A Scalable Active Framework for Region Annotation in 3D Shape Collections

Li Yi[1] Vladimir G. Kim[1,2] Duygu Ceylan[2] I-Chao Shen[3] Mengyan Yan[1]
Hao Su[1] Cewu Lu[1] Qixing Huang[4,5] Alla Sheffer[3] Leonidas Guibas[1]
[1]Stanford University [2]Adobe Research [3]University of British Columbia [4]TTI Chicago [5]UT Austin

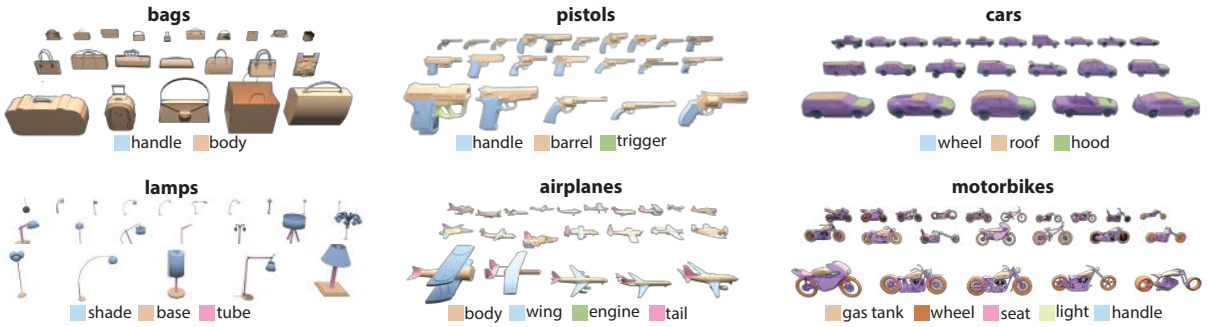

Figure 1: *We use our method to create detailed per-point labeling of 31963 models in 16 shape categories in ShapeNetCore.*

Abstract

Large repositories of 3D shapes provide valuable input for data-driven analysis and modeling tools. They are especially powerful once annotated with semantic information such as salient regions and functional parts. We propose a novel active learning method capable of enriching *massive* geometric datasets with *accurate* semantic region annotations. Given a shape collection and a user-specified region label our goal is to correctly demarcate the corresponding regions with minimal manual work. Our active framework achieves this goal by cycling between manually annotating the regions, automatically propagating these annotations across the rest of the shapes, manually verifying both human and automatic annotations, and learning from the verification results to improve the automatic propagation algorithm. We use a unified utility function that explicitly models the time cost of human input across all steps of our method. This allows us to jointly optimize for the set of models to annotate and for the set of models to verify based on the predicted impact of these actions on the human efficiency. We demonstrate that incorporating verification of all produced labelings within this unified objective improves both accuracy and efficiency of the active learning procedure. We automatically propagate human labels across a dynamic shape network using a conditional random field (CRF) framework, taking advantage of global shape-to-shape similarities, local feature similarities, and point-to-point correspondences. By combining these diverse cues we achieve higher accuracy than existing alternatives. We validate our framework on existing benchmarks demonstrating it to be significantly more efficient at using human input compared to previous techniques. We further validate its efficiency and robustness by annotating a massive shape dataset, labeling over 93,000 shape parts, across multiple model classes, and providing a labeled part collection more than *one order of magnitude* larger than existing ones.

Keywords: shape analysis, active learning

Concepts: •**Computing methodologies** → **Active learning settings; Shape analysis;**

SA '16 Technical Papers,, December 05-08, 2016, , Macao
ISBN: 978-1-4503-4514-9/16/12
DOI: http://dx.doi.org/10.1145/2980179.2980238

1 Introduction

In recent years, large and growing online repositories of 3D shapes have proved to be a rich resource for data-driven techniques in computer graphics, vision, and robotics. Recent 3D modeling interfaces, e.g. [Chaudhuri et al. 2011], and advances in text and image understanding [Lin et al. 2014] demonstrate that data-driven tools become especially effective when they rely on big, curated datasets with detailed semantic annotations. However, existing annotated datasets for 3D shapes [Chen et al. 2009] contain only a few of hundreds of models and are also largely composed of manifold, watertight meshes that are not representative of the data available in public repositories such as 3D Warehouse or ShapeNet [Chang et al. 2015]. This significantly limits their practical applications in real-world scenarios. Filling this void requires an efficient tool for *reliably* annotating massive, diverse and growing datasets of polygon soup 3D models. Automatic annotation can never be perfectly reliable as such annotation requires knowledge of detailed shape semantics, yet the size and constant evolution of 3D model databases renders fully manual annotation impractical. We view accuracy as the most critical metric, as all subsequent processing breaks down with inaccurate solutions. Consequently we provide a middle-ground approach: an active-learning method that combines human annotation, algorithmic annotation propagation, and human *verification* of *every* generated annotation.

We focus on generating semantic per-point region labels, such as object parts and salient regions, since they have proved to be extremely useful for various downstream applications. Given a collection of 3D shapes and a user-prescribed label of interest, our system produces human-verified per-point labels for each shape, indicating which points belong to the region of interest (see Figure 1).

While one can clearly add verification as a post-process for any active learning method, such naive addition would lead to a significant increase in human work. Instead, we integrate verification into the active learning framework, by using verification output as part of our feedback loop. This integration ensures accurate annotations, allows to identify human errors, andresults in substantial time savings compared to an equivalent verification-less approach (see Figure 2a and 2b). The number of human annotations required to reliably label the input data goes down dramatically, replaced by a

combination of automatic annotation and positive verification. This substitution leads to x100 time saving per each avoided annotation without any loss in quality, as substantially less time is required by a human to verify a region than to annotate it.

Our method alternates between four main steps: annotation, propagation, verification, and learning. It obtains manual region annotations for a selected set of models; then automatically propagates these annotations across the input shape database; acquires human verification of both human and automatic annotations for selected models; and finally uses the verification result to improve the automatic propagation algorithm.

When choosing the shapes to present to human workers for annotation and verification we seek to maximize *annotation efficiency*. We define it as the ratio between the number of annotations verified to be correct and the total time spent by human workers. Different from existing work on semi-supervised shape annotation [Wang et al. 2012], we explicitly model the cost of human input, i.e. semantic region demarcation and verification. We use this model to define the annotation efficiency as a utility function, and repeatedly optimize its expected value as we select the next batch of shapes for manual annotation or labeling verification. Our studies show that human verification is as accurate as human annotation (see Section 9), so to detect both human errors and ambiguous models we verify both automatic and manual annotations.

Our algorithm propagates labels from the annotated shapes to unlabeled ones by exploiting both local geometric features and global shape structure. We combine feature-based classifiers, point-to-point correspondences, and shape-to-shape similarities into a single CRF optimization over the network of shapes. Previous techniques that only rely on point-wise shape features [Wu et al. 2014] are sensitive to outliers and large shape variations often observed in massive datasets. By taking advantage of higher-level structural similarities between shapes in addition to low-level point-wise geometric cues, our method achieves superior performance on large and heterogenous datasets (see Figure 2a and 2c).

Our learning step leverages the verified annotation results to update our shape network and improve the performance of the propagation technique in subsequent iterations. In particular we learn a better shape-to-shape similarity metric, and learn better weights for CRF terms, enabling our network of shapes to adapt dynamically to the analyzed category and the label of interest (see Figure 13).

We compare our method to existing alternatives and demonstrate that it can reach the same accuracy with half the human input and scales to much bigger datasets. We test our annotation tool on ShapeNetCore [Chang et al. 2015], a massive and challenging dataset obtained from online repositories. We annotate 16 different shape categories containing 31963 shapes, and produce 93625 verified annotations with only an estimated 110 hours worth of crowdsourced work, which would have taken 780 hours without our tool.

Contribution. Our key contribution is the development of a novel scalable method for efficient and accurate geometric annotation of massive 3D shape collections. Our approach explicitly models and minimizes the human cost of the annotation effort. This contribution is made possible due to two main technical innovations. First, by embedding the verification step as part of our unified throughput maximization framework, we ensure result accuracy and dramatically improve annotation speed. Second we develop and exploit a versatile annotation propagation algorithm that leverages both global inter-shape and local point-wise features over a dynamic shape network, evolving under user feedback. Jointly these contributions significantly reduce the cost of labeling, and allow collecting verified annotations on a massive dataset that is 30 times larger than existing curated 3D shape datasets.

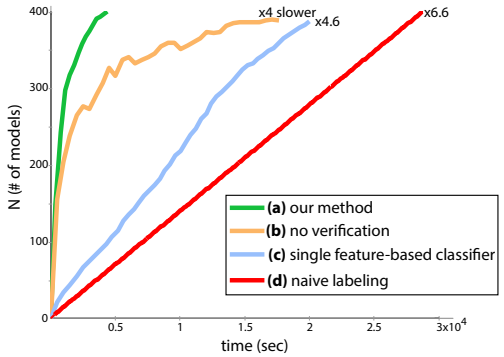

Figure 2: *This figure illustrates the number of correctly-labeled models (y-axis) as people spend more time providing input (x-axis) for a representative collection. Our result corresponds to the highest-performing curve (a), and we provide two variants of our method, one that does not include verification (b), and one that only uses local geometric features to train a single classifier for the entire dataset (c). We also show baseline cost of manually labeling every model (d). Note that all variants take substantially longer to annotate the entire dataset.*

2 Previous Work

Our work builds on previous techniques for analyzing collections of 3D shapes, as well as methods that utilize active learning to address problems such as object detection and classification.

