix $C \in \mathbb{R}^{NK \times NK}$ to store computed samples for matched pairs of points. Note that in an ideal case, if we assume that a point on a model corresponds to exactly one unique point on every other model, then the rank of C is independent of the number of models (and equals to the number of points, K). We use diffusion maps to compute a spectral embedding of C that maps each point on a shape to a Euclidean space whose coordinates are the eigenvectors of C scaled by the eigenvalues. We expect the embedded points to lie on a low-dimensional manifold where corresponding points are close to each other, and thus we estimate fuzzy correspondence based on distances in the embedded space (also called diffusion distances). This approach has two main advantages: (i) the embedding

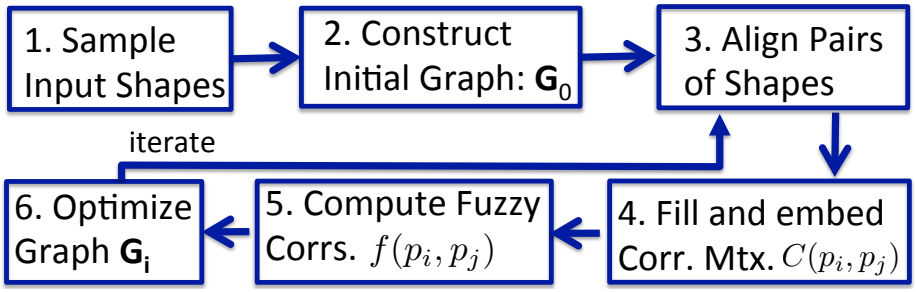

Figure 4.2: ***Computational Pipeline.*** *In our optimization procedure, we first construct an initial alignment graph* $\mathbf{G}_0$*, which is further used to fill the correspondence matrix* C *by aligning shapes connected by an edge. The spectral embedding of* C *defines the fuzzy correspondence function* f*, which is used to optimize the alignment graph. We iterate until the process converges.*

(fuzzy correspondences) can be computed without aligning all point pairs, and (ii) it is robust with respect to noise, and thus it overcomes the problems of methods based solely on pairwise alignments.

We now describe our computational pipeline (see Figure 4.2).

4.3.1 Step 1: Sample input shapes

We represent any shape S_i by a discrete set of sample points P_i. The set is produced by starting with a random vertex and then iteratively adding vertices that are farthest from the set until $|P_i| = K = 128$ [29]. We only estimate the fuzzy correspondence function for pairs of points in P_i across different models, and then interpolate the function to the rest of the shape using nearest-neighbor interpolation.

4.3.2 Step 2: Construct an initial alignment graph

Next, we construct an initial graph $\mathbf{G}_0(\S, \mathbf{A}_0)$ that has the following two properties. First, we want edges in $\mathbf{A}_0$ to only connect shapes that are similar enough to be matched automatically. The importance of this property is illustrated in Figure 4.3 where airplane models

are the nodes of the graph and edges are produced by an affine alignment that minimizes surface distance. Note that the airplanes with wings closer to the nose are misaligned with airplanes that have wings closer to the tail. These misalignments add noise to the complete graph $\mathbf{G}_A$, which results in an erroneous blend of correspondences from one wing to the other (left column). In contrast, a noise-free alignment graph $\mathbf{G}_B$ where only similar models are matched leads to a more accurate embedding (right column). The second desirable property for $\mathbf{G}_0$ is that every pair of shapes should have multiple paths between them so that the embedding is robust to misalignments. Given these properties, we construct the initial alignment graph as follows. First, we initialize $\mathbf{G}_0$ as the complete graph over all shapes. We then compute the spherical harmonics shape descriptor of the GEDT function [67] for every shape and use the L_2 distance between descriptors as edge weights. These weights predict whether the connected shapes are similar enough to be matched automatically (smaller weights suggest higher similarity). Based on these weights, we update $\mathbf{G}_0$ to be the minimal spanning tree. Next, we add edges to improve the graph connectivity. For each node, we select the $3M$ lowest weight edges as candidates, and for each candidate, we compute its *edge rank*, which is a metric proposed by Heath et al. [51] that estimates the importance of an edge to the overall connectivity of a graph. We then add the highest ranking $N \cdot M$ edges (roughly M edges per shape) to $\mathbf{G}_0$. Choosing a larger value for M improves the connectivity of the alignment graph; we use $M = 5$ in all of our experiments.

4.3.3 Step 3a: Align pairs of shapes extrinsically

Our method can use any pairwise alignment method, and some algorithms may be more appropriate for certain types of collections. For diverse datasets with varying topology (e.g., typical data from Google 3D Warehouse), we find affine transformations to be an effective alignment method. Step 3b describes how to use an intrinsic mapping technique

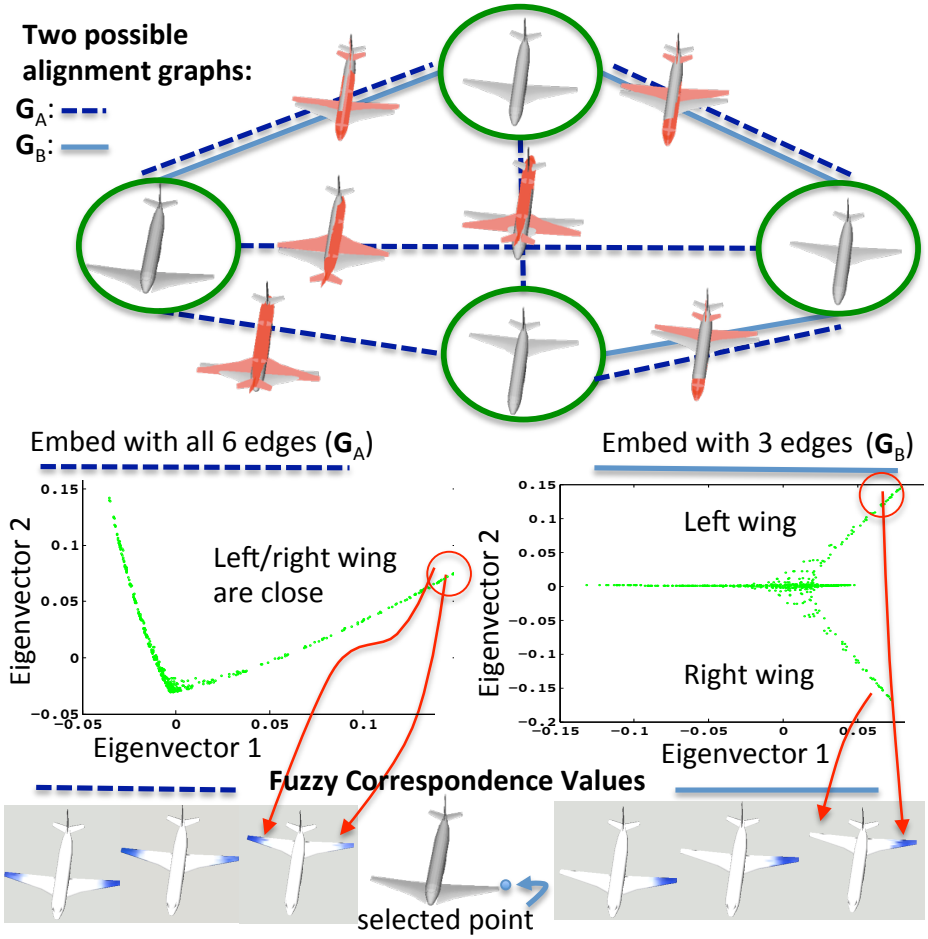

Figure 4.3: ***Example alignment graphs.*** *A denser alignment graph might result in more noise due to alignment of dissimilar models. A complete graph* $\boldsymbol{G}_A$ *has more misalignments that result in blending of wings in the embedded space (left), while the linear graph* $\boldsymbol{G}_B$ *resolves the issue (right).*

for other types of collections. Given two shapes S_k and S_l, an alignment score $a_{S_k,S_l,\phi}(p_i \in S_k, p_j \in S_l) : S_k \times S_l \rightarrow \mathbb{R}$ is defined over some parameter space ϕ (e.g., all affine transformations T). The score a is defined per pair of points and depends on the quality of the local alignment (L_a) and the global alignment (B_a):

$$a_{S_k,S_l,\phi}(p_i, p_j) := L_a(p_i, p_j) \cdot B_a(S_k, S_l). \tag{4.1}$$

We define the local term based on Euclidean distance after the transformation $\phi = T$, while the global term represents how well T aligns two shapes globally:

$$L_a(p_i, p_j) := \exp\left(-\frac{D_{\text{Eucl.}}(p_i, T(p_j))^2}{\sigma(S_l)^2}\right) \tag{4.2}$$

$$\begin{aligned} B_a(S_k, S_l) := &\left(\frac{1}{K}\sum_{p_1 \in P_k} \exp\left(-\frac{D_{\text{Eucl.}}(p_1, T(S_l))^2}{(0.5 \cdot \sigma(S_l))^2}\right)\right) \times \\ &\left(\frac{1}{K}\sum_{p_2 \in P_l} \exp\left(-\frac{D_{\text{Eucl.}}(p_2, T^{-1}(S_k))^2}{(0.5 \cdot \sigma(S_k))^2}\right)\right). \end{aligned} \tag{4.3}$$

Note that $\sigma(S)$ depends on the expected ambiguity of pairwise alignments (higher σ captures fuzzier pairwise matching), we use $\sigma(S) = \text{Diam}(S)/5$ for all examples, where $\text{Diam}(S)$ is an average distance between all pairs of points, i.e., $\text{Diam}(S) := \sum_{p_j, p_j \in S} D_{\text{Eucl.}}(p_i, p_j)/K^2$. The global term roughly estimates what fraction of surfaces align under a tighter threshold $\sigma(S)/2$. Finally, we find the aligning transformation T for a pair of models. In our experience, the models in Google Warehouse have consistent upward orientation, so we only look for optimal 2D rotation and a scale. To avoid optimizing over the space of real-valued parameters, we test all 4 alignments of principal components for a pair of shapes as an initial guess, and locally refine the transformation using ICP optimization [48]. We choose the transform that maximizes the global alignment score, i.e., $\arg\max_T B_a$. Note that we store all the four aligning transforms with the edge and the transformation can change during the graph optimization (see Step 6).

4.3.4 Step 3b: Align pairs of shapes intrinsically

In cases where the input collection is known to contain smooth, near-isometric manifold surfaces, intrinsic mapping techniques are more suitable. For such datasets, we use

blended conformal maps [71] as pairwise alignment method. We further define an alignment score based on an intrinsic map $m : S_k \rightarrow S_l$. Similarly to the original work, we use a map's area distortion to quantify alignment confidence at a point: $c_m(p \in S_k) := 2 \Big/ \left[\frac{area(N_p)}{area(f(N_p))} + \frac{area(f(N_p))}{area(N_p)}\right]$, where N_p is a 1-ring vertex neighborhood around vertex p. We define local and global terms for the alignment score in Equation 4.1, as:

$$L_a^{\text{blend}}(p_i, p_j) := c_m(p_i) \cdot \exp\left(-\frac{D_{\text{geod.}}(m(p_i), p_j)^2}{\sigma_{\text{intrinsic}}(S_l)^2}\right) \tag{4.4}$$

$$B_a^{\text{blend}}(S_k, S_l) := \frac{1}{K} \sum_{p_1 \in P_k} c_m(p_1) \tag{4.5}$$

where, $\sigma_{\text{intrinsic}}(S_l) = 0.3 \cdot \sqrt{area(S_l)}$.

Note that the method of Kim et al. [71] produces several intrinsic maps in a presence of near-symmetry (5-10 for examples presented in this paper). Although during the initial alignment we pick a map with the highest score B^{blend}, we make use of all the candidate maps in the graph optimization (see Step 6).

4.3.5 Step 4: Fill and embed the correspondence matrix C

We populate a correspondence matrix C using alignment scores for pairs of shapes connected by an edge in the alignment graph **G**. As each row in C is associated with a mapped point, we row-normalize C, labeling it $\tilde{C}$, so that every point is mapped somewhere in the collection with a constant energy:

$$\forall (k, l) \in \mathbf{G} : \qquad \tilde{C}(p_i \in S_k, p_j \in S_l) = \frac{a_{S_k, S_l, \phi}(p_i, p_j)}{\sum_{p_j} a_{S_k, S_l, \phi}(p_i, p_j)}. \tag{4.6}$$

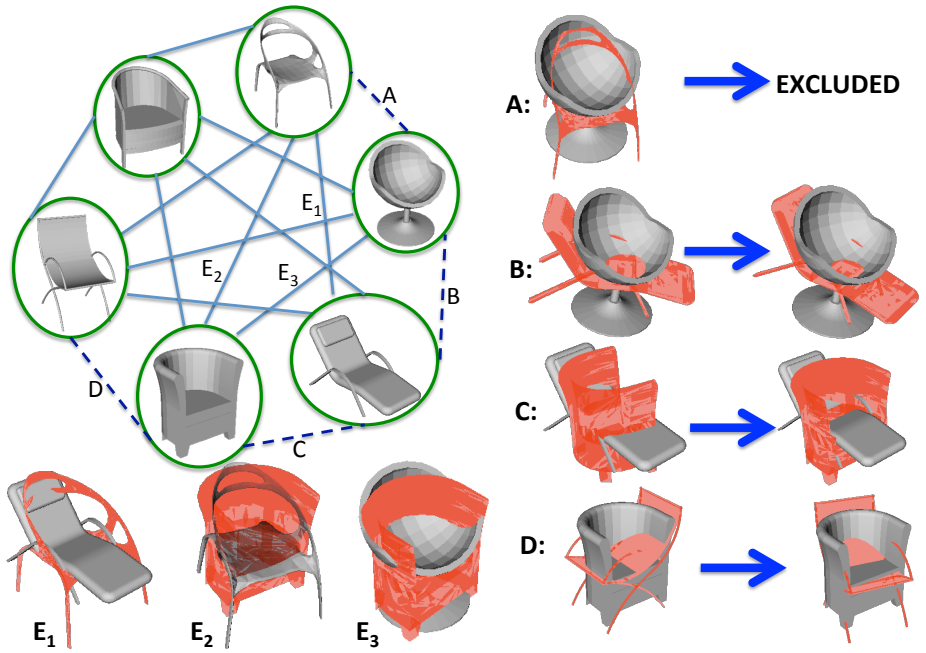

Figure 4.4: ***Graph optimization.*** *For a small example graph of 6 nodes we show how the edges change during the graph optimization. Edges that correctly align models (solid lines) compensate for the noise introduced by edges where pairwise matching fails (A-D). The graph optimization either re-aligns some models (B-D) or excludes an alignment (A) if no good re-alignment was found. The main issue with (A) is that all possible alignments map seat too close to other chair's base.*

We further perform spectral analysis of $\tilde{C}$. Let ψ_n be n^{th} eigenvector and λ_n be n^{th} eigenvalue of the matrix $\tilde{C}$. The eigenvectors ψ are normalized as proposed by Nadler et al. [92], setting $\Psi := D^{-1/2}U$, where U are the original eigenvectors and D is a diagonal matrix with the row sums of C as its entries. The diffusion map at time t ($t = 10$ in our examples, set to a higher value to reduce the influence of noisy alignments) defines the spectral embedding:

$$\Pi_t(p_i) := \left(\lambda_1^t\psi_1(p_i), \lambda_2^t\psi_2(p_i), \ldots, \lambda_{NK-1}^t\psi_{NK-1}(p_i)\right) \tag{4.7}$$

where, $\psi_n(p_i)$ is the i^{th} component of n^{th} eigenvector corresponding to the point p_i. We further define the diffusion distance as Euclidean distance in the embedded space:

$$D_t(p_i, p_j)^2 := \sum_n \lambda_n^{2t} (\psi_n(p_i) - \psi_n(p_j))^2. \tag{4.8}$$

4.3.6 Step 5: Compute fuzzy correspondence.

We define the fuzzy correspondence function as

$$f(p_i, p_j) := \exp\left(-D_t(p_i, p_j)^2 / \tau(p_i)^2\right) \tag{4.9}$$

Algorithm 1 Optimize $\mathbf{G}_i$ to create new graph $\mathbf{G}_{i+1}$

$\mathbf{G}_{i+1} \leftarrow (\mathbf{G}_i.\S, \emptyset)$

Improve consistency of all edges
for $a \in \mathbf{G}_i.\mathbf{A}$ **do**
  $a_{S_k,S_l}.\phi \leftarrow \arg\max_{\phi' \in \Phi}(\text{score}_{f_i}(a_{\phi'}))$
end for

Only keep consistent edges
$\text{NE}_{\text{add}} \leftarrow 0$
for $a_{S_k,S_l} \in \mathbf{G}_i.\mathbf{A}$ **do**
  $E_{\text{adj.}} \leftarrow \{a_{S_m,S_n} \in \mathbf{G}_i.\mathbf{A} \text{ s.t. } m = k \text{ or } n = l\}$
  $best \leftarrow \max_{a \in E_{\text{adj.}}}(\text{score}_{f_i}(a))$
  if $\text{score}_{f_i}(a) \geq 0.3 \cdot \text{best}$ **then**
    $\mathbf{G}_{i+1}.\mathbf{A} \leftarrow \mathbf{G}_{i+1}.\mathbf{A} \cup a$
  else
    $\text{NE}_{\text{add}} \leftarrow \text{NE}_{\text{add}} + 1$
  end if
end for

Sample new alignments
$candidates \leftarrow a_{S_l,S_k}$ s.t. $\text{ShortestPath}_{\mathbf{G}_{i+1}}(S_l, S_k) > 3$
$candidates \leftarrow \text{IFCBest}_{f_{i+1}}(candidates, 3 \cdot \text{NE}_{\text{add}})$
$candidates \leftarrow \text{EdgeRankBest}(candidates, \text{NE}_{\text{add}})$
$\mathbf{G}_{i+1}.\mathbf{A} \leftarrow \mathbf{G}_{i+1}.\mathbf{A} \cup candidates$

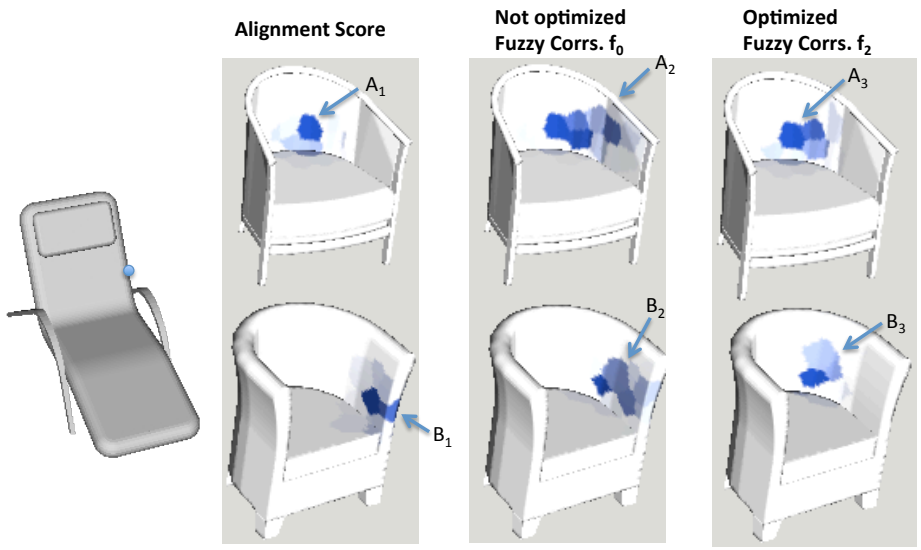

Figure 4.5: ***Graph optimization: fuzzy correspondences.*** *Fuzzy correspondences before and after the graph optimization shown in Figure 4.4 – note that the bottom chair is misaligned, see Figure 4.4C. Note that diffusion improves the result for the bottom chair by using indirect paths (B_1 to B_2), however, the results for the top chair get fuzzier due to incorrect alignments (A_1 to A_2). Finally, after the pairwise alignments are fixed, the fuzzy correspondences on the right are more accurate (A_3 and B_3).*

where, $\tau(p_i)$ is point-specific normalization set equal to the distance to the farthest of 15% of the nearest points (to p_i) in the embedded space. To efficiently find f from the spectral embedding, we set the fuzzy correspondence of all points that are farther than $2\tau(p_i)$ to 0, and based on the low-rank assumption only consider the top K eigenvectors. We search for the nearest neighbors in K-dimensional space using approximate nearest neighbor search (FLANN) [91].

