Retrieval on Parametric Shape Collections

ADRIANA SCHULZ
Massachusetts Institute of Technology
ARIEL SHAMIR
The Interdisciplinary Center Herzliya
ILYA BARAN
Onshape Inc.
and
DAVID I. W. LEVIN, PITCHAYA SITTHI-AMORN, and WOJCIECH MATUSIK
Massachusetts Institute of Technology

While collections of parametric shapes are growing in size and use, little progress has been made on the fundamental problem of shape-based matching and retrieval for parametric shapes in a collection. The search space for such collections is both discrete (number of shapes) and continuous (parameter values). In this work, we propose representing this space using descriptors that have shown to be effective for single shape retrieval. While single shapes can be represented as points in a descriptor space, parametric shapes are mapped into larger continuous regions. For smooth descriptors, we can assume that these regions are bounded low-dimensional manifolds where the dimensionality is given by the number of shape parameters. We propose representing these manifolds with a set of primitives, namely, points and bounded tangent spaces. Our algorithm describes how to define these primitives and how to use them to construct a manifold approximation that allows accurate and fast retrieval. We perform an analysis based on curvature, boundary evaluation, and the allowed approximation error to select between primitive types. We show how to compute decision variables with no need for empirical parameter adjustments and discuss theoretical guarantees on retrieval accuracy. We validate our approach with experiments that use different types of descriptors on a collection of shapes from multiple categories.

Categories and Subject Descriptors: I.3.5 [**Computer Graphics**]: Computational Geometry and Object Modeling—*Geometric algorithms, languages, and systems*

General Terms: Algorithms

Additional Key Words and Phrases: Shape retrieval, parametric designs

ACM Reference Format:

Adriana Schulz, Ariel Shamir, Ilya Baran, David I. W. Levin, Pitchaya Sitthi-Amorn, and Wojciech Matusik. 2017. Retrieval on parametric shape collections. ACM Trans. Graph. 36, 1, Article 11 (January 2017), 14 pages. DOI: http://dx.doi.org/10.1145/2983618

1. INTRODUCTION

A fundamental problem in many applications in graphics and modeling is the retrieval of shapes from a large collection. While shape-based matching and retrieval have been widely addressed for simple (nonparametric) shape databases, little progress has been made in efficient retrieval on collections of parametric shapes.In this work, we propose a strategy for searching through a database of parametric models in which the input query is expressed as a single 3D shape.

Parametric shapes—generalized models that return different shapes for different parameter settings—are important tools in graphics and modeling. In essence, one can view a single parametric design as representing a whole family of 3D shapes (see Figure 1). Using parametric designs can save storage but more importantly, they can support customization by users. Different users in different circumstances may require different designs of the same object, or may want to explore different variations of a similar design. In many cases it is impractical to explicitly design a new model for each variation, and this is where parametric designs are most useful.

Parametric shapes are widespread because most man-made objects are designed in parametric CAD systems such as Solidworks, OpenScad, Creo, Onshape, etc. In such tools, modeling is done by specifying parametric features, which can then be modified to allow for shape variations.Collections of such designs are available on repositories such as GrabCAD. In addition to modeling using CAD systems, several approaches have been suggested to allow for automatic conversion of existing designs to shapes that can then be manipulated while preserving their structure [Gal et al. 2009; Bokeloh et al. 2012]. The results of these techniques are customizable shapes with a constrained set of valid variations.

Retrieval on parametric shape collections is challenging because the search space is both discrete (number of shapes) and continuous (parameter values). In all previous work, when matching a given query model to a parametric model, one first has to fit the parameters to best match the query and then compute the distance from

This research was funded by NSF grant 1138967. Ariel Shamir is partly supported by ISF grant 324/11.
Authors' addresses: A. Schulz, 77 Massachusetts Ave, MIT 32-D414, Cambridge, MA 02139; email: adschulz@mit.edu.
Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from Publications Dept., ACM, Inc., 2 Penn Plaza, Suite 701, New York, NY 10121-0701 USA, fax +1 (212) 869-0481, or permissions@acm.org.
© 2017 ACM 0730-0301/2017/01-ART11 $15.00
DOI: http://dx.doi.org/10.1145/2983618

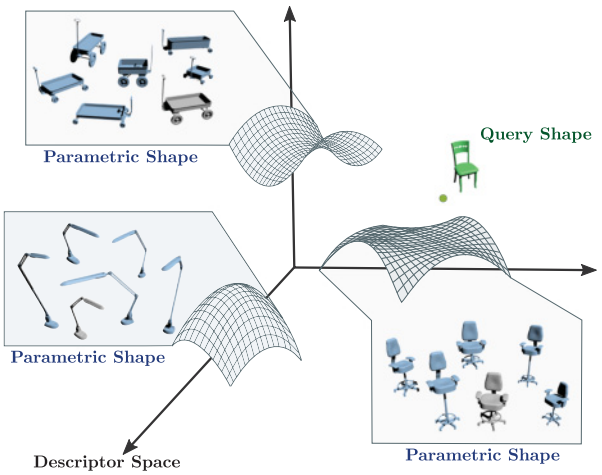


Fig. 1. We propose a method for shape retrieval from parametric shape collections that uses a descriptor space representation. While shape descriptors map single shapes to points in a descriptor space, smooth descriptors map parametric shapes to low-dimensional manifolds in this space. Our method efficiently represents these manifolds in order to allow for accurate and fast retrieval of the closest parametric model to a given query shape.

the query to the fitted shape. We call this the *fit-first* scheme. This scheme has several disadvantages. First, the process of fitting is time-consuming and has to be done for every shape in the database. It therefore does not scale well as the size of the database increases. Second, this scheme does not allow the use of descriptor space representations that have been shown to be effective for retrieval in a single (nonparametric) shape collection. The typical approach for efficient search does not rely on directly comparing a query element with every element in a database, but rather on precomputing descriptors for each shape and then performing fast retrieval by computing distances in this high-dimensional descriptor space. Because the actual geometry of each parametric shape is known only after fitting, it is not possible to perform the time-consuming task of computing descriptors a priori using the *fit-first* scheme. Descriptors must be extracted just before comparing or a direct comparison of the geometry must be used.

In our work, we propose a method for performing matching and retrieval from a collection of parametric shapes that does not follow the *fit-first* scheme. The key idea is to represent the full parametric shape, including continuous variations, in descriptor space. While single shapes can be described as points in a descriptor space, parametric shapes occupy larger "regions." To find the closest parametric shape given a query model (single point in descriptor space) we need to efficiently compute the distance from this point to each shape "region" and retrieve the closest one. We address this problem by creating a compact representation for these regions that allows minimal storage and fast evaluations, all the while guaranteeing accurate distance measurements. We observe that, for smooth descriptors, these regions are bounded low-dimensional manifolds embedded in high-dimensional space. The dimensionality of these manifolds is given by the number of parameters and the bounds are given by the feasible set of parameter values. We also have access to the actual function that defines the manifold, given by the composition of the parametric shape function and the signature function of the descriptor (see Figure 2).

We propose an algorithm for covering each manifold with a set of primitives that can be efficiently used for retrieval. We use two

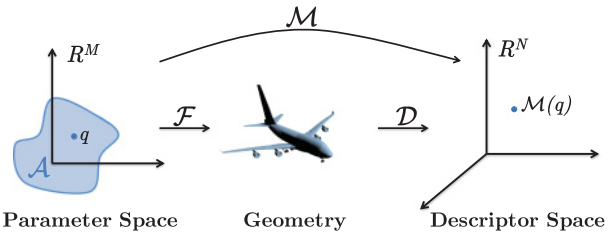


Fig. 2. The function $\mathcal{M}(q) = (\mathcal{D} \circ \mathcal{F})(q)$ is a composition of the mapping function $\mathcal{F}$ from parameter values to a geometry with the signature function $\mathcal{D}$, which generates a descriptor for a given geometry.

types of primitives: *points* and *bounded tangent spaces*. We discuss methods for creating these primitives (specifically, defining the bounds for the tangent spaces) and selecting between them to guarantee efficient storage and retrieval. The general idea is that flatter regions should be covered by tangent spaces, while more curvy ones should be covered by points. However, since different primitives have different storage and retrieval costs, the optimal cover depends not only on the geometry of the manifold, but also on the desired amount of accuracy. We therefore define an approximation error for our classification application and propose a method for primitive selection based on curvature, boundary evaluation, and allowed approximation error. Our theoretical analysis allows us to compute threshold values with no need for empirical parameter adjustments and provides guarantees on retrieval accuracy.

In addition to proposing the first retrieval algorithm for parametric shapes that exploits descriptor space representations, our work makes the following technical contributions:

—A representation of manifolds as a mixture of point and tangent primitives and a strategy for optimally selecting between primitive types for efficient coverage.
—A method for determining bounds of the tangent primitives based on target fitting error, curvature, and distance to the boundary.

We evaluate our method in terms of performance and accuracy using three types of descriptors and a collection of parametric shapes from multiple categories.

2. RELATED WORK

Our work draws from a number of methods in data-driven modeling, shape retrieval, template-driven exploration of shape collections, representations of point clouds in high dimensions, and manifold distances.

Data-driven Modeling with Parametric Shapes. Data-driven modeling exploits composition of new designs from a collection of shapes or shape parts. Such systems often require partial shape deformation for constraint satisfaction and part composition [Funkhouser et al. 2004; Huang et al. 2015]. However, these shape manipulations need to preserve structure and a variety of feasibility constraints, for example in the context of fabrication [Schulz et al. 2014]. By explicitly defining a feasible set in parameter space, the parametric representation allows for large variability while guaranteeing validity. Recent usage of parametric shapes for modeling includes reconstruction of 3D shapes from images [Xu et al. 2011] or point-set scans [Nan et al. 2012; Shen et al. 2012]. None of these works, however, address the fundamental problem of efficiently querying a collection of parametric shapes. Nan et al. [2012] propose a technique for fitting template parameters to best match a given model, but still query the shape collection using the *fit-first*

scheme. Talton et al. [2011] use a parametric grammar for procedural modeling and propose a general *fit-first* scheme for fitting a parametrized variable-dimensional model to a query.