Semantic shape labels. Many existing methods in geometry processing require semantic shape labels as part of their input. For example, libraries of 3D models with annotated semantic parts have been used for modeling-by-example [Chaudhuri et al. 2011], to synthesize shape variations [Kalogerakis et al. 2012], and to complete geometry from partial scans [Sung et al. 2015]. These methods require accurate part annotations, and currently rely on manually annotated databases. Existing work on non-expert annotation of 3D models using a crowdsourcing interface reports processing times of 3 minutes per worker to segment a shape into unlabeled parts [Chen et al. 2009], similarly, our region annotation interface requires 30 seconds per worker per region label (see Section 9). These timings render full manual annotation of large and growing collections impractical. Our active learning approach that incorporates verification allows generating massive collections of labeled region data with the same accuracy as full manual annotation (see Table 3).

Unsupervised shape segmentation. Previous work on unsupervised shape analysis has largely focused on segmenting models into compatible and geometrically separable parts. While earlier approaches segmented individual shapes [Shamir 2008], several recent works have shown that joint analysis of a group of related objects provides better geometric cues for what constitutes a semantically meaningful part [Sidi et al. 2011; Huang et al. 2011; Hu et al. 2012; Huang et al. 2014; Shu et al. 2016]. These methods typically do not associate the resulting segments with semantic labels, making their as-is outputs less useful for semantics driven applications. In contrast, we seek to identify semantic regions, which may contain multiple parts, or a fraction of a part, and may have boundaries that do not align with sharp geometric features. More importantly, it is typically not possible to generate perfectly accurate segmentations automatically, whereas our goal is to produce accurate annotations by including a human verification step.

Supervised shape analysis. Supervised learning is frequently used to obtain semantic annotations of 3D shapes [Kalogerakis et al. 2010; Lv et al. 2012; Xie et al. 2014; Makadia and Yumer 2014; Guo et al. 2015]. These methods employ traditional machine learn-

ing techniques and construct classifiers based on geometric features. Since any classifier can generate erroneous labels we provide an efficient technique that allows humans to verify every output. The performance of supervised techniques greatly depends on the availability of annotated training data that covers all shape variations. Currently, such datasets are assembled manually and are limited in scale. We develop an efficient tool to reliably annotate massive and diverse datasets from scratch, providing valuable data for future supervised analysis research.

Active and interactive image analysis. Active methods have been widely explored in image analysis. Their main advantage is that they acquire training data incrementally, enabling a classifier to predict which samples need to be labeled next to improve their performance [Vijayanarasimhan and Grauman 2008; Branson et al. 2011; Vezhnevets et al. 2012; Branson et al. 2014]. While these methods focus on improving classifiers, we focus on the end-goal of generating accurate, verified results, within a fixed human-effort budget. To achieve this goal, unlike previous active learning techniques we explicitly model human behavior in our annotation and verification phases through an integrated objective.

Recently, Russakovsky et al. [2015] proposed using a mixture of verification and refinement tasks for manually fixing failures of automatic image tagging algorithms. They use human input to detect and correct errors, but do not take advantage of this input to improve automatic labeling. In contrast, our approach utilizes both types of human input to improve label propagation. Our ability to harness verification output within an active learning framework is particularly useful for geometry processing where human verification is two orders of magnitude faster than annotation and where each annotation takes 30s to complete (see Section 9). For comparison, for images Russakovsky et al. [2015] cite a ratio of two between these tasks (10s for annotation and 5s for verification).

Active learning for shape analysis. There are very few methods that leverage active learning to annotate or classify 3D models, and, as pointed out in these works, 3D raises very different challenges from images and other 2D data. Wang et al. [2012] use active learning to segment shapes into labeled parts. They iteratively ask users to select pairs of patches and indicate whether they have the same label to actively co-segment 3D shapes. In contrast to their framework where many models are segmented with no direct human input we verify each result, confirming accuracy. We further show that even when using verification in-the-loop our method can annotate same-scale datasets using about half the human input time (see Section 9 for detailed comparison). Wu et al. [2014] offer a simpler painting interface to query the users, using a similar setup and yielding similar accuracy and efficiency to Wang et al. Boyko et al. [2014] use group verification to speed up object classification in urban point clouds. Their approach is not applicable in our setting, since their input is pre-segmented into objects, and they focus on classifying these segments.

Correspondence networks. Our automatic label propagation technique builds on recent advances in correspondence networks that enable establishing semantic relations in collections of 3D models [Kim et al. 2012] and images [Rubinstein et al. 2012] via pairwise matching and various notions of cycle consistency [Huang et al. 2014]. These works focus on designing static networks that remain fixed throughout the subsequent processing. In contrast, we develop and employ a dynamic network that changes as we acquire more human input, gradually improving our shape-to-shape similarity and relative weighting of point-wise shape features and global correspondences. At the end we learn a network that works the best for the label of interest on the input shape collection.

3 Active Learning Framework Overview

Our active learning framework produces human-verified per-point annotations of collections of 3D shapes while minimizing user supervision. The input to our system is a set of shapes in the same category (e.g., chairs) and a label of interest (e.g., "back"), and the output is a per-point boolean value that indicates whether the point belongs to the label of interest. To generate these labels for polygon soup inputs, we uniformly sample 3000 points on each shape and project per-point labels to per-face labels in a post-processing step.

Our approach reduces annotation effort by automatically propagating acquired annotations to unlabeled shapes. To achieve accurate results all labels are further verified by a human. For each automatically computed labeling that is verified to be correct we avoid an expensive human annotation, replacing it by a cheaper verification task. We use human input to improve the automatic propagation, and use the propagation algorithm to guide the decision of what human input to obtain next. Our approach iteratively alternates between these two steps, until all models are positively verified or the human-work budget is spent.

At each iteration we address four critical problems: issuing most budget-effective human tasks, quickly obtaining human input, effectively propagating the input to unlabeled shapes, and leveraging automatic propagation results and human input to improve the propagation in the next iteration. Below we provide an overview of the methods that we use to address each of these problems; Figure 3 depicts our pipeline.

Our system optimizes for two types of human tasks, annotation and verification, with the goal of maximizing framework efficiency, measured as a ratio between verified correct annotations and invested human time. To enable crowd-sourcing we execute the human tasks in batches, and thus at each iteration m we separately optimize for an annotation set $\mathcal{A}^m$ and a verification set $\mathcal{V}^m$. Since we do not have a-priori knowledge of which annotations will be supplied or verified as correct, we model the propagation of annotated or verified labels using a probabilistic formulation, and maximize the expected efficiency of the framework. This approach enables us to unify both tasks within a single utility function, which we optimize twice in our pipeline: first to decide which shapes to annotate for and second, after annotations are collected and automatically-propagated, to decide which shapes to verify (see Section 4).

To obtain human input we devise a web-based interface for each task. We designed our interface to be suitable for non-expert users and issued simple tasks to facilitate crowd-sourcing. For the annotation task, our interface presents workers with shapes in the annotation set $\mathcal{A}^m$ one-by-one and asks the user to highlight the part of interest, producing per-point labels h^m. For verification tasks we expect the majority of models in verification set $\mathcal{V}^m$ to have correct labels, and thus, to save worker time, we show viewers the entire set of models to verify at once and only ask them to select the incorrectly labeled ones. For each shape in the verification set our interface produces a value indicating whether its prediction is correct Q^m. See supplemental video and Section 5 for details.

We propagate human annotations to unlabeled shapes over a similarity network between shapes by leveraging multiple cues. First, we train classifiers for each human annotated shape that utilize local geometric feature similarities. Second, we incorporate point-to-point correspondences estimated among shapes connected in the shape network to provide global structural and contextual information. Finally, we leverage smoothness priors to favor regions of interest that are compact and continuous. We learn which of these cues are more trustworthy for a particular label as we acquire more human input. This novel combination is made possible with an

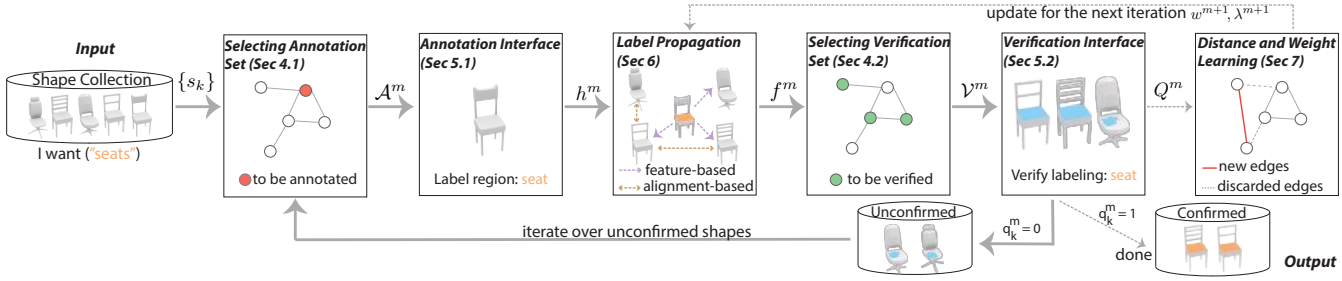

Figure 3: *This figure summarizes our pipeline. Given the input dataset we select annotation set and use our UI to obtain human labels. We automatically propagate these labels to the rest of the shapes and then query the users to verify most confident propagations. We then use these verifications to improve our propagation technique.*

efficient CRF formulation where the nodes are the shape points. Our CRF data term is defined based on the output of the feature-based classifier trained on the most similar annotated shape, and the pairwise term is derived from (i) point-to-point correspondences weighted by network-based shape similarity and (ii) spatial proximity. Our propagation procedure leverages per-point human labels h^m obtained via the annotation interface to predict labels for all points f^m that have no annotations. The setup and the optimization are described in details in Section 6.

