CROSSLINK: Joint Understanding of Image and 3D Model Collections through Shape and Camera Pose Variations

Moos Hueting
University College London

Maks Ovsjanikov
LIX, École Polytechnique

Niloy J. Mitra
University College London

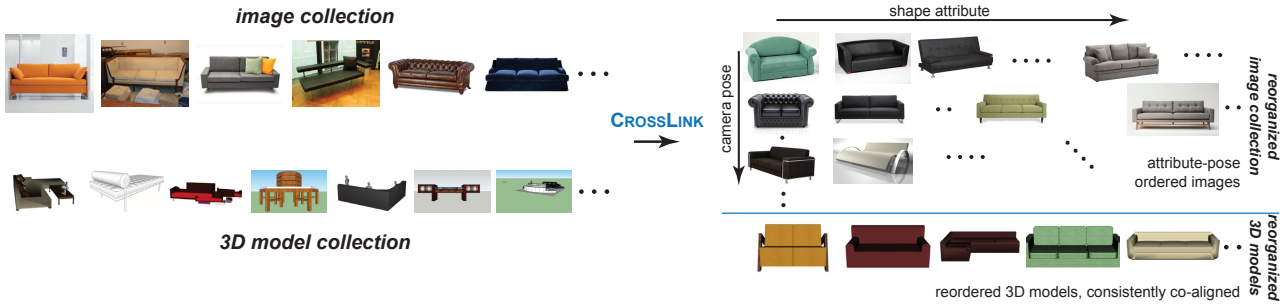

Figure 1: *We present a scalable analysis framework that jointly investigates class-labeled image and noisy 3D model collections ('couch' class in this example) to organize them by filtering outliers and factoring out shape and camera pose variations. The input images are consistently resorted along extracted view- and attribute-axes; while the 3D models are automatically reordered and consistently co-aligned.*

Abstract

Collections of images and 3D models hide in them many interesting aspects of our surroundings. Significant efforts have been devoted to organize and explore such data repositories. Most such efforts, however, process the two data modalities separately, and do not take full advantage of the complementary information that exist in different domains, which can help to solve difficult problems in one by exploiting the structure in the other. Beyond the obvious difference in data representations, a key difficulty in such joint analysis lies in the significant variability in the structure and inherent properties of the 2D and 3D data collections, which hinders cross-domain analysis and exploration. We introduce CROSSLINK, a system for joint image-3D model processing that uses the complementary strengths of each data modality to facilitate analysis and exploration. We first show how our system significantly improves the quality of text-based 3D model search by using side information coming from an image database. We then demonstrate how to consistently align the filtered 3D model collections, and then use them to re-sort image collections based on pose and shape attributes. We evaluate our framework both quantitatively and qualitatively on 20 object categories of 2D image and 3D model collections, and quantitatively demonstrate how a wide variety of tasks in each data modality can strongly benefit from the complementary information present in the other, paving the way to a richer 2D and 3D processing toolbox.

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

Keywords: shape analysis, model collections, image search, pose estimation, shape variations

1 Introduction

Image and model collections are ubiquitous and continue to grow rapidly. Analyzing and processing such collections has been the focus of a large body of work in computer graphics, computer vision and related fields over the past several decades. However, despite a great amount of progress, several tasks remain challenging, including text-based 3D model search and camera pose estimation in images, in large part due to the significant noise (e.g. in tags or annotations of 3D models), or the lack of training data (e.g. in camera pose estimation) in the unorganized online repositories.

At the same time, several recent techniques have been proposed for co-analysis and exploration of 3D model collections, leading to the area of *structure-aware shape processing* [Mitra et al. 2013]. Works in this direction are motivated by the fact that the structure and relations between 3D objects are best understood within the context of other related models in a large repository. However, the vast majority of these techniques require a pre-filtered set of models falling within the same category ('car', 'chair', 'bicycle', etc.), and obtaining such a set often requires manual intervention, especially because most existing text-based 3D search approaches are based on user-generated tags, which can be noisy and unreliable.

Similarly, in the image domain, estimating *shape attributes*, such as geometric properties including height or width of the object, or the viewing angle/camera pose, is difficult even for the largest image collections, since these attributes are rarely provided as part user labelings. Therefore, training image classifiers that would be able to discriminate across object *views* or provide geometric information about the object in an image still remains challenging (e.g., 'show an image of a long bicycle from a particular viewing angle').