Shape Retrieval. Efficient retrieval of 3D shapes has drawn the attention of the graphics community for many years. For a survey of shape retrieval methods we refer the reader to Tangelder and Veltkamp [2008]. For more recent advances in the field we refer the reader to SHREC [2014]. One of the most common approaches for fast retrieval is the use of descriptors that represent geometric models as points in a high-dimensional feature space. In this approach, the main computational cost is performed in preprocessing by evaluating the descriptors for each shape. Retrieval at runtime is reduced to a high-dimensional nearest neighbor search in descriptor space that can be performed quite efficiently. There is vast literature on descriptors for 3D shapes, ranging from simple histogram methods [Osada et al. 2001] to light transport functions [Chen et al. 2003]. Benchmarks for comparing these descriptors have also been proposed [Shilane et al. 2004] and the choice of descriptor is usually done based on the trade off between accuracy and computational cost. Some approaches also propose descriptors that are independent of certain shape transformations, such as articulations [Gal et al. 2007; Bronstein et al. 2011]. However, none of these methods can capture the variability of parametric shapes. Parametric models that return a different geometry for different parameter settings cannot be represented as points since they cover large regions of the descriptor space. We propose a method to efficiently represent these regions.

Template-Driven Shape Exploration. Our problem is also related to works that represent a category of discrete designs using a parametric 3D template. In this case, the template is not a parametric design, but a description that generalizes a set of models. Ovsjanikov et al. [2011] construct a single template to generalize a particular shape category and use it to explore the variability of the collection. Kim et al. [2013] produce a set of probabilistic templates that group large shape collections into clusters that capture the shape variations. These exploration tools have also been used for shape synthesis [Averkiou et al. 2014]. Finally, Yang et al. [2011] propose a method for exploring meshes with similar connectivity while preserving constraints. Although these works do not directly address the retrieval problem, some of the proposed techniques relate to our problem. A key observation of Ovsjanikov et al. [2011] is that since templates have a low-dimensional set of parameters, they lie near a low-dimensional manifold in a descriptor space. Following this observation, they use PCA to extract the variability of the shape collection in this space and use optimization to convert it into the variability of the template deformation. Similarly, Yang et al. [2011] define the shape space as a manifold that is navigated by local planar and quadratic approximations. In line with these works, we represent our parametric shapes as low-dimensional manifolds in descriptor space. However, in our work, each manifold is defined by a single parametric shape and not a set of nonparametric shapes. Moreover, we aim to represent a collection of such manifolds, defined by a collection of parametric shapes, and support distance queries from all of them to allow efficient retrieval.

Point Clouds in High Dimensions. Since we represent parametric shapes as low-dimensional manifolds in a descriptor space, our work is related to compact representations of low-dimensional data in high dimensions. Manifold learning is a strategy that aims at finding meaningful low-dimensional structures in high-dimensional data using nonlinear dimensionality reduction methods such as ISOMAP [Tenenbaum et al. 2000] and LLE [Roweis and Saul 2000]. In these approaches, we assume that the K-dimensional manifold is represented as a point cloud in an N-dimensional space ($K \ll N$) and no additional information is known. The result of such techniques is a map $A : \mathbb{R}^N \mapsto \mathbb{R}^K$ that allows projecting points into this low-dimensional space. This representation, however, cannot be used for retrieval since distances to query points must be computed in $\mathbb{R}^N$ allowing comparisons across manifolds.

By creating a point cloud representation of each parametric shape using sampling in parameter space, our problem is closely related to a classification problem in high-dimensional data, where each parametric shape defines a class. Among the most common approaches for this problem are Gaussian mixture models [Bishop 2006], which can be computed using Expectation-Maximization (EM) algorithms. Since parametric shapes are low dimensional, Gaussians in $\mathbb{R}^N$ cannot compactly cover each shape space and additional dimensionality reduction would be necessary to guarantee minimal overlap between class representations. Alternatively, one can use a method such as mixtures of factor analyzers [Ghahramani et al. 1996], which concurrently performs clustering and local dimensionality reduction within each cluster. In our application, however, instead of starting with a point cloud, we have access to the actual function that defines the manifold, namely, the parametric model composed with the descriptor evaluation. We also know the underlying dimensionality, which is defined by the number of parameters. We take advantage of this in our algorithm, measuring geometric properties such as derivatives and curvatures on sampled points, which are not present in a point cloud representation.

Distances to Manifolds. Our approach relies on an estimate of distances from points to manifolds with a known parametrization map, a problem that has also been addressed in several research areas. Pottmann and Hofer [2003] propose a method for constructing smooth functions that approximate the distance from a a point $x \in \mathbb{R}^N$ (variable) to a given manifold (fixed). These functions have second-order accuracy with respect to x and can therefore be used in optimization tools that have the position x as a variable and the distance to the manifold as part of the cost function. This has been applied, for example, in the context of registration [Pottmann et al. 2004; Mitra et al. 2004] and surface approximation [Wang et al. 2006]. In our work, however, since in any retrieval experiment the x position is given by a query shape (fixed), second-order accuracy with respect to x adds no information to our measurement. Instead, we prefer simpler functions that approximate the first-order distance metric and allow for fast estimation of the closest manifold to the query point. This involves efficient representation of the manifold to allow for fast distance estimation given a fixed query point. Vural and Fossard [2011] have proposed a method for discretizing manifolds to allow for distance estimation and classification. Their algorithm has similar goals to ours: they sample each manifold, all the while attempting to determine the number of samples that should be retained to maximize classification accuracy. This work, however, is restricted to point sampling. Tangent approximations have also been widely used to approximate manifold distances [Vasconcelos and Lippman 2005; Srivastava et al. 2005]. These are known to provide a more compact representation, but only work locally since they are equivalent to the first-order Taylor approximation. In our approach, we combine these two ideas by proposing a hybrid approach where the manifold is represented by a set of primitives that can be either point samples or bounded tangent spaces. In our method, we address the question of how to select between the primitive types in order to optimally allocate resources and discuss theoretical and empirical bounds on retrieval accuracy.

3. REPRESENTATION OF PARAMETRIC SHAPES

We define a parametric shape as $T = \{\mathcal{F}, \mathcal{A}\}$, where $\mathcal{A} \subset \mathbb{R}^K$ is the feasible set that constrains the parameter values, and $\mathcal{F}$ is a function from parameter values $q \in \mathcal{A}$ to a geometry (e.g., a mesh).

Given a query shape s, we would like to compute the distance from s to T. Formally, this distance is defined by

$$\text{dist}(s, T) = \min_{q \in \mathcal{A}} (\text{dist}(s, \mathcal{F}(q))),$$

where the distance between two fixed shapes $\text{dist}(s, \mathcal{F}(q))$ can be defined by a given shape descriptor. However, instead of finding the optimal value of q and computing the distance for this parameter (i.e., *fit-first*), we will find this distance by defining a representation in a descriptor space of the whole parametric shape. Similar to the previous work [Osada et al. 2001; Chen et al. 2003], we represent a geometry using a descriptor that takes a 3D mesh and computes a signature vector (typically signature vectors are high dimensional). This signature vector compactly represents a single geometry as a high-dimensional point in a descriptor space. However, this approach is not obviously applicable to parametric shapes because parametric shapes span a large set of possible geometries and therefore occupy a larger region of the descriptor space.

As shown in Figure 2, we define $\mathcal{M}(q) = (\mathcal{D} \circ \mathcal{F})(q)$, where $\mathcal{D}$ is the signature function that generates a descriptor for a given geometry. We can interpret $\mathcal{M}(q)$ as a parametrization from $\mathcal{A} \subset \mathbb{R}^K$ to $\mathbb{R}^N$, where the number of shape parameters K is much smaller than the dimensionality of the descriptor space N. Our method assumes that $\mathcal{F}$ is smooth. This holds for the models that are automatically converted from single geometries and for most CAD models since these shapes are typically designed such that parameter variations smoothly deform geometries. As a result, for smooth descriptors we can assume that the image $\mathcal{M}(\mathcal{A}) = \bigcup_{q \in \mathcal{A}} \mathcal{M}(q)$ lies on a manifold. Therefore, given a query shape s, we can apply the signature function to compute its value in descriptor space $x = \mathcal{D}(s)$ and define $\text{dist}(s, T) = \text{d}_2(x, \mathcal{M}(\mathcal{A}))$, where d_2 is the Euclidean distance in $\mathbb{R}^N$.

Our goal is to efficiently evaluate the distance from x to a collection of manifolds that represent each parametric shape in our database in order to retrieve the closest one (see Figure 1). Our approach is to construct a compact representation of each manifold that is an approximation with a certain allowed error. We aim at finding an approximation that has minimal storage requirements and allows for distance evaluations that are both fast and accurate.

3.1 Manifold Approximation

We approximate each manifold $\mathcal{M}(\mathcal{A})$ as a set of I primitives that locally describe the manifold: $\bar{\mathcal{M}}(\mathcal{A}) = \{P_1, \ldots, P_I\}$. For convenience, we will drop the parenthetical $(\mathcal{A})$ in the notation of $\mathcal{M}$ and $\bar{\mathcal{M}}$.