We dynamically adapt our shape network depending on type of label that is being propagated and shape category. In particular, after each verification stage, shape similarities and the network are updated to reflect this human input. We similarly learn the relative weighting of the different terms in our CRF to ensure that it adapts well to new data. We start with empirically-chosen initial parameter values and then update them at every iteration (see Section 7).

Pipeline. We combine these four fundamental modules into a joint pipeline. At each iteration m we first construct an annotation set $\mathcal{A}^m$ and obtain human input h^m for each shape in the set. We then propagate these annotations to the remaining models and get automatic predictions f^m. We select a subset of these predictions and all unverified human annotations to be verified $\mathcal{V}^m$ and obtain human verifications for those predictions Q^m ($q_k^m = 1$ if shape s_k is positively verified by the current iteration m). Finally, we compare automatic predictions f^m and human verifications Q^m to update shape-to-shape similarities $\omega_{i,j}^{m+1}$ and CRF weights λ^{m+1}. Once a model is positively-verified, we do not select it for subsequent annotation or verification. If a shape was manually annotated, but not verified, we mark it as ambiguous and exclude from analysis. Our pipeline terminates once all models are positively verified.

4 Selecting Annotation and Verification Sets

The goal of our active learning optimization is to select shapes for *annotation* and *verification* to maximize the human work efficiency at each iteration. We measure this efficiency as the ratio between positively verified shapes N_{good}^m and time investment T^m at the end of the iteration. Our challenge is that N_{good}^m is a function of verification result Q^m, which cannot be obtained before we compute the optimal human tasks and collect human annotations. To address this challenge we treat human input Q^m as random variables that depend on annotation and verification sets $\mathcal{A}^m, \mathcal{V}^m$. Hence both N_{good}^m and T^m are also random variables with respect to $\mathcal{A}^m, \mathcal{V}^m$, and we define our utility function based on their expected values:

$$E_U^m(\mathcal{A}^m, \mathcal{V}^m) = \frac{\mathbb{E}[N_{\text{good}}^m | \mathcal{A}^m, \mathcal{V}^m]}{\mathbb{E}[T^m | \mathcal{A}^m, \mathcal{V}^m]}. \quad (1)$$

We observe that all variables at the previous iteration $m-1$ are known at iteration m, and thus we only need to compute the expected value of the change from the previous iteration:

$$\mathbb{E}[N_{\text{good}}^m | \mathcal{A}^m, \mathcal{V}^m] = N_{\text{good}}^{m-1} + \sum_k (1 - q_k^{m-1}) \mathbb{E}[q_k^m | \mathcal{A}^m, \mathcal{V}^m].$$

Here we write $Q^m = \{q_k^m\}$, where $q_k^m = 1$ if the shape s_k is verified as correct by iteration m and 0 otherwise. The summation term adds the expectation over whether a shape will be verified $\mathbb{E}[q_k^m | \mathcal{A}^m, \mathcal{V}^m]$ for each previously unverified shape.

We compute the change in total time spent by humans as:

$$\mathbb{E}[T^m | \mathcal{A}^m, \mathcal{V}^m] = T^{m-1} + \tau_{\text{ann}} |\mathcal{A}^m| + \sum_k t_{\text{ver}}(\mathbb{E}[q_k^m | \mathcal{A}^m, \mathcal{V}^m]) v_k^m.$$

We define the verification set V^m as $V^m = \{v_k^m\}$, where $v_k^m = 1$ if the shape s_k is selected to be verified in iteration m. We use $\tau_{\text{ann}} = 30$s as a constant time to annotate a shape, as estimated via our user study. For every verified shape, we model the cost of verification by observing that a person can quickly skim through the list of correct results, but would need to spend extra effort to click on an incorrect one:

$$t_{\text{ver}}(q_k^m) = \tau_{\text{ident}} + (1 - q_k^m) \tau_{\text{click}}, \quad (2)$$

We use $\tau_{\text{ident}} = 0.3$s as the time required to identify whether the annotation is correct (as the crowd worker scans through a sequence of results) and $\tau_{\text{click}} = 1.1$s denoting the time required to select the incorrectly annotated shape by a click action. Both constants are average values estimated via user studies.

Our remaining challenge for evaluating the utility function in Equation 1 is to compute the expected verification $\mathbb{E}[Q^m | \mathcal{A}^m, \mathcal{V}^m]$. We approximate this value based on the confidence of our automatic propagation procedure $C^m[k]$ defined for every shape s_k. Note that estimating this confidence when selecting the annotation set is challenging since we have not yet executed the propagation algorithm but a more reliable value can be computed when we optimize for verification set. Thus, we use two different approximations for our confidence value, pre-propagation confidence C_{ann}^m when we optimize for the annotation set, and post-propagation confidence C_{ver}^m when we optimize for the verification set. We then use isotonic regression [Zadrozny and Elkan 2002], denoted by function $r(\cdot)$ to calibrate the confidences and obtain expected values: $\mathbb{E}[Q^m | \mathcal{A}^m, \mathcal{V}^m] = r(C^m)$. We utilize positively-verified models as training data for this regression using F1 scores (harmonic mean of precision and recall) as true prediction confidences.

Next, we provide details for estimating confidences and selecting the annotation and verification sets (see Figure 4 for examples).

4.1 Selecting the Annotation Set

We optimize for an annotation set that maximizes the utility function: $\mathcal{A}^m = \text{argmax}_{\mathcal{A}} E_U^m(\mathcal{A}, \mathcal{V}^m)$.

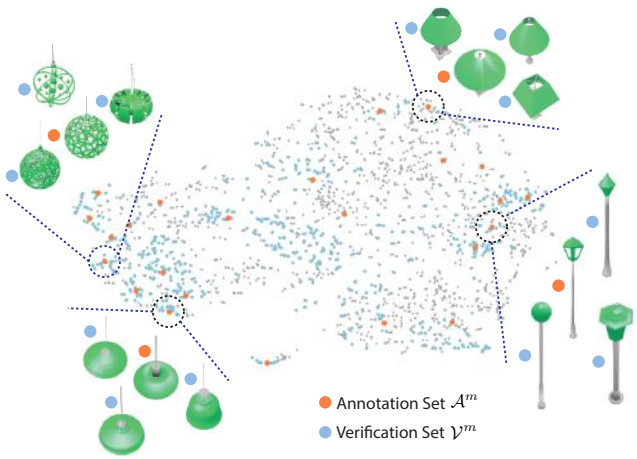

Figure 4: *At each iteration our method selects an annotation set (in orange) and a verification set (in blue) from the shape network (in gray). The models selected for annotation are distributed over the network to provide good coverage of shape variations. In contrast, models selected for verification tend to cluster close to annotated models since labels can be more reliably propagated between them.*

Pre-propagation confidence. In order to optimize our utility function we need to compute the expected verification outcome for each shape. This value depends on how confidently the automatic predictions are made for each shape. Our method leverages two terms to propagate annotations between shapes: (i) feature-based propagations from the most similar annotated shape, and (ii) point-to-point correspondences weighted by global shape similarity. Intuitively, in our first term the confidence of the predictions, c_k, for each shape s_k is likely to increase as the similarity, $\omega_{k,x}$, between this shape, s_k, and its most similar annotated shape, s_x, increases. The second term, on the other hand, is more global, enforcing similar predictions, and thus similar prediction confidences, for shapes that share correspondences. Combining these two intuitions, we estimate, $C^m_{\text{ann}} = \{c_k\}$, the prediction confidence for each shape as:

$$C^m_{\text{ann}} = \arg\min_c \sum_k \| \max_{x \in \mathcal{A}^m} \omega^m_{k,x} - c_k \|^2_2 + \lambda^m_1 \sum_{k,j} \omega^m_{k,j} \|c_k - c_j\|^2_2,$$

We weight the second term by λ_1, the same weight that is used for correspondence term in the CRF optimization. Figure 5a demonstrates the correlation between our estimate C^m_{ann} and true q^m values obtained from ground truth at iteration $m = 0$ for a representative dataset. Please note that no human input was collected at this stage to produce this confidence estimation.

Given any candidate annotation set, we can use this confidence estimate to compute the optimal verification set as described in Section 4.2 and compute the utility. Annotation set optimization thus can be performed by generating multiple candidate sets and choosing the one with maximum utility, as described next.

