# Learning Part-based Templates from Large Collections of 3D Shapes

Vladimir G. Kim[1] Wilmot Li[2] Niloy J. Mitra[3] Siddhartha Chaudhuri[1] Stephen DiVerdi[2,4] Thomas Funkhouser[1]
[1]Princeton University [2]Adobe Research [3]University College London [4]Google

## Abstract

As large repositories of 3D shape collections continue to grow, understanding the data, especially encoding the inter-model similarity and their variations, is of central importance. For example, many data-driven approaches now rely on access to semantic segmentation information, accurate inter-model point-to-point correspondence, and deformation models that characterize the model collections. Existing approaches, however, are either supervised requiring manual labeling; or employ super-linear matching algorithms and thus are unsuited for analyzing large collections spanning many thousands of models. We propose an automatic algorithm that starts with an initial template model and then jointly optimizes for part segmentation, point-to-point surface correspondence, and a compact deformation model to best explain the input model collection. As output, the algorithm produces a set of probabilistic part-based templates that groups the original models into clusters of models capturing their styles and variations. We evaluate our algorithm on several standard datasets and demonstrate its scalability by analyzing much larger collections of up to thousands of shapes.

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

**Keywords:** shape analysis, model collections, part-based template, segmentation, correspondence

**Links:** DL PDF WEB DATA CODE

## 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 Trimble 3D 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 [Kim et al. 2012b], model completion [Shen et al. 2012], model-based object recognition [Shapira et al. 2010; Nan et al. 2012], shape synthesis [Kalogerakis et al. 2012; Zheng et al. 2013], etc. In addition, analyzing the diversity of shapes in this database could yield insights about the geometric variations of real-world objects. Unfortunately, most Web-based repositories, including the Trimble 3D Warehouse, do not contain the necessary structural and semantic information required to support such applications. They

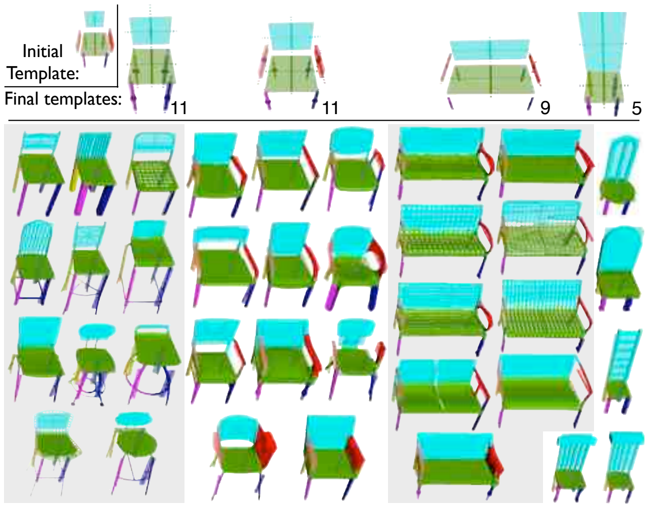

*Figure 1: Analysis results for a collection of 36 chairs. Starting from an initial template (top left), we capture the main modes of variations within the collection by the final templates (top row). In this example, the algorithm extracted template clusters for chairs without arms and with arms; a 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 individual task.*

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 necessary information for a given application.

We provide an analysis tool to derive *structure* from large, unorganized, diverse collections of 3D polygonal models (e.g., thousands of shapes within the same general object class, like chairs). 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. This allows us to group the models into clusters with similar structure; to learn a part-based deformable model of the shape variations within each such cluster; to provide consistent segmentations for all models with similar parts; and to provide correspondences between semantically equivalent points across all models in the collection. Our analysis results can then be directly used for many data-driven geometric acquisition, analysis, processing, and modeling tasks. For example, it enables the use of Trimble 3D Warehouse collections for assembly-based synthesis of new models by combining parts of existing models, which previously has only been demonstrated with much smaller model collections that require significant manual annotation [Kalogerakis et al. 2012].

Existing efforts on analyzing collections of polygonal models consider segmentation, deformation, and correspondence *separately*. For example, recent work on surface correspondence establishes links between related points on different surfaces while ignoring the part structure of objects [Kim et al. 2012a]; consistent segmentation algorithms decompose polygonal models into consistent sets of parts but do not compute point-to-point correspondences [Golovinskiy and Funkhouser 2009] or model shape variation [Sidi et al. 2011]. Finally, prior work [Fisher et al. 2011; Kalogerakis et al. 2012] on probabilistic modeling of shape variations requires manually annotated data sets with parts and correspondences already labeled. Furthermore, such algorithms do not scale to handle the thousands or tens of thousands of models in many object classes within large repositories.

In this paper, 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 tightly interconnected, 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 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 1).

The part-based deformable model is similar to the one used in [Kalogerakis et al. 2012], but it is learned from unstructured and unlabeled data, rather than labeled examples. While our 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 examples presented in this paper, the user never had to spend more than a few minutes per shape class.

Our algorithm has a linear complexity with the number of models in the collection, 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, which also results in increased accuracy. Additionally, our algorithm performs favorably in comparison on standard benchmarks to specialized algorithms for consistent segmentation and surface correspondence.