Our work is motivated by the fact that often, 3D models and 2D images provide *complementary* information about the same objects or object classes, and therefore the information can be used to improve the performance of techniques in each domain. This way, we follow several recent approaches in both computer graphics and computer vision that exploit cross-domain information for part-based model synthesis [Xu et al. 2011], object detection and camera pose estimation [Aubry et al. 2014a], and consistent 3D shape segmentation [Wang et al. 2013] among several others. However, unlike these techniques, which so far have mainly used one-directional cross-domain links, our system integrates information and combines the strengths of *both* domains to facilitate tasks in each. Moreover, our

approach is *fully automatic*, and does not require tedious manual pre-filtering, assume a clean set of annotations, or explicitly establish point- or patch-level correspondences.

We concentrate on several key problems, for which we demonstrate significant quantitative improvement in performance over existing baseline methods: text-based 3D search, 3D model coalignment, as well as camera pose estimation and geometric property (height or width of an object) in 2D images. For each of these problems we demonstrate how to exploit the strengths of different types of data to co-analyze image and 3D model collections for common object categories. Our ultimate objective is to allow better understanding and joint exploration of image and 3D model collections. We achieve this by improving 3D search results, consistently aligning the models, and exposing to the user an exploration interface, which allows image retrieval based on pose and shape, learned from 3D models.

In creating CROSSLINK[1], we solve several key technical challenges: (i) show how image-based feature representations can be used to learn efficient classifiers on 3D shapes for better text-based search, (ii) propose an efficient 3D coalignment procedure, based on a hybrid 2D-3D representation, (iii) demonstrate how this representation leads to efficient pose estimation by using the inherent periodic (spherical or circular) nature of the space of views of a 3D model, and finally (iv) develop an object shape estimation method in images, using 3D models for training. Note that although to solve the problems we heavily use classical techniques such as Support Vector Machines and non-linear regression, we make significant technical contributions to exploit the particular and novel structure of multimodal data at hand.

We extensively evaluated the proposed framework on 20 object categories obtained using the Bing Image Search and the Trimble 3D Warehouse. We propose several quantitative metrics to evaluate each step of our system, setup appropriate ground truth datasets, and compare the performance over existing baseline approaches. In summary, we:

- introduce and study the problem of *joint analysis* of 2D image and 3D model collections, while factoring out significant shape and camera pose variations; and
- propose a framework for multi-modal data analysis across collections for *search and exploration*, without explicitly solving for point- or patch-level correspondences, requiring background detection, or assuming manual filtering.

2 Related Work

2D image search. Many supervised and unsupervised methods have been developed for image retrieval (see recent surveys [Datta et al. 2008; Zhang and Rui 2013]). Broadly, the majority of methods rely on image tags or accompanying annotations and/or extracted image features (e.g., HOG, SIFT, PCA-SIFT, SURF, etc.) to train category-specific classifiers. More recently, correlation across multiple information channels (e.g., text, images) has been explored for better retrieval performance [Weston et al. 2011; Masci et al. 2014; Pereira et al. 2014]. With a similar motivation, we propose a method to link and utilize information coming from 3D models for richer image search and exploration.

3D shape search. There has also been significant work on 3D shape retrieval from large collections. These techniques can be classified according to the type of query, and include, *text-based*, or *content-based*, where a sample 3D shape is given and the goal is to retrieve similar ones, either *image-based*, or *sketch-based*. Although, in practice, text-based search is both simplest and most accessible, the poor quality of user-assigned tags in public 3D model collections means that the quality pure text-based retrieval has so far been unsatisfactory [Min et al. 2004]. At the same time, content-based and sketch-based retrieval approaches, while often accurate [Eitz et al. 2012; Li et al. 2015], assume a 2D or 3D query, which can be non-trivial to obtain for a casual user.

Image analysis. In the context of image collections, the grand goal is to annotate the content of each image and link it to an ontology of semantic concepts (e.g., ImageNet Visual Recognition Challenge [Russakovsky et al. 2014] and references therein), often by using a large repository of ground truth annotations. While this line of work is fundamental, we argue that some properties, such as geometric attributes of objects (a 'narrow' chair) or camera pose are rarely present in the annotations of even the largest image collections. Therefore, using side information from a different modality can contribute to better overall image understanding. In the context of image collections of a single scene, learning discriminative patches has been proposed to characterize the underlying 3D scene. For example, Srivastatava et al. [2011] proposed exemplar SVM to establish cross-domain image matching, while Aubry et al. [2014b] factor out various sketching effects to facilitate painting-to-3D alignment.