4.3.7 Step 6: Optimize alignment graph

Given the fuzzy correspondence function f_0 created by the embedding the graph $\mathbf{G}_0$, we update the graph $\mathbf{G}_0$ such that pairwise alignments (i.e., samples) are consistent with the estimated embedding, i.e., f_0. Thus, we want to optimize the graph by (i) detecting and pruning noisy samples, and (ii) adding new samples to the under-sampled areas.

To detect noisy samples, we search for pairwise alignments that are inconsistent with the aggregated contribution of all other paths in the alignment graph. More specifically, given an alignment graph $\mathbf{G}_i$ and resulting fuzzy correspondences f_i, we define a *consistency score* for each aligning edge a as a simple correlation:

$$\text{score}_{f_i}(a_{S_l,S_k,\phi}) := \sum_{p_1 \in P_l} \sum_{p_2 \in P_k} \frac{a(p_1,p_2) f_i(p_1,p_2)}{\sum_{p \in P_l} |f(p_1,p)|}. \tag{4.10}$$

Since our choice for the alignment parameter ϕ in the initial graph only depended on the quality of the pairwise matching, it is possible that another parameter ϕ that would align models with higher consistency score. We compute score_{f_i} for all precomputed alternative alignments (e.g., 4 ICP initializations for extrinsic matching, or other low-distortion blended maps). For each edge, we pick the parameter ϕ that maximizes the consistency score. If the best score for an edge is below 30% of the node's best score (for both nodes) we assume that shapes are too dissimilar and prune the edge.

Finally, we want to include the same number of edges that was pruned (NE_{add}), preferably choosing pairs of models that can be matched more robustly, and such that matching them improves the embedding. Unlike during initialization, at this point we already have an approximation to the function f which we use to guide the sampling. We expect to get more information by aligning pairs that are separated by long paths in the alignment graph $\mathbf{G}$, thus we use a candidate set of all edges such that the shortest path in the pruned graph between two nodes has more than 3 edges. Additionally, we rank all the edges by the integrated fuzzy correspondence value $IFC(S_k, S_l) := \sum_{p_1 \in S_k, p_2 \in S_l} f(p_1, p_2)$ and pick the 3NE_{add} highest-ranked edges. Finally, we use edge rank to order them and add the top NE_{add} that improve the graph connectivity (see Figures 4.4, 4.5). Algorithm 1 summarizes a single iteration of this optimization.

4.4 Results

To evaluate the quality and speed of our method for computing fuzzy correspondences, we ran a set of experiments with benchmark data comprising ground truth one-to-one surface correspondences. While our system is not designed for such data (we allow more ambiguous correspondence relationships), the experiments provide means to compare with previous work. Using this benchmark data, we investigate if our algorithms indeed discover a low-dimensional manifold in shape space and leverage it effectively.

4.4.1 Data Sets

We test our method on two benchmarks. The first involves smooth manifold surfaces that we used in Section 3.4.1 (also [71]). In addition, we created a second benchmark by downloading diverse collections of 111 chairs and 86 commercial airplanes from Google 3D Warehouse. As ground truth, we manually annotated 10 and 6 feature points respectively on each model of the chairs and airplanes datasets (see Figures 4.7, 4.14, and 4.15). Note that these models have large variations in the number of connected components (from 1 to 74,796) and polygons (from 816 to 2,622,379).

4.4.2 Evaluation metric

Semantic correctness of fuzzy correspondence values is hard to quantify since it requires us to prescribe values for semantic similarity for all pairs of corresponding feature points. Instead, we assume that annotated feature points are in perfect correspondence and simply project the fuzzy correspondences to the space of point-to-point maps by choosing the points with the best correspondence value. More specifically, given a pair of shapes S_k, S_l for any point $p_i \in P_k$ we assign the closest point in the embedded space: $corr(p_i) :=$

$\arg\min_{p_j \in P_l} D_t(p_i, p_j)$. We use nearest-neighbor interpolation based on such a map of K samples to capture large-scale correspondences for the benchmark.

We measure the quality of a map as proposed in Section 3.4.2 (also in [71]): we record distance from the predicted correspondence to the true correspondence $d_{S_l}(corr(p_i), f_{\text{true}}(p_i))$, and plot our results as a curve showing the fraction of correspondences mapped correctly within a threshold $d_{S_l} < D$, where different thresholds D are mapped to the x-axis. We use geodesic distances for the benchmark of intrinsically similar shapes and Euclidean distances for a diverse set of shapes. We now report the findings of our experiments based on these evaluation methods.

4.4.3 Correspondence space is low-dimensional

First, we investigate if correspondences among diverse models can indeed be effectively embedded in a low-dimensional space. For this test, we pick a relatively uniform set of 11 chairs (Figure 4.6) and use our method to compute the correspondence matrix C (note that all pairs are matched with default parameters and all alignments are good). Although the correspondence matrix is very high-dimensional ($11 \cdot 128 \times 11 \cdot 128$), note the spectral gap in the distribution of the eigenvalues. On the left we further map each of the $11 \cdot 128$ points on all shapes to a 2D plane using just the top two non-constant eigenvectors. The structure of that manifold resembles the shape of a generic chair, and corresponding points are close to one another in the embedded space, which agrees with our hypothesis that the correspondence space is low-dimensional $O(K)$.

4.4.4 A small subset of pairwise alignments suffices

Next, we test how sparsity of the alignment graph **G** affects the fuzzy correspondences. In Figure 4.6-right we demonstrate embedding due to an alignment graph with 20 pairs of

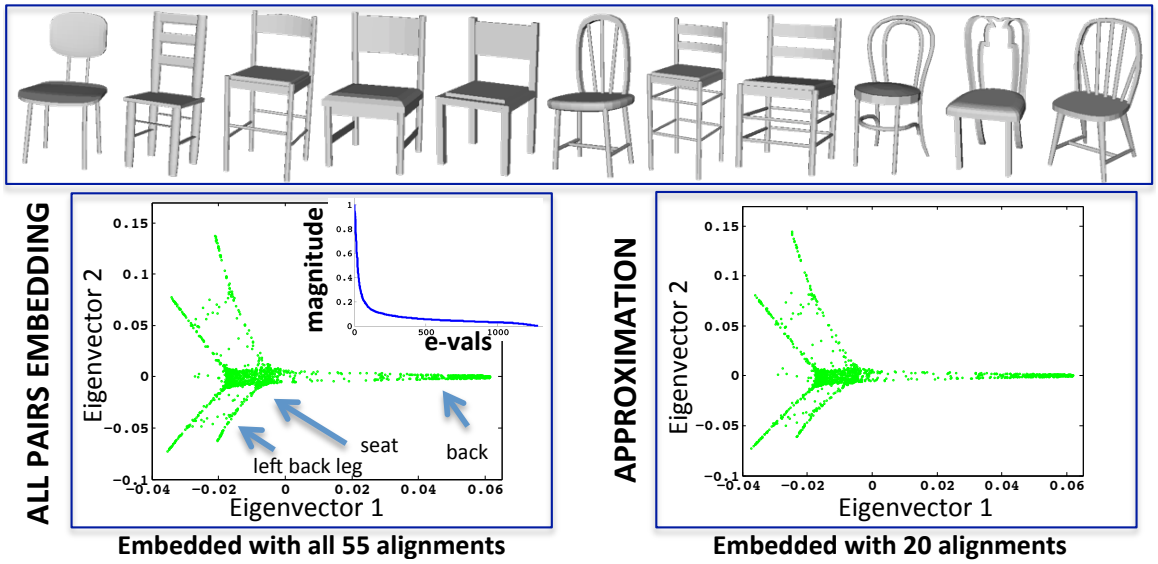

Figure 4.6: ***Top two non-constant eigenvectors of*** C***.*** *This figure demonstrates eigenvalues and the embedding of a collection of 11 chairs. A green dot in the embedded space corresponds to one of* 128×11 *points in the data. The correspondence matrix is sampled by aligning all 55 pairs, and arbitrary 20 pairs of shapes.*

models. Note that the embedding is robust to a sparser sampling. In another experiment (see Figure 4.7), we only use correct affine transformations that best align the prescribed feature correspondences in the least squares sense (note that this is the only experiment where we use the ground truth to align models). The dashed curve (manual affine) shows the accuracy obtained just by the ground truth alignment. We further compute fuzzy correspondences from the alignment graph created with 150, 250, 350, and 500 edges. The results indicate that even with 500 alignments we already reach the accuracy of using ground truth affine alignment for all 6105 pairs.

4.4.5 Fuzzy correspondences improve intrinsic pairwise maps

We also compare our method to work on optimizing collections of maps by Nguyen et al. [94] who also optimize blended intrinsic maps for consistency. Our evaluation is based on four classes in the SHREC dataset: animals, humans, teddy bears, and hands (we only use the collections of maps produced by the authors). For all datasets we compare our

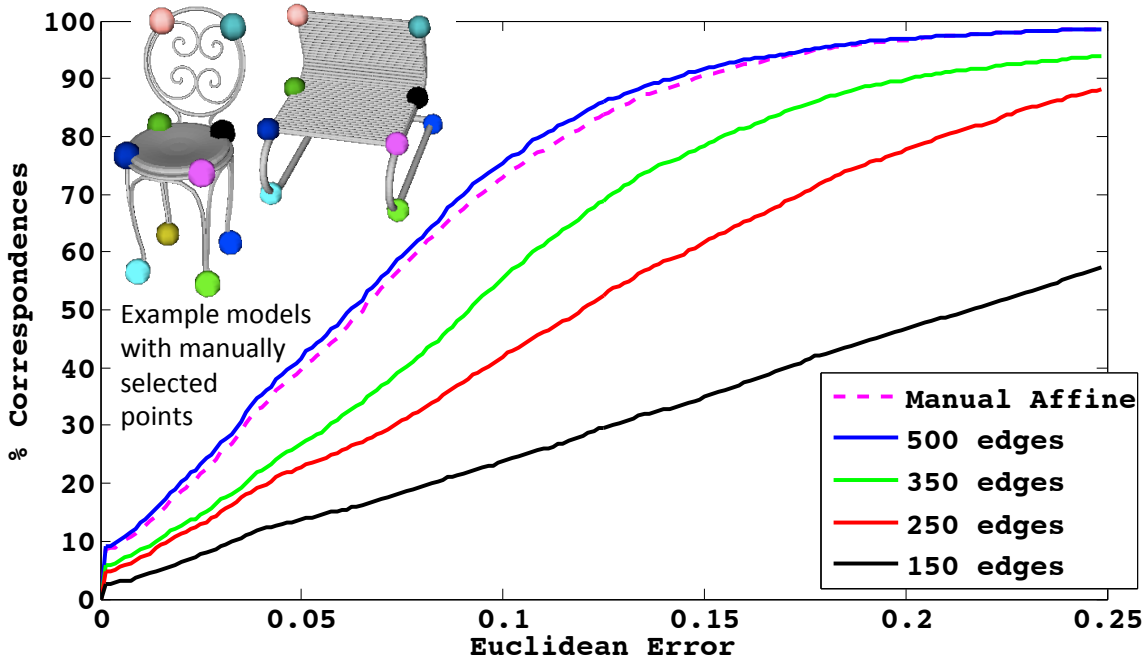

Figure 4.7: ***Chairs with manual alignments***. *We compare our method to naively aligning all pairs (dashed line) using ground truth pairwise alignments. Using fuzzy correspondences, even with 500 alignments, we get comparable results to those obtained by matching all the 6105 pairs.*

method (blue curve) to optimized maps [94] (green curve), and blended intrinsic maps [71] (red curve). We also include results for *consistent blended maps* (magenta), where we choose a blended map that is the most consistent with fuzzy correspondence values (this is equivalent to improving edge's consistency just for a single pairwise alignment). Note that unlike fuzzy correspondences a consistent blended map must be one of maps produced with pairwise alignment algorithms, which results in inferior performance when different regions have to be mapped via different diffusion paths. These results, however, do not suffer from discretizing shape into points, as fuzzy correspondences do.

For animals (Figure 4.8) and humans (Figure 4.10) datasets we successfully resolve *all* flips caused by symmetry confusion except for two higher genus human models where all blended maps fail (also Nguyen's failure case). Similar to Nguyen et al. [94], our method is confused by the symmetry in teddy bears (Figure 4.9), and we also consistently misalign some of the models.

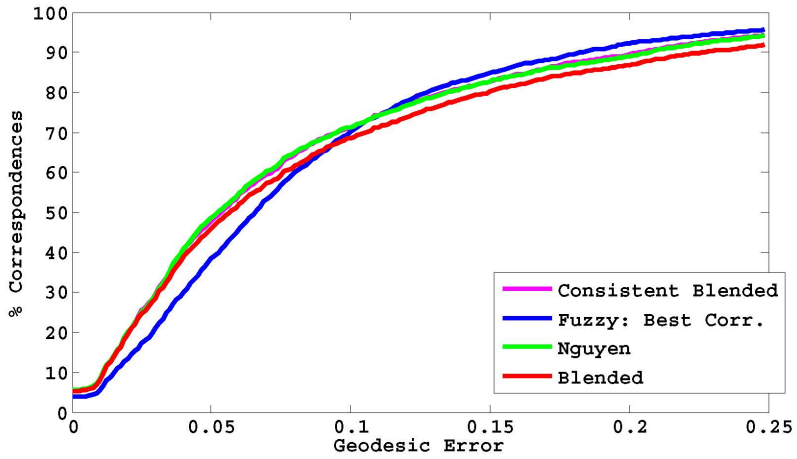

Figure 4.8: ***SHREC Animals dataset (20 models).*** *Comparison of four methods: taking the best fuzzy correspondence (blue), taking a blended intrinsic map consistent with fuzzy correspondences (magenta), method of Nguyen et al.[94] (green) and just using blended intrinsic map (red).*

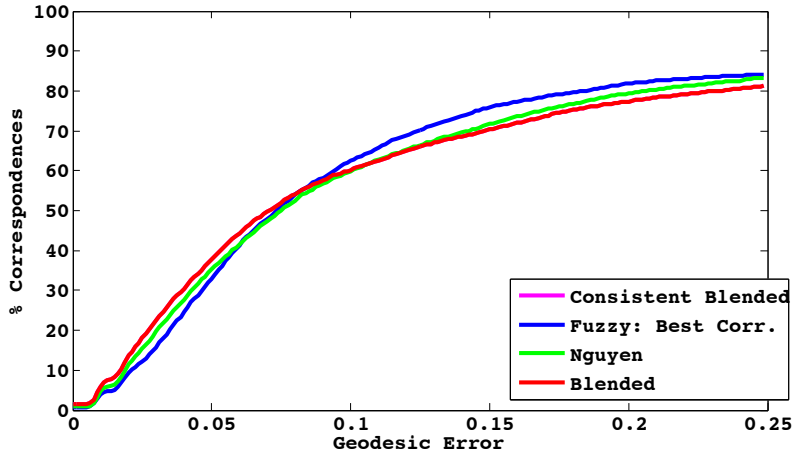

Figure 4.9: ***SHREC Teddies dataset (20 models).*** *Refer to caption of Figure 4.8 for details.*

The hand dataset (Figure 4.11) shows the real benefit of our method. Note that this dataset is different from the other three in the main source of error for the pairwise matching: instead of globally inconsistent alignments due to symmetry, it usually has local inconsistencies in mapping fingers. Note that in such cases their method is limited to finding a concatenation of full model-to-model maps that eventually aligns fingers correctly — this may be impossible if none of the maps are perfect. On the contrary, we optimize per point, and thus aggregate several maps to produce a consistent alignment. Thus we can correctly

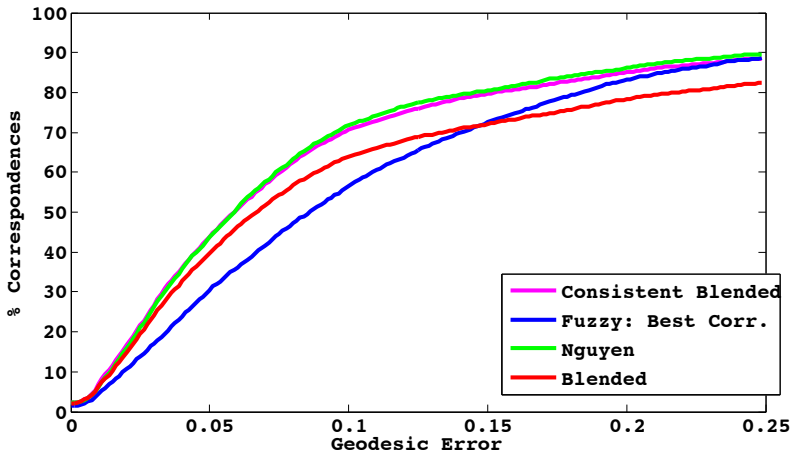

Figure 4.10: ***SHREC Humans dataset (20 models)****. Refer to caption of Figure 4.8 for details.*

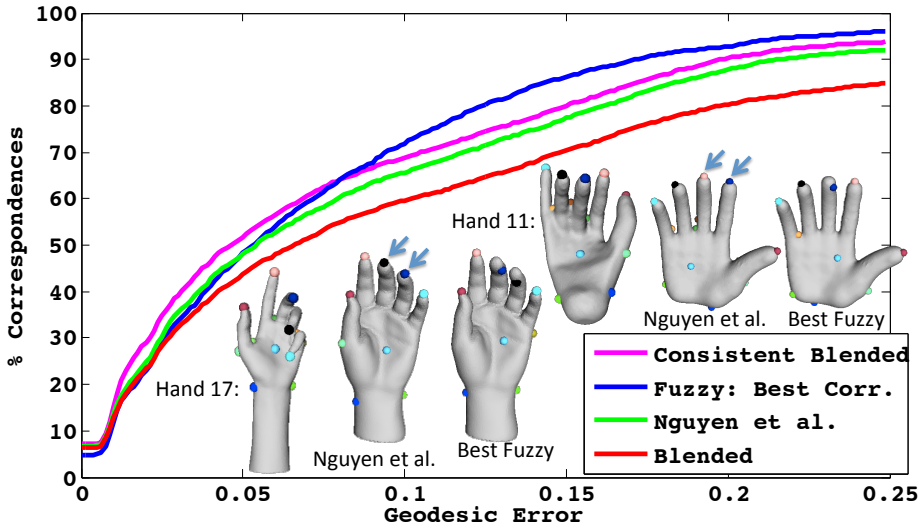

Figure 4.11: ***SHREC hands dataset****. Comparison of four methods: taking the best fuzzy correspondence (blue), taking a blended intrinsic map consistent with fuzzy correspondences (magenta), method of Nguyen et al.[94] (green) and just using blended intrinsic map (red). Note that hands 11 and 17 are consistently misaligned by the method of Nguyen et al. to all the other 18 models in the database.*

align *all* the hands, as opposed to their method (see hands 11 and 17 in Figure 4.11 and these hands are consistently misaligned to *all* the other 18 hands in their results).

Note that default implementation of Nguyen et al. [94] requires maps between all pairs of shapes as an input to their algorithm, and thus their method is too expensive to use for comparison on larger datasets. We compare our approach to just using blended intrinsic

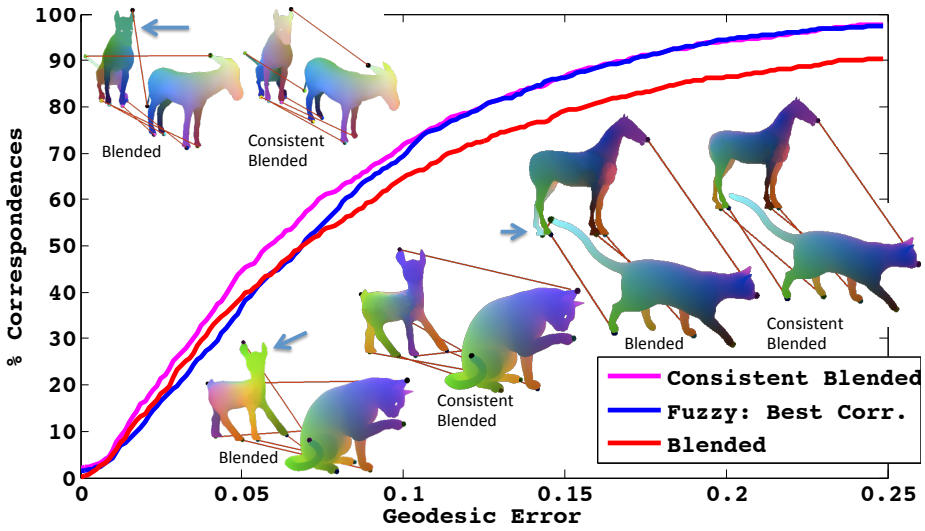

Figure 4.12: ***Animals dataset****. Comparison of the three methods: choosing blended map that is consistent with fuzzy correspondences (magenta), taking the best fuzzy correspondence (blue), and just using blended intrinsic map (red). Three examples show results obtained with the blended map, and results obtained by choosing a blended map consistent with fuzzy correspondences. The right surface is colored by mapping xyz coordinates to rgb colors and the left surface is colored by transferring color using a map. Regions misaligned by the blended map are highlighted by arrows.*

maps to map selected pairs of shapes among 71 human models (SCAPE) and 51 animals. We compute error the same pairs of shapes as in Section 3.4.3 (and [71]), overall testing 71 pairs in the first dataset and 51 pairs in the second. Figures 4.12 and 4.13 show that for larger tolerable geodesic error (right side of plots), mapping a point to the closest point in the embedded space (blue) performs better than blended maps (red) and almost all correspondences are accepted. This suggests that consistency gives leverage to fuzzy correspondences over blended intrinsic maps, creating maps that are correct at a large scale.