**Contributions.** In summary, we present an algorithm to

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

## 2 Related Work

This paper builds upon a large body of previous work in shape analysis of 3D model collections, including consistent segmentation, surface correspondence, and part-based deformable models.

**Surface correspondences.** Many researchers have investigated how to compute consistent alignments and correspondences within surface pairs [van Kaick et al. 2011b] or more recently on collections of 3D models [Nguyen et al. 2011; Huang et al. 2012; Kim et al. 2012a]. For example, Kim et al. [2012a] propose a diffusion-based method to compute "fuzzy correspondences" between every pair of surface points in a collection. While these methods are effective for small collections of models with limited shape variation, they do not explicitly leverage the part structure of objects, identify clusters of related shapes, or learn a model of shape deformation within the collection. As a result, they are unable to handle the diversity (arising from deformations) of shapes considered in this paper. Moreover, such methods rely on algorithms that are super-linear in complexity with respect to the number of input models and thus cannot scale to handle large collections.

**Consistent segmentation.** Although there has been significant recent work on consistent segmentation of 3D model collections, the methods are limited by the size and/or diversity of the input collections they can handle. Golovinskiy and Funkhouser [2009] present 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 small 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. [2010] 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 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 [Kalogerakis et al. 2010; van Kaick et al. 2011a; Huang et al. 2011; Sidi et al. 2011; Hu et al. 2012; Wang et al. 2012]. 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. Further, 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., [Kalogerakis et al. 2010; van Kaick et al. 2011a]) require 15% to 95% of the collection to be segmented and labeled as training data, while Wang et al. [2012] require users to interactively specify tens to hundreds of constraints. Finally, most of these analysis algorithms rely on comparing all pairs of shapes with running times in the tens of hours for hundreds of shapes [Huang et al. 2011], which makes them impractical for much larger collections.

**Part-based models.** Our approach is motivated by the successful application of deformable templates for object recognition in computer vision [Jain et al. 1998]. Felzenszwalb et al. [2005] 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 [Fergus et al. 2003; Amit and Trouve 2007; Felzenszwalb et al. 2010]. Similar approaches have been used for pose estimation [Lopez-Sastre et al. 2011], image segmentation [Eslami and Williams 2012], and viewpoint classification [Gu and Ren 2010]. Yet, in most of 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 also [Weber et al. 2000a; Weber et al. 2000b] for applications in visual recognition).

Many geometry processing tasks require and assume access to part-based model information: Shen et al. [2012] use a database of segmented models to reconstruct models from Kinnect scans; Xu et al. [2012] use part based deformation model to spawn an evolutionary model towards creation of novel and interesting model variations; Kim et al.[2012b] 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 to such applications.

Kalogerakis et al. [2012] learn a probabilistic distribution over a part-based model encoding multiple object styles, part cardinalities, part shapes, and part placements, and uses it for shape synthesis. However, the 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. [2011] 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.

## 3 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 [Xu et al. 2010]. For example, the collection of 36 chairs in Figure 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

```
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 4.3)
LearnTemplate($T, S$) (Section 4)
  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 4.1)
  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($t, s$) (Section 4.2)
  foreach candidate alignment $r$
    $d = t_{mean}$
    repeat until $\ell$ converges
      $\ell$ = SegmentShape($t, s, d, r$)
      $\{r, d, e\}$ = argmin$_{\{r,d\}}$ FitError($t, s, \ell$)
  return $\{r, d, \ell, m, e\}$ with least $e$
-----
RefineTemplateSet($T, S, C, R, D, L, E$)
  $S_L$ = SelectLearningSet($S, E$)
  $T_L$ = FitLearningSet($T, S_L, C, L$)
  $T'$ = ClusterLearningSet($T, T_L$)
  return $T'$
```

***Figure 2:*** *Algorithm pseudocode.*

better describe the shape variations within each cluster, possibly spawning new templates in the process (see Figure 2). 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 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 different strategies for template initialization.

## 4 Algorithm

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

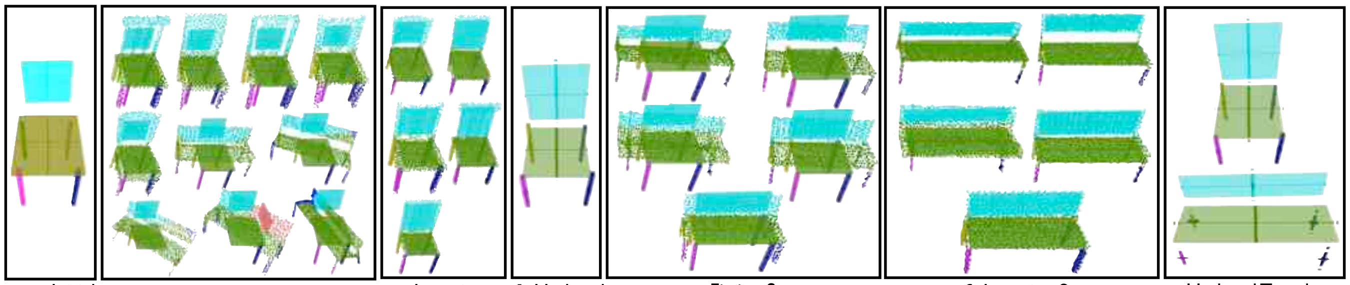

***Figure 3: Overview example.*** *Two main modes of variation are learned in a collection of 10 models (5 chairs and 5 benches). First, an initial template (a) is matched to all the models (b); however, only chairs with small fitting error are included in the Learning Set (c). Variations are learned 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 image (g) 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 by longer dashed lines.*

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 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 4.3).