Shape analysis. Analogously, in the context of 3D model collections, co-analysis approaches [Mitra et al. 2013] have focused on extracting part-level anisotropic scale variations for characterizing style [Xu et al. 2010], linking point-level correspondence detection across shape variations to learn template-based shape variations [Kim et al. 2013], semi-supervised learning strategies for fine grained labeling of shapes collections [Huang et al. 2013a], learning characteristic deformation directions from models collections [Yumer and Kara 2014], performing semantic editing [Yumer et al. 2015], or using functional maps to analyze unstructured model collections [Huang et al. 2014]. The methods rely on access to single-category collections free from outlier shapes. Often, such model collections are manually curated, which limits extensions to different classes. Here, we show how large collections of natural images can be used to automatically filter irrelevant 3D models and to co-analyse and explore collections of 3D models.

Coupled image-shape analysis. The classic work on morphable faces [Blanz and Vetter 1999] demonstrated the utility of modal analysis using point-level correspondence across shapes to computer graphics. Xu et al. [2011] used model collections to perform part-based model synthesis using photographs for style guidance, while Li et al. [2011] fuse photographs and LiDAR scans to create depth-layer decomposition of urban facades. More recently, Wang et al. [2013] analyze different 2D projections of 3D shapes to transfer information from labeled images to consistently segment the 3D shapes. In two related efforts, Vicente et al. [2014] use landmark-based correspondences to roughly estimate camera locations for images of different but related shape instances in an effort to reconstruct the VOC data, while Su et al. [2014] estimate deformation fields regularized by a network of shapes to convert segmented images to corresponding depth maps. Note that in the later effort the input images are assumed to be presegmented. With a similar motivation, Aubry et al. [2014a] use renderings of 3D models from multiple viewpoints to train a pose classifier for data-driven part-based 2D-3D alignment in a single manually curated object class.

Pose estimation. Another line of work that reasons about 3D and 2D data jointly, is research in Camera Pose Estimation, which can also be stated as alignment of a 3D object with a natural 2D image. Although a classical and well-studied, there are many variations

[1]Please visit the project page at http://geometry.cs.ucl.ac.uk/projects/2015/crosslink/ for code and benchmark data.

that range in robustness and complexity (see [Dambreville et al. 2008; Corsini et al. 2009; Prisacariu and Reid 2012] among many others, as well as recent work by Russell et al. [2011] applying this idea to align historical architectural paintings with 3D models obtained using multi-view stereo). We propose CROSSLINK that achieves both *scalability* and *robustness* to large changes in geometry and appearance, and *avoids* any explicit correspondence. Our use of classifier smoothing and interpolation in this context can also be seen as a special-case of regularized multi-task learning [Evgeniou and Pontil 2004], where we exploit the circular nature of the parameter space.

Exploration interfaces. A common exploration strategy is to embed shapes based on their estimated pairwise similarities. Such embeddings can either be computed statically [Averkiou et al. 2014], or dynamically [Kleiman et al. 2013]. Alternately, low dimensional shape variations can be revealed using template-based signature analysis, and used to order shapes along the discovered dominant modes of shape variations [Ovsjanikov et al. 2011]. In another theme of work, shapes have been locally ordered and embedded based on relations among pair of shape pairs, i.e., quartet of shapes [Huang et al. 2013b], or based on dynamically detected contextual focal points [Xu et al. 2014] in the case of collections of scenes. In an interactive setting, a set of tools was introduced by the Average Explorer system [Zhu et al. 2014] to perform supervised image clustering and alignment to improve the quality of weighted image averages. Our work is also related to the area of content-based image retrieval (CBIR) (cf., [Rafiee et al. 2010]), wherein the focus is to retrieve images using attributes like color, texture, spatial layout, etc. In contrast, our analysis by tightly coupling image and shape collections allows users to isolate shape and pose variations, and thus enable multi-modal exploration.

3 Overview

Our key hypothesis is that by jointly considering the inherent qualities of 2D and 3D collections, we can harness the power of one of the domains to improve performance of tasks in the other. However, translating this intuition into a practical framework is challenging as the connections between 2D and 3D repositories are not trivial to discover and exploit, due to the fundamental differences in the two representations. More importantly, natural images (i.e., photographs of real objects) usually have very different appearance compared to counterpart user-generated 3D models. Both the presence of the background clutter in the natural images and the variation of geometry and texture in the 3D models often lead to significant differences in the resulting representations (see Figure 2).

Figure 2: *Using a keyword search, 'airplane' in this example, we retrieve the default ordered images and 3D models from Bing and the Trimble 3D Warehouse. Note the poor quality on the bottom.*

CROSSLINK (see Figure 3) takes as input a class-labeled 2D image collection $\boldsymbol{I}$, and a class-labeled 3D model repository $\boldsymbol{M}$. We represent each model in $\boldsymbol{M}$ by a set of renderings $\boldsymbol{V}$ taken from a fixed set of viewing angles. We retrieve the class-labeled collections by respectively querying the Bing and the Trimble 3D Warehouse repositories with text queries (e.g., 'car,' 'airplane,' etc.).