Please refer to Huang et al. [58] (a work that followed our publication) for comparison of fuzzy correspondences to their map optimization method.

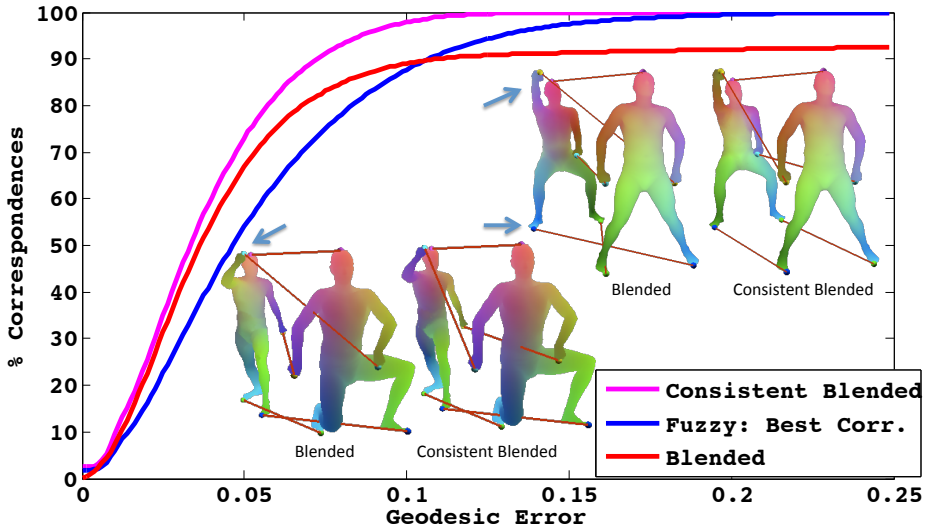

Figure 4.13: ***SCAPE dataset***. *Comparison of three methods similarly to the animals dataset (Figure 4.12) the main improvement with out method comes due to consistency optimization. We present a few representative examples where consistent blended map provides an improvement over the blended map.*

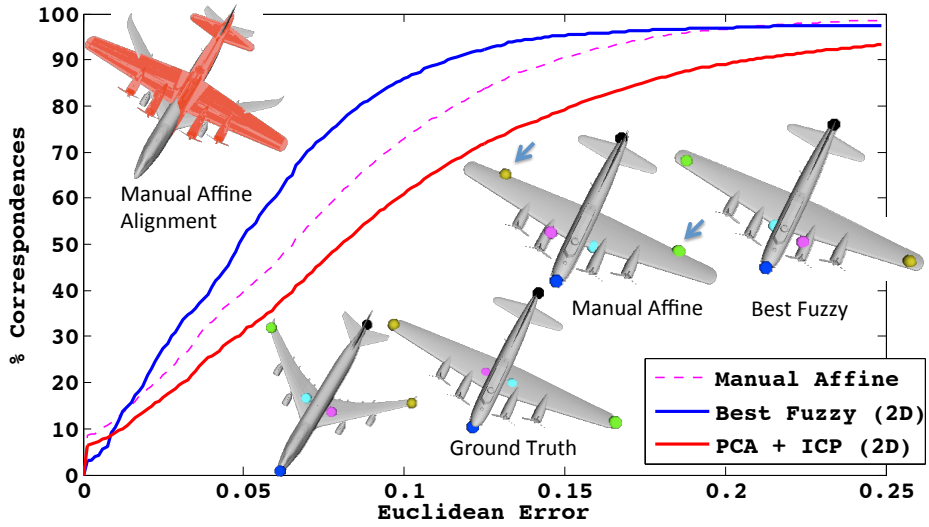

Figure 4.14: ***Airplanes dataset***. *We compare taking best fuzzy correspondence to geometric matching of all pairs. Note that fuzzy correspondences also do better than the best manual pairwise alignment because diffusion expands the allowable aligning deformations beyond affine transformations by using indirect alignments.*

4.4.6 Fuzzy correspondences improve extrinsic pairwise maps

In Figures 4.14 and 4.15 we compare a pairwise matching using just the best affine transformation (red), affine transformation produced with ground truth correspondences (magenta,

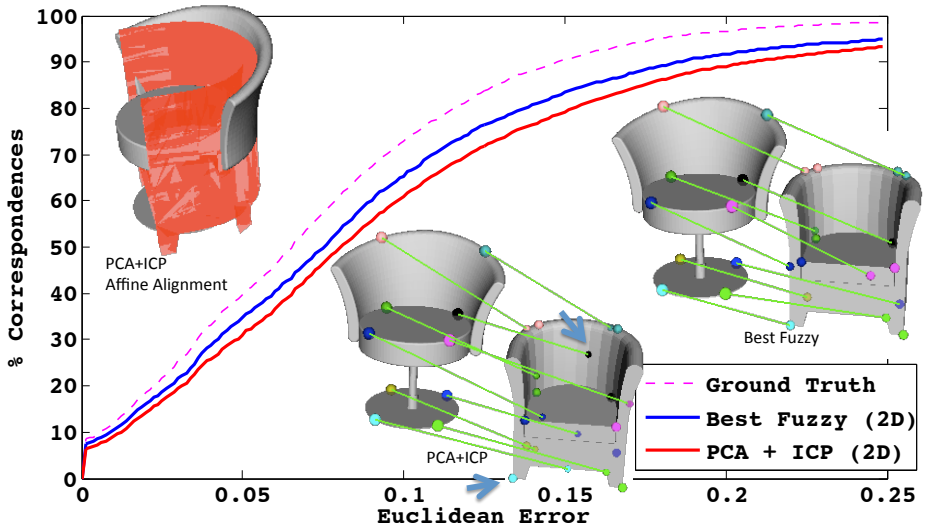

Figure 4.15: ***Chairs dataset****. In this dataset we are comparing taking best fuzzy correspondence to pairwise alignment of all pairs.*

dashed), and taking best fuzzy correspondence (blue). Similarly to intrinsic maps, we improve over pairwise matching due to consistency optimization (avoiding matching the front of an airplane to the tail). Interestingly, for some error intervals, fuzzy correspondences outperform the affine alignment with the ground-truth points. The reason for this is that wings and the body generally have very different proportions and position along the body of the airplane, thus even with ground truth alignment, the semantic points might not align (i.e., the deformations within the class of airplanes are beyond our pairwise alignment method); however, diffusion allows these correspondences to propagate closer to semantically correct values. That is, it extends the allowable deformations beyond the capability of our simple pairwise matching algorithm. Although our method on average does not perform much better than the naive ICP in Figure 4.15, it fixes issues with some unusual chairs (see Figure for an example) that do not contribute much to the global error (since there are fewer of them).

Refer to Section 5.4.5 for more comparisons of fuzzy correspondences to more recent techniques including Huang et al. [58], and method described in Chapter 5.

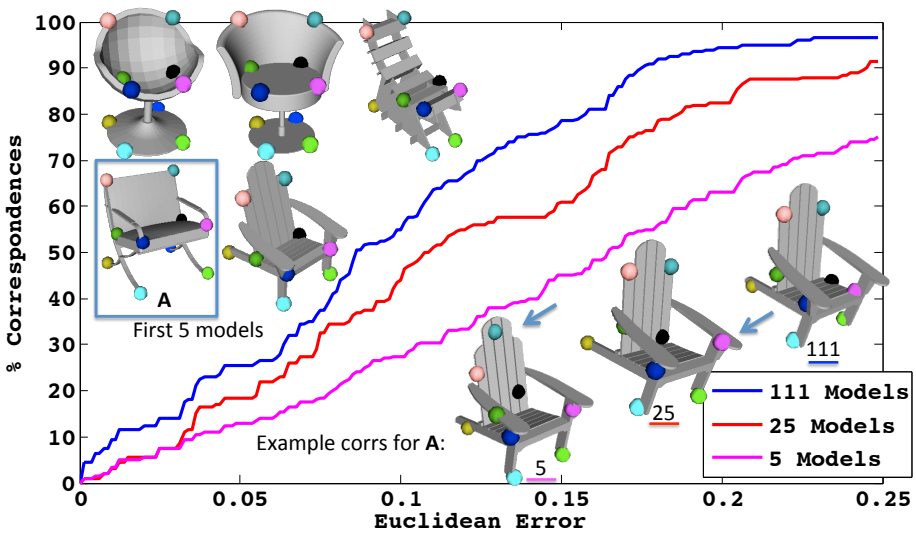

Figure 4.16: ***Chairs: 5, 25, 111 models***. *We analyze the error for 5 selected models present in all datasets and add more chair models for joint analysis. Note that increasing the size of the database improves the quality of the correspondences, as shown in the example correspondences at the bottom right.*

4.4.7 More data improves correspondences

In our final experiment, we investigate whether leveraging the "power of the set" aids computation of fuzzy correspondences between individual pairs. Figure 4.16 shows the accuracy curves as the number of models in the database changes. We selected a subset of 5 chairs from the full collection aiming at higher variation within the subset, and show the error only for the selected 5 models; similarly for a dataset with 20 additional random models from the chair dataset, and all the 111 models. These results demonstrate that increasing the size of the database improves the correspondences due to a denser sampling of shape variations within a class.

4.4.8 Timing

We executed all off-line computations of fuzzy correspondences on SunFire X4100 computer with an AMD Opteron 275 Dual-Core 2.2GHz processor. For the largest dataset of 111 chairs the whole analysis (including pairwise matching) requires about 900s, of which

about 300s are spent on spectral analysis of the correspondence matrix, 500s on graph optimization, and the rest on pairwise matching. Our algorithm used only 602 out of 6105 pairs for chairs dataset, and 430 out of 3655 airplane pairs. Although, the optimization timings are comparable to the SCAPE and animals datasets, the intrinsic pairwise matching takes up to 5 minutes per pair for SCAPE (we match 355 of 2485 pairs), and 10 to 30 minutes per pair for animals (we match 632 of 1275 pairs). We compute these maps in parallel on a cluster. In all examples our method converged within 8 iterations. On the hands dataset (Figure 4.11) fuzzy correspondences were computed with 100 intrinsic maps and the optimization process converged within 70s. Nguyen et al. [94] used all 380 pairwise alignments, and the optimization took 140s. Finding the intrinsic maps takes about 5-10 minutes. Our exploration interface runs interactively on a laptop with 2.4GHz Intel Core 2 Duo processor.

4.4.9 Parameters

We use the same parameters for all the datasets. We choose $K = 128$ samples per model based on the desired resolution of correspondences: a larger K offers higher precision in fuzzy correspondences at the cost on increased compute times. We chose $t = 10$ based on the desired trade-off between robustness and fuzziness of correspondences: higher diffusion times reduce noise due to misalignment errors and sparse connectivity in the alignment graph, while lower diffusion times provide higher discriminative power of correspondence values. In the limit, as t goes to infinity, f approaches a uniform distribution for every point.

4.4.10 Limitations

In the off-line step, we found that fuzzy correspondences are vulnerable to the presence of a strong sampling bias (e.g., due to near-symmetry). This issue usually manifests in two ways: (i) A model might be aligned incorrectly (but consistently) across the collection (e.g., a nose of an airplane is mapped to tails of all other airplanes in a collection). Among all examples datasets, we found that 1 of 86 airplanes were consistently misaligned. (ii) If a bias is too common for a collection, diffusion might propagate information along incorrect paths, causing an undesirable blending of correspondences (see Figure 4.3). The largest collection we analyzed with our current implementation has 111 models. Scaling our method to thousands of models would require the parallel implementation of some steps of the algorithm like eigenvalue decomposition and iterative graph optimization. We are also restricted to collections where for every pair of dissimilar models there is a continuous path of pairwise-similar models. While the datasets we analyzed have various types of noise, multiple disconnected components, intersecting surfaces, large holes, micro holes, etc., our method is unlikely to be suitable for processing raw scans.

4.5 Exploring 3D collections

Despite the rapid growth of 3D model repositories, the task of exploring such large 3D collections remains an important and challenging problem. In particular, while most online databases make it easy for users to select sets of similar models using text-based filtering, understanding the range of variations *within* such collections is typically much more difficult (see also [98]).

For many object classes, one key challenge is that the shape can vary in many different ways, and users may be interested in exploring different types of variations. For example, within a collection of chair models, one user may want to see how the backs of chairs at-

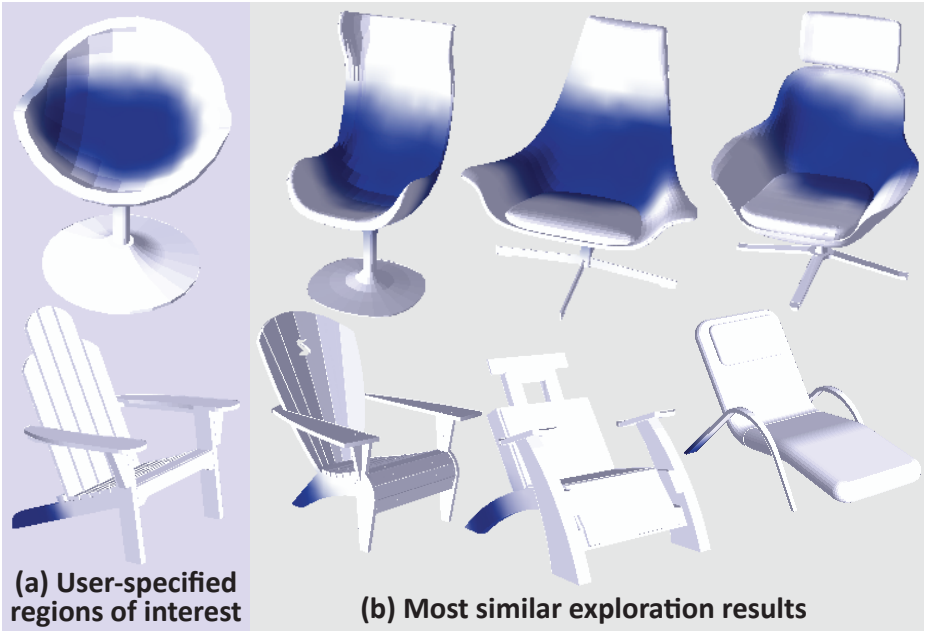

Figure 4.17: ***Region-based exploration of chairs.*** *The user specifies exploration criteria by selecting regions of interest (a), and our system sorts models based on their similarity within those regions (b). Corresponding regions are highlighted in dark-blue.*

tach to the seat (smoothly merge, right angle, etc.), while another user may only want to see chairs whose back legs are at a certain angle (see Figure 4.17). Even within a single exploration session, a user may want to define the exploration space using multiple attributes (e.g., chairs with a stem base and curved back). Given the spectrum of possible exploration criteria, a static predefined organization of the data is clearly not sufficient.

In this section we demonstrate that fuzzy correspondences can assist in the interactive exploration of shape structure in 3D collections, especially similarities and differences between shapes. To investigate this idea, we present a new analysis tool and exploration interface for 3D model collections. As a key feature, we allow users to directly specify regions of interest (ROI) on example shapes in order to guide subsequent exploration actions. Thus, we can support the browsing scenarios described above; the user selects the appropriate ROIs on one or more chairs, and the system automatically organizes the rest of the chairs based on their similarity to the specified region.

Specifically, we introduce a browsing interface that allows users to paint regions of interest on any example shape, which then act as navigation criteria to determine how the other models in the collection are organized and presented to the user. Our interface consists of two panes: the ROI pane shows the set of user-selected regions of interest, and the results pane shows a sorted view of the collection on one or more result pages (see Figure 4.19a). By selecting various ROIs and examining the results, users can quickly explore many different types of variations within the collection. In order for this type of interface to be effective, it must satisfy a few key design requirements:

Finding where variations occur: Faced with an unfamiliar collection, users may not know *a priori* which regions of a shape to explore. Thus, the browsing interface should convey where interesting variations occur within a collection to help users decide what ROIs to specify.

Visual comparison: To help users understand the similarities and differences across a set of shapes, the interface should facilitate visual comparison of the models. In particular, it should be easy for users to see shapes from a consistent viewpoint and focus on the ROI of each model.

Interactive sorting: Finally, to enable guided exploration using the selected ROIs, the browsing system must be able to interactively sort models based on their similarity to the example shape(s).

The next three sections describe the main features of our interface with respect to these requirements.

4.5.1 Finding Variations

To visualize where variations occur across a collection, we measure the amount of variation within different corresponding regions of the shapes. For each sample point on a shape, we

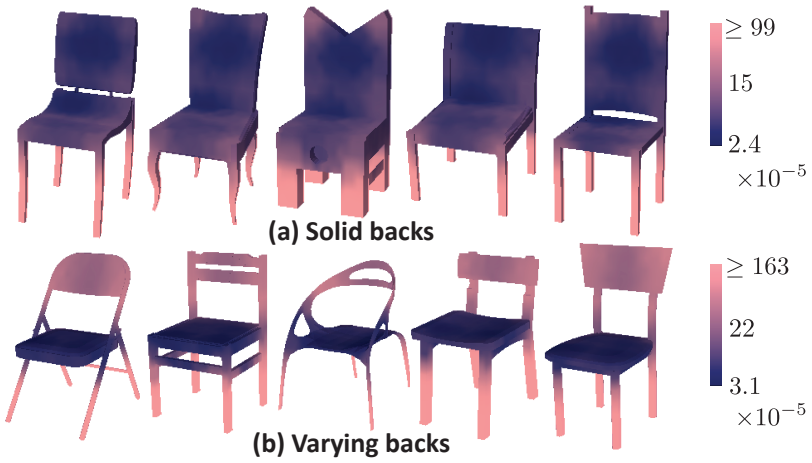

Figure 4.18: ***Variance function for two example collections.*** *The chairs in (a) have more similar backs than the chairs in (b) (dark blue regions indicate less variance). In both collections, seats exhibit less variation than the legs, which have a variety of styles.*

compute an average distance to 15% of the nearest neighbors in the embedded space. This average represents an estimate of how much variance there is in a particular region of the shape with respect to the rest of the collection. We normalize these variance values within each model and then visualize this function with a colour map (see Figure 4.18), where brighter colors indicate more variance. By examining this visualization, users can quickly see what regions of a shape vary the most or the least.

4.5.2 Visual Comparison

To help users understand the similarities and differences between the models presented in our interface, we provide several features to facilitate visual comparison. First, the system allows the user to align all the models to a consistent viewpoint, as shown in Figures 4.19a–c. We compute automatic alignments for either the entire shape (see Figure 4.19b) or just the selected region (see Figure 4.19c) using fuzzy correspondences as described in the next subsection. Once the models are aligned, the user can interactively orbit, pan, and zoom the viewpoints of all the models in conjunction, which makes it easy to inspect and compare different portions of the shapes. Furthermore, to facilitate orbiting around a

selected ROI, our system places the orbit centre for each result model at the centroid of the corresponding ROI points, which we compute as an average of the the point locations weighted by their correspondence values. Finally, to emphasize the corresponding ROIs in the result models, we highlight these regions in blue (see Figures 4.19c–f). The intensity of the highlight is scaled by the correspondence values at each sample point, interpolated across the surface. To ensure that the highlight is sufficiently visible on each model, we normalize the correspondence values by the maximum correspondence value within the model.

4.5.3 Interactive Sorting

To sort the collection based on a user-selected ROI, we need a way to measure the geometric similarity between the selected region of an example shape and all other target models in the collection. To this end, we introduce a selection-aware similarity function that uses fuzzy correspondences to determine how to align and compare target models to example shapes.