Subsequently, our main algorithm (LearnTemplate in Figure 2) 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 $N_{\text{iter}}$.

Figure 3 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 4.3.

## 4.1 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 ($\ell$).

**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:

$$e(t,s,r,d,m,\ell) = E_{\text{data}}(t,s,r,d,m,\ell) + \alpha E_{\text{deform}}(t,d) + \beta E_{\text{smooth}}(s,\ell) \quad (1)$$

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:

$$E_{\text{data}}(t,s,r,d,m,\ell) = E_{s\to t} + E_{t\to s} + \gamma E_{\text{feat}} \quad (2)$$

$$E_{s\to t}(t,s,r,d,m) = \frac{1}{|P_s|}\sum_{p_s\in P_s} E_{\text{dist}}(p_s, r(d(m(p_s))))^2 \quad (3)$$

$$E_{t\to s}(t,s,r,d,m) = \frac{1}{|P_t|}\sum_{p_t\in P_t} E_{\text{dist}}(r(d(p_t)), m^{-1}(p_t))^2 \quad (4)$$

$$E_{\text{feat}}(t,s,l) \quad = \quad \frac{1}{|P_s|}\sum_{p_s \in P_s} E_{\text{feat dist}}(p_s, \ell(p_s))^2 \qquad (5)$$

where $E_{\text{dist}}$ 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 ($\lambda_2/\lambda_1$ and $\lambda_3/\lambda_1$), 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 $B$:

$$E_{\text{deform}}(d) = \sum_{b \in t} \frac{|b_p - \mu^p(B)|^2}{\sigma^p(B)^2} + \frac{|b_s - \mu^s(B)|^2}{\sigma^s(B)^2} \qquad (6)$$

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:

$$E_{\text{smooth}}(l) = \sum_{\substack{p_1,p_2 \in P_s \\ s.t. p_2 \in \text{Nhd}(p_1) \\ \ell(p_2) \neq \ell(p_1)}} -log\left(1 - \frac{\theta_{p_1,p_2}}{\pi}\right) \cdot \exp\left(\frac{dist(p_1,p_2)}{\tau^2_{Nhd}}\right) \qquad (7)$$

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

**Optimization procedure:** Minimizing this fitting energy is difficult, because the optimal rigid alignment, template deformation, shape segmentation, and point correspondences are all interdependent. 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,

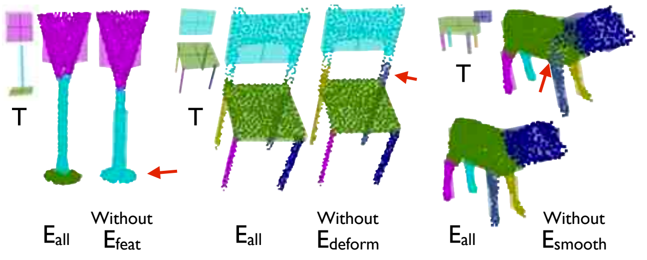

***Figure 4: Energy function example.*** *Effect of every energy term in the template fitting procedure. For all the three examples, we show a result with all the energy terms (left) and with one energy term excluded (right). (Left-to-right) The stem merges with the base if local shape features are excluded 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.*

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 $a_{\text{up}}$, and that optimal rotations around that axis are usually close to multiples of $\pi/2$. Thus, we simply align centroids and try all the four $\pi/2$ rotations around $a_{\text{up}}$. 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 $\ell$, 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 $d$ and $r$ as fixed and optimize $\ell$, 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 $\ell$. 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 $E_{\text{smooth}}$. We approximately solve the problem with the efficient graph cut algorithm described by Boykov et al. [2001].

During the second step, we treat $\ell$, $d$, and $r$ 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 $\ell$, $m$, and $d$ as fixed and only optimize $r$. We find an optimal rigid transformation that minimizes the sum of squared distances between corresponding points, using a singular value decomposition based method [Sorkine 2007].

During the fourth step, we treat $\ell$ 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 four 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.

### 4.2 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 $K_{\text{L}}$ shapes with lowest $\tilde{e}^*(t_s, s)$ to the Fitting Set, plus another $K_{\text{L}}$ 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 $\max(e_t, e_{\max})$, 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 $N_{\text{iter}}$. We have chosen conservative values of $K_L = 50$, $e_{\min} = 75$, $e_{\max} = 150$, $\tau = 4$, and $N_{\text{iter}} = 10$ empirically, and keep them the same for all results presented in this paper, noting that $K_L$ and $\tau$ could be tuned to encourage more (less) conservative evolution with smaller (larger) values.