Annotation set optimization. Our goal is to select an annotation set $\mathcal{A}^m$ that will maximize the utility function, i.e. we choose $\mathcal{A}^m$ to consist of models that are expected to *confidently annotate* their unlabeled neighbors. We select a fixed number of shapes at each iteration $|\mathcal{A}^m| = n_{\text{ann}}$ (where $n_{\text{ann}} = 0.01N$). To perform such a selection we use a variant of beam search, where at any time we keep a set of sub-optimal solutions represented by $N = |\{s_k\}|$ candidate subsets Ω_k. Initially, the subsets just include one shape ($\Omega^0_k = \{s_k\}$). At every iteration $i > 0$, we evaluate every possible k^{th} subset that includes the shape s_k and any set Ω^{i-1}_l from the

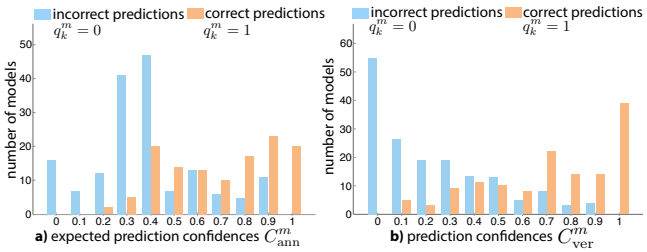

Figure 5: *Given a dataset of models with ground-truth human verifications Q^m, we demonstrate the distribution of our prediction confidences, C^m (x-axis) (blue bars for incorrectly propagated predictions and brown for correct). A good confidence predictor is expected to place higher blue bars closer to 0 on the x-axis, correctly identifying the bad predictions, and higher yellow bars closer to 1 on the x-axis, correctly identifying the good predictions. (a) While the estimates used to select the annotation set in the beginning, i.e. with no human input yet, are less accurate, they are sufficient to bootstrap our algorithm. (b) These estimates significantly improve after a single iteration of annotation propagation enabling to optimize for a more reliable verification set.*

previous iteration, and pick the one that maximizes utility function:

$$\Omega^i_k = \underset{\mathcal{A}^m = \Omega^{i-1}_l \cup \{s_k\}, l=1\ldots N}{\text{argmax}} \mathbb{E}_U(\mathcal{A}, \mathcal{V}),$$

where $\mathbb{E}_U(\mathcal{A}, \mathcal{V})$ is defined by Equation 1. When the cardinality of Ω^i_k reaches n_{ann} we greedily pick a shape to remove (maximizing the utility function for a subset with cardinality $n_{\text{ann}} - 1$), thus at next iteration when we add a new element our cardinality remains constant $|\Omega^{i \geq n_{\text{ann}}}_k| = n_{\text{ann}}$. This element removal allows us to refine our subsets and we terminate either once subsets Ω stop changing or after $10n_{\text{ann}}$ iterations. We set $\mathcal{A}^m$ to be the subset that maximizes the utility function.

4.2 Selecting the Verification Set

After we obtain user input for our annotation set and propagate the labels, we compute our verification set by maximizing the same utility function: $\mathcal{V}^m = \text{argmax}_{\mathcal{V}} E^m_U(\mathcal{A}^m, \mathcal{V})$; this time using $\mathcal{A}^m$ as a fixed constant.

Post-propagation confidence. We now refine our confidences by leveraging the results of label propagation. In particular, we use the per-point label probability $\rho_p = \Pr(f^m(p) = 1 | h^m)$ computed by our propagation step (see Section 6). We denote prediction entropy: $J(p) = \rho_p \log \rho_p + (1 - \rho_p) \log(1 - \rho_p)$, a value that increases for more confident predictions as ρ_p gets closer to 0 or 1, and we further normalize it to be a positive value in a range $[0, 1]$: $\tilde{J}(p) = 1 + \frac{J(p)}{log(2)}$. We now use this normalized entropy to get refined confidences:

$$C^m_{\text{ver}}[k] = \frac{1}{|s_k|} \sum_{p \in s_k} \tilde{J}(p) \cdot C^m_{\text{ann}}[k],$$

where $|s_k|$ is the number of points sampled on a shape s_k. Note that the correlation between the ground truth confidences and our estimations improves when optimizing for the verification set since we have more input obtained from the annotation propagation step (see Figure 5). We use these confidence estimates when optimizing for the verification set.

Verification set optimization. Since our approach directly benefits from positive verification and does not benefit from a negative one, we select a verification set $\mathcal{V}^m$ to contain annotations that are most likely to be positively-verified. We consequently initialize the

set with all the human annotated models in the current iteration. To compute the rest of the optimal verification set we sort the remaining shapes with respect to their expected verification values $\mathbb{E}(Q^m|\mathcal{A}^m, \mathcal{V}^m))$ (highest to lowest), and iteratively add models to the verification set $\mathcal{V}^m$ in that order until the utility function stops increasing. This strategy is guaranteed to generate the optimal solution with respect to our utility function and we refer the reader to the supplemental material for justification. Figure 4 demonstrates an example verification subset (blue points), note how verified models are located near the annotated ones. If a human annotated model fails verification it is marked as ambiguous and not processed further (see Section 8 for more details). Other models that fail verification are returned to the pool for further processing.

5 Obtaining Human Input

Once we find the sets of models to be annotated or verified, we interface with workers to obtain the data. Our interfaces are simple, not requiring any special expertise, and thus suitable to be distributed over the web (and crowd-sourced if necessary). We have developed two interfaces targeted at obtaining per-point labels h^m and per-shape verifications Q^m.

5.1 Annotation interface

The input to our annotation interface is a set of 3D models and a user instruction to highlight model regions corresponding to a specific label of interest; the output is the manually generated labels $h^m \in \{0, 1\}$, where $h(p \in s_k) = 1$ iff point p on shape s_k belongs to the region of interest. To avoid requiring users to perform complicated 3D manipulation we show them only 2D images of shapes they need to annotate. We generate the images by selecting two views that cover the largest surface area for each model. The user is then asked to highlight a particular region (e.g. "bag handle"). This interface is very simple, since for every task we only show a single model from a single viewpoint at a time and only ask the viewer to label a single semantic region. We provide the users with a painting interface where they use a resizable brush to highlight the region of interest (see Figure 6). We use the underlying depth image to ensure that the brush (regardless of its size) does not spill over depth discontinuities. This facilitates quick and accurate labeling of large and small areas (see supplemental video for an example). Note that the annotator can also specify that part does not exist or indicate that the part is not visible from the current view. We propagate the labels to hidden areas on the models and to other models, as explained in the next section.

5.2 Verification interface

The input to our verification interface is a set of shapes and their predicted annotations f^m, and our goal is to acquire the per-shape quality Q^m of these predictions. We expect the set to consist of largely correctly labeled models. To minimize the effort in scanning through multiple models, we render a group of models at a time and ask viewers to select all models with incorrect labels. The size of the verification set is determined by our optimization and we possibly split this set into multiple verification tasks with at most 100 models at a time to ensure consistent amount of effort for each crowd task. In order to ensure verification quality, we insert a small set of models (2 models for each labeling task in our case) with incorrect labels as a *gold standard* [Su et al. 2012] into each verification task. Verification results from the users who fail to reject these incorrectly labeled models are not used in analysis, and their tasks get re-assigned to other workers.

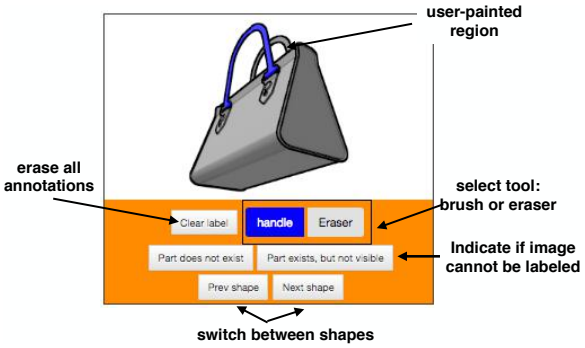

Figure 6: *User interface for acquiring semantic labeling. Our interface is designed for crowdsourcing and is therefore very lightweight and simple; the user only annotates a single label, on a single shape, from a single viewpoint.*

6 Propagating Labels

At each iteration of our pipeline we propagate manual per-point annotations h^m to the entire dataset, computing predicted per-point labels f^m on all the input models. The annotations are propagated both to unlabeled shapes and to those points on labeled shapes that were hidden from view in the annotation interface. To simplify notation we will not use the iteration superscript for cases when all variables are given at a single iteration m.

Energy Function. We optimize all labels simultaneously by minimizing an energy function that accounts for three criteria: local geometry; global correspondence; and label continuity or smoothness. The first criterion evaluates at each point a local feature based labeling metric, learned on similar, user-annotated models. The second incorporates point-to-point correspondences that are estimated with global shape matching techniques. Correspondences provide additional global structural and contextual cues (e.g., chair legs typically share the same relative location), which cannot be captured with feature-based point classifiers. Lastly, our smoothness term encodes the expectation for regions of interest to be compact and continuous. We encode these criteria using a CRF-based energy function $E(f) = -\log(\Pr(f^m|h^m))$, maximizing the likelihood of overall prediction f^m given human labels h^m. We decompose our objective as follows:

$$E(f) = \sum_{p \in S} \psi_{\mathrm{f}}(p) + \sum_{p_1, p_2 \in S \times S} \lambda_1 \psi_{\mathrm{c}}(p_1, p_2) + \sum_{p_1, p_2 \in S} \lambda_2 \psi_{\mathrm{s}}(p_1, p_2), \tag{3}$$

where $p \in S$ is the set of all points on all shapes (we sample $|s_k| = 3000$ points per shape). The feature-based term, $\psi_{\mathrm{f}}(p)$, encourages the label of each point p to be similar to the classification result obtained by an ensemble of feature-based classifiers built from the annotated shapes; ψ_{c} promotes corresponding points across different shapes to have similar labels, whereas ψ_{s} promotes nearby points on the same shape to have similar labels. The weight λ_i are learned from the annotation output as discussed in Section 7. For estimating per-point label probabilities $\Pr(f^m(p) = 1|h^m)$ in Section 4 we use marginal inference over the CRF.