Given a selected region $R \subset S_i$ on example shape S_i we compute the distance $D_R(S_j)$ to target shape S_j as follows. First, we determine the best rigid alignment $T_R(S_j)$ of the target shape to the selected region. Specifically, for every selected point on the example shape, we find all fuzzy correspondences on the target shape and then compute $T_R(S_j)$ by finding the optimal affine transformation that aligns the corresponding points, where the error is weighted by the correspondence values and minimized in the least-squares sense. To ensure that the alignment remains stable even for small selected regions, we add fuzzy correspondences for *unselected* points to the computation as well, but we multiply their values with a damping factor $\alpha \ll 1$ set to $1/4K$, where K is the number of point samples per model. Next, we find a single correspondence value f_R for each point on the target shape by taking the maximum correspondence value to any point in the selected

Figure 4.19: ***Exploration interface and exploration results.*** *The original collection of chairs (a) is automatically aligned to a canonical viewpoint (b) and then to a selected region of interest (c). While exploring a collection of animals (d), the user queries for a specific arrangement of paws and the system returns all animals in a sitting pose as the most similar results. If the user does not select the right paw, a cat with one paw up appears among the top matching results. To combine exploration criteria, the user selects regions on multiple example shapes. For example, she browses for chairs with high curved backs and stems (e), bikes with large front wheels and straight handlebars (f), and humans with an upright posture and arms extended away from the torso (g).*

region $f_R(p) := \max_{p' \in R}(f(p', p)f(p, p'))$. Finally, we use these correspondence values as weights to compute the Euclidean distance between the aligned target shape and the example shape as $D_R(S_j) := \sum_{p \in S_j} D_{\text{Eucl.}}(T_R(p), S_i) f_R(p) / \sum f_R(p)$.

4.5.4 Discussion

We use our interactive tool to explore and visualize variations within several example collections (see Figure 4.19 and supplementary video): chairs, bikes, commercial airplanes, animals, and SCAPE humans. The ability to browse the collection based on specific regions of interest enables us to uncover interesting characteristics within all of these datasets. For example, by selecting the curved back and arms of a chair (Figure 4.19e) we discover that several other chairs have a similar arrangement of these features. Selecting the straight handlebars of a bike (Figure 4.19f) reveals a variety of shapes that share this property, including the "bike" with no wheels that is ranked as the most similar result. In addition to emphasizing similarities in the collection, our system also reveals diversity. For example, by sorting the tops of chair backs in variations mode, we immediately see a wide range of results that include boxy, organic, upright and tilted shapes (see Figure 4.19c).

Note that the ability for users to select arbitrary regions rather than just predefined parts represents a significant advantage for our exploration system. For diverse collections such as the chairs, selecting portions or combinations of standard "parts" often yields informative and discriminative navigation criteria, such as the seat/back region in Figure 4.17. Furthermore, for collections where our analysis considers intrinsic geometry, such as the animals and humans, selecting regions that span several limbs or joints is an intuitive way to guide exploration based on pose. For instance, Figure 4.19g shows how selecting the mid-section of a human can restrict navigation to upright poses, while a selection across the shoulders and chest returns humans with their arms extended away from their sides. As another example, selecting all four paws of the dog in Figure 4.19d retrieves other sitting

animals, but if we do not select one of the front paws, we also discover the sitting cat with one paw in the air.

Finally, the ability to combine exploration criteria into facets provides additional flexibility and control during navigation (see Figures 4.19e–g). Faceted browsing not only allows the user to narrow the exploration space (e.g., only show chairs with both a stem and high curved back) it also gives the user a sense for what types of variations are independent of each other within the collection. For example, the sequence of facets in Figure 4.19e shows that the shape of chair backs and the type of base vary independently, and the queries in Figure 4.19g reveal a similar independence between the different aspects of human poses in the SCAPE dataset. These types of insights are especially valuable for understanding the variations across diverse shape collections.

Limitations: In some cases our exploration tool might return unintuitive results due to limitations of the interactive sorting algorithm, which does not capture variations in some interesting geometric features (see Figure 4.20). Developing tunable and more discriminative geometric descriptors that employ fuzzy correspondences is an interesting topic for future work.

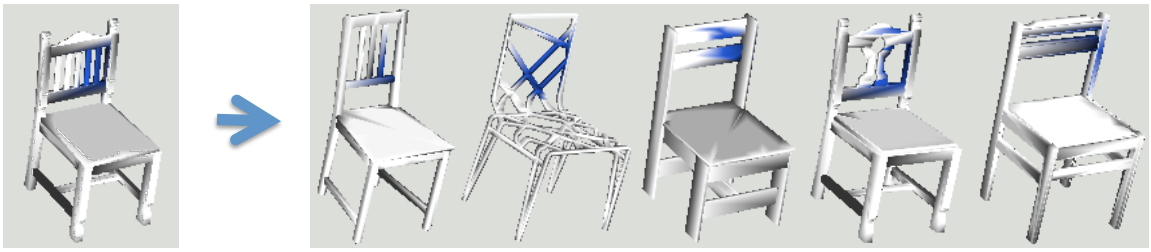

Figure 4.20: ***Sorting limitation***. *The similarity metric we use for the interactive sorting is too simple to capture some aspects of geometry like vertical bars on chair's back.*

Chapter 5

Learning part-based templates to establish structure in large collections of 3D shapes

5.1 Introduction

Large repositories of 3D models that have become available in recent years offer an unprecedented amount and diversity of data, posing a need for automated algorithms to establish structural relationships between shapes. There are three important goals of structural analysis: understanding correspondences between shapes, segmentation of shapes into semantic parts, and modeling deformations of individual parts. These analysis results find applications in shape synthesis [61], surface reconstruction [73], and object recognition [93].

Existing efforts to analyze collections of polygonal models consider correspondence, segmentation, and deformation *separately*. For example, the surface correspondence establishes links between related points on different surfaces while ignoring the part structure of objects [68, 58]; consistent segmentation algorithms decompose polygonal mod-

els into consistent sets of parts but do not compute point-to-point correspondences [48] or model shape variation [115]. Finally, prior work on probabilistic modeling of shape variations requires manually annotated datasets with parts and correspondences already labeled [36, 61]. Moreover, such algorithms do not scale to handle the thousands or tens of thousands of models in many object classes found in large repositories.

We provide an algorithm that simultaneously segments polygonal models into parts, learns a probabilistic model of part-based template variations, and establishes point-to-point surface correspondences in large collections of shapes. The rationale for this approach is that these three problems are all inter-connected, i.e., segmentations help predict correspondences and deformations; correspondences help predict segmentations and deformations; and deformation models help predict segmentations and correspondences. Attacking the three problems together leads to more efficient, more accurate, and more consistent analysis results.

Our algorithm is based on a probabilistic, part-based, deformable model, which encodes clusters of shape styles, cardinalities of parts, shapes of parts, correspondences across clusters, and alignments of a template to each model. It starts from a repository of polygonal models and an initial deformable model, which is represented by a template encoding the types of parts expected in the repository and an initial guess for the locations and scales of each part. It proceeds by iteratively evolving the set of deformable templates, fitting them to the polygonal models (implicitly aligning and co-segmenting the polygonal models), updating the distributions of shape deformations, and refining part- and point-level correspondences across the shapes (see Figure 5.1).

The part-based deformable model is similar to the one used in [61], but it is learned from unstructured and unlabeled data, rather than labeled examples. While the learning algorithm can run in a fully automatic mode, we found that the analysis results for diverse collections can substantially improve with user-assisted template initialization. For all ex-

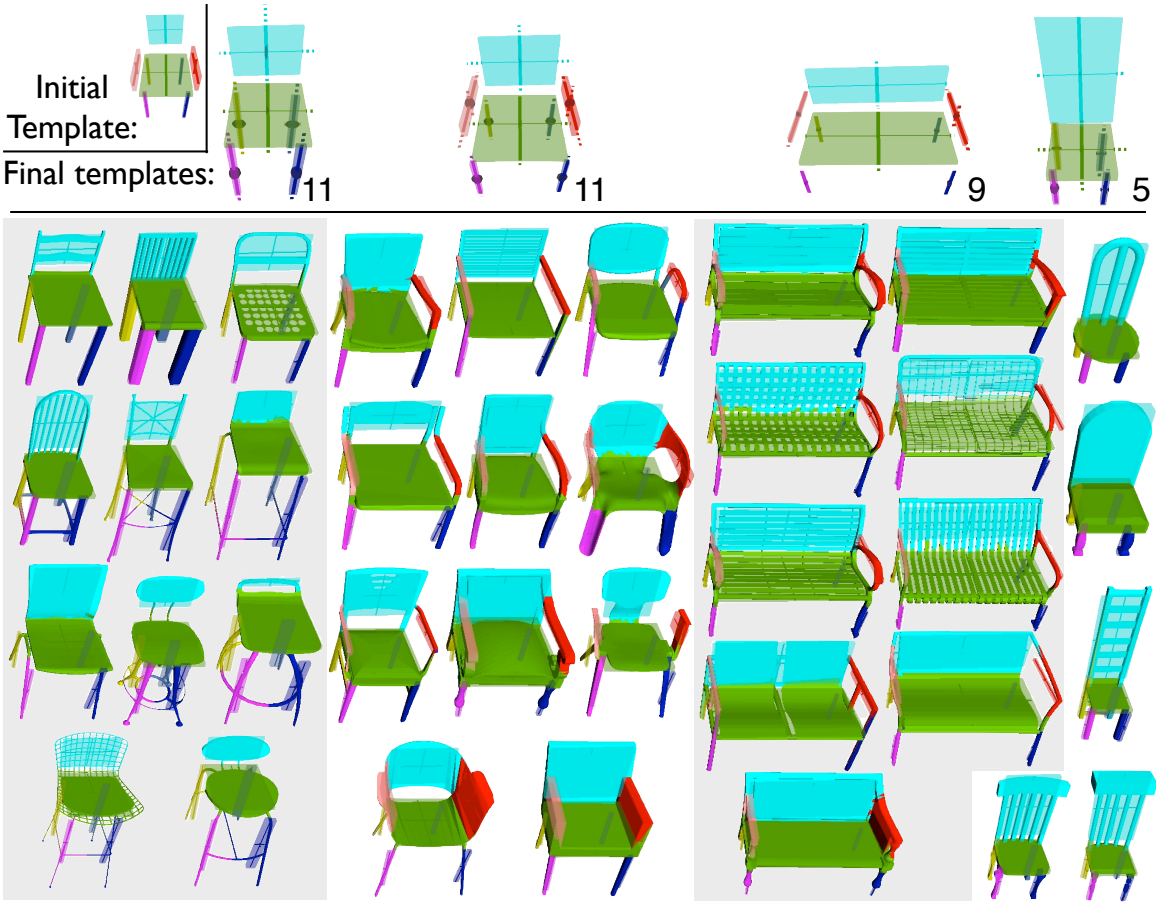

Figure 5.1: *Analysis results for a collection of 36 chairs. Starting from an initial template (top left), we capture the main modes of variations within the collection by the final templates (top row). In this example, the algorithm extracted template clusters for chairs without arms and with arms; a cluster for wide benches; and a cluster for tall chairs. By jointly solving for model deformations, part segmentation, and inter-model correspondence, our algorithm achieves higher accuracy for each task.*

amples presented in this paper, the user never had to spend more than five minutes per shape class.

Additionally, our algorithm has a linear complexity regarding the number of models in the collection that can be executed out-of-core, and is highly parallel. As a result, we can analyze polygonal model collections an order of magnitude larger than most previous geometry processing datasets (e.g., 7K+ models in one case). It also provides joint segmentations, correspondences, and deformation models, all of which attribute to increased accuracy.

Additionally, our algorithm performs favorably on standard benchmarks in comparison to specialized algorithms for consistent segmentation and surface correspondence.

Contributions: In summary, we present an algorithm to

- learn structure from any large, unorganized, unlabeled shape collection,
- simultaneously recover segmentation, point-level correspondence, and a probabilistic part-based deformable model for shape collections in order to reveal key information for many data-driven geometric modeling and synthesis tasks, and
- efficiently realize an out-of-core framework to handle collections spanning thousands of models at a scale never demonstrated before.

5.2 Related Work

Many algorithms have been developed to find interesting structural relationships in collections of shapes. Let us overview the methods that work under the assumption that shapes can be decomposed into semantically equivalent parts.

5.2.1 Consistent segmentation

Recently, there has been significant work on the consistent segmentation of 3D model collections. Golovinskiy and Funkhouser [48] presented an approach that first aligns models, then builds correspondences, and finally segments all models into parts. Since their method relies on rigid alignments of model pairs to establish correspondences, they obtain good results only for collections with little shape diversity. Moreover, their algorithm computes alignment, correspondence, and segmentation in sequence, without feedback between the steps, and thus fails to leverage information learned in later steps to correct earlier mistakes. The "deform-to-fit" consistent segmentation technique of Xu et al. [126] suffers

from a similar limitation. Their algorithm computes initial segmentations independently for each mesh, those segmentations are clustered without alignment or correspondence, the co-segmentation for each style cluster is computed independently without accounting for the statistics of shape variation, and the co-segmentations for different style clusters are not adapted as inter-style part correspondences are found. In addition, the overall complexity of the algorithm is quadratic in the number of models in the collection.

Several existing approaches address the problem of consistent segmentation and labeling by clustering points in an embedded space of local shape features [62, 120, 59, 115, 56, 123]. However, these methods do not learn or leverage the statistics of part cardinalities, shapes, or geometric arrangements, which are relevant types of structure for many applications. Furthermore, the algorithms make one or more of the following assumptions: input collections have low shape variations/deformations; access to clean and manifold models; patch-based feature signatures are robust across model variations; and access to labeled training data. Moreover, as output, they produce only labels (e.g., labels indicate leg, but not which leg) and possibly inconsistent correspondences across the entire data set. In terms of scalability, the supervised methods (e.g., [62, 120]) require 15% to 95% of the collection to be segmented and labeled as training data, while Wang et al. [123] require users to interactively specify tens to hundreds of constraints. Finally, most of these analysis algorithms rely on comparing all pairs of shapes, which leads to running times in the tens of hours for hundreds of shapes [59], making them impractical for much larger collections.

5.2.2 Part-based models

Our approach is motivated by the successful application of deformable templates for object recognition in computer vision [60]. Felzenszwalb et al. [32] and others have developed part-based models to encode distributions of appearances and spatial arrangements of parts, and used them for recognition of objects in images [34, 4, 33]. Similar approaches have

been used for pose estimation [84], image segmentation [30], and viewpoint classification [50]. Yet, in all this work, the part-based model is given or is learned from previously segmented and labeled training data. In contrast, we learn the part-based model from unlabeled data, evolving a set of templates to best fit the data (see [124, 125] for applications in visual recognition).

Many geometry processing tasks require and assume access to part-based model information: Shen et al. [112] use a database of segmented models to reconstruct models from Kinnect scans; Xu et al. [127] use part-based deformation model to spawn an evolutionary model aimed at creation of novel and interesting model variations; Kim et al.[73] searches over allowed deformations in part-based template models to fit object labels and pose attributes to sparse noisy point cloud scans. The output of our algorithm can directly be used as input for such applications.

Kalogerakis et al. [61] learn a probabilistic distribution over a part-based model encoding multiple object styles, part cardinalities, part shapes, and part placements, and use it for shape synthesis. However, this method assumes access to manually segmented and labeled examples. Since we focus on analyzing very large model collections, manually annotating even a small fraction of such collections is infeasible and, thus, calls for a different approach.

Ovsjanikov et al. [98] describe a part-based method to explore shape variations within a collection of polygonal models. Their algorithm forms a template from a single, manually segmented model and then describes shape variations in terms of how part translations and scales affect a global D2 shape descriptor. This approach handles deformations that reveal themselves as low-dimensional structures (e.g., 1D curves) in the global descriptor space but fails to discover part-level shape variations. Also, their analysis does not explicitly map template parts to surface regions on models and, thus, is not directly useful for applications that require segmentations and/or correspondences.

5.3 Method

5.3.1 Overview

At the core of our algorithm is a probabilistic, part-based model that uses a set of deformable templates to describe shape variations within a collection. Each template represents a distribution (or "style") of shapes as a set of oriented parts with random variables indicating their positions, sizes, and local geometric features. This part-based model matches the modes of variation found in many collections of man-made objects, which often contain a few clusters of shapes with more-or-less the same types of parts arranged in more-or-less the same locations with specific instances differing only slightly in the positions, sizes, and shapes of parts [126]. For example, the collection of 36 chairs in Figure 5.1 has four clusters with different sets of parts (arms versus no arms) and/or parts with different relative sizes and shapes (e.g., benches versus chairs). To model such variations, we introduce an automatic method for learning part-based deformable templates to an input shape collection.

Starting from an initial set of templates, we use an iterative algorithm that (i) deforms templates to fit the shapes, (ii) clusters the shapes based on their fits, and (iii) refines the set of templates to better describe the shape variations within each cluster, possibly spawning new templates in the process (see Figure 5.3). We repeat these steps in sequence until the family of learned templates stops evolving, or a maximum number of iterations is reached. The resulting clusters of shapes, one for each template, represent the discrete set of shape styles across the collection. Within each cluster, the random variables of the associated template describe continuous shape variations, and the template-to-shape fitting information provides non-rigid alignments, point-to-point correspondences, and consistent part segmentations. By default, we compute the initial templates based on an automatic segmentation of

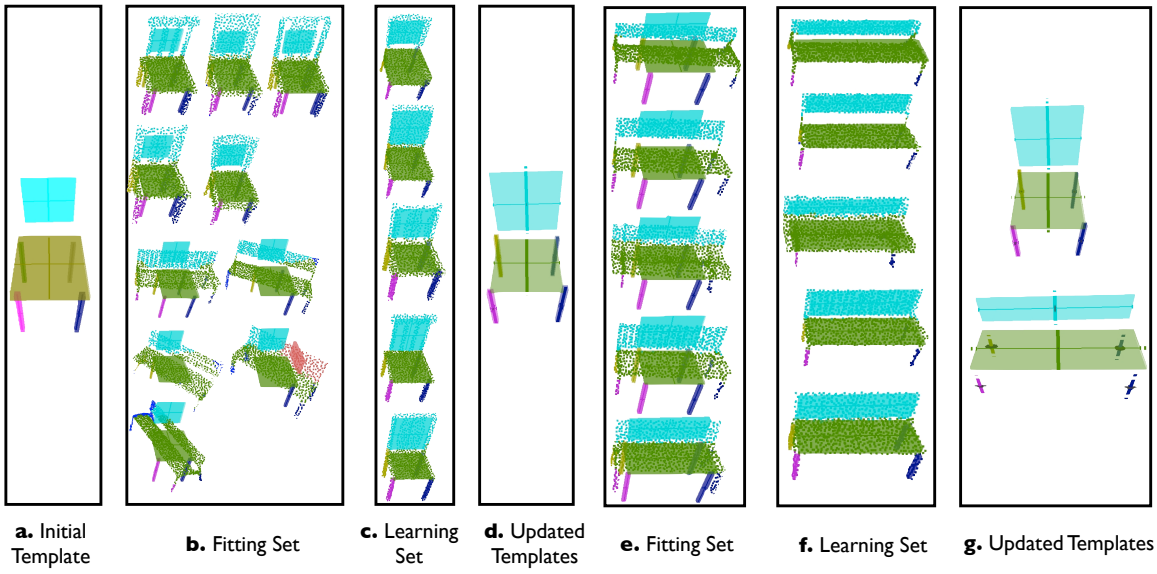

Figure 5.2: ***Overview example.*** *Two main modes of variation are learned in a collection of 10 models (5 chairs and 5 benches). First, an initial template (a) is matched to all the models (b), however, only chairs with small fitting error are included in the Learning Set (c). Variations are learned from the set and the template is updated (d), which is then fit to the remaining shapes (e). Note that the learned part-wise deformations subsequently lead to better segmentation and alignment of benches. Variations among benches (f) are too dissimilar from the chairs, and hence, a second template is spawned to the final set of templates (g). In the rightmost image and throughout the paper, a higher variance in part positions is depicted by larger ellipsoids at part centers, while higher variance in anisotropic scales is depicted with longer dashed lines.*

a set of randomly selected shapes. The user can select/refine these templates, or create her own.

The following section describes in detail our part-based template representation, the individual steps of our iterative algorithm, and strategies for template initialization.

This section describes the steps to learn a set of templates for a collection of shapes.