Figure 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 $e_{\text{data}}(t, s, r, d_\mu(t), m, \ell) \le e_{\min}$. For

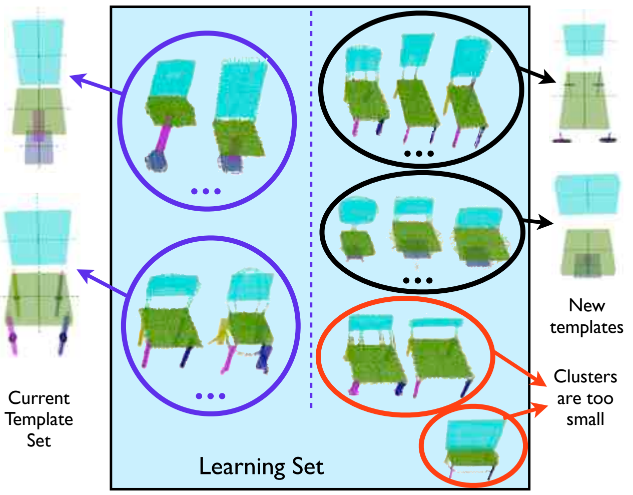

*__Figure 5: Template refinement.__ This figure demonstrates a single iteration of template refinement step. Shapes in the learning set (shaded area) are either used to update existing templates (blue), spawn new templates (black) or do not participate in learning (red).*

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 $d^*(t, s)$. Note that the spawned templates do not introduce new parts, but may only include a subset of parts of their parents.

**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 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.

## 4.3 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 $N_{\text{seg}}$ clusters generated by Voronoi diagram of iteratively farthest points. We initialize $N_{\text{seg}}$ 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).

**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.

We found that the user refinements can be valuable, especially to provide semantic cues and to make sure that all possible parts are included in the initial template. 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 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. Please refer to the supplementary material for further details.

**Datasets.** We use several datasets in our evaluation: the COSEG dataset [Sidi et al. 2011; Wang et al. 2012] containing several classes of smooth manifold shapes with ground truth per-face labeling, the correspondence benchmark of [Kim et al. 2012a] that includes collections of polygon soups obtained from the 3D Warehouse [Trimble 2012] 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 [Amazon 2012], 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 2 for more details on classes of shapes used in our analysis and sizes of the collections. Note that all of these datasets have a canonical 'up' direction, which is used by the template fitting procedure. We also scale every shape, normalizing the average distance to its centroid.

**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 6).

**Segmentation accuracy.** We evaluate quality of part-wise labeling $\ell$ produced by our method using the COSEG dataset [Sidi et al. 2011; Wang et al. 2012]. Similarly to previous methods we evaluate labeling accuracy, measuring 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 [Katz and Tal 2003], 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 1 presents labeling accuracy of our method. We test two versions of template initialization described in Section 4.3: 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.

We further include results reported by the previous methods of [Sidi et al. 2011] (Sidi) and [Hu et al. 2012] (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

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

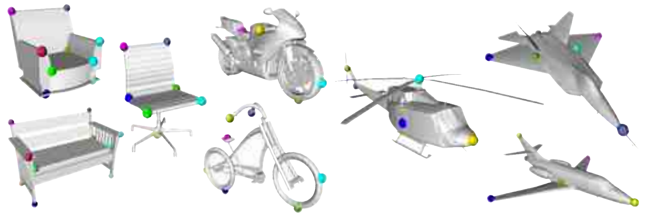

*Figure 6: Ground truth. This image illustrates some example models with ground truth feature points. Note that each model can have only a subset of ground truth points (e.g., arm rests points of chairs are absent in some chair shapes).*