Our goal is to find the closest parametric shape in a collection given a query shape, that is, find the closest manifold $\mathcal{M}$ given a query point x. Accordingly, we have a good approximation $\bar{\mathcal{M}}$ if the distance from x to $\mathcal{M}$ and the distance from x to $\bar{\mathcal{M}}$ are approximately the same. We therefore say the approximation error of the manifold is δ, if

$$\forall x \in \mathbb{R}^N, \quad |\text{d}_2(x, \mathcal{M}) - \text{d}_2(x, \bar{\mathcal{M}})| \leq \delta.$$

We can write this as

$$\text{d}_2(x, \mathcal{M}) - \delta \leq \text{d}_2(x, \bar{\mathcal{M}}) \leq \text{d}_2(x, \mathcal{M}) + \delta.$$

The inequality on the right is satisfied if

$$\text{d}_2(y, \bar{\mathcal{M}}) \leq \delta \ \ \forall y \in \mathcal{M}, \qquad \text{(Coverage Lemma)}$$

while the inequality on the left is satisfied if

$$\text{d}_2(\bar{y}, \mathcal{M}) \leq \delta \ \ \forall \bar{y} \in \bar{\mathcal{M}}. \qquad \text{(Tightness Lemma)}$$

Proof of the Coverage Lemma. Given $x \in \mathbb{R}^N$, $\exists y \in \mathcal{M}$ such that $\text{d}_2(x, \mathcal{M}) = \text{d}_2(x, y)$. If the Coverage Lemma holds, then there $\exists \bar{y} \in \bar{\mathcal{M}}$ such that $\text{d}_2(y, \bar{y}) \leq \delta$. By the triangle inequality we get $\text{d}_2(x, \bar{y}) \leq \text{d}_2(x, y) + \text{d}_2(y, \bar{y})$. Since, $\text{d}_2(x, \bar{\mathcal{M}}) \leq \text{d}_2(x, \bar{y})$, we conclude that $\text{d}_2(x, \bar{\mathcal{M}}) \leq \text{d}_2(x, y) + \text{d}_2(y, \bar{y})$, which, in turn, gives us $\text{d}_2(x, \bar{\mathcal{M}}) \leq \text{d}_2(x, \mathcal{M}) + \delta$. □

Proof of the Tightness Lemma. Analogous to the proof of the Coverage Lemma. □

The Coverage Lemma states that every point on $\mathcal{M}$ is sufficiently close to $\bar{\mathcal{M}}$. This means that every point on the original manifold can be represented by a point on our approximation. This guarantees that if $x \in \mathbb{R}^N$ is close to $\mathcal{M}$, then it will be close to $\bar{\mathcal{M}}$. The Tightness Lemma states that every point on $\bar{\mathcal{M}}$ is sufficiently close to $\mathcal{M}$, which means that there is no point on the approximation that is far from the manifold. This guarantees that if $x \in \mathbb{R}^N$ is far from $\mathcal{M}$, then it will be far from $\bar{\mathcal{M}}$. Together, the coverage and tightness mean that the Hausdorff distance between $\bar{\mathcal{M}}$ and $\mathcal{M}$ is bounded by δ.

4. ALGORITHM

Each primitive P_i of the approximation $\bar{\mathcal{M}} = \{P_1, \ldots, P_I\}$ is defined as either a *point* or a *bounded tangent space*, which is formed by the intersection of a tangent space at a given point with an ellipsoid $\mathcal{E}_i \subset \mathbb{R}^N$ centered at that point. We write

$$P_i = \begin{cases} p_i \text{ or} \\ \{x \in \mathcal{E}_i | x = p_i + \sum_j a_j^i \mathbf{t}_j^i\}, \end{cases} \qquad (1)$$

where p_i is a point on $\mathcal{M}$, $\{\mathbf{t}_1^i, \ldots, \mathbf{t}_K^i\}$ are the normalized directional derivatives that form a basis to the tangent space of $\mathcal{M}$ at p_i, and $a_j^i \in \mathbb{R}$ are weights.

To define $\bar{\mathcal{M}}$, we propose an algorithm that samples points y on $\mathcal{M}$ at random and then adds a primitive to $\bar{\mathcal{M}}$ if $D(y, \bar{\mathcal{M}}) > \delta$. Random sampling of points on $\mathcal{M}$ is done by randomly selecting points $q \in \mathcal{A}$ and computing $y = \mathcal{M}(q)$. The added primitive could either be a single point or a bounded tangent space as defined in Equation (1). We argue that in the limit, this sampling algorithm assures that we get a complete coverage of the manifold. In our experiments, we terminate sampling after 2,000 rejections. This does not provide a technical guarantee of complete coverage, but it is a good approximation as shown in Figure 8. This is because the rejection sampling scheme we use will keep all points that are not covered by the approximation. Tightness is always satisfied when P_i is a point, but not when P_i is a tangent space. In this case, we use the Tightness Lemma to define a rule on how to determine the bounding ellipsoid $\mathcal{E}_i$ for P_i, as will be discussed in the following.

When our rejection sampling scheme chooses to add a primitive to $\bar{\mathcal{M}}$, the primary decision is to determine whether it should be represented as a single point primitive or a bounded tangent space with the point as its center. This choice is done to maximize efficiency. A bounded tangent primitive requires storing K tangent vectors in addition to the center point; we therefore say that its cost is $K + 1$ times the cost of the point primitives. This also roughly corresponds to the increase in query computation time (see Section 5). Hence,

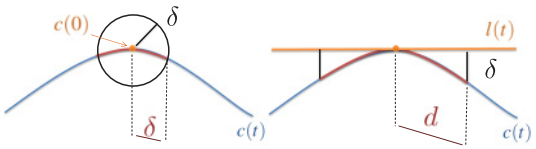


Fig. 3. The coverage of a point (left) and of a tangent line (right) is defined by the region of the manifold (here illustrated as a curve $c(t)$) that is well approximated by this primitive given the allowed approximation error δ. While the coverage of the point $c(0)$ is directly proportional to δ, the coverage of the tangent line $l(t)$ is proportional to d, which depends on the curvature.

if the bounds of a tangent primitive are tight enough such that the region it covers is smaller than the region that can be covered by $K+1$ points, then it is not worthwhile to use this primitive. To make this decision, we need to define and measure the coverage of both point and tangent primitives. We will first consider the case where the manifold is unbounded (i.e., $\mathcal{A} \equiv \mathbb{R}^K$) and later we will take into account the additional bounds imposed by the feasible set $\mathcal{A}$.

4.1 Unbounded Manifolds

If a manifold does not have bounds, the only aspect that determines how well it can be locally represented by tangent spaces is how much it deviates from being flat. We will measure how well a tangent space can locally approximate a manifold based on the Coverage Lemma. Then, we will discuss how we define the bounding hyperellipse $\mathcal{E}_i$ based on the Tightness Lemma.

Coverage. First, let us consider the one-dimensional case where $\mathcal{M} = c(t)$ is a curve in $\mathbb{R}^2$ and assume without loss of generality that the sample point is $p = c(0)$. In this case, the tangent approximation is then given by the line

$$l(t) = c(0) + tc'(0).$$

Since we allow an error of size δ, then once a point is sampled, any point on the circle of radius δ centered at that point is well represented by the sampled point according to the Coverage Lemma. On the other hand, if we take the tangent line on that point, then any point on the curve that is within distance δ of this line is covered by the line representation. So, while the coverage of a point is proportional to δ, the coverage of the tangent line is proportional to d, where d is the distance from the point p to the furthest point on the curve $c(t)$ that is sufficiently close to the tangent line (see Figure 3).

We can approximate d using the Taylor expansion. If $c(t) = c(0) + tc'(0) + \frac{1}{2}t^2c''(0)$, then the distance from a point $c(t)$ to the line l is given by

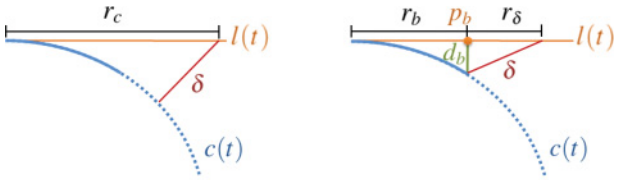
$$D(c(t), l) = \frac{1}{2}t^2 \left\| c''(0) - c'(0)\frac{\langle c''(0), c'(0)\rangle}{\|c'(0)\|^2} \right\|.$$

We can make this distance smaller than δ by bounding t as follows:

$$t \leq \sqrt{2\delta / \left\| c''(0) - c'(0)\frac{\langle c''(0), c'(0)\rangle}{\|c'(0)\|^2} \right\|}.$$

The distance d can then be approximated by $T_{\max}\|c'(0)\|$ from which we get

$$d = \sqrt{2\delta / \frac{\|c''(0)\langle c'(0), c'(0)\rangle - c'(0)\langle c''(0), c'(0)\rangle\|}{\|c'(0)\|^4}}. \quad (2)$$



(a) Unbounded case (b) Bounded case

Fig. 4. Computation of the bounding radius for a tangent space primitive $l(t)$ on the manifold $c(t)$. In the illustration, the dotted line represents the part outside the boundary of the manifold and δ is the allowed approximation error. Left: when we do not take the boundary into account the bounding radius is determined uniquely by the curvature constraint r_c. Right: when we are close to the boundary, the radius is computed as $r_b + r_\delta$, where r_b is the distance to the boundary and r_δ is the amount by which we can expand the radius preserving tightness constraints. We can compute r_δ from δ and d_b, which is the distance from the boundary point p_b on $l(t)$ to the manifold.

We observe that the denominator inside the square root of this expression is precisely the definition of curvature κ for the curve $c(t)$ at $t = 0$ [Do Carmo 1976]. From this we can write $d = \sqrt{\frac{2\delta}{\kappa}}$.

A lower bound on the number of points needed to cover the same region as the tangent line is given by the ratio of the two coverages, d/δ. Hence, we should store a tangent primitive if this ratio is larger than the extra storage requirement, $K + 1$. That is, we should store a tangent primitive if

$$k \leq \frac{2}{\delta(K+1)^2}. \quad (3)$$

This result is quite intuitive, since the curvature measures the amount by which the curve deviates from being flat. In our algorithm, we therefore measure the curvature at the point and if the curvature is small, then we store the bounded tangent primitive; if it is too big, we store the point primitive. The preceding equation defines how we determine this threshold based our the original parameter δ and the dimension of the parametric shape, so no additional empirical parameter estimation is needed.