The main input is a collection of shapes, each represented by a discrete set of points. Note that our algorithm *does not* require manifold polygonal meshes, or even meshes at all. Rather, it can take almost any shape representation as input, including polygon soups, voxel grids, and point set surfaces – it simply samples points on the surfaces of other representations to form point sets for its analysis. This choice is disadvantageous for most steps of

Inputs:
  Shape collection S
  An optional template T
Outputs:
  Updated set of templates $\{T\}$
  Template↔shape clusters C
  Template→shape global rigid transformations R
  Template→shape per-part deformations D
  Template↔shape point mappings M
  Shape→template point labelings L
  Template→shape fit errors E

———-

If no T is provided, CreateAutoTemplate(S) **(Section 5.3.4)**
LearnTemplate(T, S) **(Section 5.3.3)**
  foreach iteration
    $\{R, D, M, L, E\}$ = FitTemplateSet(T, S)
    C = ClusterShapes(T, S, E)
    T = RefineTemplateSet(T, S, C, R, D, L, E)
  return $\{T, C, R, D, M, L, E\}$

———-

FitTemplateSet(T, S) **(Section 5.3.2)**
  foreach $S[j] \in S$
    foreach $T[i] \in T$
      $\{R[i,j], D[i,j], M[i,j], L[i,j], E[i,j]\}$ = FitTemplate$(T[i], S[j])$
  return $\{R, D, M, L, E\}$

FitTemplate(t, s)
  foreach candidate alignment r
    $d = t_{mean}$
    repeat until ℓ converges
      ℓ = SegmentShape(t, s, d, r)
      $\{r, d, e\}$ = argmin$_{\{r,d\}}$ FitError(t, s, ℓ)
  return $\{r, d, \ell, m, e\}$ with least e

———-

RefineTemplateSet(T, S, C, R, D, L, E)
  S_F = SelectFittingSet(S, E)
  T_F = FitSet(T, S_F, C, L)
  S_L = SelectLearningSet(T_F, S, E)
  T' = ClusterLearningSet(T, T_L)
  return T'

Figure 5.3: *Algorithm pseudocode.*

our algorithm (e.g., computing segmentations with good boundaries is simpler with connected surface meshes), but it is necessary to allow processing the wide variety of shape representations found in repositories on the Web (e.g., polygon soups).

A second input, which is optional, is one (or more) initial template(s). The user can manually create such templates. If not, the system can create template(s) automatically by choosing from segmentations of several candidate shapes (see Section 5.3.4).

Subsequently, our main algorithm (LearnTemplate in Figure 5.3) proceeds by interleaving three steps: fitting, clustering, and refinement. During the fitting step, every template is deformed to fit to every shape. Then, during the clustering step, shapes are associated with their best-fitting template. Finally, during the refinement step, the set of templates is updated to reflect the shape deformations observed during the fitting step, possibly spawning new templates to describe clusters of shapes represented poorly by the deformations of the initial templates. We terminate when either template cannot be updated from any new shapes, or after reaching the maximal number of iterations N_{iter}.

Figure 5.2 illustrates two iterations of the template learning procedure. The example collection of 10 models is bi-modal and includes 5 chairs and 5 benches. Starting with the initial template (a) we first fit it to every model (b), note how the intermediate alignments and segmentations produced with the initial template are not accurate since it does not cover the space of all shape deformations. We further learn deformation parameters from a subset of models (c) and use these parameters to refine the set of templates (d). Iterating the fitting and learning steps (e, f) allows our method to identify the second mode of shape variation (g).

Template definition: In our implementation, we treat a template as a collection of k boxes $\{B_1, \ldots, B_k\}$ with each template part being abstracted as a box. We model each box by a Gaussian distribution capturing its centroid position $(\mu^p(B_i), \sigma^p(B_i))$, a Gaussian distribution capturing its anisotropic scale $(\mu^s(B_i), \sigma^s(B_i))$, and a Gaussian distribution of per-point local shape features $(\mu^f(B_i), \sigma^f(B_i))$. Thus, template deformation amounts to relative movements of the boxes and their respective anisotropic scaling (we do not consider rotation in our implementation). Finally, as template initialization, a user can mark

certain parts as must-exist to enforce semantic structure of the learned templates (e.g., a seat must-exist in a chair template).

We now describe the various stages of our algorithm, focusing on template fitting and refinement, while deferring template initialization to Section 5.3.4.

5.3.2 Template Fitting

Given a set of templates T, our next task is to "fit" each of them to a set of shapes S. For each pair of template $t \in T$ and shape $s \in S$, our goal is to find the segmentation of the shape and the deformation of the template that optimizes an energy function e measuring the alignment of their parts.

Free variables: The free variables in this optimization are: (i) a rigid transformation aligning the template to the shape (r), (ii) a set of deformation parameters for the template (existences, position, and scales of parts) that best fit the shape (d), (iii) a mapping from points in the shape to corresponding points on the template (m) and vice-versa, and (iv) a labeling of points in the shape according to their corresponding part in the template (ℓ).

Energy function: The goal is to minimize an energy function $e(t, s, r, d, m, \ell)$ measuring the fit of the template parts to the shape segmentation. The energy function is designed to favor segmentations that are consistent with both the shape geometry and the template structure, while penalizing implausible deformations of the template to fit the shape. To achieve these goals, we define the fitting energy e to be a sum of three terms:

$$\begin{aligned} e(t, s, r, d, m, \ell) = E_{\text{data}}(t, s, r, d, m, \ell) + \alpha E_{\text{deform}}(t, d) \\ + \beta E_{\text{smooth}}(s, \ell) \end{aligned} \tag{5.1}$$

The data term measures distances and dissimilarities in local shape features between points on the shape and corresponding points on the template. To compute it, we suppose that the

shape and the template are both uniformly sampled with discrete sets of points, P_s and P_t, respectively, and sum error estimates at those points:

$$E_{\text{data}}(t, s, r, d, m, \ell) = E_{s\to t} + E_{t\to s} + \gamma E_{\text{feat}} \quad (5.2)$$

$$E_{s\to t}(t, s, r, d, m) = \frac{1}{|P_s|} \sum_{p_s \in P_s} E_{\text{dist}}(p_s, r(d(m(p_s))))^2 \quad (5.3)$$

$$E_{t\to s}(t, s, r, d, m) = \frac{1}{|P_t|} \sum_{p_t \in P_t} E_{\text{dist}}(r(d(p_t)), m^{-1}(p_t))^2 \quad (5.4)$$

$$E_{\text{feat}}(t, s, l) = \frac{1}{|P_s|} \sum_{p_s \in P_s} E_{\text{feat dist}}(p_s, \ell(p_s))^2 \quad (5.5)$$

where E_{dist} measures the Euclidean distance between points (normalized by the shape radius R, computed as the furthest distance between any pair of points in a shape), and $E_{\text{feat dist}}$ measures the squared difference between a local shape feature vector at a point on the shape with the average local shape feature vector for all points in the corresponding part of the template divided by the variance of those features. To account for potential noise or outliers a point can be labeled as $null$, the distance penalties are 0, and we set high penalty $E_{\text{feat dist}}(p_s, null) = 1$ in this case.

In our implementation, we compute local shape features f at a point p by analyzing the covariance matrix of its local neighborhood $Nhd(p)$ which is defined by all points within the distance $\tau_{Nhd} = 0.15R$. Suppose sorted eigenvalues (decreasing order) and eigenvectors are $\lambda_{1,2,3}$ and $v_{1,2,3}$, we produce six features, including ratios of eigenvalues (λ_2/λ_1 and λ_3/λ_1), and normalized angles between axes and eigenvectors $acos(v_1 \cdot a_{\text{up}})/\pi$, $acos(v_1 \cdot a_2)/\pi$, $acos(v_2 \cdot a_{\text{up}})/\pi$, $acos(v_2 \cdot a_2)/\pi$.

The data term produces a low error if surfaces are well-aligned, have similar part structures, and have similar local shape features at corresponding points.

The deformation term penalizes solutions that have statistically unlikely positions b_p and scales b_s of template parts B:

$$E_{\text{deform}}(d) = \sum_{b \in t} \frac{|b_p - \mu^p(B)|^2}{\sigma^p(B)^2} + \frac{|b_s - \mu^s(B)|^2}{\sigma^s(B)^2} \tag{5.6}$$

where b is a part in template t, b_p, and b_s are the position and scale for that part, and $\mu^p(B)$ and $\sigma^p(B)$ are the mean and standard deviation of positions for all instances of part b learned from shapes assigned to t (and similarly for b_s). This term penalizes extreme deformations of t to fit s. Note that a user can specify 'must include' parts. If no points are mapped to such a part its deformation penalty is set to $E_{\text{exist penalty}} = \infty$ to avoid learning from topologically invalid shapes.

Finally, the smoothness term penalizes shape segmentations in which nearby points with similar surface normals are assigned to different template parts:

$$E_{\text{smooth}}(l) = \sum_{\substack{p_1, p_2 \in P_s \\ s.t. p_2 \in \text{Nhd}(p_1) \\ \ell(p_2) \neq \ell(p_1)}} -log\left(1 - \frac{\theta_{p_1,p_2}}{\pi}\right) \cdot \exp\left(\frac{dist(p_1, p_2)}{\tau^2_{Nhd}}\right) \tag{5.7}$$

where $\theta(p_1, p_2)$ is the angle between surface normals at two points within local neighborhood p_1 and p_2.

We use $\alpha = 0.5$, $\beta = 2$, and $\gamma = 2$ to weight these three energy terms for all results presented in this paper.

Figure 5.4 illustrates how the different energy terms influence the final fitting. These examples demonstrate that excluding local shape features E_{feat} fails to discriminate between some parts, missing deformation priors E_{deform} results in incorrect and implausible segmentation and alignment of parts; and absence of smoothness E_{smooth} generates labeling with noisy boundaries.

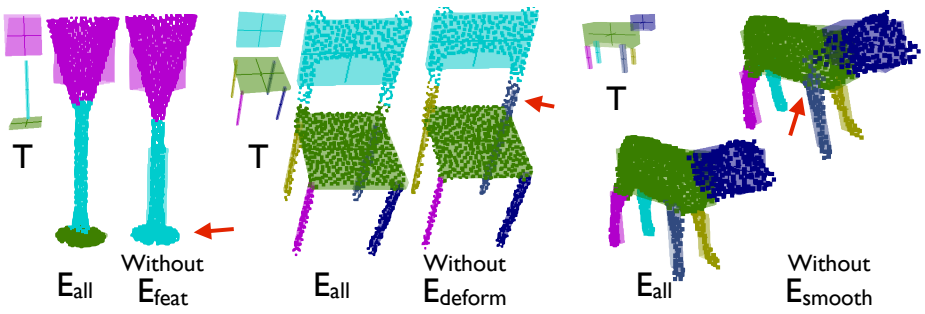

Figure 5.4: ***Energy function example.*** *Effect of every energy term in the template fitting procedure. For all 3 examples, we show a result with all the energy terms (left) and with one energy term excluded (right). (Left-to-right) The stem merges with the base if local shape features are excluded for goblets; chair legs are extended towards back support if we exclude statistical plausibility of a template deformation; and the segmentation boundary between the head and the torso of an animal is noisy in the absence of the smoothness term.*

Optimization procedure: Minimizing this fitting energy is difficult, because the optimal rigid alignment, template deformation, shape segmentation, and point correspondences are all inter-dependent. If the optimal shape segmentation was given, it would be easier to find the rigid alignment and template deformation that best aligns parts. Alternatively, if the optimal rigid alignment and template deformation were known, it would be easier to find the optimal point correspondences and shape segmentation. However, neither is known in advance. We address this problem with an iterative approach.

Our algorithm starts by searching for the rigid alignment r that best aligns the mean template surface to the shape. In practice, we search a discrete number of rigid alignments, noting that geometric repositories commonly have a default up direction a_{up}, and that optimal rotations around that axis are usually multiples of $\pi/2$. Thus, we simply align centroids and try all the four $\pi/2$ rotations around a_{up}. For each of those four starting transformations, r is fixed – we return the one that provides the best e after all other free parameters are estimated.

For each rigid transformation, we optimize $e(t, s, r, d, m, \ell)$ with an algorithm that alternatively optimizes the discrete shape segmentation ℓ, then the discrete point correspondences m, and finally the continuous template deformation parameters m. These three steps are

iterated until the labeling remains unchanged in consecutive iterations (with a maximum number of iterations set to 100).

During the first step, we treat d as fixed and optimize ℓ, the assignment of shapes points to template parts. Since our energy function requires correspondences m, for any point label $\ell(p)$ we set its correspondence to the nearest point on $\ell(p)$. We further exclude the template to points distances $E_{t \to s}$ since m^{-1} is only defined when all points are labeled, and thus cannot be computed without optimizing ℓ. The remaining non-constant terms can be formulated as conditional random field with the unary terms defined by $E_{s \to t} + E_{\text{feat}}$ and the binary terms defined by E_{smooth}. We approximately solve the problem with the efficient graph cut algorithm described by Boykov et al. [15].

During the second step, we treat ℓ and d as fixed and optimize m. This is a classic surface correspondence problem, with the special properties that points are constrained to correspond only to other points with the same part label. Since surfaces have already been aligned by the optimal rigid transformation r and template deformation d (after the first iteration), we simply use closest-point algorithm to estimate point correspondences. Specifically, for each point in s, we find the closest point sampled from the template part with the same label, and vice-versa.

During the third step, we treat ℓ and m as fixed and optimize d. This is a classic regression problem that requires minimizing a quadratic energy function, $E_{\text{data}} + E_{\text{deform}}$. We can solve for critical points $\frac{\partial (E_{\text{data}} + E_{\text{deform}})}{\partial b_p} = 0$ and $\frac{\partial (E_{\text{data}} + E_{\text{deform}})}{\partial b_s} = 0$ for all parts b. Note that we can solve independently for scale and position of every part w.r.t. every dimension, reducing the problem to finding inverses of 2x2 matrices.

As these three steps are iterated, the shape segmentation is refined and the template is deformed to align its parts with corresponding segments on the shape. The final fitting energy provides an estimate for how well the template fits to the shape.

5.3.3 Template Refinement

Once we fit all the shapes to all the templates, our final step is to evolve the set of templates to better represent shape variations within the collection. Our specific goals are to re-estimate the distributions of part positions and scales within the existing templates and to create new templates to represent salient clusters of shapes not fit well by any existing template.

Selecting a learning set: The first challenge is to select a subset of shapes from which the parameters of each template should be re-estimated. This is important for two reasons: (i) to avoid learning template parameters from shapes that fit poorly, and (ii) to avoid the undue computational burden of considering every shape for every refinement of the templates. To address this challenge, we build a *Learning Set* S_L that includes only those shapes that fit best to their associated templates, and then we re-estimate the template parameters only using those shapes.

Our method for selecting the Learning Set is quite simple. First we re-estimate the fitting error for a subset of models, which we call a *Fitting Set*, S_F. In the first iteration the Fitting Set includes all the models in the collection, in the subsequent iterations we sort every shape s by the most recently computed fitting error $\tilde{e}^*(t_s, s)$. Then, we add the K_{L} shapes with lowest $\tilde{e}^*(t_s, s)$ to the Fitting Set, plus another K_{L} shapes chosen at random. We re-fit the current set of templates to the models in the Fitting Set, and update fitting errors $e^*(t_s, s)$ associated with the best fitted template t_s. For each of these shapes, we add it to the Learning Set if its $e^*(t_s, s)$ is less than either a global threshold, e_{min}, indicating an almost-certainly good fit, or less than $\max(e_t, e_{\text{max}})$, where $e_t = \tau \cdot argmin_{s'} e^*(t, s')$, indicating that the fit for s is among the best for t. We terminate the learning procedure when the Learning Set is empty or after reaching the maximal number of iterations N_{iter}. We have chosen conservative values of $K_L = 50$, $e_{\text{min}} = 75$, $e_{\text{max}} = 150$, $\tau = 4$, and $N_{\text{iter}} = 10$ empirically, and keep them the same for all results presented in this paper, noting that K_L

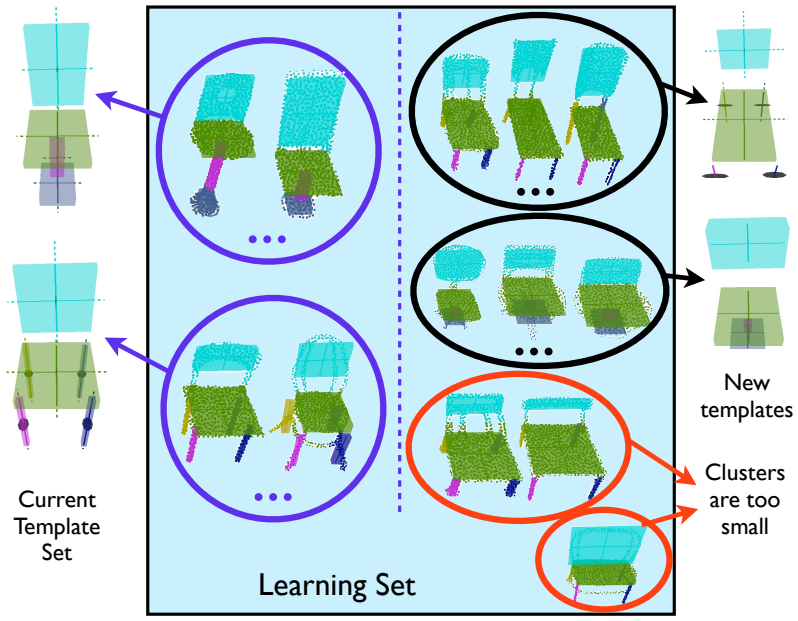

Figure 5.5: ***Template refinement.*** *This figure demonstrates a single iteration of template refinement step. Shapes in the learning set (shaded area) are either used to update existing templates (blue), spawn new templates (black) or do not participate in learning (red).*

and τ could be tuned to encourage more (less) conservative evolution with smaller (larger) values.

Figure 5.5 illustrates the template refinement step. For an example Learning Set (middle) the shapes can either contribute to re-estimating template deformation parameters (blue clusters), spawn new shapes (black clusters), or be disregarded as outliers (red clusters). Note how chairs that are different from the current set of templates (with elongated seats and with short stems) form new clusters potentially expanding the space of geometries represented by the set of templates.

Re-estimating template deformation parameters: Next, we re-estimate the distribution of part deformations for each template based on its fits to examples in the Learning Set S_L.

Specifically, for each template t, we want to update it from all matched shapes that are similar to its mean part arrangement $d_\mu(t) = \{\mu^p(B), \mu^s(B)\}$. We consider the subset S_t of the Learning Set associated with t, and with $e_{\text{data}}(t, s, r, d_\mu(t), m, \ell) \leq e_{\text{min}}$. For each

shape $s \in S_t$, we estimate the template part deformations parameters $d^*(t, s)$ that minimize $E_{\text{data}}(t, s, r, d, m, \ell)$. Then, we build a Gaussian distribution over all $d^*(t, s)$ for all $s \in S_t$ (re-estimating the means and variations of the centroids, scales, and shape descriptors for every part in the template t).

Spawning new templates: Next, we create new templates to represent clusters of deformations that are similar to one another, but poorly modeled by the existing templates. To do so, we perform agglomerative clustering of all shapes that are fitted to templates with the same part cardinalities. We measure dissimilarity for a pair of shapes s_1, s_2 as $E_{\text{data}}(s_1, s_2, r, d, m, \ell) + E_{\text{data}}(s_2, s_1, r, d, m, \ell)$, where deformations and labeling are fixed and m is defined by nearest points with the same label.

We iteratively merge nearest clusters as long as maximal dissimilarity between their elements is below e_{min}. We use clusters with more than $5\% \cdot K_{\text{LS}}$ elements to add new templates to the set with parameters estimated from optimal deformations $d^*(t, s)$.

Rejecting outliers: Finally, remaining shapes are considered to be outliers (deformation-wise) and we exclude them from future learning steps. Note that while these shapes are explained by the current set of templates, we assume that they do not represent sufficient fraction of the collection to expand the template set.

The final result is a new set of templates with re-estimated deformation parameters, and possibly with additional templates describing new shapes and arrangements of parts.

Consider, for example the collection of 36 chairs in Figure 5.1: there are four different clusters, two sets with arms, and two sets without arms. Even among those with arms, the algorithm groups chairs and benches separately based on their corresponding continuous deformation parameters. Our algorithm learns the structure of this collection automatically: starting from an initial template (top left), it learns that the collection has fours different clusters of shapes, each with relatively tight distributions of part positions and sizes, and

accordingly produces four representative deformable templates (second row). The resulting templates can further be used to analyze similar collections.

5.3.4 Template Initialization

We support two main modes for initial template creation: (i) automatic shape segmentations, and (ii) user assisted refinement.

Automatic shape segmentations: Our system starts by creating an initial template based on automatic shape segmentations. We start by segmenting every shape using a modification of our template fitting procedure. More specifically, we use seed with N_{seg} clusters generated by Voronoi diagram of iteratively farthest points. We initialize N_{seg} bounding boxes oriented along PCA directions with anisotropic scales set to one standard deviation. Finally, we use our template fitting optimization to re-label points and find optimal deformation parameters optimizing $E_{\text{data}} + E_{\text{smooth}}$. We observe that in the most cases the template with the smallest fitting energy produces the best results. The user can further pick a template from the set of best-fitted examples. In the fully automatic mode, we execute our template learning procedure initialized from 10 best-fitted initial templates, and then we pick the resulting template with the smallest average fitting score (see example in Section 5.4).