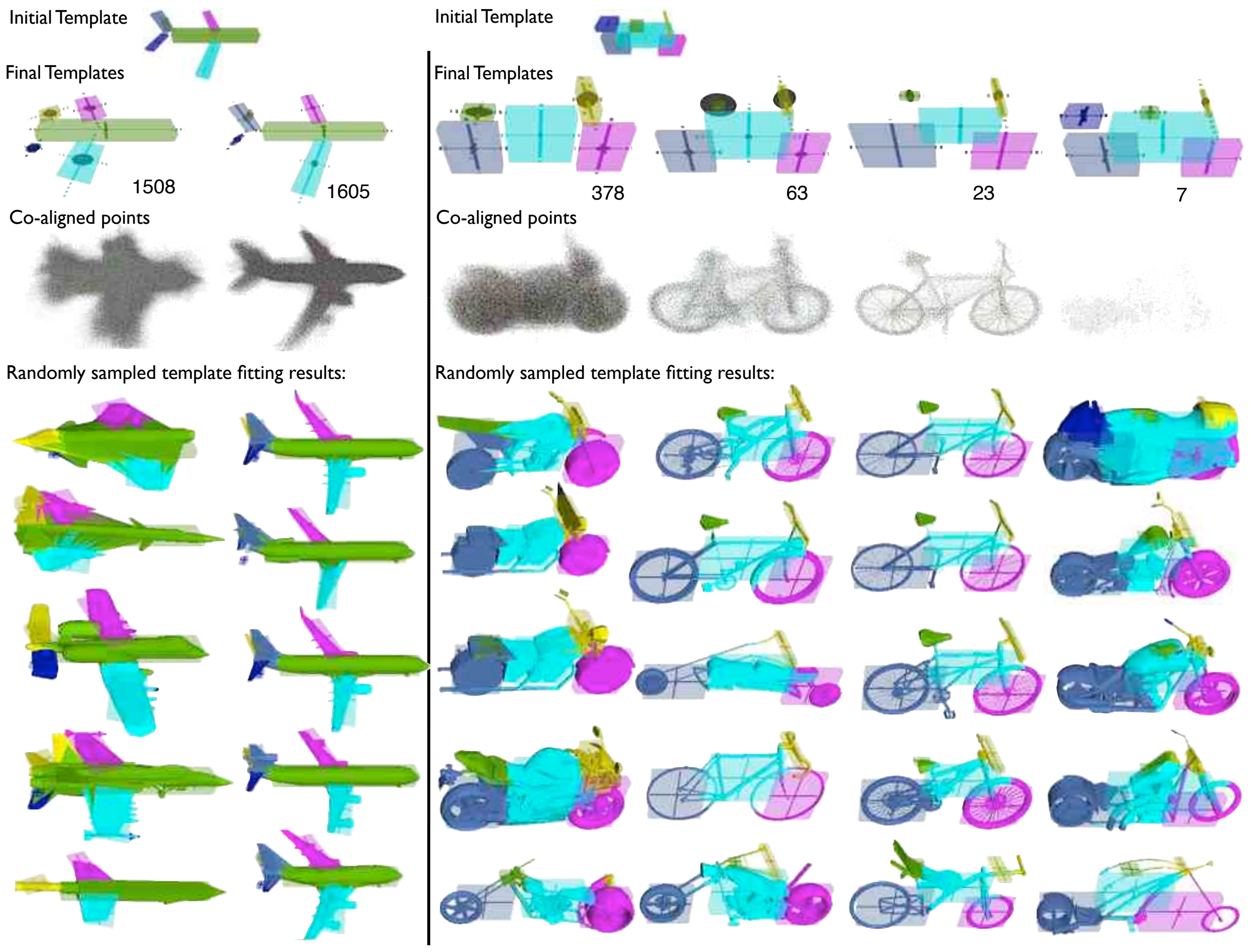

***Figure 7: Trimble 3D Warehouse data.*** *Some example templates learned from 3D Warehouse datasets along with corresponding models. Note how we automatically identify that the airplane dataset has military planes with wider variation in positions of wings (left column) and commercial airplanes with relatively uniform part scales. We also automatically identify the difference between bicycles and motorcycles.*

| 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 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).*

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 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. Figure 9 demonstrates our segmentation results for a set of lamps, note how we automatically identify three types of deformable templates and successfully label most models in the collection.

**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 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 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 8 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

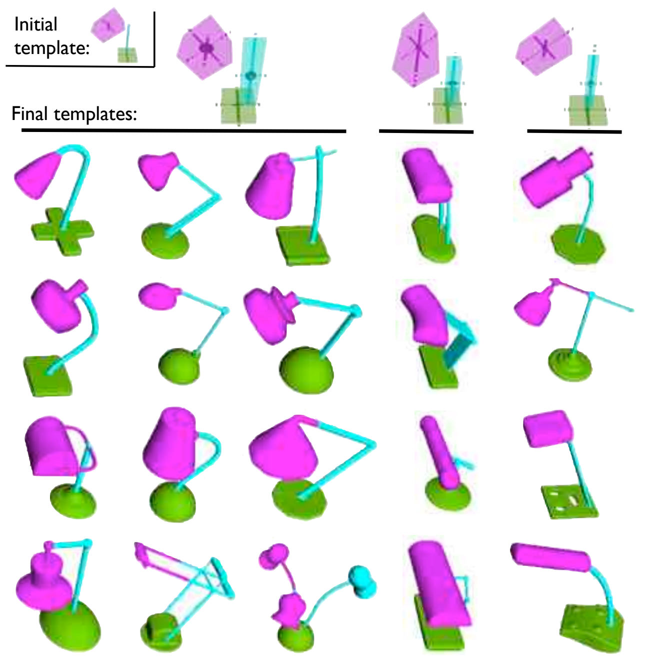

***Figure 9: Segmentation example.*** *This example illustrates analysis of a collection of 20 lamps. Note how resulting templates capture variations in position and size of lamps' head.*

significant amount of geometric variations (see also supplementary material).