Feature-based Classification. In general, one can expect points with similar local geometric features to have a similar label (e.g., a chair seat is typically flat). However, this correlation between label similarity and local geometric feature similarity holds only for overall relatively similar objects (see Figure 7). Motivated by this observation we use our already annotated models to learn a feature based per-point classifier, which assigns each point a likelihood of being associated with our label of interest. To prioritize label propagation between more similar shapes, instead of building a single

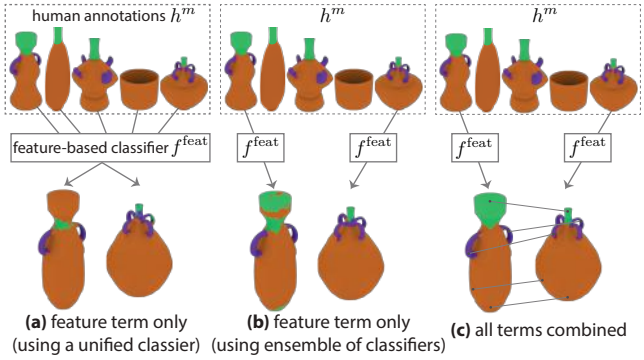

Figure 7: *Given a set of human annotated models (top) and two unlabeled shapes (bottom), the labels can be propagated by (a) using only the feature-based classifier trained on all annotated models, (b) using only the most similar model to train the classifier, and (c) using correspondences and smoothness terms. Incorporating both (b) and (c) significantly improves the result in comparison to currently used alternatives (a).*

feature-based classifier for the entire dataset, we build a separate classifier from each annotated shape s_k. We use the ensemble of these classifiers to annotate the unlabeled shapes and hidden points on labeled shapes. For each shape to be processed, we use the classifier built from its nearest annotated shape. Figure 7 demonstrates how building a single classifier that takes all training data into account (a) is inferior to using the nearest shape (b).

To build a classifier, we view s_k as a composition of positive examples $\{p \in s_k : h(p) = 1\}$, negative examples $h(p) = 0$, and hidden points $h(p) = \emptyset$ (not used in training) that were not visible from any of the annotated views. We compute the following local geometric features for each point: shape diameter function (SDF), absolute curvature, height, and spectral features in fixed-size Euclidean ball (we use the publicly available implementation of Kim et al. [2014] to compute these features). We then train a per-point classifier, $f^{\text{feat}}_{s_k}$ on negative and positive samples using logistic regression. Each such classifier generates a probability distribution function $\Pr(f^{\text{feat}}_{s_k}(p) = l)$, where $l \in \{0, 1\}$.

To propagate the labels reliably in a diverse collection, for each unlabeled shape s_i we pick the most similar annotated shape s_k. This choice is based on a shape similarity measure ω, discussed in Section 7, indicating how reliably each pair of shapes can exchange labels. We set:

$$\psi_{\text{f}}(p \in s_i) = -\log(\Pr(f^{\text{feat}}_{s_k}(p) = f(p))), \text{where} \quad (4)$$
$$k = \underset{j}{\operatorname{argmax}}\, \omega_{j,i}.$$

Cross-Shape Correspondence. We encourage corresponding points on similar shapes to have similar labels. While our method can work with any shape matching algorithm we use a simple and efficient co-alignment technique suitable for meshes, polygon soups, and point clouds, well suited for our setup. Similar to previous work, we construct a shape network $\mathcal{N}$ where we connect each shape to its $n_{\text{nhd}} = 30$ most similar shapes based on global shape descriptors $\omega^0_{i,j}$ (Section 7). We jointly deform all shapes so that the neighbors in the network are aligned. In particular, we use the hierarchical joint alignment algorithm of Chang et al. [2015], and then refine the alignments with free-form deformations [Huang et al. 2013]. We define the weighted correspondence for a pair of points as:

$$\text{corr}(p_1 \in s_i, p_2 \in s_j) = \omega^m_{i,j} e^{-||\text{pos}_{p_1} - \text{pos}_{p_2}||^2_2} \quad (5)$$

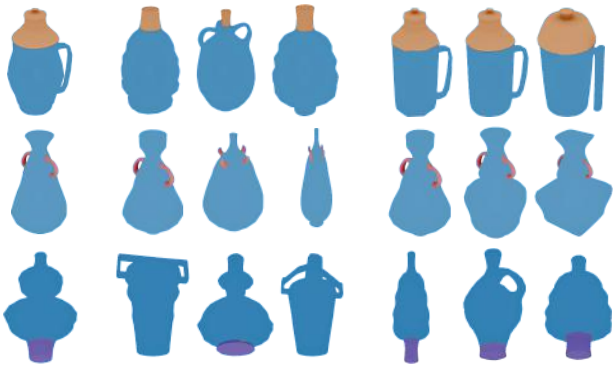

Figure 8: *We illustrate the effectiveness of learning shape similarity on the "vases" dataset. Given a query model (a), we show its three most similar models with respect to the metric ω^0 (b). As we get more human input our metric ω^1 becomes more part-aware (c).*

iff there is an edge between s_i and s_j in $\mathcal{N}$, and p_1 and p_2 are mutually closest points after the alignment, and set $\text{corr}(p_1, p_2) = 0$ otherwise. The point coordinates $\text{pos}_{p_1}, \text{pos}_{p_2}$ are computed on the co-deformed shapes, with their common bounding box normalized to a unit cube. Our correspondence term penalizes corresponding points that have different labels:

$$\psi_{\text{c}}(p_1, p_2) = \delta\left(f(p_1) \neq f(p_2)\right) \cdot \text{corr}(p_1, p_2) \quad (6)$$

Figure 7 demonstrates that adding the correspondence term (c) leads to superior performance.

Smoothness Term. Our smoothness term encourages adjacent points on the same shape to have similar labels. For each point p on a shape s_i we find $n_{\text{smooth}} = 5$ nearest points, and define smoothness analogously to correspondence:

$$\psi_{\text{s}}(p_1, p_2) = \delta\left(f(p_1) \neq f(p_2)\right) \cdot e^{-||\text{pos}_{p_1} - \text{pos}_{p_2}||^2_2} \quad (7)$$

iff p_2 is among n_{smooth} nearest neighbors of a point p_1. Note that adding the smoothness term leads to a marginal quality improvement in comparison to correspondence term which significantly boosts performance (see Figure 13).

Optimization. We solve Equation 3 using a spectral optimization method [Leordeanu and Hebert 2005], obtaining per-point labels f for all shapes. To scale the optimization to big datasets, we perform k-means clustering using global shape descriptors, breaking the optimization into several subproblems, aiming for average cluster size of about 300 shapes. Note that our optimization does not require each cluster to contain annotated models since the feature-based classifiers can be applied across clusters.

Per-face Segmentation. While all our processing is performed on points, our initial models are typically either polygon soups or closed meshes. To use our output in downstream applications we project per-point labels to per-face labels using a graph-cut based solution similar to Sidi et al. [2011]. Specifically, we assign a constant data cost of 1 if the label of a face is not consistent with the label of its closest point. We assign a constant smoothness cost of 0.2 for each pair of neighboring faces that are assigned different labels. On extra coarse meshes a single triangle can span several labels; to avoid such cases we iteratively split the longest edges until all the edge lengths are at most 0.02 of the mesh radius.

7 Learning Similarities and Weights

The shape-to-shape similarity metric, $\omega_{i,j}$, and the weights λ_i of the terms in Equation 3 play an important role in our system and

are learned incrementally, as more user input becomes available. At each iteration of our method, we update these parameters after annotations have been propagated and verified, since we then possess additional information about which propagations succeeded. We improve our similarity metric $\omega_{i,j}$ by learning how to best map global shape features to similarities. We also learn better weights λ_i that help us to identify which cues (i.e. feature-based or correspondence-based) should be given more weight for a particular shape category and label.

Learning Shape Similarity. Estimating accurate shape similarity is the key to successfully propagating annotations. Before collecting any human input (at iteration $m = 0$) we simply define the shape-to-shape distance in the space of global shape descriptors. We set $\omega_{i,j}^{m=0} = e^{-(d(x_i,x_j)/\sigma)^2}$ where x_i and x_j are light field descriptors of shapes s_i and s_j [Chen et al. 2003], $d(\cdot,\cdot)$ is the cosine distance between vectors, and $\sigma = 0.2$. Once we collect human annotations and verification results we refine our estimate of pairwise shape similarity based on the success of automatic predictions Q^m. We use this data to learn a better distance metric for global shape descriptors using large margin nearest neighbor (LMNN) [Weinberger et al. 2005] as our metric learning algorithm. We first build feature based classifiers for every shape with verified labels, and compute pairwise prediction F1 scores among them. The computed F1 scores serve as true pairwise similarity score and we use 3 shapes with the best F1 score as target neighbors in LMNN. We constrain the transformation matrix optimized in LMNN to be a diagonal matrix (this is done since light field descriptors yield big feature vectors, so we want to avoid overfitting). We update the shape similarity measure as $\omega_{i,j}^m = (1-\alpha)\omega_{i,j}^{m-1} + \alpha e^{-(d(M^m x_i, M^m x_j)/\sigma)^2}$, where $\alpha = 0.3$ is the learning rate. Figure 8 illustrates how nearest neighbors are refined as we learn a better metric for a collection of shapes.

Learning Weights. The weighting between feature, correspondence, and smoothness based terms in the objective function (Equation 3) plays a significant role in the quality of final predictions; the optimal weight values λ_1 and λ_2 can differ drastically depending on the shapes and regions of interests. To ensure reasonable performance at iteration $m = 0$ we empirically initialize $\lambda_1^{m=0} = 10$ and $\lambda_2^{m=0} = 2$, and then update the weights at each iteration, setting:

$$\lambda_i^m = (1-\alpha)\lambda_i^{m-1} + \alpha\lambda_i^{*m}, i = 1, 2. \quad (8)$$

Here λ_i^{*m} is the optimal parameter estimated via cross validation and $\alpha = 0.3$ is the learning rate. To estimate λ_i^{*m} we take the set of correctly labeled shapes: $\Omega^{\text{verified}} = \{k : q_k^m = 1\}$, and sample 10 subsets of Ω^{verified} each containing $b|\Omega^{\text{verified}}|$ shapes, where $b = \frac{\sum_k a_k^m}{N}$ (we want these subsets to be proportional in size to the number of annotated shapes). We run our joint optimization on each subset multiple times with 21 different values λ_i sampled uniformly on a log scale in the range $[0.1\lambda_i^{m-1}...10\lambda_i^{m-1}]$, we then set λ_i^{*m} to the value that gives the best average performance.