This curvature interpretation can be easily expanded to $\mathcal{M} : \mathbb{R}^K \to \mathbb{R}^N$. In the multidimensional case, we use the maximal principal curvature [Do Carmo 1976], which measures the curvature at the direction where it is maximized. Since the coverage ratio is now given by $(d/\delta)^K$, we get

$$\frac{1}{k^{max}} \geq \frac{\delta}{2}(K+1)^{2/K}. \quad (4)$$

In our implementation, we approximate the maximal principal curvature k^{max} by the largest curvature in the K derivative directions. The curvature in each direction is computed using the expression for κ inside Equation (2), replacing the derivatives of curve c with the partial derivatives of the manifold $\mathcal{M}$.

Tightness. To bound the tangent space we have to ensure that the Tightness Lemma is satisfied. As we did in the previous section, we will first look at the one-dimensional case and will use the Taylor approximation. Then, the distance from a point $l(t)$ to the curve c can be bounded by the distance from a point $l(t)$ to the point $c(t)$ (see Figure 4(a)):

$$D(l(t), c) \leq D(l(t), c(t)) = \frac{1}{2}t^2\|c''(0)\|.$$

To ensure that this is smaller than δ, we bound t, such that

$$t \leq \sqrt{2\delta/\|c''(0)\|},$$

from which we get that the tangent space should be bounded by a circle of radius:

$$r_c = \sqrt{2\delta\|c'(0)\|^2/\|c''(0)\|}.$$

Again, in the multidimensional case, we use a hypersphere and take the first and second derivatives in the direction of the maximal principal curvature.

4.2 Bounded Manifolds

Next, we discuss how to incorporate the feasible set $\mathcal{A}$ into our representation. Because the feasible set induces boundaries on the manifold in descriptor space $\mathbb{R}^N$, we need to incorporate this effect into $\mathcal{E}_i$ in order to guarantee tightness. This, in turn, affects the coverage of the tangent primitives and should also be taken into account when choosing which primitive to store.

Tightness Constraints. Once again, we will start by looking at the one-dimensional case. As previously discussed, the curvature of the manifold defines a bound r_c to the tangent primitive, as shown in Figure 4(a). We define boundary constraint r_b, as the largest radius that guarantees that the projection of the bounded tangent line onto the curve falls on points $c(t)$, such that $t \in \mathcal{A}$ (see Figure 4(b)).

If the point $y = c(0)$ is close to the boundary, then r_c could be larger than r_b. To guarantee tightness in this case, the tangent line has to be bounded by $r_b + r_\delta$, where r_δ is the amount by which we can expand the curve to guarantee that the distance from it to the bounded manifold is smaller than the allowed approximation error δ.

In the multidimensional case, we consider a direction $\mathbf{v}$ in descriptor space in which to compute the distance from a sample point $y = \mathcal{M}(q)$ to the boundary. We assume that we have a set of analytic expressions that represent the boundary constraints in the parameter space and then map them to the descriptor space using the Jacobian $\mathbf{J_q}$ at the sampled point $\mathbf{q}$. Then, if a boundary constraint in the parameter space is written as a function $g(\mathbf{x})$, in the descriptor space it becomes $\mathbf{J_q}g(\mathbf{x}) + \mathcal{M}(\mathbf{q})$. We can find the distance to the boundary $g(\mathbf{x})$ in the direction $\mathbf{v}$ by solving

$$\min_{\alpha,\mathbf{x}} \|\mathbf{J_q}g(\mathbf{x}) - \alpha\mathbf{v}\|. \tag{5}$$

If the ray along the direction $\mathbf{v}$ intersects the boundary constraint $g(\mathbf{x})$, then the value of this minimization will be zero and the resulting α will return the distance from $\mathcal{M}(\mathbf{q})$ to this boundary constraint. The distance to the boundary, r_b, along $\mathbf{v}$ is then determined by computing this for every constraint $g(\mathbf{x})$ and taking the minimum.

To compute r_δ we first need to evaluate the distance d_b from the point $p_b = \mathcal{M}(\mathbf{q}) + r_b\mathbf{v}$ to the manifold. Then, as illustrated in Figure 4(b), we can compute r_δ so that $\delta^2 = d_b^2 + r_\delta^2$. To compute d_b we use the second-order Taylor approximation in a similar manner as explained earlier.

The computed distance to the boundary depends on the direction $\mathbf{v}$. Shooting rays in multiple directions and taking the minimum radius would determine a bounding hypersphere. This, however, is very restrictive, since a point can be close to the boundary in one direction and not in others. Therefore, we have chosen to bound the tangent spaces using ellipsoids instead of hyperspheres.

Naturally, the area covered by the ellipsoid depends on its orientation. Choosing optimal orientations for the ellipsoids can reduce the number of primitives needed to represent the manifold (see Figure 5). To determine a good basis for the orientation of the ellipsoids, we aim at aligning it with the least constrained directions of the manifold. We do this using a greedy approach. We start with a set of directions on the tangent space. First, we compute the distance from each of them to the boundary (using the method described earlier). Second, we take the direction that has the minimum distance to the boundary and set it as a basis vector. We then restrict our search to the orthogonal space of the current basis and repeat the first step. The algorithm ends after we complete a full basis.

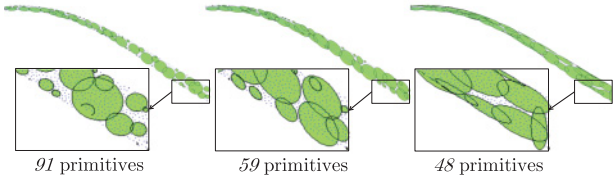


Fig. 5. From left to right: covering the manifold with tangent spaces bounded by hyperspheres, nonoriented ellipsoids, and oriented ellipsoids. This example illustrates that the number of primitives needed to represent the manifold for the same value of δ is reduced when we use better primitives. We notice that even in this example with a two-dimensional parameter space there is a significant improvement when oriented primitives are used. The blue dots represent the underlying manifold represented via super sampling. (Please note that these are high-dimensional primitives projected to 2D for visualization and therefore appear slightly distorted.)

Coverage. To choose between points and tangent primitives, we first verify Equation (4) and then compare coverage taking into account the constraints imposed by the boundary. For each direction, we set the coverage radius to be $r^i = \min(r_b^i + r_\delta^i, r_c^i)$. Then, following Equation (3), we choose to add a bounded tangent instead of a point if

$$\prod_{i=1}^{K} \frac{r^i}{\delta} \geq (K+1). \tag{6}$$

5. RETRIEVAL

Our retrieval method determines the closest parametric shape by finding the closest primitive to the query shape. Distances to points are measured using standard Euclidean norms and distances to bounded planes are measured by first projecting the query point x onto the tangent space and then computing the distance from the projection p to the ellipsoid. This distance is approximated using a scaling function S that maps the ellipsoid to the unit hypersphere centered at the origin. If $S(p) < 1$, then the distance is given by the projection error $d_p = \|x - p\|$. Otherwise, we approximate the distance from p to the ellipsoid as $d_e \approx \|p - S^{-1}(\frac{S(p)}{\|S(p)\|})\|$ and thus the final error is given by $\sqrt{d_p^2 + d_e^2}$ (see Figure 6).

While computing the distance from a query point to a point primitive in R^N is $\Theta(N)$, computing the distance to a tangent space primitive involves additional computation for evaluating the projection onto the tangent space p and its distance from the ellipsoid d_e. We can precompute the $N \times K$ projection matrix for each tangent primitive and store it as part of our data structure. This does not affect our previous storage discussion since the ratio of the storage cost for tangent primitives as compared to points remains on the order of K. With this, computing the projection of the query point onto the tangent space is $\Theta(KN)$. Using the simplification discussed previously, the computation of d_e is $\Theta(K)$. Therefore, while the distance to a point primitive is $\Theta(N)$, the distance to a tangent primitive is $\Theta(KN)$. From this, we conclude that, similar to storage requirement, the additional retrieval time for tangent primitives

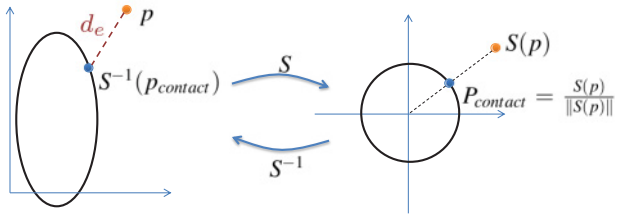


Fig. 6. Approximating the distance d_e from the projected point p to the hyperellipse. Let S be a scaling function that maps the hyperellipse to the unit hypersphere centered at the origin. The point on the hypersphere that is the closest to $S(p)$ is given by $p_{contact} = \frac{S(p)}{\|S(p)\|}$. We use the inverse mapping and approximate the distance from p to the hyperellipse as $d_e \approx \|p - S^{-1}(p_{contact})\|$.

Table I. Parametric Designs in Our Collection

Category	Number of Models
lamps	17
boats	11
chairs	15
planes	9
carts	10
tables	15

when compared to points is proportional to the number of parameters, K.

Although our method finds the closest parametric model to the query, finding the closest match still involves the final step of fitting parameters. Since we find the closest primitive, we can use the parameters of this primitive as an initial guess and use existing search methods to refine it. This problem has been addressed in previous work with an Iterative Closest Point (ICP) method [Nan et al. 2012].

6. EXPERIMENTAL SETUP

We tested our algorithm on a collection of models from multiple categories using three different descriptors.

6.1 Database

Although repositories of parametric CAD shapes are available (e.g., GrabCAD), evaluating geometry for a given parameter configuration requires access to the proprietary CAD software (e.g., Solidworks). Therefore, in order to run experiments with our retrieval method, we created a collection of parametric shapes using two procedures. First, we used a free CAD software (OpenSCAD) to model objects and expose design parameters. Second, we used an automatic method to convert single geometries into parametrized objects based on a simple method inspired by previous work on manipulation of man-made shapes [Gal et al. 2009].

Our collection of parametric shapes spans multiple categories, as shown in Table I. We use two CAD designs and 74 automatically converted models. We plan to release this collection along with the article to encourage future research on parametric shape collections. Figure 1 illustrates some of the models in our collection and their variations. We refer the reader to the supplemental materials for a detailed description of these designs. The number of parameters for each design ranges from 2 to 9. We argue that this is a descriptive range, even considering complex parametric CAD designs. Although parametric CAD software allows for many independent variables, these are often constrained by manufacturing considerations and the need to interface with other models. Therefore, in practice, most CAD designs only have a small set of meaningful parameters that can be directly exported by designers to allow for further user-driven customization (typically less than 10).