**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. [2012a], 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 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. [2012] 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. [2012a] (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.,

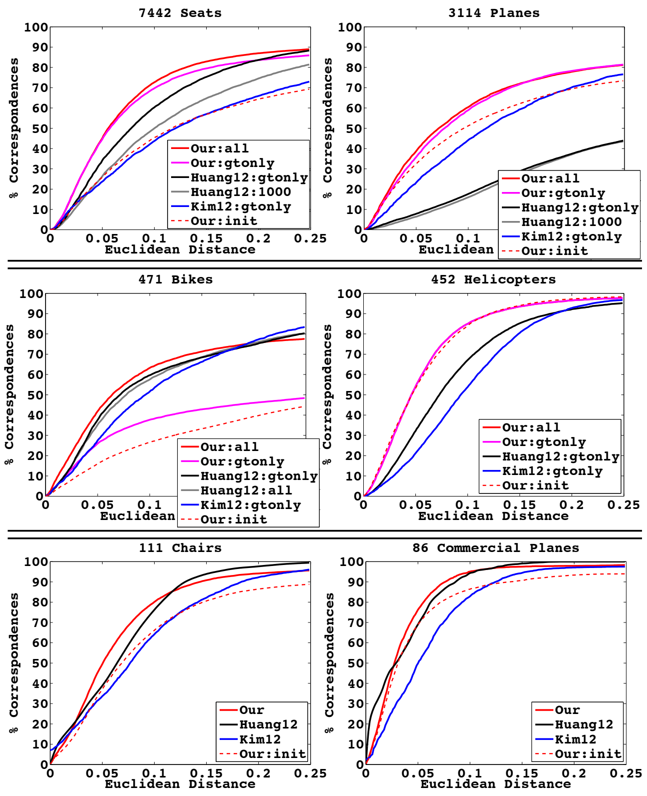

***Figure 10: Correspondence benchmark.*** *This figure demonstrates quality of correspondences produced by our method (red,magenta) relative to prior work: Huang et al. [2012] (black,gray), Kim et al. [2012a] (blue). Each point on a curve indicates fraction of correctly predicted correspondences for a given Euclidean error threshold. We execute each algorithm on ground truth models only (gtonly) and on the largest subset for which it could be run (all, 1000). Note that while we perform similar to previous techniques on small datasets with fewer variations, we gain substantially on datasets with significant part-level deformations (e.g., seats).*

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 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. [2012a] dataset in comparison to about 10 minutes for Huang et al. [2012]), 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. [2012] 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.

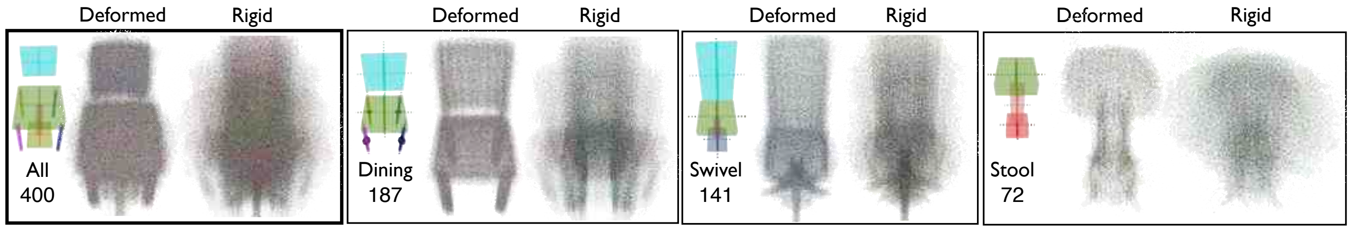

***Figure 8: Co-aligned shapes.*** *We use template deformation parameters $(r, d)$ to align all shapes from the set of 400 chairs in COSEG dataset to their corresponding mean part arrangements and scales (left image). And compare it to just using rigid transformations $r$ to co-align shapes (right image). Note that our deformation model leads much sharper co-alignments by factoring out the dominant deformation modes.*

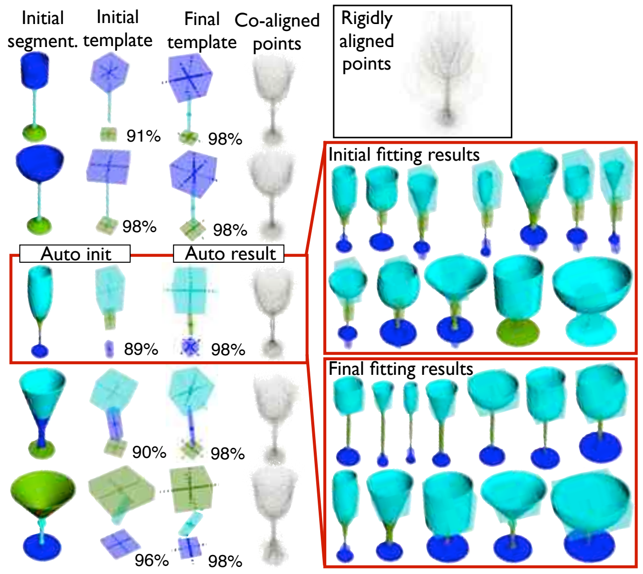

***Figure 11: Robustness to initial template.*** *Each row corresponds to six different automatic segmentations that produce different initial templates. The percentages under each template indicate labeling accuracy of the whole dataset of goblets. Our algorithm further automatically learns parameters 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.*

**Robustness to template initialization** Figure 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.

**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 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 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.