8 Annotation in the Wild

To test the robustness and effectiveness of our method we aim to test it on a massive and diverse dataset of typical real-life models. Since existing benchmarks are too small and have limited data variety, we use ShapeNetCore [Chang et al. 2015] as a source of data. This database contains shapes from real public repositories such as 3D Warehouse [Trimble 2015] and all models are classified into several categories. Our allocated budget for this experiment was roughly 100 human hours, thus we selected a subset of categories from ShapeNetCore which we estimated could be processed in this time frame. We consequently *fully* annotate 16 diverse categories

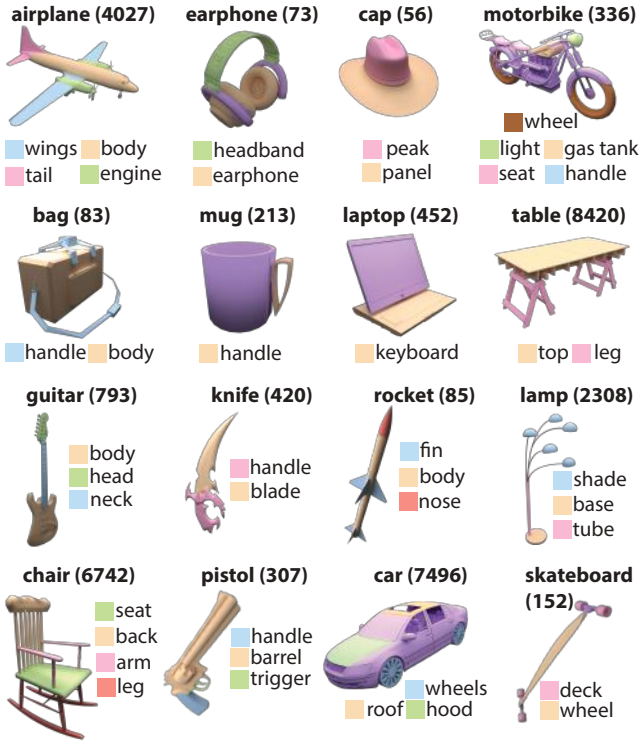

Figure 9: *We use our method to get part labels for more than 30,000 models in 16 shape categories in ShapeNetCore. We denote the number of models in each category in parentheses.*

Category	N	L	Ann	Ver	T	FMF
Guitar	793	3	270	2380	2.8	7
Knife	420	2	149	828	1.5	4.6
Pistol	307	3	176	893	1.7	4.5
Lamp	2308	3	696	6783	7.4	7.6
Chair	6742	4	2112	26152	26.2	8.3
Table	8420	2	999	16022	14.6	9.1
Mug	213	1	33	207	0.4	3.9
Airplane	4027	4	2142	15757	22.6	5.8
Bag	83	2	18	160	0.2	7.7
Cap	56	2	18	109	0.2	5.4
Earphone	73	2	25	143	0.2	5
Laptop	452	1	18	444	0.2	14.8
Skateboard	152	2	24	304	0.3	8.6
Rocket	85	3	35	240	0.5	4.2
Motorbike	336	5	264	1568	2.9	4.5
Car	7496	3	2215	21635	27.8	6.5
Total	31963	42	9194	93625	109.6	7.1

Table 1: *Annotation statistics: For each category we report number of models (N), number of labels (L) (to label the entire dataset one needs $N{\cdot}L$ annotations), number of human-annotated labels (Ann), number of human-verified correct labels (Ver), estimated total human work in hours (Time), and force-multiplication factor (FMF).*

(see Table 1) containing 31963 models. We specify 1-5 salient regions per object category identifying parts that are shared among multiple models (see Figures 1, 9 for some examples). Note that our labeling does not partition the surface; we can process overlapping labels or labels that do not jointly cover the shape, detecting only "interesting" regions.

Statistics and FMF. We present our results in Table 1 demonstrating for each category the number of shapes, labels, acquired annotations, positively-verified annotations, and estimated human work time (in hours). Our method is estimated to utilize 110 hours of

worker time, while naive baseline would have taken 780 hours. Since collecting exact timings for crowdsourced tasks is problematic, due to potential user interruptions, the time above is computed based on the number of different tasks performed and their estimated completion times evaluated in our user study (Section 9). In total we obtained 93625 verified correct annotations (out of those 9564 are confirmation that part does not exist on an object). Figure 10 demonstrates how the number of positively-verified annotations grows as our system acquires more user input for several representative labels. We use force-multiplication factor (FMF) to measure how much our system amplifies user efficiency. We compute FMF as the ratio between total one-by-one annotation time and our annotation time. Our method results in average FMF=7.1 across all categories. This factor naturally drops as larger fraction of data is annotated: initial propagations are usually successful among similar models, and more diverse models have to be annotated one-by-one. We depict this behavior in Figure 11 where x-axis is the fraction of annotated data and y-axis is the FMF. Note that we recompute FMF after each iteration of our algorithm.

Our system is remarkably efficient in the context of similar systems designed for other types of data. For a reference, Boyko et al. [2014] have FMF=1.7 for classification of point clouds, and Russakovsky et al. [2015] report FMF=2.8 for object detection in images. While both of these tasks are not directly comparable to ours, these factors highlight the complexity of obtaining verified results on data obtained "from the wild," and suggest that the shape labeling problem greatly benefits from *appropriate* active supervision.

Verification. We evaluate the quality of the obtained data by asking an expert to re-verify a subset of the final results (see next section for details). We find that that false positive rate for crowd-verifications is very small (less than 4%) which suggests that our dataset can be used for various downstream applications. The expert re-verified results can also be used as a reliable benchmark for future shape analysis algorithms.

We also found that various categories have 2-7% ambiguous geometry, i.e., cases when none of the human annotations passed verification. Most often these are shapes that have a degenerate version of the queried feature region, which some may consider as non-existent. The overall percentage of such labels across all the processed models was 3%. Please refer to the supplementary material for an example set of such ambiguous shapes.

Applications. The created dataset is more than an order of magnitude larger than existing segmented geometric datasets (e.g., compare to PSB with 380 segmented shapes [Chen et al. 2009] and COSEG with 1090 [Wang et al. 2012]), drastically expanding the capabilities of downstream applications. It can serve as a benchmark, a training dataset, or as a source for data-driven modeling and semantics-aware reconstruction techniques.

9 Evaluation and Comparison

We validate our method by comparing it against previous work and evaluating significance of various design choices.

Dataset. We evaluate our method on the COSEG benchmark used by previous techniques [Wang et al. 2012]. It contains several collections of co-segmented manifold meshes with ground truth per-face labels. In addition, we have created a ShapeNetCore-benchmark dataset by randomly selecting a subset of 12000 models for 4 categories (chair, lamp, airplane, table) from ShapeNetCore. We ensure the ground truth labelings for these datasets by asking an expert to re-verify the results collected with our method and exclude incorrect labelings (less than 4% of the examples).

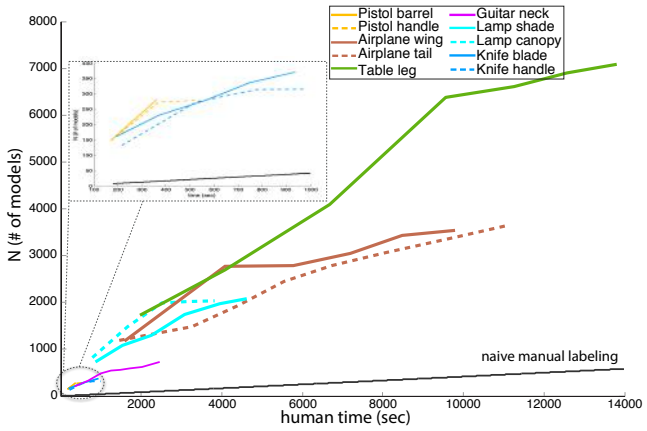

Figure 10: *Number of positively-verified labeled shapes (y-axis) as a function of human input time (x-axis) for representative labels in our data. As expected the graph is monotonically increasing, and flattens out as time progresses and the algorithm encounters more diverse models. Note the relative inefficiency of manual labeling.*

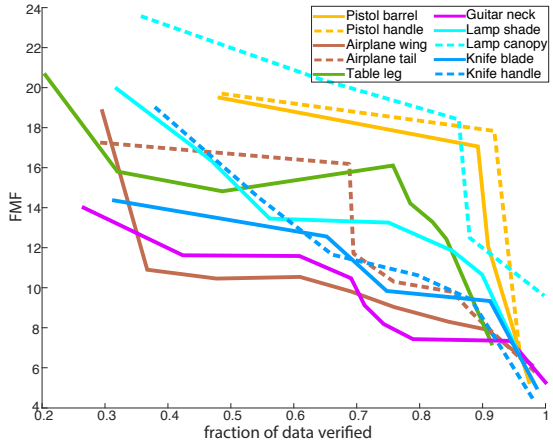

Figure 11: *Force-multiplication factor (FMF, y-axis) over time. Predictably FMF drops as the system annotates a higher fraction of the model collection (x-axis).*

Evaluation Metrics. There are two relevant dimensions to evaluating a semi-supervised method: amount of supervision and accuracy. We measure the amount of user supervision based on invested time. Since this time varies between different users we use estimated average times for annotation and verification to obtain a single time estimate. We also use force-multiplication factor, FMF, to estimate the efficiency of different methods. Accuracy is commonly computed as a fraction of correctly labeled faces, and we use this metric when comparing to previous work. It is worth noting that achieving 100% accuracy is not possible even with full manual annotation since some parts have fuzzy boundaries allowing for many equally plausible solutions, e.g. where exactly does a nose of a rocket end and its body begins? This fuzziness is acknowledged in previous work, as Wang et al. [2012] report timings for achieving "close to error-free" results allowing some deviation from ground truth.