User-refined templates: The user can also refine the initial suggestions made by the automatic shape segmentation. Effectively, using the proposed template(s) as scaffold, the user refines the arrangement of boxes using a simple interactive tool that allows orienting, positioning, and scaling boxes. The process, which is fairly intuitive, typically takes about 5 minutes (e.g., user updates/adds boxes to define an airplane template with fuselage, wings and a tail; or creates a bicycle template with wheels, body, seat and handles). In less obvious datasets, one can look at a few example shapes from the input collection and just align bounding boxes to the semantic parts. For initial sets containing multiple templates (only

chairs in our examples), we started with a single template, investigated outlier shapes in the results (based on the fitting score), and added new arrangement of parts that would cover the outliers. We believe, such a workflow is appropriate for analyzing large collections without relying on any prior knowledge about the dataset.

While our algorithm can run in the automatic mode, we found that the user refinements can be valuable, especially to provide semantic cues. We expect that the users interested in analyzing a collection of shapes are able and willing to spend a couple of minutes to interactively explore proposed initial templates, and/or fixup and select from among a set of templates produced automatically.

5.4 Results

In this section, we evaluate how well our algorithm recovers the shape structure from an unlabeled collection. First, we evaluate the quality of segmentations, deformations, and correspondences, and then visualize variations learned from shape collections containing thousands of models. We further investigate different aspects of our algorithm such as its sensitivity to the initial template, generality of the learned variations, and scalability of the method.

5.4.1 Datasets

We use several datasets in our evaluation: the COSEG dataset [115, 123] containing several classes of smooth manifold shapes with ground truth per-face labeling, the correspondence benchmark introduced in Section 4.4.1 (also in [68]), that includes collections of polygon soups obtained from the 3D Warehouse [119] with consistently selected feature points. Further, to validate applicability of our method to analysis of very large and diverse datasets we created a set of large scale collections containing thousands of models. We obtained this

dataset by crawling the 3D Warehouse for particular keywords[1] and grouped the retrieval results into collections of semantically-similar classes. We further presented all models rendered as a grid of thumbnails to AMT workers [3], and asked them to select images that contain only a single object of interest. We finally pruned geometries that did not receive a majority vote among 5 users. We refer to Table 5.2 for more details on classes of shapes used in our analysis and sizes of the collections.

5.4.2 Correspondence benchmark for large collections

We build a new correspondence benchmark for the newly acquired datasets. Due to the scale of the data, we randomly sample 100 models from each of the collections and select feature points only for those models. Since the collections are very diverse it is hard to define a consistent set of feature points that are present in all models, thus we allow some points to be missing on some models (see Figure 5.6).

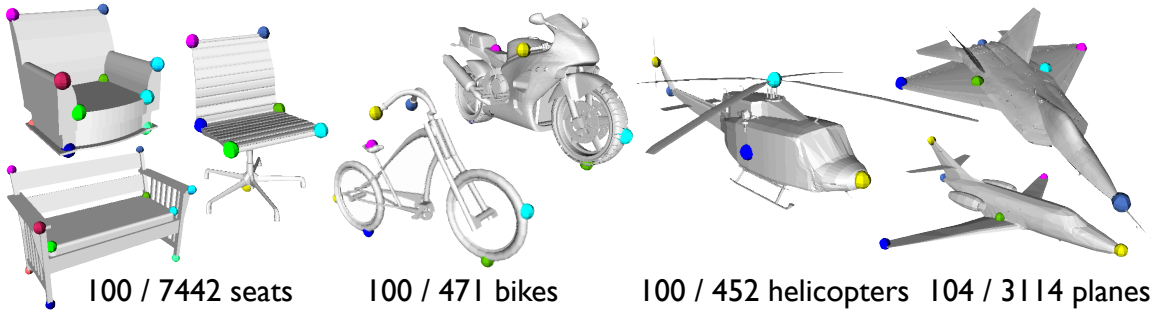

Figure 5.6: ***Ground truth.*** *This image illustrates some example models with ground truth feature points. Note that each model can have only a subset of ground truth points (e.g., arm rests points of chairs are absent in some chair shapes).*

5.4.3 Segmentation accuracy

We evaluate quality of part-wise labeling ℓ produced by our method using the COSEG dataset [115, 123]. Similarly to previous methods we evaluate labeling accuracy, measur-

[1] "chair", "bench", "plane", "airplane", "jet", "bike", "bicycle", "motorcycle", and "helicopter'.'

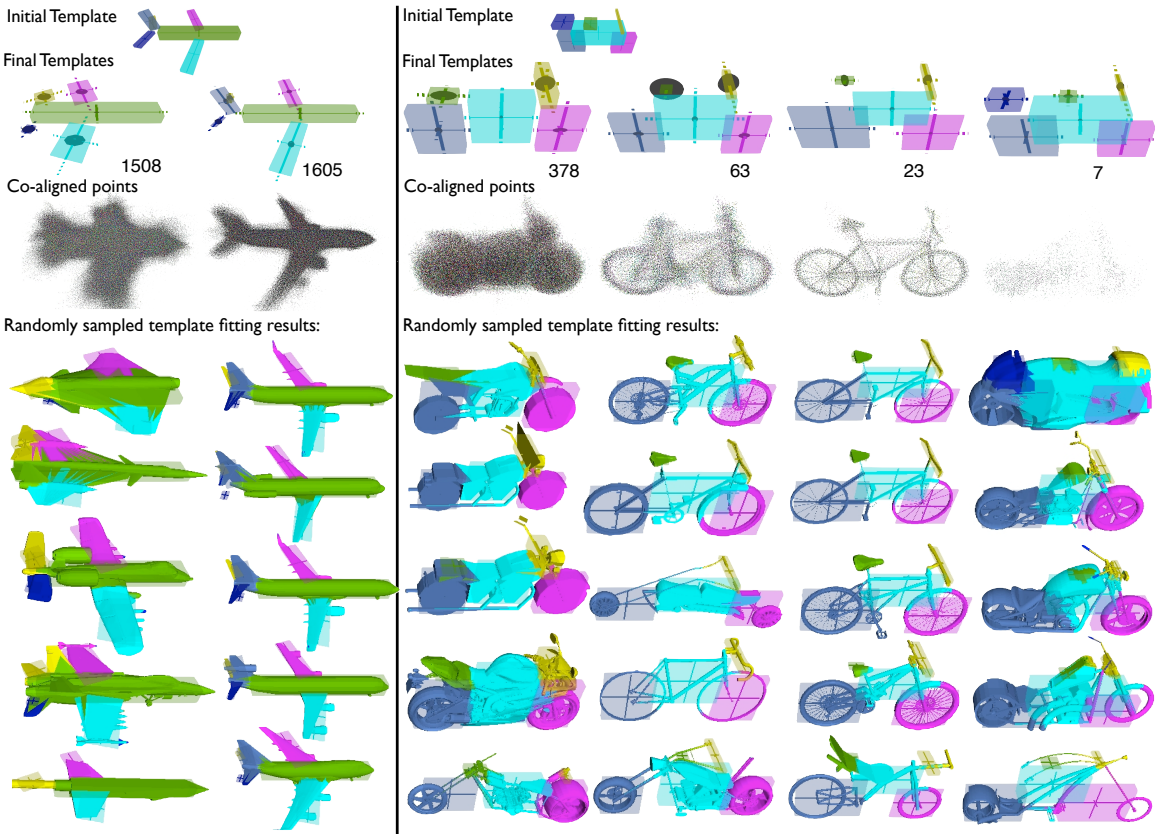

Figure 5.7: ***Trimble 3D Warehouse data.*** *Some example templates learned from 3D Warehouse datasets along with corresponding models. Note how we automatically identify that the airplane dataset has military planes with wider variation in positions of wings (left column) and commercial airplanes with relatively uniform part scales. We also automatically identify the difference between bicycles and motorcycles.*

ing the fraction of faces with correct labels (where labels are assigned manually to the corresponding segments). Since accuracy is measured at a face level, we project our per-point segmentations to mesh faces using a variant of fuzzy cuts algorithm [64], where fuzzy cut boundaries are defined by the labeled points. For non-manifold meshes, we simply assign face label by voting with the nearest labeled points.

Table 5.1 presents labeling accuracy of our method. We test two versions of template initialization described in Section 5.3.4: fully automatic (Auto) and manually-refined (Man). In both cases, we also include labeling results that were obtained prior to template learning procedure (Init). Note that for simple classes of shapes initial template is sufficient to understand part-wise decomposition (e.g., small dataset of chairs). However, shapes with

more geometric diversity (e.g., four-legged creatures) demonstrate significant improvement after learning deformations.

Class	Sidi	Hu	Auto init	Auto result	Man. init	Man. result
Lamps	94.3	90.7	95.1	95.2	81.8	97.6
Chairs	84.8	89.6	96.7	97.6	97.9	98.4
Vase	87.4	80.2	80.7	81.3	81.7	83.2
FourLegged	77.3	88.7	81.6	86.9	84.6	87.1
Guitars	87.2	98.0	86.6	88.5	95.9	96.5
Goblets	98.2	99.2	89.4	97.6	98.4	98.1
Candelabra	84.4	93.9	82.9	82.4	85.7	87.9
Chairs (400)	XX	XX	80.4	91.2	91.3	96.9
Vase (300)	XX	XX	85.7	85.6	85.9	81.2

Table 5.1: ***Segmentation accuracy***. *Each entry records fraction of the area that was labeled correctly by a segmentation technique, where rows correspond to datasets and columns correspond to methods. We compare to the consistent segmentation techniques by Sidi et al. [2011] and Hu et al. [2012]. We also execute our method with two types of initial templates: fully automatic (Auto) and manually-refined (Man). In both cases, we show results of using just the initial template for matching (init) and results after we execute our template learning procedure (result).*

We further include results reported by the previous methods of [115] (Sidi) and [56] (Hu). The quality of our labels is comparable to the previous techniques, and we provide improvement for classes where spacial arrangement of parts is important (e.g., lamps, chairs). Our method demonstrates a slightly inferior performance on classes that do not have well-defined parts (e.g., vases). Another common problem that we encountered with automatically-generated templates is that the final templates might not have the same granularity as the ground truth parts (e.g., the neck and head of a guitar are commonly put into single part, and the body is segmented into two pieces). Despite lower labeling accuracy in these cases, the co-alignment and learned variations for these classes is still meaningful. Note that unlike previous techniques, our method does not require manifold meshes as input and our results are disadvantaged by this generality.

Figure 5.8 demonstrates our segmentation results for a set of lamps, note how we automatically identify three types of deformable templates and successfully label most models

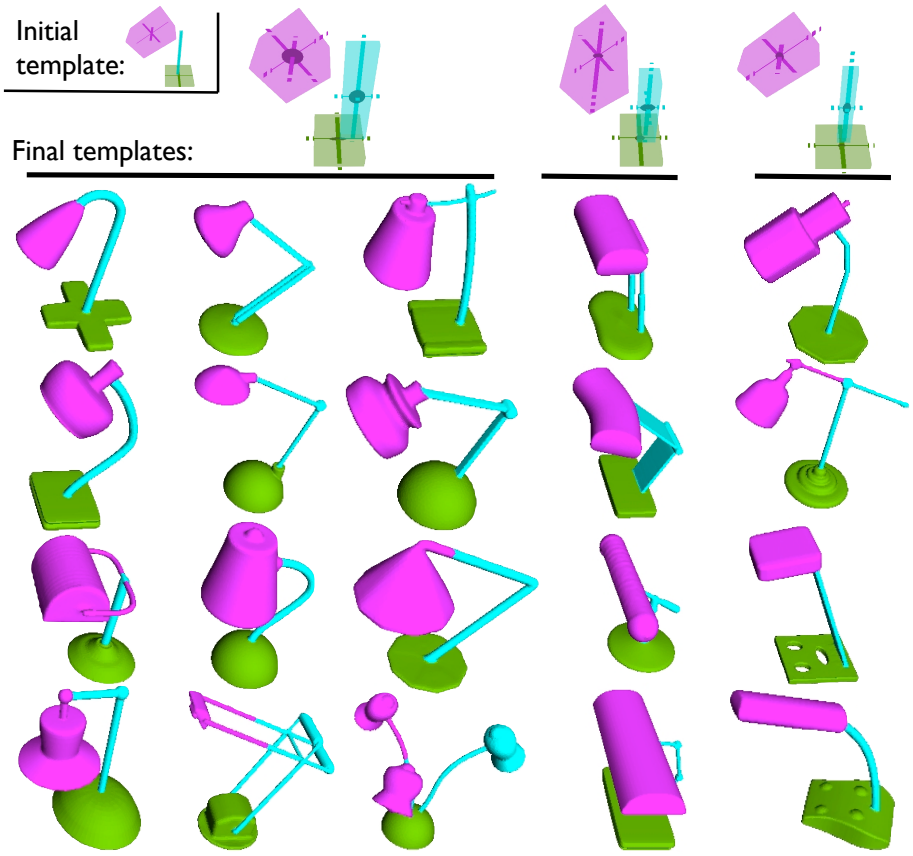

Figure 5.8: ***Segmentation example.*** *This example illustrates analysis of a collection of 20 lamps. Note how resulting templates capture variations in position and size of lamps' head.*

in the collection. Figure 5.7 demonstrates randomly selected segmentations and template alignments from much larger datasets containing polygon soups. Note how our method is effective even for these non-manifold surfaces.

5.4.4 Deformation quality

We visually verify the quality of per-model template alignment and deformation parameters r, d, by rendering deformed boxes aligned to the matched models (e.g., see bottom of Figure 5.7). We also use the template deformation to co-align all models to a canonical space, thus relating all the shapes. More specifically, for every shape s associated with template t with deformation parameters d_t, we align every point p_s to corresponding point in the canonical

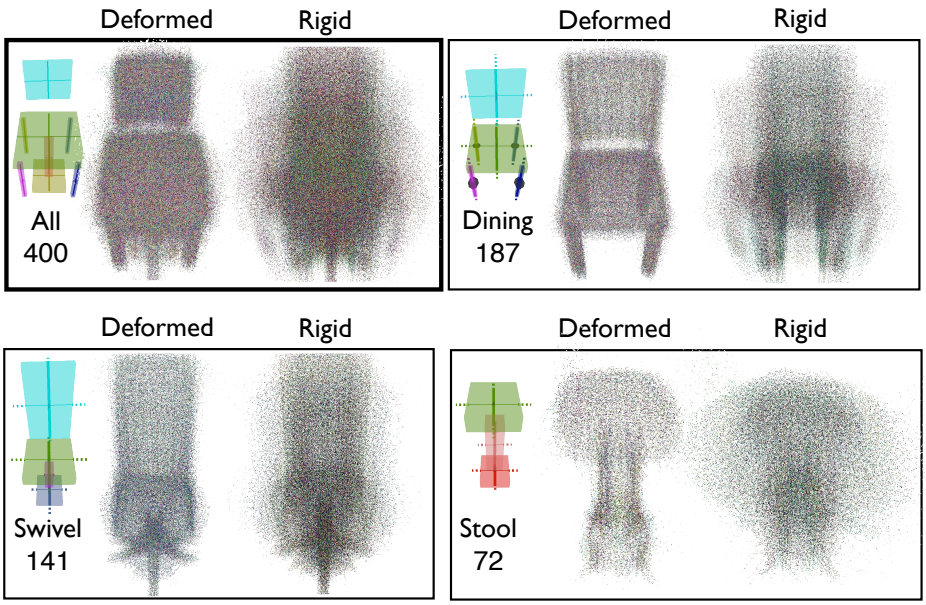

Figure 5.9: ***Co-aligned shapes.*** *We use template deformation parameters (r, d) to align all shapes from the set of 400 chairs in COSEG dataset to their corresponding mean part arrangements and scales (left image). And compare it to just using rigid transformations r to co-align shapes (right image). Note that our deformation model leads much sharper co-alignments by factoring out the dominant deformation modes.*

space $m(p_s) + o_s$, where o_s is the offset by which point is displaced from a deformed template: $o_s = d(m(p_s)) - p_s$.

Figure 5.9 demonstrates co-aligned points from *all* the models side by side with rigidly aligned points that do not account for part-wise deformations. Note that our result is much sharper than rigidly aligned models significantly reducing the variance in point distributions, demonstrating that anisotropic part scales account for a very significant amount of geometric variations (see also supplementary material).

5.4.5 Correspondence accuracy

We evaluate the quality of the resulting map between models m using standard correspondence benchmarks. To provide pairwise correspondences required for the benchmarks, we co-align all the shapes to a canonical domain and use nearest neighbors to define shape-to-

shape correspondences. We evaluate our results on the datasets proposed in Kim et al. [68], and our newly created benchmark for larger and more diverse collections. Similarly to previous methods we compute correspondences for selected feature points and measure Euclidean error between predicted point and ground truth, normalized by mesh diameter (maximal distance between a pair of points). We further aggregate this information into plots that show fraction of correspondences predicted correctly (y-axis) for a given Euclidean error threshold (x-axis).

Figure 5.10 demonstrates these results on all benchmarks used in our evaluation. We demonstrate results for our method trained on all data (red), our method only trained on models that have ground truth correspondences (magenta). We further compare the method proposed by Huang et al. [58] trained on ground truth models (black), and on the largest dataset it could handle (gray): it successfully analyzed all bikes, a subset of 1000 planes, a subset of 1000 seats, and crashed for larger subsets of helicopter dataset. We also execute the method of Kim et al. [68] (blue), which is not able to handle collections much larger than one hundred models.

Note that the performance of our method is comparable on datasets with less part-wise variations (e.g., chairs, commercial planes), while our method performs significantly better on datasets with part-wise deformations (e.g., seats have significant deformation between chairs and benches). Both previous techniques rely on rigid alignment to compute correspondences for pairs of models and use transitivity to recover larger variations. These approaches are sensitive to multi-modal variations in collections that contain clusters that are not connected by a smooth path of rigid alignments (e.g., there is a deformation gap between chairs and benches in the seat dataset). Our method improves pairwise correspondences (w.r.t. to the template) by modeling part-wise deformations, and thus is more capable of bridging such a gap by extrapolating deformation parameters. We also observe that modeling deformations allows more accurate correspondences at finer scales (i.e., the

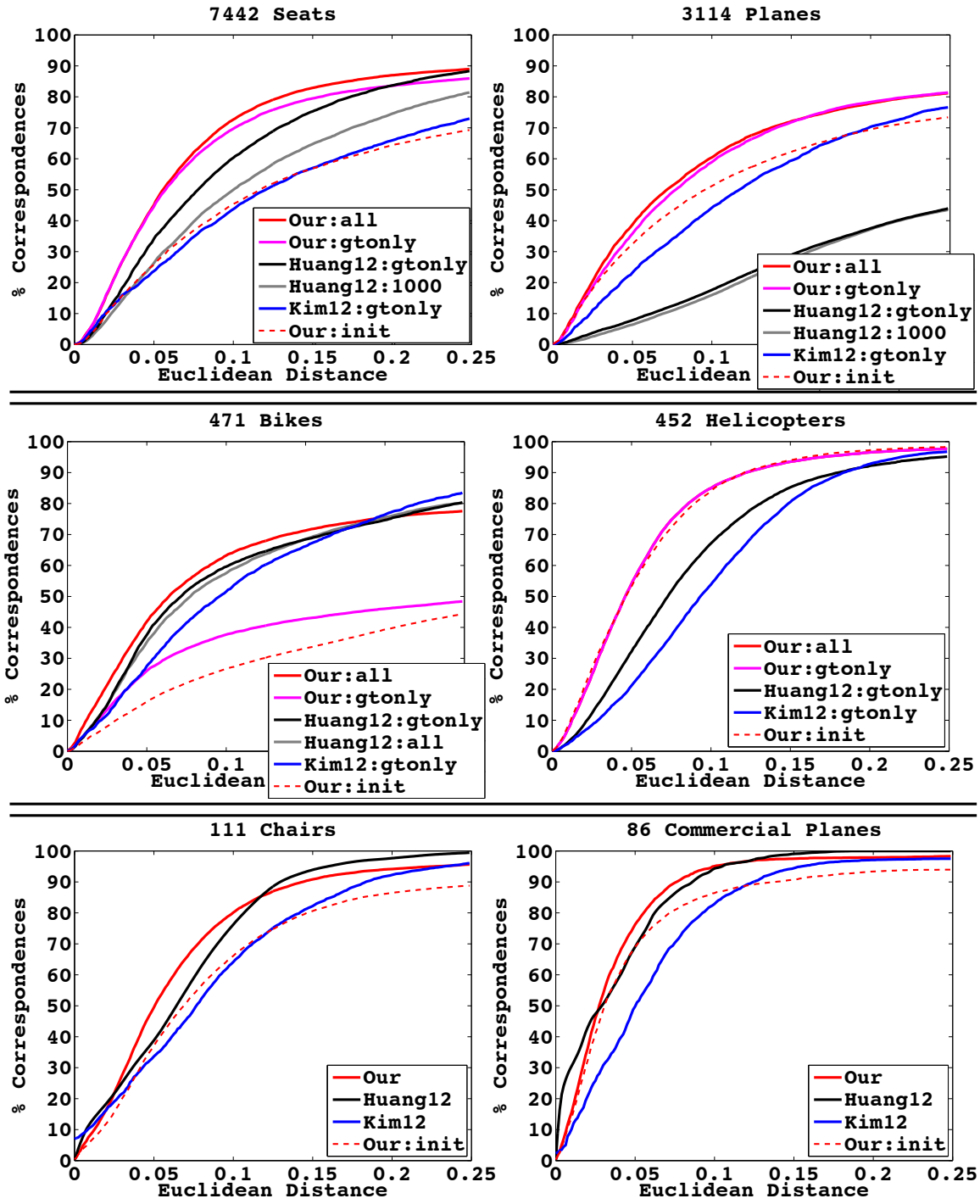

Figure 5.10: ***Correspondence benchmark.*** *This figure demonstrates quality of correspondences produced by our method (red,magenta) relative to prior work: Huang et al. [58] (black,gray), Kim et al. [68] (blue).*

left side of the curve). Note that the most common source of error for bikes and planes is near-symmetry under 180° rotation, learning local shape features allow distinguishing between front and back parts. Unfortunately with too little data our method can learn incorrect variations in the first few iterations, as it happens for bikes (magenta).