For the parametric CAD models, the ranges for the exposed parameters were hand annotated by the designer. While generating the manifold representations in descriptor space, we call the CAD software to compute a 3D mesh for each parameter configuration. For models defined by our automatic conversion procedure, we represented each vertex of the mesh explicitly as a linear function of the parameters. This makes geometry evaluations very fast, especially when compared to the CAD models where each evaluation involves several nontrivial operations.

In our designs, the feasible set of parameter values are linear: we defined ranges for exposed parameters of CAD designs and our automatic method defines the boundary of the feasible set using a set of linear constraints. With this assumption, the optimization shown in Equation (5) is a least squares problem that can be solved efficiently. We stress, however, that the mathematical model discussed in this article does not depend on the linearity assumption. In addition, the implementation speedups given by the linearity assumption are only relevant during the precomputation step that generates the manifold approximations and they do not affect retrieval time.

6.2 Descriptors

The algorithm we propose is independent of the choice of descriptor. The only assumption we make is that a descriptor should be quite smooth, so that the image of the parametric space lies close to a manifold in the descriptor space. We use three different descriptors for our experiments. The first one is the **D2 Shape Distribution**, which is defined by a histogram of distances between pairs of points on the surface of the model [Osada et al. 2001]. The second is the **VOXEL Shape Histogram**, which is a shape histogram descriptor [Ankerst et al. 1999] and describes the distribution of a model area as a function of the distances from voxel centers. Since these descriptors are not necessarily smooth, we interpolate the feature signal with a Gaussian kernel, following the approach of Ovsjanikov et al. [2011]. The third descriptor is the **Light Field Descriptor** [Chen et al. 2003], which captures geometry detail from rendered images of the shape and is known to have good retrieval precision [Shilane et al. 2004].

For the D2 descriptor, we sample 3,000 points on the surface of the model and express them as a function of the shape parameters. For a given parameter setting, we measure the pairwise distances (normalized by the average distance) between all sampled points and convolve this distribution with a set of Gaussian kernels of a fixed width σ and means distributed uniformly between 0 and 3. Since in our collection each sampled point can be written as a linear expression of the parameters, derivatives can be computed analytically. We have also experimented with finite differences, which are faster to compute and comparable in terms of accuracy. In our experiments, we set $\sigma = 0.1$ and use 300 Gaussian means. We have also used PCA on our dataset to reduce our descriptor space to 24 dimensions.

For the VOXEL descriptor, we sample one million points on the surface for the model. We take the difference from each sample point to the center of mass and normalize them by maximal distance. We convolve this distribution with a set of isotropic 3D-Gaussian kernels that have variance σ and means distributed uniformly on a 3D grid. For this experiment, we made sure to resample all the points for each parameter configuration and use finite differences

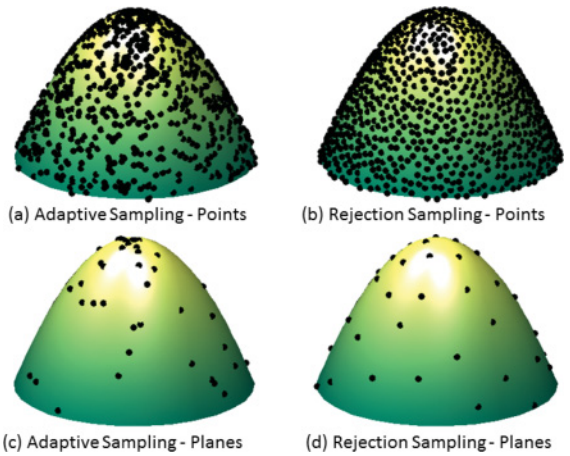


Fig. 7. Comparison between adaptive sampling and rejection sampling on a simple paraboloid example. Our rejection sampling scheme was done for both point and planes for a fixed approximation error δ. The number of samples for the adaptive sampling schemes was chosen to be the same as the result of the rejection sampling for both points and planes. The top row shows the results for point samples. Although both methods return a uniform distribution, in the adaptive sampling scheme points tend to clump together and leave gaps. The bottom row results for approximating with tangent spaces (we only display the center of the tangent space for simplification). Once again both methods display the desired distribution (based on curvature) and rejection sampling covers the space more effectively.

to compute first and second derivatives. We have used $\sigma = 0.2$ and 1,000 Gaussians distributed on a $10 \times 10 \times 10$ grid. As with the D2 descriptor, we use PCA on the dataset to reduce the descriptor to 64 dimensions. Although this descriptor is not rotation invariant, we exploit the fact that our models are CAD designs that have upright orientation and are aligned with one of the four principal axes. Therefore, we perform retrieval on four rotated versions of the query and keep the best one.

Construction of the Light Field descriptor involves transforming a model to be centered at the origin and inside of a unit sphere. The model is then rendered from a number of viewpoints, sampled from the vertices of a dodecahedron. Image features are computed as in Chen et al. [2003], combining Zernike moments with Fourier coefficients. Again, in this setting, we handle rotation invariance by performing retrieval on four rotated versions of the query and keeping the best one. We use PCA on the dataset to reduce the descriptor to 280 dimensions.

7. EVALUATION

We present results on evaluating the accuracy and efficiency of our manifold representation for individual parametric shapes, as well as on the overall retrieval method from a shape collection.

7.1 Manifold Representation

Sampling Scheme. Alternatives to our rejection sampling approach are adaptive sampling schemes based either on curvature or on surface areas. While a method based on curvature is suitable for approximating the manifold with tangent planes, adaptive sampling based on surface area approximates a uniform distribution of points in descriptor space. We have implemented both of these approaches using a Metropolis–Hastings algorithm, where the probability density function given a current sample is given by a Gaussian centered

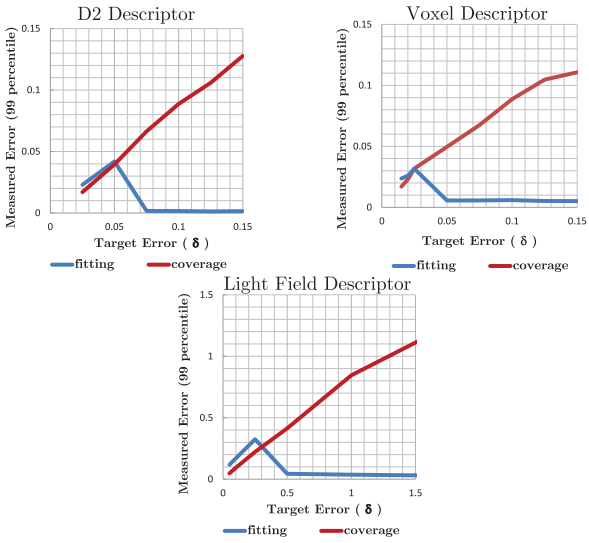


Fig. 8. Measuring *fitting* and *coverage* errors as a function of the target parameter δ for the implemented descriptors. We observe that both measured errors are within the bounds of δ. For large values of δ we observe that the fitting error drops to zero. This is because for very coarse approximations, our algorithm prefers points to tangents—the coverage of points becomes larger with δ, while plane coverage is still limited by the curvature and the boundary of the manifold. Since absolute distance values depend on descriptors (and are much larger for the Light Field descriptor), the ranges of the target errors for this experiment were chosen so that the number of samples were similar for all descriptors.

at that point with variance proportional to the curvature or surface area measured at that point. We compare the results from these approaches to our rejection sampling scheme using only point or tangent plane primitives and illustrate the results in Figure 7. We observe that we get similar distributions for curvature based adaptive sampling and rejection sampling for tangent primitives, and similar distributions for surface area adaptive sampling and rejection sampling using only point primitives.

However, while the adaptive sampling approaches are good at approximating the desired distributions, the randomness in the algorithm makes it unsuitable for minimal coverage, especially when the number of primitives is small. Figure 7 shows how points tend to clump together and leave gaps. On the other hand, the rejection sampling scheme guarantees that only points that contribute to coverage are added. In addition, our method determines the number of samples based on a unique user-specified parameter that reflects the retrieval error. Although the parameters of the adapting sampling schemes may be tweaked to vary how densely the sampling covers the space, these cannot be easily mapped to a global approximation error. Since sampling is part of a preprocessing step, this justifies a more expensive approach (rejection sampling) that results in lower storage, more controlled approximation error, and faster runtime. Another important aspect is that the criteria for adaptive sampling depends on the primitive type, while the rejection sampling method we propose can handle a hybrid representation. Finally, we can incorporate the boundary information to the rejection sampling algorithm, which allows us to create a compact representation for our bounded manifolds.