Annotating COSEG. We use our system to annotate the COSEG benchmark by running it for each part label independently and using graph-cuts at each iteration to project per-point labels to per-face labels. Since segmentation partitions a surface, we only collect (#parts-1) labels, and the missing part is estimated after the graph-cut. We report average human input time (in minutes) and FMF to reach about 95% average accuracy in Table 2, the timing is computed from a user study conducted on Candelabra, Big Chairs and

Big Vases datasets. We also run our system on other categories and compute the number of annotation and verification tasks required to annotate all models, and then estimate total timing based on the average timings from the user study. These estimated timings are denoted with a (*) in the table. We annotate all categories of this dataset in 74 minutes; this would have taken 12 times longer with manual annotation.

Comparison to ACA. Active co-analysis (ACA) [Wang et al. 2012] is the state-of-the-art active learning technique for 3D segmentation of manifold meshes into a known, small number of components. Knowing the number of labels that they have to partition the surface allows them to do a warm start by running an unsupervised co-segmentation algorithm first. For efficiency they pre-segment their meshes to super-patches such that each such patch is assigned to only a single part (resulting in about 7 patches per shape) and use active-learning to classify these patches. Our framework is more general – it is not limited to manifold meshes, supports labels that do not partition the input shape, and extends to any labeling granularity (e.g. it can be used to detect regions that do not conform to geometric patch boundaries such as the hood of the car in Figure 1). Below we focus on measurable comparisons between the methods and their components.

Wang et al. [2012] report measured timing with a user study on only two datasets: candelabras and vases. Our method is faster on these two datasets (Table 2). ACA iteratively queries users to provide "must-link" and "cannot-link" relationships between patches. These interactions require users to find good exemplar pairs among multiple choices selected based on active learning criteria. It then uses similarity in local geometric features of patches to propagate labels across the dataset. Since Wang et al. [2012] report the number of pairwise constraints and total number of patches on their models, we can extrapolate additional times as well as FMF from their numbers. We compute average time for a user to provide a constraint from the given two time measurements, and estimate approximate timing on other datasets based on the reported number of constraints, denoting these timings with a (*) in Table 2. We estimate the FMF of their method, measured as $\frac{2*\#\text{constraints}}{\#\text{patches}}$, since one needs to click on two patches to specify constraints (assuming that one could provide all labels by clicking on every patch). Note that we do not account for the fact that specifying patch-to-patch constraints takes longer than naive one-by-one patch labeling, and thus our estimate is biased towards higher-than-true FMF for the ACA method. While we make some assumptions to project both approaches to comparable metric, we believe that these statistics are representative of the performance of the two methods, indicating that our framework is twice more efficient in using human-time.

Another critical difference between the two methods is the time efficiency of the methods they use for propagating human input. ACA propagates labels between patches using a spring-based embedding model; this approach takes 15s per optimization on the largest dataset that contains 3000 patches. In contrast, our CRF-based optimization takes 10s for optimization with 750,000 points. Applying the mass-spring model to our data would slow the processing down by two orders of magnitude. On the other hand, pre-segmentation into patches would restrict the granularity of our labeling, and may be challenging on polygon soups.

Comparison to supervised labeling. We compare the performance of our automatic propagation algorithm to the supervised labeling methods of Kalogerakis et al. [2010] and Wu et al. [2014]. We test these methods on lamps and chairs in the ShapeNetCore-benchmark dataset. Some of the point-wise features used by previous techniques can only be computed for manifold, well-tessellated surfaces that form a single connected component, and thus cannot be used to analyze our data. Thus, in this comparison we use our point-wise features for all three methods. While these previous methods do not leverage correspondences, to make the comparison more fair we also provide 3D positions of points after shape co-alignment step as additional features. A random subset of 500 models is chosen from ShapeNetCore for each category during the comparison. For each plot sample, we generate 10 random train/test splits and report the average performance. Figure 12 shows that our propagation method outperforms existing alternatives even in a traditional supervised labeling setup.

Dataset	Meshes	FMF ACA	Time ACA	FMF Ours	Time Ours
Candelabra	20	3.4	7.0	**8.8**	**1.4**
Chairs	20	3.3	10.5*	**35.5**	**0.9***
Four-legged	20	1.9	20.1*	**3.5**	**11.5***
Goblets	12	6.1	1.2*	**21**	**0.7***
Guitar	44	27.5	**1.8***	**28**	1.9*
Lamps	20	**24.3**	**0.6***	10.6	2.3*
Vases	28	2.5	9.9*	**5.4**	**8.3***
Irons	18	2.7	7.6*	**3.0**	**7.2***
Big Chairs	400	8.7	73.6*	**22.3**	**21.4**
Big Vases	300	**17.4**	20.0	10.4	**18.1**
Cumulative	882	7.1	152.3*	**12.3**	**73.6***

Table 2: *We compare to the approach of Wang et al.[2012] (ACA). Higher FMF and smaller times indicate more efficient performance and are highlighted in bold. We use (*) to denote estimated timing (see text for details).*

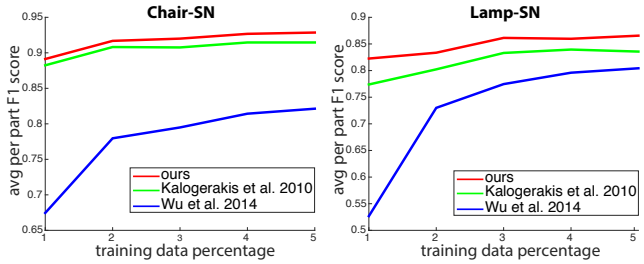

Figure 12: *Our automatic label propagation tool demonstrates a superior performance in comparison to previous techniques (adapted to handle ShapeNetCore data).*

User Study. We conduct an in-lab user study to (i) obtain timing for comparisons to ACA, and (ii) estimate timing constants used in our utility function. We selected 5 users with at least basic computer literacy. We provided them the same tools and instructions as we used for online crowdsourcing tasks. We issued them the tasks required to annotate chairs, vases, and candelabra categories from the COSEG dataset and report total timing in Table 2. We also used bigger datasets and some examples from ShapeNetCore categories to estimate average timing for constants used in our human model: average per-model annotation time τ_{ann} (labeling two views), time to identify whether labeling is correct τ_{ident}, and time to click on incorrect labeling τ_{click}. Since the later two come from the same verification task, we estimate them using the best fit to our human model t_{ver} (Equation 2). We use conservatively rounded averages $\tau_{\text{ann}} = 30\text{s}$, $\tau_{\text{ident}} = 0.3s$, $\tau_{\text{click}} = 1.1s$.

Quality of verification. To evaluate the quality of crowd-sourced verifications we asked an expert (i.e., a geometry processing researcher) to re-verify results for a subset of categories. Table 3 reports the accuracy of crowd workers on both annotation and verification tasks. We measure annotation accuracy (fraction of models that were annotated correctly), verification precision (the fraction of labels identified by workers as correct which are indeed correct), and verification recall (the fraction of correct labels identified by workers as such). We observe that annotation accuracy is close to

Dataset	Part	h^m	Q^m	Q^m
		Acc	Pr	Rc
Plane SN	body	96.04	96.82	92.31
	wing	97.47	97.36	95.01
	engine	98.25	97.73	88.87
	tail	97.17	97.23	93.20
Chair SN	back	95.23	98.98	98.51
	seat	95.11	95.51	98.66
	leg	97.53	92.78	99.14
	arm	99.10	97.80	96.35
Table SN	top	96.74	98.57	96.86
	leg	97.08	92.02	99.10
Lamp SN	base	96.91	98.17	92.59
	canopy	89.72	97.29	86.42
	lampshade	95.71	95.20	99.46
Average		96.29	96.56	95.36

Table 3: *This table presents the annotation (h^m) and verification (Pr and Rc) accuracies of crowd workers.*

verification precision and recall, suggesting that verifying whether a labeling is correct is as reliable as labeling the region.

Evaluating design choices. We run several studies demonstrating the significance of different design decisions and contributions we have made. We report average per-part F1 score (harmonic mean of precision and recall) for Big Vases and Chairs as a function of user time in Figure 13. We chose average per-part F1 score since it up-plays the role of small salient parts in comparison to average per-face accuracy. In this evaluation we compare our method (red solid line) and alternatives with one of our features disabled. We run the following variants:
– No correspondence term - We exclude cross-shape correspondence term from our per-point CRF ($\lambda_1 = 0$).
– No active selection - We disable annotation set optimization and instead select a random set of shapes of the same size to be labeled.
– No verification step - We only annotate models and do not query users for verification. Interestingly, the performance does not necessarily increase monotonically since we have no mechanism to understand which propagations worked well.
– No feature-based term - We exclude feature-based classifiers from our per-point CRF, thus excluding local shape cues.
– No ensemble learning - We build a single feature-based classifier instead of ensemble of classifiers, thus excluding global shape-to-shape similarity cues.
– No learning of weights - We do not update our weights $\omega, \lambda_1, \lambda_2$, which makes our method less adaptable to novel data (e.g., new categories and labels of interest).
– No smoothness term - We exclude smoothness term from our per-point CRF ($\lambda_2 = 0$).

As we can see from this evaluation all our features yield significant performance boost, while incorporating a verification step and the correspondence terms plays the most significant role.