**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 $|S_{\mathrm{L}}| = 0$). Thus, our algorithm executes for at most $N$ iterations, where each iteration involves re-fitting models in Fitting Set, $O(K_{\mathrm{L}} T_{\max})$ and, possibly, an agglomerative clustering $O(K_{\mathrm{L}}^2)$. 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 $O(T_{\max} \cdot N')$. Note that except for agglomerative clustering, all the other steps can be performed in parallel.

| Class | $N$ | $\lvert T_{\mathrm{init}} \rvert$ | Total $t_{\mathrm{learn}}$ sec | Ave. $t_{\mathrm{label}}$ 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 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.*

The right plot in Figure 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 number of iterations). Table 2 further includes compute times and statistics for all the datasets.

## 6 Conclusion

We presented a shape analysis tool to derive structure from large, unorganized, diverse collections of 3D polygonal models. Given a collection of polygonal models (and optionally an initial template),

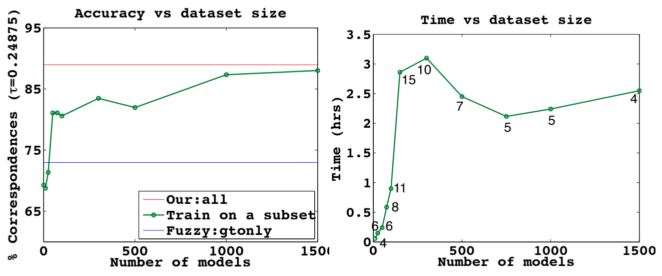

*Figure 12: **Learning templates from a subset of models.** We learned a set of templates on subsets of different size of 7442 seats dataset. The left plot shows how size of the subset influences fraction of correct correspondences for a fixed Euclidean error threshold. The right plot shows the time 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.*

we jointly partition the models into clusters with similar structure, learn a part-based deformable model of the shape variations within each cluster, provide consistent segmentations for all the models with similar parts, and provide correspondences between semantically equivalent points across all models in the collection. Our algorithm executes out-of-core, has time complexity linear in the size of the collection, and thus scales to handle very large data sets. It also performs favorably on benchmark data sets for consistent segmentation and surface correspondence in comparison to previous work.

**Limitations.** Despite these features, 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 [Fergus et al. 2003], 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. While currently we expect the user to refine the initial template and, possibly, add more parts to the initial template, but in the future, we would like to develop a fully automatic alternative. Fourth, our method is greedy, and thus is not guaranteed to converge to an optimal set of templates.

In the future, we plan to investigate new tools enabled by our analysis for applications including exploration of geometric collections, part-based shape modeling, and shape reconstruction. Furthermore, we would like to generalize our part-based templates to other types of 3D data with other types of part relationships such as hierarchical decompositions, contextual information, etc. We expect that the structural understanding of 3D collections can find many applications in robotics, computer vision, and graphics.

**Acknowledgements.** We acknowledge Qi-Xing Huang for distributing code and data. We thank Aleksey Efros and the anonymous reviewers for their comments and suggestions. The project was partially supported by NSERC, NSF (CCF-0937139 and CNS-0831374), Adobe, Intel (ISTC-VC), Google, and Marie Curie Career Integration Grant 303541.

## References

AMAZON, 2012. Amazon mechanical turk, https://www.mturk.com/.

AMIT, Y., AND TROUVE, A. 2007. POP: patchwork of parts models for object recognition. *IJCV 75*, 2, 267–282.

BOYKOV, Y., VEKSLER, O., AND ZABIH, R. 2001. Efficient approximate energy minimization via graph cuts. *IEEE transactions on PAMI 20*, 12, 1222–1239.

ESLAMI, S. M. A., AND WILLIAMS, C. 2012. A generative model for parts-based object segmentation. In *NIPS*.

FELZENSZWALB, P. F., AND HUTTENLOCHER, D. P. 2005. Pictorial structures for object recognition. *IJCV 61*, 1, 55–79.

FELZENSZWALB, P., GIRSHICK, R., MCALLESTER, D., AND RAMANAN, D. 2010. Object detection with discriminatively trained part-based models. *IEEE PAMI 32*, 9 (sept.), 1627 –1645.

FERGUS, R., PERONA, P., AND ZISSERMAN, A. 2003. Object class recognition by unsupervised scale-invariant learning. In *IEEE CVPR*.

FISHER, M., SAVVA, M., AND HANRAHAN, P. 2011. Characterizing structural relationships in scenes using graph kernels. *ACM SIGGRAPH 30*, 34:1–34:12.

GOLOVINSKIY, A., AND FUNKHOUSER, T. 2009. Consistent segmentation of 3D models. *Proc. SMI 33*, 3, 262–269.

GU, C., AND REN, X. 2010. Discriminative mixture-of-templates for viewpoint classification. In *ECCV*.

HU, R., FAN, L., , AND LIU, L. 2012. Co-segmentation of 3d shapes via subspace clustering. *Computer Graphics Forum (Proc. SGP) 31*, 5, 1703–1713.