Accuracy. We evaluate the accuracy of our manifold representation by measuring the actual *fitting* and *coverage* error for different

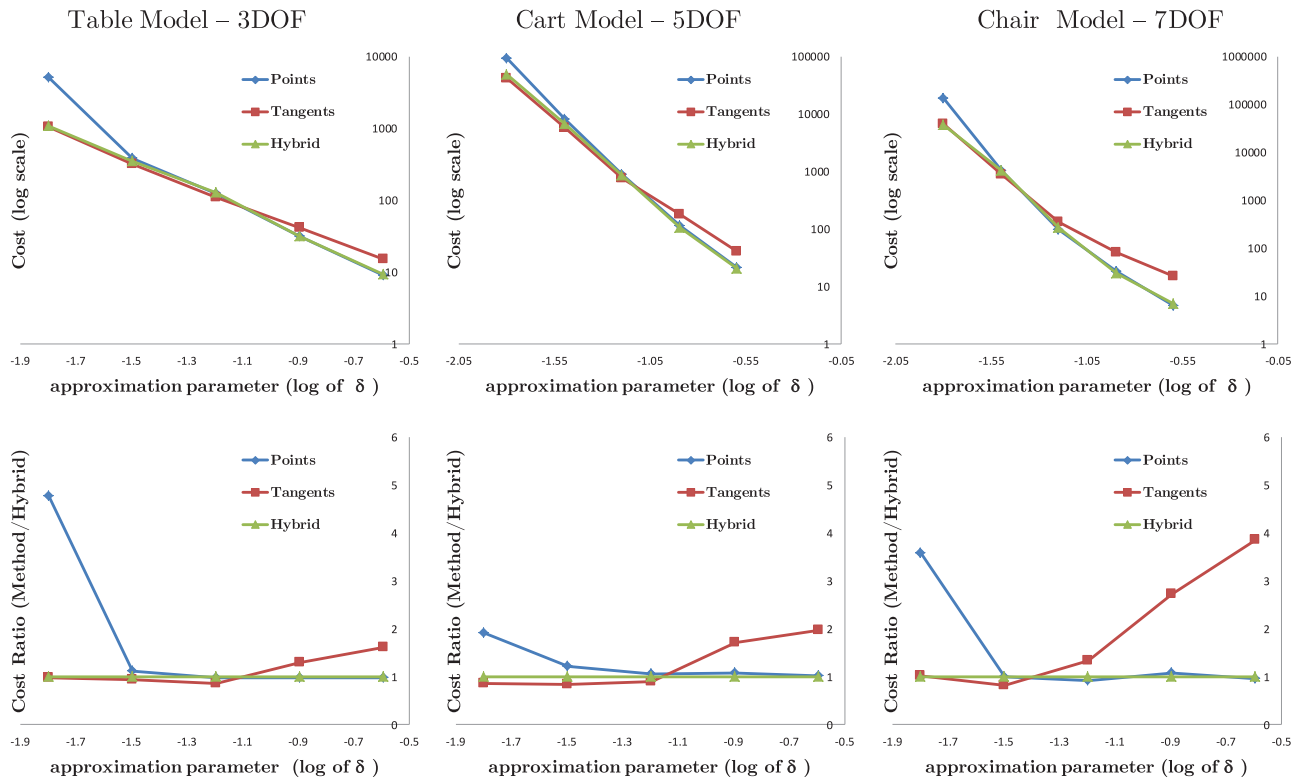


Fig. 9. Comparison between our hybrid method and using a single primitive. Top row: shows the storage cost of each representation across different target parameters δ in log scale. Bottom row: the relative cost of the single primitive methods while compared to our method.

values of the target parameter δ. We measure coverage error by sampling points from the ground truth manifold $\mathcal{M}$ and computing the distances to the representation $\bar{\mathcal{M}}$. We measure fitting error by sampling points on $\bar{\mathcal{M}}$ and computing distances to the ground truth $\mathcal{M}$. As ground truth we use a dense super sampling of the manifold $\mathcal{M}$, namely, a point-only (no tangent approximations) rejection sampling with very small $\delta = 0.005$. Figure 8 shows results of an experiment on a parametric chair with two parameters for all three descriptors. We plot across different values of δ the 99 percentile error (the worst error discarding the worst 1%).

Efficiency. To evaluate the efficiency of our representation we compare our method with a rejection sampling scheme that uses just point primitives and one that uses just tangent primitives. Figure 11 shows the storage cost of each representation across different values of the target parameter δ (that defines the accuracy of the approximation). The cost depends on the number of stored primitives and their storage costs. We set the cost for point primitives to 1 and the cost of tangent primitives to $K + 1$, where K is the number of parameters (Degrees of Freedom (DOF)) of the shape. Note that counting ellipsoids as $K + 1$ times more expensive also roughly corresponds to the increase in query computation time. We present evaluation results on three different shapes: a table with 3DOF, a cart with 5DOF, and a chair with 7DOF.

The top row in Figure 9 shows the cost versus δ on a log scale. We observe the trend of preferring tangents for small values of δ and points for large values of δ. As shown in the graphs, our hybrid representation can optimally select the transition between points and planes so that its cost is constantly below the other two alternatives. The second row shows the relative cost of using a single primitive compared to our method. As shown in the graphs, the relative cost of our hybrid method is close to one of the two at a certain δ range, while the other methods have a cost up to five times larger, depending on the shape and primitive type. The small oscillations in this graph are mostly due to the randomness in the sampling algorithm, but are also related to some approximations in our implementation.

Note that our representation uses both primitives and does not change between them at a specific point. It has a higher percentage of points for larger values of δ, and a higher percentage of ellipsoids for smaller values of δ. Note also that the crossover value where point primitives start to outperform tangents varies depending on the shape. This happens because different shapes have different numbers of parameters and also because the sizes of the feasible sets vary. These parameters influence the coverage of a tangent approximation. Therefore, using a representation that picks one primitive type depending on the target parameter δ is not feasible as the transition value depends on the shape. Given a specific δ, some shapes may be better represented with points, while others with planes. In contrast, our hybrid representation can adapt to the specific shape and choice of δ, automatically choosing the right combination of the two primitives. This is especially important while representing a collection of shapes, as only a hybrid scheme can optimize across all the different models in the database.

7.2 Retrieval

Next, we evaluate how well our representation works for retrieving models in a collection of parametrized shapes. First, we motivate the importance of taking into account the parametrized model. We

Fig. 10. Demonstration of query failures when representation only consists of the mean shape. From left to right: the mean shape of some parametric models in our database, query shape given by a random parameter setting of each parametric model, and the closest mean shape retrieved from the database. Since changes in parameter settings significantly alter the geometry, the closest mean shapes are usually not from the parametric models that originate the queries.

evaluate what happens when we do not represent the full manifold but instead, use a shape with the default template parameters (we will call this the mean shape). For the query, we use random parameter configurations and search the database for the closest mean shape. We have used the D2 descriptor for this experiment. Figure 10 illustrates a few of the retrieved results. In this figure, we show the query shape, the mean shape of the models that originates the query, and the closest mean shape retrieved. We observe that the changes on the parameters significantly affect the geometry. For example, when we flatten a cart it resembles a coffee table; if we shrink the feet of a stool it resembles a lamp. We ran the mean shape retrieval test on 20 random parameter samples for each model in our database. In this experiment, we managed to retrieve the correct mean shape only 29% of the time.

We also compare our sampling scheme with the naive approach that represents the manifold by randomly sampling a fixed number of points from the *parameter space* of each model. The advantages of our approach are threefold. First, distributing samples uniformly over the *descriptor space* rather than over the parameter space provides better coverage of the manifold. Second, fixing the value of δ for each shape allows the number of samples per shape to vary according to the size of the corresponding manifold in the descriptor space. Third, by storing both points and tangent primitives we reduce the storage cost.

To compare the two methods we evaluate their performance on retrieval of points sampled from our parametric shape collection. Naturally, if we sample these points randomly over parameter space the naive approach will have a better performance on average since it matches the distribution. Alternatively, if we sample uniformly over descriptor space, our approach does much better on average. For a fair comparison, we sample uniformly over the parametric space, but evaluate the worst-case, rather than the average error, measured as the distance to the closest primitive on the correct manifold minus the distance to the closest primitive on a wrong manifold. As a result, the error does not depend on the query distribution, but on how well our samples cover the space. Since the error is measured in descriptor space and needs a reference to interpret, we plot it against the target error δ (also in descriptor space) for our method in Figure 11. For an "equal cost" comparison, we plot the error of the naive method using the same amount of storage as our method. We plot the results sampling both over the full database and over individual categories.

The results show that the error for our method is not only smaller (close to δ), but is also consistent. For example, airplane models present very small variations in the D2 descriptor space and can therefore be well represented with only a few samples. On the other hand, chair and lamp variations are much more dramatic, and therefore need more samples. Our method automatically handles this difference and allocates more storage to represent larger manifolds. We therefore notice that our method performs consistently better across all categories.

Finally, we show retrieved results for models in each category that were collected from online shape repositories. In these experiments, retrieval times for each model were in the order of 10 milliseconds. Figure 12 shows the top result for both descriptors for varying targeted errors, δ. Results for additional query shapes are included in the supplemental material. The retrieved results are shown with the parameter settings of the closest primitive. In the case of tangent primitives we simply use the center of the ellipsoids and no additional projection. As discussed in Section 5, an additional fitting step would be required to select the optimal parameter configuration.

The quality of our retrieval results is obviously heavily dependent on the descriptor (Figure 12). For example, while the Voxel descriptor visually outperforms the D2 descriptor with chairs, the D2 descriptor appears to do better at retrieving lamps. As expected, the Light Field descriptor retrieves better results in every category. The figure also illustrates results for different target errors. Using measurements based on the descriptor space metric, we confirm that with smaller target errors the retrieved shapes are closer to the query. However, the relationship between the visual similarity and proximity in descriptor space highly depends on the quality of the descriptors. For example, using the D2 and Voxel descriptors, smaller δ's result in shapes that are visually less similar for the

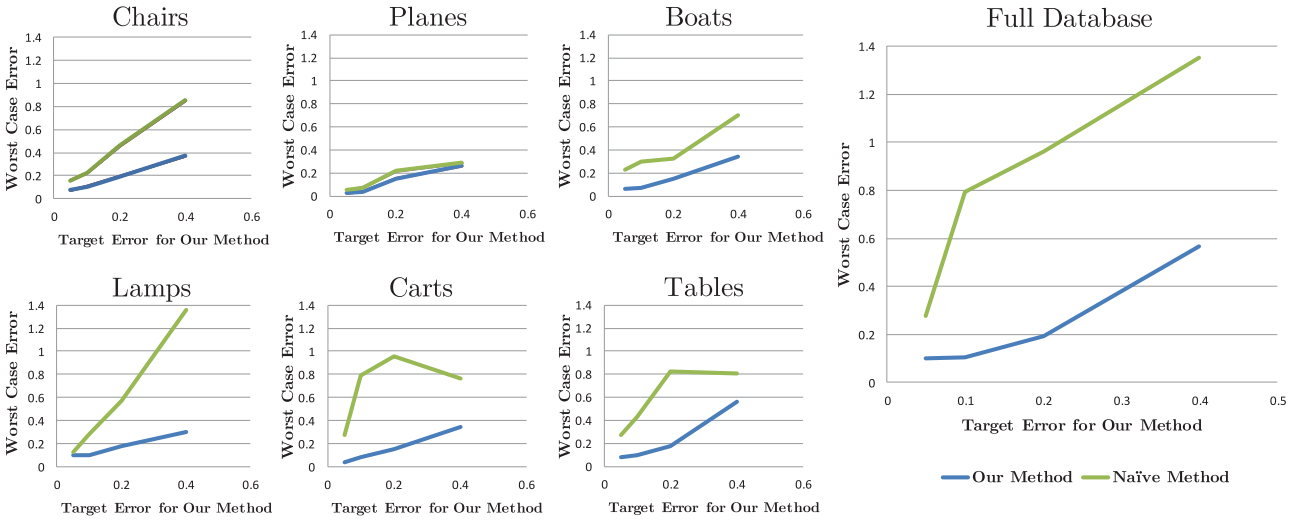


Fig. 11. Comparison between our approach and the naive one. We measure the difference between the distance to the closest primitive in the collection and the distance to the correct manifold and show the worst case for both approaches for querying points sampled on the full database and on individual categories. From these results we verify that our method has a better performance across all categories.