While compute times for our method are comparable to previous techniques on small datasets (e.g., 5-30 minutes for our method on Kim et al. [68] dataset in comparison to about 10 minutes for Huang et al. [58]), our method offers significant improvements for larger datasets (e.g., a random subset of 1000 chairs was analyzed by our method in 2.6 hrs in contrast to Huang et al. [58] which took 6 hrs.). Since the amortized efficiency of our algorithm improves with increasing data volumes, we expect this gap to widen further as the shape collections grow in size.

We also show our results obtained using the initial template (red dotted line). Note how results improve significantly after learning since quality of correspondences greatly benefits from understanding part-wise deformations.

5.4.6 Robustness to template initialization

Figure 5.11 shows results initialized from different seed templates. Note that while initial templates are very diverse and commonly do not match well to the rest of the collection, the final templates are very similar in learned variations, labeling accuracy and final co-alignments of all points.

5.4.7 Generality of learned parameters

We further validate whether the template that we learn can be generalized to new data. We select random subsets of shapes from 7442 chairs dataset, such that smaller sets are always included in larger sets, and learn a set of templates using the subset. We further

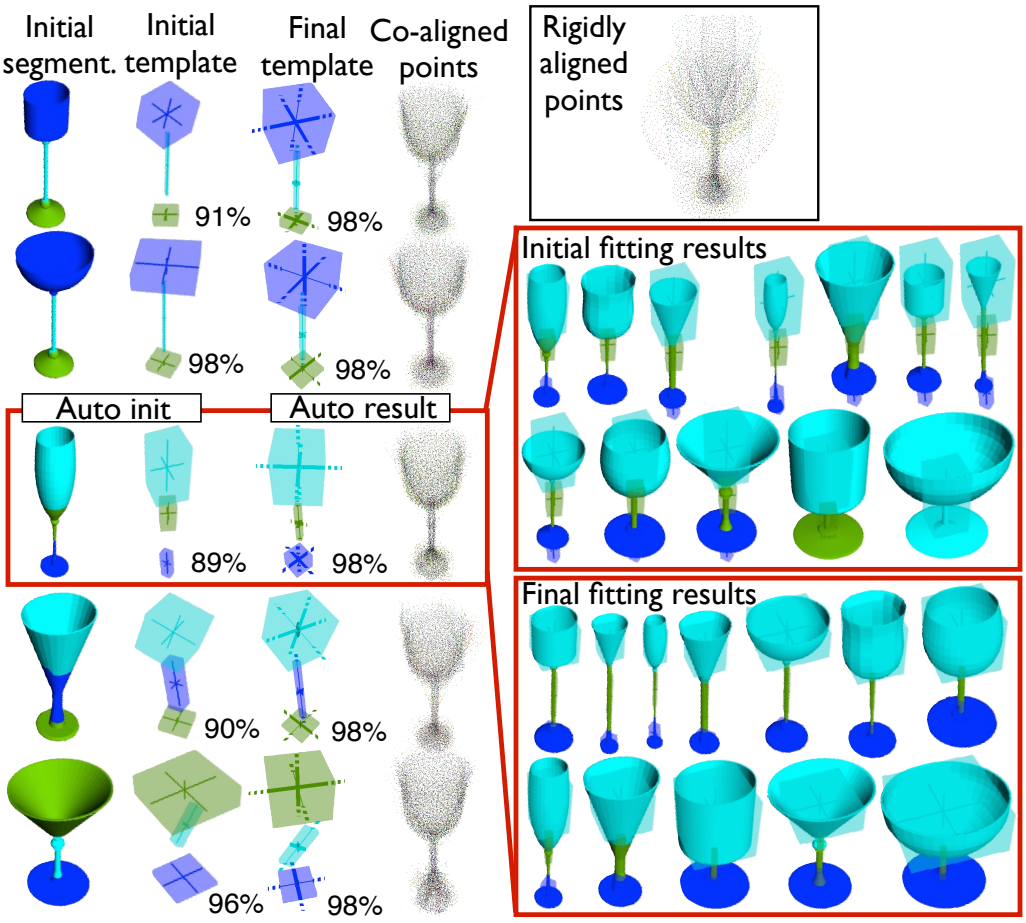

Figure 5.11: ***Robustness to initial template.*** *Each row corresponds to six different automatic segmentations that produce different initial templates. The percentages under each template indicate labeling accuracy of the whole dataset of goblets. Our algorithm further automatically learns parameters for each initial template resulting in different final templates. Note that all initial templates converge to very similar final template parameters, labeling accuracies and co-alignments regardless of the initial quality. The inset further illustrates example segmentations produced with the automatically-picked template with the minimal average fitting score.*

use the learned set of templates to establish correspondences among a different set of 100 models (none of these models were part of the training set). The accuracies of resulting correspondences are presented in Figure 5.12 (left plot, dark green curve). Note that the accuracy increases as size of the dataset grows, and almost achieves the quality of correspondences learned from all 7000 models (red horizontal line) after training on just a 1000. This suggests that the learned templates can be efficiently used to analyze new data.

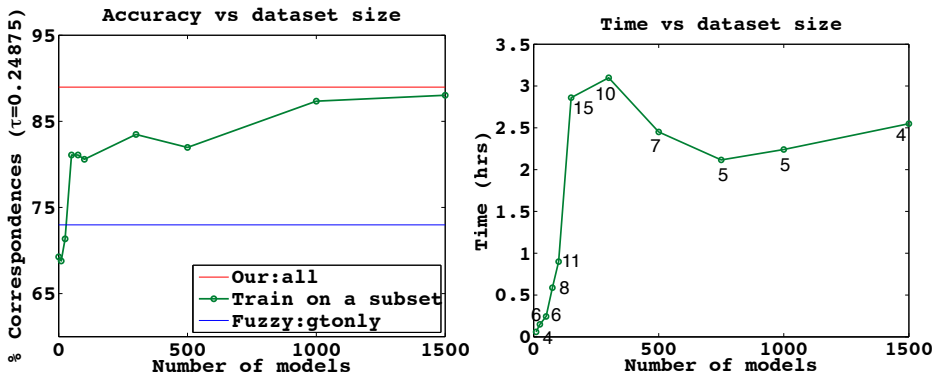

Figure 5.12: ***Learning templates from a subset of models.*** *We learned a set of templates on subsets of different size of 7442 seats dataset. The left plot shows how size of the subset influences fraction of correct correspondences for a fixed Euclidean error threshold. The right plot shows the time require for learning the variations (does not include the labeling time), and numbers next to data points indicate number of final templates. These results suggest that variations can be learned in near constant time from a randomly selected subset of all the shapes.*

5.4.8 Scalability and timing

Finally, we discuss scalability and computational complexity of our algorithm. Let us define an input collection of N models that can be described by T_{max} templates. Note that every model is added to a Learning Set exactly once, moreover at least one model is included in a Learning Set at every iteration (our algorithm terminates when $|S_{\text{L}}| = 0$). Thus, our algorithm executes for at most N iterations, where each iteration involves re-fitting models in Fitting Set, $O(K_{\text{L}}T_{\text{max}})$ and, possibly, an agglomerative clustering $O(K_{\text{L}}^2)$. Thus, our method is linear in the number of models N as long as number of templates that describe all geometric variations T_{max} does not depend on the collection size. Finally, fitting any new collection of N' models to existing set of templates is $O(T_{\text{max}} \cdot N')$. Note that except for agglomerative clustering, all the other steps can be performed in parallel.

The right plot in Figure 5.12 demonstrates that number of templates T_{max} and the time required to compute them do not depend on the number of models in the collection (note that the curve flattens out because our procedure terminates after reaching the maximal

Class	N	$\lvert T_{\text{init}}\rvert$	Total t_{learn} sec	Ave. t_{label} sec
Trimble 3D Warehouse (polygonal soups)				
Chairs	111	2	1.8K	22
CommPlane	86	1	344	4
Seats	7442	2	38K	32
Planes	3114	1	12K	7
Helicopters	471	1	15K	18
Bikes	452	1	5.4K	11
Co Seg (manifold shapes)				
FourLegged	20	1	95	4.6
Lamps	20	1	145	8.6
Candelabra	20	1	101	8.4
Guitars	44	1	219	7.7
Chairs	20	2	95	6.4
Goblets	12	1	39	1.4
Vase	28	1	130	1.2
Chairs (400)	400	3	14K	37
Vase (300)	300	1	3K	8.4

Table 5.2: ***Data statistics***. *This table provides number of models N, number of initial templates used in analysis of a collection T_{init}, total learning time t_{learn} and* average *per-shape fitting time with the final set of templates. The last three columns correspond to experiments with manually-refined initial templates.*

number of iterations). Table 5.2 further includes compute times and statistics for all the datasets.

5.4.9 Limitations

The current algorithm has several limitations that require further investigation. First, the shape of each part is currently represented by an oriented box, and correspondences are assigned based on closest points. While this make our method well suited for "boxy" parts that protrude away from the rest of the shape (e.g., airplane wings, chair legs, etc.), the method does not fare so well in presence of parts with complex shapes in close proximity to others (e.g., bike frames). It would be interesting to investigate the accuracy/speed trade-offs of alternative shape representations (e.g., a set of meshes for each part). Second,

our template does not explicitly model relative spatial relationships between parts, which is sufficient for many man-made objects where parts often appear in consistent global arrangements, but would not be as good for articulated shapes. A constellation model is an alternative [34], which might provide better results for some object classes at greater computational expense. Third, our template learning procedure requires initial template that includes all possible parts. Since our automatic template initialization procedure only creates templates from a single segmentation, it is not suitable for diverse collections where there might be no shape that includes all parts. Our current solution is to ask the user to refine the initial template and, possibly, add more parts to the initial template, but we would like to investigate fully automatic alternatives in the future. Fourth, our method is greedy, and thus is not guaranteed to converge to an optimal set of templates.

Chapter 6

Conclusion and Future Work

This dissertation offers several tools for finding structure in large collections of diverse 3D shapes. These tools can handle a larger space of geometric variations and allow for substantially bigger datasets in comparison to previous techniques.

6.1 Summary

In this dissertation we described several algorithms to derive structure in diverse and unorganized 3D collections.

First, we created a method for finding a map between pairs of surfaces by blending a collection of low dimensional maps. This method blends near-isometric maps into the final blended map, which can significantly deviate from a perfect isometry. Our approach is fully automatic, efficient, and it provides robust mapping between non-isometric surfaces. In practice, this method is suitable for discovering geometric similarities between pairs of shapes that undergo non-rigid and non-uniform deformations, such as articulation and different scaling of parts. We evaluated our method on several benchmarks, demonstrating significant improvement over previous matching techniques.

Second, we developed fuzzy correspondences, a fully automatic computational tool to find and encode semantic relationships between points in a diverse collection of 3D models. Our framework relies on the transitivity of correspondences to improve the quality of pairwise alignments between shapes related by non-uniform deformations (including different geometries and multiplicities of parts). Our representation encodes inherent ambiguity in matching very dissimilar shapes with a continuous fuzzy correspondence function. We evaluated our computational framework on several benchmarks, showing that it is both efficient, in that it generates all correspondences by matching only a subset of pairs of models, and accurate, in that it, improves the semantic alignment of points in comparison to existing mapping methods.

Third, we developed a shape analysis tool to derive structure from large, unorganized, diverse collections of 3D polygonal models. Given a collection of 3D shapes (and optionally an initial template), our method jointly partitions the shapes into clusters with similar structure, learns a part-based deformable model of the shape variations within each cluster, provides consistent segmentations for all the shapes with similar parts, and provides correspondences between semantically equivalent points across all shapes in the collection. Our algorithm executes out-of-core, has time complexity that is linear in the size of the collection, and thus can handle very large datasets. It also performs favorably on benchmark datasets for consistent segmentation and surface correspondence in comparison to previous work.

6.2 Future Work

We identify several research challenges for future work on analysis of geometric structure.

One direction is to enhance the methods that treat high-level understanding of the data and structure computation as two inter-related problems. Explicitly modeling and representing

a larger space of *geometric variations* and *functionality of a shape* can substantially simplify and improve the structure detection. For example, we can augment our deformable templates from Chapter 5 with other types of part relationships, such as hierarchical decompositions, context information, etc. Representing the space of valid shapes in a concise and generalizable form significantly simplifies the model-to-collection mapping problem by restricting the search space. Furthermore, encoding and discovering the functionalities of parts can improve the robustness of such an approach by ensuring that the geometry of each part is consistent with its functional purpose.

Another interesting research direction is to investigate new tools enabled by our analysis, which can be used for applications such as the exploration of geometric collections, data-driven shape modeling, and shape reconstruction. Furthermore, discovering structure in scans of large-scale indoor and outdoor environments can find many applications in robotics, computer vision and computer graphics.

The long-term goal of this research is to represent and organize the world's geometric data using high-level analysis tools. Our current research to discover correspondences, segmentations and geometric variations is a first step towards that goal. We plan to further develop data-driven techniques for recovering and generalizing geometric structures in classes of shapes, ranging from everyday objects to scientific datasets. Ultimately, all digitized shapes can be organized in a single repository that captures low-level geometric details as well as high-level structural understanding of every class of shapes. This representation should include information about the decomposition of shapes into semantically meaningful regions, functional and geometric relationships between these regions, their hierarchical categorizations, and more. Such a repository can be made accessible and useful for scientists, experts and the general public with appropriate high-level tools for manipulation, exploration, understanding and detection of interesting functional relationships between shapes. Efficiently mapping these data to geometry acquired in real time

would also enable reasoning about shapes in a dynamic setting and the discovery of many exciting applications.

Bibliography

[1] Marc Alexa. Recent advances in mesh morphing. *Computer Graphics Forum*, 21(2):173–198, 2002.

[2] Brett Allen, Brian Curless, and Zoran Popović. The space of human body shapes: reconstruction and parameterization from range scans. In *ACM SIGGRAPH*, pages 587–594, 2003.

[3] Amazon. Amazon mechanical turk, https://www.mturk.com/ 2012.

[4] Y. Amit and A. Trouve. POP: patchwork of parts models for object recognition. *IJCV*, 75(2):267–282, 2007.

[5] D. Anguelov, P. Srinivasan, D. Koller, S. Thrun, H. Pang, and J. Davis. The correlated correspondence algorithm for unsupervised registration of nonrigid surfaces. *Proc. NIPS*, 2004.

[6] Dragomir Anguelov, Praveen Srinivasan, Daphne Koller, Sebastian Thrun, Jim Rodgers, and James Davis. Scape: shape completion and animation of people. In *ACM SIGGRAPH*, pages 408–416, 2005.

[7] M. Ankerst, G. Kastenmuller, H.-P. Kriegel, and T. Seidl. Nearest neighbor classification in 3d protein databases. *ISMB*, 1999.

[8] Marco Attene, Bianca Falcidieno, and Michela Spagnuolo. Hierarchical mesh segmentation based on fitting primitives. *THE VISUAL COMPUTER*, 22:181–193, 2006.

[9] Paul J. Besl and Neil D. McKay. A method for registration of 3-d shapes. *IEEE PAMI*, 14(2):239–256, February 1992.

[10] V. Blanz, K. Scherbaum, and H.-P. Seidel. Fitting a morphable model to 3d scans of faces. *ICCV*, 2007.

[11] Volker Blanz and Thomas Vetter. A morphable model for the synthesis of 3d faces. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '99, pages 187–194, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co.

[12] D. Blei. Probabilistic topic models. *Communications of the ACM*, 55(4):77–84, 2012.

[13] D. Blei and J. Lafferty. Topic models. *Text Mining: Classification, Clustering, and Applications*, 2009.

[14] Doug M. Boyer, Yaron Lipman, Elizabeth St. Clair, Jesus Puente, Biren A. Patel, Thomas Funkhouser, Jukka Jernvall, and Ingrid Daubechies. Algorithms to automatically quantify the geometric similarity of anatomical surfaces. *PNAS*, 108(45), 2011.

[15] Yuri Boykov, Olga Veksler, and Ramin Zabih. Efficient approximate energy minimization via graph cuts. *IEEE transactions on PAMI*, 20(12):1222–1239, 2001.

[16] A. M. Bronstein, M. M. Bronstein, and R. Kimmel. *Numerical geometry of non-rigid shapes*. Springer, 2008.

[17] Alexander M. Bronstein, Michael M. Bronstein, Leonidas J. Guibas, and Maks Ovsjanikov. Shape google: Geometric words and expressions for invariant shape retrieval. *ACM Trans. Graph.*, 30(1):1:1–1:20, February 2011.

[18] Alexander M. Bronstein, Michael M. Bronstein, and Ron Kimmel. Generalized multidimensional scaling: A framework for isometry-invariant partial surface matching. *PNAS*, 103(5):1168–1172, 2006.

[19] Benedict J. Brown and Szymon Rusinkiewicz. Global non-rigid alignment of 3-d scans. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 2007.

[20] N. Canterakis. 3d zernike moments and zernike affine invariants for 3d image analysis and recognition. *In 11th Scandinavian Conf. on Image Analysis*, 1999.

[21] L. Cao and L. Fei-Fei. Spatially coherent latent topic model for concurrent object segmentation and classification. In *Proceedings of IEEE Intern. Conf. in Computer Vision (ICCV).*, 2007.

[22] U. Castellani, M. Cristani, S. Fantoni, and V. Murino. Sparse points matching by combining 3d mesh saliency with statistical descriptors. *Computer Graphics Forum*, 27(2):643–652, 2008.

[23] S. Chaudhuri, E. Kalogerakis, L. Guibas, and V. Koltun. Probabilistic reasoning for assembly-based 3D modeling. *ACM SIGGRAPH*, pages 35:1–35:10, 2011.

[24] D.-Y. Chen, M. Ouhyoung, X.-P. Tian, and Y.-T. Shen. On visual similarity based 3d model retrieval. *CGF*, (223–232), 2003.

[25] Xiaobai Chen, Aleksey Golovinskiy, and Thomas Funkhouser. A benchmark for 3D mesh segmentation. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 28(3), August 2009.

[26] Xiaobai Chen, Abulhair Saparov, Bill Pang, and Thomas Funkhouser. Schelling points on 3d surface meshes. *ACM Trans. Graph.*, 31(4):29:1–29:12, July 2012.

[27] Haili Chui and Anand Rangarajan. A new point matching algorithm for non-rigid registration. *Comput. Vis. Image Underst.*, 89(2-3):114–141, 2003.

[28] M. Eitz, R. Richter, T. Boubekeur, K. Hildebrand, and M. Alexa. Sketch-based shape retrieval. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 31(10):1–31, 2012.

[29] Y. Eldar, M. Lindenbaum, M. Porat, and Y. Zeevi. The farthest point strategy for progressive image sampling. *IJCV*, 40(2):99–121, 1997.

[30] S. M. Ali Eslami and C. Williams. A generative model for parts-based object segmentation. In *NIPS*, 2012.