HUANG, Q., KOLTUN, V., AND GUIBAS, L. 2011. Joint shape segmentation with linear programming. In *ACM SIGGRAPH Asia*, 125:1–125:12.

HUANG, Q.-X., ZHANG, G.-X., GAO, L., HU, S.-M., BUTSCHER, A., AND GUIBAS, L. 2012. An optimization approach for extracting and encoding consistent maps. *SIGGRAPH Asia*.

JAIN, A., ZHONG, Y., AND DUBUISSON-JOLLY, M.-P. 1998. Deformable template models: A review. *Signal Processing 71*, 2, 109 – 129.

KALOGERAKIS, E., HERTZMANN, A., AND SINGH, K. 2010. Learning 3D mesh segmentation and labeling. In *ACM SIGGRAPH*, 102:1–102:12.

KALOGERAKIS, E., CHAUDHURI, S., KOLLER, D., AND KOLTUN, V. 2012. A probabilistic model for component-based shape synthesis. *SIGGRAPH*.

KATZ, S., AND TAL, A. 2003. Hierarchical mesh decomposition using fuzzy clustering and cuts. *ACM Trans. Graph. 22*, 3, 954–961.

KIM, V. G., LI, W., MITRA, N., DIVERDI, S., AND FUNKHOUSER, T. 2012. Exploring collections of 3D models using fuzzy correspondences. *Trans. on Graphis (Proc. of SIGGRAPH)*.

KIM, Y. M., MITRA, N. J., YAN, D., AND GUIBAS, L. 2012. Acquiring 3d indoor environments with variability and repetition. *SIGGRAPH Asia*.

LOPEZ-SASTRE, R., TUYTELAARS, T., AND SAVARESE, S. 2011. Deformable part models revisited: A performance evaluation for object category pose estimation. In *ICCV Workshop on Challenges and Opportunities in Robot Perception*.

NAN, L., XIE, K., AND SHARF, A. 2012. A search-classify approach for cluttered indoor scene understanding. *ACM Trans. Graph. (Proc. SIGGRAPH Asia) 31*, 6.

NGUYEN, A., BEN-CHEN, M., WELNICKA, K., YE, Y., AND GUIBAS, L. 2011. An optimization approach to improving collections of shape maps. *SGP 30*, 5, 1481–1491.

OVSJANIKOV, M., LI, W., GUIBAS, L., AND MITRA, N. J. 2011. Exploration of continuous variability in collections of 3D shapes. *ACM SIGGRAPH 30*, 4, 33:1–33:10.

SHAPIRA, L., SHALOM, S., SHAMIR, A., COHEN-OR, D., AND ZHANG, H. 2010. Contextual part analogies in 3d objects. *IJCV 89*, 2-3, 309–326.

SHEN, C.-H., FU, H., CHEN, K., AND HU, S.-M. 2012. Structure recovery by part assembly. *SIGGRAPH Asia*.

SIDI, O., VAN KAICK, O., KLEIMAN, Y., ZHANG, H., AND COHEN-OR, D. 2011. Unsupervised co-segmentation of a set of shapes via descriptor-space spectral clustering. *ACM SIGGRAPH Asia 30*, 6, 126:1–126:9.

SORKINE, O., 2007. Least-squares rigid motion using svd, http://igl.ethz.ch/projects/ARAP/svd_rot.pdf.

TRIMBLE, 2012. Trimble 3D warehouse, http://sketchup.google.com/3dwarehouse/.

VAN KAICK, O., TAGLIASACCHI, A., SIDI, O., ZHANG, H., COHEN-OR, D., WOLF, L., , AND HAMARNEH, G. 2011. Prior knowledge for part correspondence. *CGF Eurographics 30*, 2, 553–562.

VAN KAICK, O., ZHANG, H., HAMARNEH, G., AND COHEN-OR, D. 2011. A survey on shape correspondence. *CGF 30*, 6, 1681–1707.

WANG, Y., ASAFI, S., VAN KAICK, O., ZHANG, H., COHEN-OR, D., AND CHENAND, B. 2012. Active co-analysis of a set of shapes. *SIGGRAPH Asia*.

WEBER, M., WELLING, M., AND PERONA, P. 2000. Towards automatic discovery of object categories. In *IEEE CVPR*.

WEBER, M., WELLING, M., AND PERONA, P. 2000. Unsupervised learning of models for recognition. In *ECCV*.

XU, K., LI, H., ZHANG, H., DANIEL COHEN-OR, Y. X., AND CHENG, Z.-Q. 2010. Style-content separation by anisotropic part scales. *SIGGRAPH Asia*.

XU, K., ZHANG, H., COHEN-OR, D., AND CHEN, B. 2012. Fit and diverse: Set evolution for inspiring 3D shape galleries. *ACM Trans. on Graph (Proc. of SIGGRAPH) 31*.

ZHENG, Y., COHEN-OR, D., AND MITRA, N. J. 2013. Smart variations: Functional substructures for part compatibility. *CGF Eurographics*.