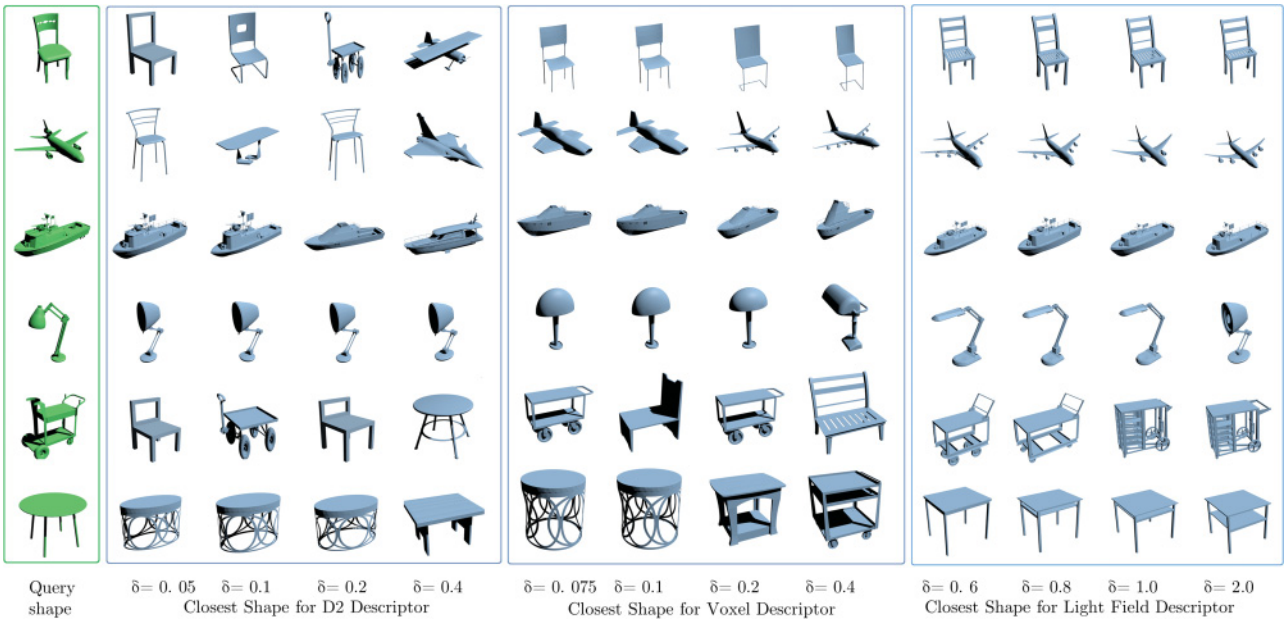


Fig. 12. Results of retrieval. From left to right: query shape, results for D2 descriptor (for increasing target errors), results for Voxel descriptor (for increasing target errors), and results for the Light Field descriptor (for increasing target errors).

airplane query example. Nevertheless, we argue that since the retrieved results are actually closer according to the descriptor metric, better descriptors will retrieve more visually accurate results. In fact, this is what happens with the Light Field descriptor.

Classification. Searching in the space of parametric shapes allows capturing structure preserving variations during retrieval. When parametric variations are taken into account using our manifold representation, each shape covers a much larger area on the search space. This change of the search space can affect classification in nontrivial ways and is also very dependent on the choice of descriptor. We analyze the effects in classification using Table II and Figure 13.

Table II compares the search space for different descriptors when single mean shapes are used and when the full manifolds are represented. The distance between categories is measured as the average of the pairwise minimal distance between categories. The category size is measured by the maximal distance between two shapes in a category and the average across all categories is shown. This result shows that when parametric representations are used the classes

Table II. Comparison between Coverage Regions in Descriptor Space

Descriptor	Distance Between Categories		Average Category Size	
	Mean Shape	Manifold	Mean Shape	Manifold
D2	0.16	0.02	0.98	1.83
Voxel	0.40	0.16	1.17	1.54
Light Field	1.75	1.12	3.61	5.37

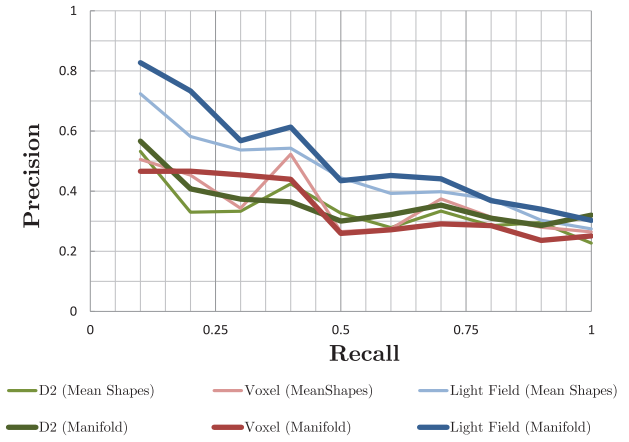


Fig. 13. Precision-recall plots evaluating classification accuracy for our method compared to using only mean shapes for different descriptors.

become closer to each other and the space covered by each class becomes larger. This is expected since parametric shapes include structure preserving variations. There are, however, significant variations depending on the descriptor. While the average distance between categories is reduced by 93% for the D2 descriptor, the reduction is only 31% for the Light Field descriptor.

Figure 13 shows the standard precision recall plot, which measures classification accuracy. Curves closer to the horizontal line at precision $= 1.0$ indicate superior retrieval results. Since classification depends on the descriptor, we notice a clear improvement in performance in the Light Field descriptor when compared to the D2 and Voxel descriptor. This result is consistent with mean shapes and the manifold representation. We notice, however, that while the manifold representation outperforms the mean shape on the Light Field descriptor, the results are equivalent (or slightly worse) for the other two descriptors.

From Figure 13 and Table II we conclude that when low quality descriptors are used, classifiers have poor predictive performance and the additional complexity added by the deformation parameters cannot be captured. Therefore, they do not help performance and can even act as noise, increasing the error. However, when high quality descriptors are used, the variations of the parametric representations allow better coverage of the space, improving classification performance.

We emphasize, however, that the application of retrieval in parametric shape collections goes beyond classification. This is illustrated in Figure 14, which uses the Light Field descriptor and compares the mean shapes and the manifold approximation. Results show that the increased variability in the search allows closer matches to be found. In some cases the retrieved results remain in the same category (see the boat, lamp, and cart examples). For the table example, however, the parametric shape space search returns a stool that although it belongs to different category (chairs), it can

Fig. 14. Comparison of retrieval with mean shapes only and manifold representation for the Light Field descriptor. From left to right: query shape (green), closest mean shape retrieved (gray), closest parametric shape retrieved with parameter fitting (blue) with its corresponding mean shape (gray). We observe that using the parametric shapes we retrieve models that are more similar in geometry but may lie on a different class.

be deformed to resemble a table. Although in terms of classification this is an inaccurate result, we notice an improvement in geometric proximity when comparing it to the table retrieved by querying the collection of mean shapes. In the case of a bench query, since we have no database models in this category, the mean shape search finds a boat that has similar dimensions. Our approach, on the other hand, can represent variations of the chair category that make it resemble a bench. This added capability of our technique is not captured by simply using precision-recall classification metrics.

8. LIMITATIONS AND FUTURE WORK

Although the main focus of this work is a method to represent a manifold created by parametric shapes in descriptor space, the results of retrieval will always rely on the quality of the actual

descriptor. We have tested the retrieval on three different descriptors and observed a large variation in performance. Other descriptors could be tested in our approach as long as they are smooth, that is, that the region covered by the parameters in descriptor space is close to a manifold.

Another important limitation to discuss is scalability. In our algorithm, storage size is not directly determined by the number of parameters (i.e., dimensionality) but by the volume of descriptor space relative to the tolerance. This volume indicates the variability of valid shapes for a given parametric model, which depends not only on the number of dimensions but also on the ranges of the parameters. For example, one of our airplane models with eight parameters needs less than a third of the storage of a lamp model with only three parameters. Nevertheless, in theory, the volume can increase exponentially with the number of parameters and therefore, our method, like most dimension-dependent representations, would not scale. We argue, however, that, in practice, models with a large number (and volume) of meaningful parameters are not frequently encountered. This is true because although parametric CAD allows for many independent variables, these are often constrained by manufacturing considerations and the need to interface with other models. Therefore, the volume of useful variations of a single design tends to be relatively small.

Another assumption that we make is that the feasible set $\mathcal{A}$ is connected. This is mostly relevant for approximating regions of the manifold using tangent spaces and computing boundaries of the ellipsoids. An extension of our work can represent $\mathcal{A}$ as a union of connected sets. It would also be interesting to handle complex boundaries (originated by an arbitrary number of nonlinear constraints) as well as a mixture of continuous and discrete parameters. These cases would require more primitives since tangent approximations can only be used on the continuous regions.