Segmentation vs labeling. Our method is designed for labeling semantic regions. This is a different problem from segmentation which focuses on detecting boundaries between segments [Chen et al. 2009]. We focus on labeling because most applications require segments to have consistent labels across the dataset (e.g., modeling-by-parts interfaces need to swap compatible parts with the same label), and not all applications require partitioning of the surface (e.g., affordances of a bicycle can be predicted by detecting seats, pedals, and handles, while the rest of the shape is irrelevant). While we provide a simple procedure for converting labels to segments (see end of Section 6), a more elaborate technique could be used that takes pairwise features into account to create better boundaries (e.g., see Kalogerakis et al. [2010]). To validate the quality of our boundaries we tested our method on PSB [Chen et al. 2009] using a leave-one-out setup. We found that our method produces reasonable boundaries, as confirmed by low Rand Index (see supplemental material for details).

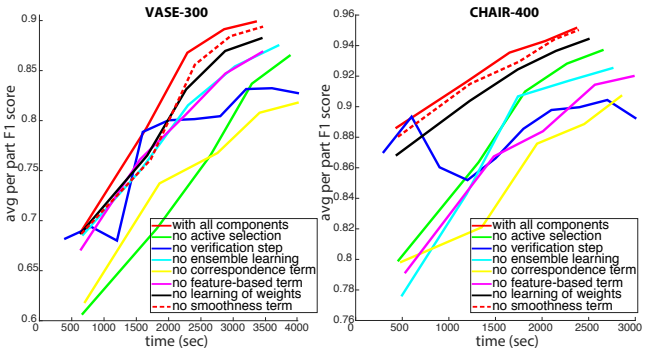

Figure 13: *Comparison of different variants of our method where each curve corresponds to a result obtained without some feature, and x-axis is human input time and y-axis is average per-part F1-score. See text for details.*

Computation Time. We estimate computation times on a representative dataset of 400 chairs. We expect timings to scale linearly for bigger datasets since we use a divide-and-conquer strategy. The pre-processing time for our method is about 45s per shape, which includes 10s for point sampling and feature computation, and 35s for shape alignment via free form deformation. Our active subset selection takes 5s. Label propagation takes 10s for 750,000 points. Learning λ_i is a relatively expensive step as it requires re-optimizing our CRF multiple times, thus it takes 2 min on the chairs dataset. All timings are reported on an 8-core 2.6 GHz Intel Core i7 machine with 8GB RAM. We use a cluster of 5 machines (150 cores) for annotating the subset of ShapeNetCore.

10 Conclusion

We provide an efficient active learning framework for annotating massive geometric datasets, based on two key elements. By incorporating human verification of automatically-generated results in our optimization we simultaneously improve the efficiency of our tool and obtain accurate, human-verified results. By leveraging global shape-to-shape similarities in an evolving network, as well as point-to-point correspondences and local geometric features, we were able to achieve reliable and accurate automatic inference of per-point labels in shape collections. Jointly these contributions lead to an annotation tool that is more efficient than state-of-the art techniques and is suitable for annotating massive and diverse shape collections from public repositories. We have demonstrated this by annotating a dataset that is more than an order of magnitude larger than what was available before.

Limitations and Future Work. There are many remaining challenges and opportunities for future research on detailed shape annotation. While our 2D annotation and verification interfaces enable quick and easy processing by non-expert crowd-workers and provide a good accuracy vs efficiency tradeoff, there may be areas that are hidden in all provided views resulting in minor labeling inaccuracies. Designing efficient interfaces for visualizing complex shapes is an interesting future work. When labeling complex scenes, the presence of small parts and clutter make it infeasible to select views independently from the label of interest. Future work might consider optimizing for a viewpoint trying to focus annotator attention on interesting regions. Our current pipeline labels one region at a time. Interactions and inter-relations between the labels can be exploited in a more joint framework. Finally, our current method is designed to acquire surface labels; a different approach

may be required for richer annotations such as computing human-object interaction poses as in Kim et al. [2014], or for labeling the mechanical structure of an object (e.g., joints and their parameters).

Acknowledgments

The authors thank Justin Solomon and Nicholas Vining for their help. This research was supported by NSF grants DMS-1521608 and IIS-1528025, UCB MURI grant N00014-13-1-0341, Sheffer's NSERC Discovery and DAS grants, a Google Focused Research award, and gifts from Adobe.

References

BOYKO, A., AND FUNKHOUSER, T. 2014. Cheaper by the dozen: Group annotation of 3D data. *UIST*.

BRANSON, S., PERONA, P., AND BELONGIE, S. 2011. Strong supervision from weak annotation: Interactive training of deformable part models. In *IEEE ICCV*.

BRANSON, S., VAN HORN, G., WAH, C., PERONA, P., AND BELONGIE, S. 2014. The ignorant led by the blind: A hybrid human–machine vision system for fine-grained categorization. *IJCV 108*, 1-2, 3–29.

CHANG, A. X., FUNKHOUSER, T., GUIBAS, L., HANRAHAN, P., HUANG, Q., LI, Z., SAVARESE, S., SAVVA, M., SONG, S., SU, H., XIAO, J., YI, L., AND YU, F. 2015. ShapeNet: An Information-Rich 3D Model Repository. Tech. Rep. arXiv:1512.03012 [cs.GR].

CHAUDHURI, S., KALOGERAKIS, E., GUIBAS, L., AND KOLTUN, V. 2011. Probabilistic reasoning for assembly-based 3D modeling. *ACM SIGGRAPH*, 35:1–35:10.

CHEN, D.-Y., TIAN, X.-P., SHEN, Y.-T., AND OUHYOUNG, M. 2003. On visual similarity based 3d model retrieval. *Eurographics 22*, 3, 223–232.

CHEN, X., GOLOVINSKIY, A., AND FUNKHOUSER, T. 2009. A benchmark for 3D mesh segmentation. *ACM SIGGRAPH 28*, 3.

GUO, K., ZOU, D., AND CHEN, X. 2015. 3D mesh labeling via deep convolutional neural networks. *ACM TOG 35*, 1.

HU, R., FAN, L., AND LIU, L. 2012. Co-segmentation of 3d shapes via subspace clustering. *CGF 31*, 5 (Aug.), 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., SU, H., AND GUIBAS, L. 2013. Fine-grained semi-supervised labeling of large shape collections. *SIGGRAPH Asia 32*, 6, 190:1–190:10.

HUANG, Q., WANG, F., AND GUIBAS, L. 2014. Functional map networks for analyzing and exploring large shape collections. *SIGGRAPH 33*, 4, 36:1–36:11.

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*.

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

KIM, V. G., CHAUDHURI, S., GUIBAS, L., AND FUNKHOUSER, T. 2014. Shape2Pose: Human-Centric Shape Analysis. *SIGGRAPH 33*, 4.

LEORDEANU, M., AND HEBERT, M. 2005. A spectral technique for correspondence problems using pairwise constraints. In *ICCV*, vol. 2, IEEE, 1482–1489.

LIN, T.-Y., MAIRE, M., BELONGIE, S., HAYS, J., PERONA, P., RAMANAN, D., DOLLÁR, P., AND ZITNICK, C. L. 2014. Microsoft coco: Common objects in context. *ECCV*.

LV, J., CHEN, X., HUANG, J., AND BAO, H. 2012. Semi-supervised mesh segmentation and labeling. *CGF 31*.

MAKADIA, A., AND YUMER, M. E. 2014. Learning 3d part detection from sparsely labeled data. In *3DV*.

RUBINSTEIN, M., LIU, C., AND FREEMAN, W. T. 2012. Annotation propagation in large image databases via dense image correspondence. *ECCV*, 85–99.

RUSSAKOVSKY, O., LI, L.-J., AND FEI-FEI, L. 2015. Best of both worlds: human-machine collaboration for object annotation. In *IEEE CVPR*.

SHAMIR, A. 2008. A survey on mesh segmentation techniques. *Computer Graphics Forum 27*, 6, 1539–1556.

SHU, Z., QI, C., XIN, S., HU, C., WANG, L., ZHANG, Y., AND LIU, L. 2016. Unsupervised 3d shape segmentation and co-segmentation via deep learning. *Comp. Aided Geom. Des. 43*.

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.

SU, H., DENG, J., AND FEI-FEI, L. 2012. Crowdsourcing annotations for visual object detection. In *Workshops at the Twenty-Sixth AAAI Conference on Artificial Intelligence*.

SUNG, M., KIM, V. G., ANGST, R., AND GUIBAS, L. 2015. Data-driven structural priors for shape completion. *SIGGRAPH Asia*.

TRIMBLE, 2015. Trimble 3D warehouse.

VEZHNEVETS, A., BUHMANN, J., AND FERRARI, V. 2012. Active learning for semantic segmentation with expected change. In *IEEE CVPR*, 3162–3169.

VIJAYANARASIMHAN, S., AND GRAUMAN, K. 2008. Multi-level active prediction of useful image annotations for recognition. In *NIPS*, 1705–1712.

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*.

WEINBERGER, K. Q., BLITZER, J., AND SAUL, L. K. 2005. Distance metric learning for large margin nearest neighbor classification. In *NIPS*, 1473–1480.

WU, Z., SHOU, R., WANG, Y., AND LIU, X. 2014. Interactive shape co-segmentation via label propagation. *CAD/Graphics*.

XIE, Z., XU, K., LIU, L., AND XIONG, Y. 2014. 3d shape segmentation and labeling via extreme learning machine. *SGP*.

ZADROZNY, B., AND ELKAN, C. 2002. Transforming classifier scores into accurate multiclass probability estimates. In *SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*.