[31] L. Fei-Fei and L.-J. Li. What, Where and Who? Telling the Story of an Image by Activity Classification, Scene Recognition and Object Categorization. *Studies in Computational Intelligence- Computer Vision*, pages 157–171, 2010.

[32] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Pictorial structures for object recognition. *IJCV*, 61(1):55–79, 2005.

[33] P.F. Felzenszwalb, R.B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *IEEE PAMI*, 32(9):1627 –1645, sept. 2010.

[34] R. Fergus, P. Perona, and A. Zisserman. Object class recognition by unsupervised scale-invariant learning. In *IEEE CVPR*, 2003.

[35] Matthew Fisher, Daniel Ritchie, Manolis Savva, Thomas Funkhouser, and Pat Hanrahan. Example-based synthesis of 3d object arrangements. In *ACM SIGGRAPH Asia 2012 papers*, SIGGRAPH Asia '12, 2012.

[36] Matthew Fisher, Manolis Savva, and Pat Hanrahan. Characterizing structural relationships in scenes using graph kernels. *ACM SIGGRAPH*, 30:34:1–34:12, 2011.

[37] Thomas Funkhouser, Michael Kazhdan, Philip Shilane, Patrick Min, William Kiefer, Ayellet Tal, Szymon Rusinkiewicz, and David Dobkin. Modeling by example. *ACM SIGGRAPH*, pages 652–663, 2004.

[38] Thomas Funkhouser and Philip Shilane. Partial matching of 3d shapes with priority-driven search. *Symp. on Geom. Processing*, 2006.

[39] Ran Gal and Daniel Cohen-Or. Salient geometric features for partial shape matching and similarity. *ACM Trans. Graph.*, 25(1):130–150, January 2006.

[40] Michael Garland, Andrew Willmott, and Paul S. Heckbert. Hierarchical face clustering on polygonal surfaces. *ACM Symposium on Interactive 3D Graphics*, 2001.

[41] Natasha Gelfand, Niloy J. Mitra, Leonidas Guibas, and Helmut Pottmann. Robust global registration. *Symp. on Geom. Processing*, 2005.

[42] Deboshmita Ghosh, Andrei Sharf, and Nina Amenta. Feature-driven deformation for dense correspondence. *Proc. SPIE 7261*, 2009.

[43] D. Giorgi, S. Biasotti, and L. Paraboschi. Shrec:shape retrieval contest: Watertight models track. *http://watertight.ge.imati.cnr.it/*, 2007.

[44] D. Giorgi, P. Frosini, M. Spagnuolo, and B. Falcidieno. 3D relevance feedback via multilevel relevance judgements. *Vis. Comput.*, 26(10):1321–1338, 2010.

[45] A. Golovinskiy, V.G. Kim, and T. Funkhouser. Shape-based recognition of 3d point clouds in urban environments. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 2154 –2161, 29 2009-oct. 2 2009.

[46] A. Golovinskiy, W. Matusik, S. Rusinkiewicz H. Pfister, , and T. Funkhouser. A statistical model for synthesis of detailed facial geometry. *SIGGRAPH*, 2006.

[47] Aleksey Golovinskiy and Thomas Funkhouser. Randomized cuts for 3D mesh analysis. *ACM Transactions on Graphics (Proc. SIGGRAPH ASIA)*, 2008.

[48] Aleksey Golovinskiy and Thomas Funkhouser. Consistent segmentation of 3D models. *Proc. SMI*, 33(3):262–269, 2009.

[49] Google. Google 3D warehouse, http://sketchup.google.com/3dwarehouse/ 2011.

[50] C. Gu and X. Ren. Discriminative mixture-of-templates for viewpoint classification. In *ECCV*, 2010.

[51] K. Heath, N. Gelfand, M. Ovsjanikov, M. Aanjaneya, and L. J. Guibas. Image webs: Computing and exploiting connectivity in image collections. In *IEEE CVPR*, 2010.

[52] Masayuki Hisada, Alexander G. Belyaev, and Tosiyasu L. Kunii. A skeleton-based approach for detection of perceptually salient features on polygonal surfaces. *Computer Graphics Forum*, 21(4):689–700, 2002.

[53] Donald D Hoffman and Manish Singh. Salience of visual parts. *Cognition*, 63(1):29 – 78, 1997.

[54] Matthew Hoffman, David M. Blei, and Perry R. Cook. Content-based musical similarity computation using the hierarchical dirichlet process. In Juan Pablo Bello, Elaine Chew, and Douglas Turnbull, editors, *ISMIR 2008, 9th International Conference on Music Information Retrieval, Drexel University, Philadelphia, PA, USA, September 14-18, 2008*, pages 349–354, 2008.

[55] B. Horn. Extended gaussian images. *In Proceedings of the IEEE*, 72(1656–1678), 1984.

[56] R. Hu, L. Fan, , and L. Liu. Co-segmentation of 3d shapes via subspace clustering. *Computer Graphics Forum (Proc. SGP)*, 31(5):1703–1713, 2012.

[57] Qi-Xing Huang, Bart Adams, Martin Wiche, and Leonidas J. Guibas. Non-rigid registration under isometric deformations. *SGP*, 2008.

[58] Qi-xing Huang, Guo-Xin Zhang, Lin Gao, Shi-Min Hu, Adrian Butscher, and Leonidas Guibas. An optimization approach for extracting and encoding consistent maps. *SIGGRAPH Asia*, 2012.

[59] Qixing Huang, Vladlen Koltun, and Leonidas Guibas. Joint shape segmentation with linear programming. In *ACM SIGGRAPH Asia*, pages 125:1–125:12, 2011.

[60] Anil Jain, Yu Zhong, and Marie-Pierre Dubuisson-Jolly. Deformable template models: A review. *Signal Processing*, 71(2):109 – 129, 1998.

[61] Evangelos Kalogerakis, Siddhartha Chaudhuri, Daphne Koller, and Vladlen Koltun. A probabilistic model for component-based shape synthesis. *SIGGRAPH*, 2012.

[62] Evangelos Kalogerakis, Aaron Hertzmann, and Karan Singh. Learning 3D mesh segmentation and labeling. In *ACM SIGGRAPH*, pages 102:1–102:12, 2010.

[63] S. Kang and K. Ikeuchi. Determining 3-d object pose using the complex extended guassian image. *CVPR*, pages 580–585, 1991.

[64] Sagi Katz and Ayellet Tal. Hierarchical mesh decomposition using fuzzy clustering and cuts. *ACM Trans. Graph.*, 22(3):954–961, 2003.

[65] M. Kazhdan. Thesis. *Princeton University*, 2004.

[66] M. Kazhdan, T. Funkhouser, and S. Rusinkiewicz. Symmetry descriptors and 3d shape matching. *SGP*, pages 116–125, 2004.

[67] Michael Kazhdan, Thomas Funkhouser, and Szymon Rusinkiewicz. Rotation invariant spherical harmonic representation of 3D shape descriptors. In *Proc. SGP*, pages 156–164, 2003.

[68] Vladimir G. Kim, Wilmot Li, Niloy Mitra, Stephen DiVerdi, and Thomas Funkhouser. Exploring collections of 3D models using fuzzy correspondences. *Trans. on Graphis (Proc. of SIGGRAPH)*, 2012.

[69] Vladimir G. Kim, Wilmot Li, Niloy J. Mitra, Siddhartha Chaudhuri, Stephen DiVerdi, and Thomas Funkhouser. Learning part-based templates from large collections of 3d shapes. *Trans. on Graphics (Proc. of SIGGRAPH 2013)*, to appear, 2013.

[70] Vladimir G. Kim, Yaron Lipman, Xiaobai Chen, and Thomas Funkhouser. Mobius transformations for global intrinsic symmetry analysis. *Computer Graphics Forum (Proc. of SGP)*, 2010.

[71] Vladimir G. Kim, Yaron Lipman, and Thomas Funkhouser. Blended intrinsic maps. In *ACM SIGGRAPH*, pages 79:1–79:12, 2011.

[72] Vladimir G. Kim, Yaron Lipman, and Thomas Funkhouser. Symmetry-guided texture synthesis and manipulation. *ACM Trans. Graph.*, 31(3):22:1–22:14, June 2012.

[73] Young Min Kim, Niloy J. Mitra, Dongming Yan, and Leonidas Guibas. Acquiring 3d indoor environments with variability and repetition. *SIGGRAPH Asia*, 2012.

[74] J. Knopp, M. Prasad, G. Willems, R. Timofte, and L. Van Gool. Hough transform and 3d surf for robust three dimensional classification. In *Proceedings of the European Conference on Computer Vision*, 2010.

[75] A. Kovnatsky, M. M. Bronstein, A. M. Bronstein, K. Glashoff, and R. Kimmel. Coupled quasi-harmonic bases. *eg*, 32(2), 2013.

[76] Vladislav Kraevoy and Alla Sheffer. Cross-parameterization and compatible remeshing of 3D models. In *ACM SIGGRAPH*, pages 861–869, 2004.

[77] Chang Ha Lee, Amitabh Varshney, and David W. Jacobs. Mesh saliency. *ACM Trans. Graph.*, 24(3):659–666, July 2005.

[78] Hao Li, Robert W. Sumner, and Mark Pauly. Global correspondence optimization for non-rigid registration of depth scans. In *Proc. SGP*, pages 1421–1430, 2008.

[79] Yaron Lipman, Xiaobai Chen, Ingrid Daubechies, and Thomas Funkhouser. Symmetry factored embedding and distance. In *ACM SIGGRAPH*, pages 103:1–103:12, 2010.

[80] Yaron Lipman and Thomas Funkhouser. Mobius voting for surface correspondence. In *ACM SIGGRAPH*, pages 72:1–72:12, 2009.

[81] Yaron Lipman, Vladimir G. Kim, and Thomas A. Funkhouser. Simple formulas for quasiconformal plane deformations. *ACM Trans. Graph.*, 31(5):124:1–124:13, September 2012.

[82] Rong Liu and Hao Zhang. Segmentation of 3d meshes through spectral clustering. In *Proceedings of the Computer Graphics and Applications, 12th Pacific Conference*, PG '04, pages 298–305, Washington, DC, USA, 2004. IEEE Computer Society.

[83] Tianqiang Liu, Vladimir G. Kim, and Thomas Funkhouser. Finding surface correspondences using symmetry axis curves. *Computer Graphics Forum*, 31(5):1607–1616, 2012.

[84] R.J. Lopez-Sastre, T. Tuytelaars, and S. Savarese. Deformable part models revisited: A performance evaluation for object category pose estimation. In *ICCV Workshop on Challenges and Opportunities in Robot Perception*, 2011.

[85] Facundo Mémoli and Guillermo Sapiro. Comparing point clouds. In *SGP '04: Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, 2004.

[86] Patrick Min. A 3d model search engine. *Ph.D. Thesis, Princeton University*, 2004.

[87] Patrick Min. A comparison of text and shape matching for retrieval of online 3d models, with statistical significance testing. *Utrecht University, Technical Report UU-CS-2004-026*, 2004.

[88] N. J. Mitra, L. Guibas, and M. Pauly. Partial and approximate symmetry detection for 3d geometry. *ACM Transactions on Graphics (SIGGRAPH)*, 25(3):560–568, 2006.

[89] Niloy J. Mitra, Leonidas Guibas, and Mark Pauly. Symmetrization. *ACM Transactions on Graphics (SIGGRAPH)*, 26(3):#63, 1–8, 2007.

[90] Niloy J. Mitra, Mark Pauly, Michael Wand, and Duygu Ceylan. Symmetry in 3d geometry: Extraction and applications. In *EUROGRAPHICS State-of-the-art Report*, 2012.

[91] Marius Muja and David G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *Proc. VISSAPP*, pages 331–340, 2009.

[92] Boaz Nadler, StÈphane Lafon, Ronald R. Coifman, and Ioannis G. Kevrekidis. Diffusion maps, spectral clustering and reaction coordinates of dynamical systems. *Applied and Computational Harmonic Analysis*, 21(1):113 – 127, 2006.

[93] Liangliang Nan, Ke Xie, and Andrei Sharf. A search-classify approach for cluttered indoor scene understanding. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)*, 31(6), 2012.

[94] Andy Nguyen, Mirela Ben-Chen, Katarzyna Welnicka, Yinyu Ye, and Leonidas Guibas. An optimization approach to improving collections of shape maps. *SGP*, 30(5):1481–1491, 2011.

[95] M. Novotni and R. Klein. Shape retrieval using 3d zernike descriptors. *Computer Aided Design*, 36(11):1047–1062, 2004.

[96] R. Osada, T. Funkhouser, B. Chazelle, and D. Dobkin. Matching 3d models with shape distributions. *SMI*, (154–166), 2001.

[97] Maks Ovsjanikov, Mirela Ben-Chen, Justin Solomon, Adrian Butscher, and Leonidas Guibas. Functional maps: a flexible representation of maps between shapes. *ACM Trans. Graph.*, 31(4):30:1–30:11, July 2012.

[98] Maks Ovsjanikov, Wilmot Li, Leonidas Guibas, and Niloy J. Mitra. Exploration of continuous variability in collections of 3D shapes. *ACM SIGGRAPH*, 30(4):33:1–33:10, 2011.

[99] Maks Ovsjanikov, Quentin Mérigot, Facundo Mémoli, and Leonidas J. Guibas. One point isometric matching with the heat kernel. *SGP*, 29(5):1555–1564, 2010.

[100] Maks Ovsjanikov, Jian Sun, and Leonidas Guibas. Global intrinsic symmetries of shapes. *Computer Graphics Forum*, 27(5):1341–1348, July 2008.

[101] M. Pauly, N. J. Mitra, J. Giesen, M. Gross, and L. Guibas. Example-based 3D scan completion. In *Symposium on Geometry Processing*, pages 23–32, 2005.

[102] M. Pauly, N. J. Mitra, J. Wallner, H. Pottmann, and L. Guibas. Discovering structural regularity in 3D geometry. *ACM Transactions on Graphics*, 27(3):#43, 1–11, 2008.

[103] Ulrich Pinkall and Konrad Polthier. Computing discrete minimal surfaces and their conjugates. *Experimental Mathematics*, 2:15–36, 1993.

[104] J. Pokrass, A. M. Bronstein, M. M. Bronstein, P. Sprechmann, and G. Sapiro. Sparse modeling of intrinsic correspondences. *eg*, 32(2), 2013.

[105] E. Praun, W. Sweldens, and P. Schröder. Consistent mesh parameterizations. *Proc. of SIGGRAPH 2001*, 2001.

[106] Yusuf Sahillioğlu and Yücel Yemez. Coarse-to-fine combinatorial matching for dense isometric shape correspondence. *SGP*, 30(5):1461–1470, 2011.

[107] N. Scaringella, G. Zoia, and D. Mlynek. Automatic genre classification of music content: a survey. *Signal Processing Magazine, IEEE*, 23(2):133 –141, march 2006.

[108] John Schreiner, Arul Asirvatham, Emil Praun, and Hugues Hoppe. Inter-surface mapping. *ACM SIGGRAPH*, 23(3):870–877, 2004.

[109] Ariel Shamir. A survey on mesh segmentation techniques. *Computer Graphics Forum*, 27(6):1539–1556, 2008.

[110] Tianjia Shao, Weiwei Xu, Kun Zhou, Jingdong Wang, Dongping Li, and Baining Guo. An interactive approach to semantic modeling of indoor scenes with an rgbd camera. *ACM Trans. Graph.*, 31(6):136:1–136:11, November 2012.

[111] L. Shapira, S. Shalom, A. Shamir, D. Cohen-Or, and H. Zhang. Contextual part analogies in 3d objects. *IJCV*, 89(2-3):309–326, 2010.

[112] Chao-Hui Shen, Hongbo Fu, Kang Chen, and Shi-Min Hu. Structure recovery by part assembly. *SIGGRAPH Asia*, 2012.

[113] Philip Shilane and Thomas Funkhouser. Distinctive regions of 3D surfaces. *ACM Transactions on Graphics*, 26(2):Article 7, June 2007.

[114] Philip Shilane, Patrick Min, Michael Kazhdan, and Thomas Funkhouser. The princeton shape benchmark. In *Proc. SMI*, pages 167–178, 2004.

[115] Oana Sidi, Oliver van Kaick, Yanir Kleiman, Hao Zhang, and Daniel Cohen-Or. Unsupervised co-segmentation of a set of shapes via descriptor-space spectral clustering. *ACM SIGGRAPH Asia*, 30(6):126:1–126:9, 2011.

[116] Justin Solomon, Mirela Ben-Chen, Adrian Butscher, and Leonidas Guibas. Discovery of intrinsic primitives on triangle meshes. In *Proc. Eurographics 2011*, 2011.

[117] Jian Sun, Maks Ovsjanikov, and Leonidas Guibas. A concise and provably informative multi-scale signature based on heat diffusion. In *Proceedings of the Symposium on Geometry Processing*, SGP '09, pages 1383–1392, Aire-la-Ville, Switzerland, Switzerland, 2009. Eurographics Association.

[118] A. Tevs, M. Bokeloh, M.Wand, A. Schilling, and H.-P. Seidel. Isometric registration of ambiguous and partial data. *In: Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[119] Trimble. Trimble 3D warehouse, http://sketchup.google.com/3dwarehouse/ 2012.

[120] Oliver van Kaick, Andrea Tagliasacchi, Oana Sidi, Hao Zhang, Daniel Cohen-Or, Lior Wolf, , and Ghassan Hamarneh. Prior knowledge for part correspondence. *CGF Eurographics*, 30(2):553–562, 2011.

[121] Oliver van Kaick, Hao Zhang, Ghassan Hamarneh, and Daniel Cohen-Or. A survey on shape correspondence. *CGF*, 30(6):1681–1707, 2011.

[122] Yanzhen Wang, Kai Xu, Jun Li, Hao Zhang, Ariel Shamir, Ligang Liu, Zhiquan Cheng, and Yueshan Xiong. Symmetry hierarchy of man-made objects. *Computer Graphics Forum (Special Issue of Eurographics)*, 30(2):287–296, 2011.

[123] Yunhai Wang, Shmulik Asafi, Oliver van Kaick, Hao Zhang, Daniel Cohen-Or, and Baoquan Chenand. Active co-analysis of a set of shapes. *SIGGRAPH Asia*, 2012.

[124] M. Weber, M. Welling, and P. Perona. Towards automatic discovery of object categories. In *IEEE CVPR*, 2000.

[125] M. Weber, M. Welling, and P. Perona. Unsupervised learning of models for recognition. In *ECCV*, 2000.

[126] Kai Xu, Honghua Li, Hao Zhang, Yueshan Xiong Daniel Cohen-Or, and Zhi-Quan Cheng. Style-content separation by anisotropic part scales. *SIGGRAPH Asia*, 2010.

[127] Kai Xu, Hao Zhang, Daniel Cohen-Or, and Baoquan Chen. Fit and diverse: Set evolution for inspiring 3D shape galleries. *ACM Trans. on Graph (Proc. of SIGGRAPH)*, 31, 2012.

[128] Kai Xu, Hao Zhang, Wei Jiang, Ramsay Dyer, Zhiquan Cheng, Ligang Liu, and Baoquan Chen. Multi-scale partial intrinsic symmetry detection. *ACM Transactions on Graphics (Special Issue of SIGGRAPH Asia)*, 31(6):to appear, 2012.

[129] Kai Xu, Hao Zhang, Andrea Tagliasacchi, Ligang Liu, Guo Li, Min Meng, and Yueshan Xiong. Partial intrinsic reflectional symmetry of 3d shapes. *sigga*, 28(5), 2009.

[130] Andrei Zaharescu, Edmond Boyer, Kiran Varanasi, and Radu P. Horaud. Surface feature detection and description with applications to mesh matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Miami Beach, Florida, June 2009.

[131] Eugene Zhang, Konstantin Mischaikow, and Greg Turk. Feature-based surface parameterization and texture mapping. *ACM Trans. Graph.*, 24(1):1–27, January 2005.

[132] Hao Zhang, Alla Sheffer, Daniel Cohen-Or, Qingnan Zhou, Oliver van Kaick, and Andrea Tagliasacchi. Deformation-drive shape correspondence. *SGP*, 27(5):1431–1439, 2008.

[133] Youyi Zheng, Hongbo Fu, Daniel Cohen-Or, Oscar Kin-Chung Au, and Chiew-Lan Tai. Component-wise controllers for structure-preserving shape manipulation. In *CGF Eurographics*, volume 30, 2011.

[134] Yinan Zhou and Zhiyong Huang. Decomposing polygon meshes by means of critical points. In *Multimedia Modelling Conference, 2004. Proceedings. 10th International*, pages 187 – 195, jan. 2004.