Lastly, another limitation of our method is that the time for computing queries scales linearly with the size of our database since we currently use a naive search approach. In high-dimensional descriptor spaces, algorithms based on Locality Sensitive Hashing (LSH) [Datar et al. 2004] can solve the nearest neighbor problem in sublinear time. While LSH algorithms typically work with points, one can imagine using ellipsoid centers with LSH to prune obviously far-away regions of the descriptor space and take advantage of such search structures. The effectiveness and feasibility of such methods would need to be tested by experiments.

9. CONCLUSIONS

In this work, we present the first approach for efficient retrieval on a collection of parametric shapes that improves upon the standard scheme of first fitting parameters and subsequently computing the distance to the query shape. We address this problem by using shape descriptors and representing parametric shapes as manifolds in this space.

Using a metric for manifold approximation error based on retrieval performance, we propose an algorithm for approximating a parametric shape given a target error. Our approximation consists of a mixture of points and bounded tangent primitives. We discuss how to bound the tangent primitives based on curvature and distance to the boundary. We also define a strategy for optimally selecting primitive types to minimize storage.

Our experiments validate the accuracy of our representation and show that our proposed hybrid representation consistently outperforms approximations that use a single primitive type. Finally, we demonstrate the performance of our method using three different types of descriptors for retrieval on a database of parametric shapes of multiple categories. We observe that the method efficiently retrieves the closest geometry according to the descriptor metric. These may lie outside the original shape categories because of the significant variations imposed by the parametric changes. We envision this approach being particularly useful for systems that query for parametric parts and then assemble them [Shen et al. 2012].

We observe a trend of using parametric shapes in both commercial software and research works. We anticipate that there will soon be large repositories of parametric shapes available and that they will be increasingly used in data-driven modeling systems. In this context, analysis tools that deal with these types of shapes will be of great importance. We hope that this work, together with the database we are releasing will inspire future work in this area.

ACKNOWLEDGMENTS

The authors would like to thank Professor Charles K. Smart for helpful suggestions and discussions; Baker Logan, Marie P. E. Moudio, and Jacob Haip for designing the models in the database; and Megan C. Chao for help with renderings.

REFERENCES

Mihael Ankerst, Gabi Kastenmller, Hans-Peter Kriegel, and Thomas Seidl. 1999. Nearest neighbor classification in 3D protein databases. *In Proceedings of ISMB* (1999), 34–43.

Melinos Averkiou, Vladimir Kim, Youyi Zheng, and Niloy J. Mitra. 2014. ShapeSynth: Parameterizing model collections for coupled shape exploration and synthesis. *Computer Graphics Forum (Special Issue of Eurographics 2014)* (2014), 10.

Christopher M. Bishop. 2006. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ.

Martin Bokeloh, Michael Wand, Hans-Peter Seidel, and Vladlen Koltun. 2012. An algebraic model for parameterized shape editing. *ACM Transactions on Graphics* 31, 4 (2012), 78:1–78:10.

Alexander M. Bronstein, Michael M. Bronstein, Leonidas J. Guibas, and Maks Ovsjanikov. 2011. Shape google: Geometric words and expressions for invariant shape retrieval. *ACM Transactions on Graphics* 30, 1, Article 1 (2011), 1:1–1:20.

Ding-Yun Chen, Xiao-Pei Tian, Yu-Te Shen, and Ming Ouhyoung. 2003. On visual similarity based 3D model retrieval. *Computer Graphics Forum* 22, 3 (2003), 223–232.

Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S. Mirrokni. 2004. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the 20th Annual Symposium on Computational Geometry*. ACM, 253–262.

Manfredo Perdigao Do Carmo. 1976. *Differential Geometry of Curves and Surfaces*. Vol. 2. Prentice-Hall, Englewood Cliffs.

Thomas A. Funkhouser, Michael M. Kazhdan, Philip Shilane, Patrick Min, William Kiefer, Ayellet Tal, Szymon Rusinkiewicz, and David P. Dobkin. 2004. Modeling by example. *ACM Transactions on Graphics* 23, 3 (2004), 652–663.

Ran Gal, Ariel Shamir, and Daniel Cohen-Or. 2007. Pose oblivious shape signature. *IEEE Transactions of Visualization and Computer Graphics* 13, 2 (2007), 261–271.

Ran Gal, Olga Sorkine, Niloy J. Mitra, and Daniel Cohen-Or. 2009. IWIRES: An analyze-and-edit approach to shape manipulation. *ACM Transactions on Graphics* 28, 3 (2009).

Zoubin Ghahramani, Geoffrey E. Hinton, et al 1996. *The EM Algorithm for Mixtures of Factor Analyzers*. Technical Report CRG-TR-96-1, University of Toronto.

Qixing Huang, Hai Wang, and Vladlen Koltun. 2015. Single-view reconstruction via joint analysis of image and shape collections. *ACM Transactions on Graphics* 34, 4, Article 87 (July 2015), 10 pages. DOI: http://dx.doi.org/10.1145/2766890

Vladimir G. Kim, Wilmot Li, Niloy J. Mitra, Siddhartha Chaudhuri, Stephen DiVerdi, and Thomas Funkhouser. 2013. Learning part-based templates from large collections of 3D shapes. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2013)* (2013).

Niloy J. Mitra, Natasha Gelfand, Helmut Pottmann, and Leonidas Guibas. 2004. Registration of point cloud data from a geometric optimization perspective. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*. ACM, 22–31.

Liangliang Nan, Ke Xie, and Andrei Sharf. 2012. A search-classify approach for cluttered indoor scene understanding. *ACM Transactions on Graphics* 31, 6, Article 137 (Nov. 2012), 10 pages. DOI: http://dx.doi.org/10.1145/2366145.2366156

Robert Osada, Thomas Funkhouser, Bernard Chazelle, and David Dobkin. 2001. Matching 3D models with shape distributions. In *Proceedings of the International Conference on Shape Modeling & Applications (SMI'01)*. IEEE Computer Society, Washington, DC, 154.

Maks Ovsjanikov, Wilmot Li, Leonidas J. Guibas, and Niloy J. Mitra. 2011. Exploration of continuous variability in collections of 3D shapes. *ACM Transactions on Graphics* 30, 4 (2011), 33.

Helmut Pottmann and Michael Hofer. 2003. *Geometry of the Squared Distance Function to Curves and Surfaces*. Springer.

Helmut Pottmann, Stefan Leopoldseder, and Michael Hofer. 2004. Registration without ICP. *Computer Vision and Image Understanding* 95, 1 (2004), 54–71.

Sam T. Roweis and Lawrence K. Saul. 2000. Nonlinear dimensionality reduction by locally linear embedding. *Science* 290, 5500 (December 2000), 2323–2326. DOI: http://dx.doi.org/10.1126/science.290.5500.2323

Adriana Schulz, Ariel Shamir, David I. W. Levin, Pitchaya Sitthi-amorn, and Wojciech Matusik. 2014. Design and fabrication by example. *ACM Transactions on Graphics* 33, 4, Article 62 (July 2014), 11 pages. DOI: http://dx.doi.org/10.1145/2601097.2601127

Chao-Hui Shen, Hongbo Fu, Kang Chen, and Shi-Min Hu. 2012. Structure recovery by part assembly. *ACM Transactions on Graphics* 31, 6, Article 180 (Nov. 2012), 11 pages. DOI: http://dx.doi.org/10.1145/2366145.2366199

Philip Shilane, Patrick Min, Michael Kazhdan, and Thomas Funkhouser. 2004. The Princeton shape benchmark. In *Proceedings of the Shape Modeling International 2004*. 167–178.

SHREC. 2014. 3D Shape Retrieval Contest at EUROGRAPHICS. Retrieved June 2, 2015 from http://3dor2014.ensea.fr/SHREC2014.html.

Anuj Srivastava, Shantanu H. Joshi, Washington Mio, and Xiuwen Liu. 2005. Statistical shape analysis: Clustering, learning, and testing. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 27, 4 (2005), 590–602.

Jerry O. Talton, Yu Lou, Steve Lesser, Jared Duke, Radomír Měch, and Vladlen Koltun. 2011. Metropolis procedural modeling. *ACM Transactions on Graphics* 30, 2, Article 11 (April 2011), 14 pages. DOI: http://dx.doi.org/10.1145/1944846.1944851

Johan W. Tangelder and Remco C. Veltkamp. 2008. A survey of content based 3D shape retrieval methods. *Multimedia Tools and Applications* 39, 3 (2008), 441–471.

Joshua B. Tenenbaum, Vin de Silva, and John C. Langford. 2000. A global geometric framework for nonlinear dimensionality reduction. *Science* 290, 5500 (2000), 2319.

Nuno Vasconcelos and Andrew Lippman. 2005. A multiresolution manifold distance for invariant image similarity. *IEEE Transactions on Multimedia* 7, 1 (2005), 127–142.

Elif Vural and Pascal Frossard. 2011. Discretization of parametrizable signal manifolds. *IEEE Transactions on Image Processing* 20, 12 (2011), 3621–3633.

Wenping Wang, Helmut Pottmann, and Yang Liu. 2006. Fitting B-spline curves to point clouds by curvature-based squared distance minimization. *ACM Transactions on Graphics* 25, 2 (2006), 214–238.

Kai Xu, Hanlin Zheng, Hao Zhang, Daniel Cohen-Or, Ligang Liu, and Yueshan Xiong. 2011. Photo-inspired model-driven 3D object modeling. *ACM Transactions on Graphics* 30, 4 (2011), 80.

Yong-Liang Yang, Yi-Jun Yang, Helmut Pottmann, and Niloy J. Mitra. 2011. Shape space exploration of constrained meshes. *ACM Transactions on Graphics* 30, 6, Article 124 (Dec. 2011), 12 pages. DOI: http://dx.doi.org/10.1145/2070781.2024158

Received September 2015; revised September 2016; accepted October 2016