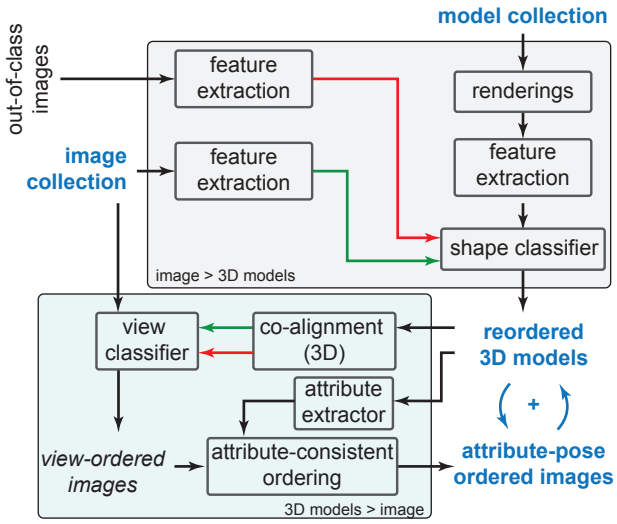

Figure 3: *Overview of the proposed framework for joint understanding of class-labeled 2D image and 3D model collections.*

Images to improve 3D model search. We observe that image collections often have more accurate labels than their 3D counterparts. For example, a keyword search of 'car' on a 2D search engine yields nearly only true positives among the top hits, whereas the same search on a 3D search engine yields more questionable results (see Figure 2). This is partly explained by the fact that online image repositories are orders of magnitude larger than 3D shape collections, which enables training significantly more accurate image labeling and classification mechanisms. We exploit this difference in quality by training a classifier using the data in $\boldsymbol{I}$, by considering them to be 'ground truth positives,' using image descriptors to first convert the images to canonical feature vectors (Section 4.2). We then re-sort the models in $\boldsymbol{M}$ using the scores of the classifier on the renderings in $\boldsymbol{V}$ and discard models based on the classification scores. This leaves us with the filtered set of 3D models $\boldsymbol{M}_f$ (Section 5). These 3D models, however, are not consistently oriented. We develop a novel image-based approach to co-align the filtered set of 3D models $\boldsymbol{M}_f$ (Section 5.1) by exploiting the circular structure in the view space. Thus, an image collection helps to re-sort and co-align a corresponding model collection.

Re-sorted models to reorder images. We then use the clean set of co-aligned 3D models for a given object class to better organize and annotate the input set of 2D images. Our motivation is that whereas 3D models can be viewed from any angle, 2D images are fixed, and extracting the viewpoint (i.e., camera pose) from a given 2D image is non-trivial. We therefore propose an approach for viewpoint estimation that exploits the set of 3D models. We train view-specific classifiers, this time with renderings of one viewpoint from each model in $\boldsymbol{M}_f$ as positives, and the other viewpoint renderings as negatives. As an interesting technical novelty, we show how a one-parameter *family* of classifiers can be obtained via fitting, thus alleviating the need to train many independent classifiers for each view independently (Section 6.2). Given a target viewpoint, we then re-sort the image collection $\boldsymbol{I}$ based on the classifier score. Further, we assign to each image in $\boldsymbol{I}$ its most likely view.

Having factored out view variation, we turn to shape variation. While both the 3D models and the natural images exhibit shape variation, we note that certain geometric shape attributes are trivial to extract from the re-sorted 3D models, while the same task is difficult in 2D images. Hence, we train a nonlinear regressor by using the models $\boldsymbol{M}_f$, regularized using a novel formulation to exploit the

circular view structure, for each shape attribute and each viewpoint. We then use the regressors to estimate the geometric properties for images in $\boldsymbol{I}$, while using the view information extracted in the previous stage (Section 6.3). Thus, the 3D model collection helps to reorganize the counterpart image collections according to view and shape variations. The extracted cross links are then used for jointly exploring the image and model collections (Section 7).

4 Input and Data Representation

The goal of our framework is to reason about 2D and 3D data concurrently. For the 2D part, we take as input a collection $\boldsymbol{I}$ of natural images, which consists of sets of images $\boldsymbol{I}_c$, obtained by issuing a text query c (e.g., 'car,' 'chair,' 'bicycle,' etc.) to a standard 2D image search engine (Bing Image Search). Similarly, we take a 3D repository $\boldsymbol{M}$ consisting of sets of models $\boldsymbol{M}_c$, obtained by issuing the same text query c to a 3D model search engine (Trimble 3D Warehouse). One of our goals is to use the set $\boldsymbol{I}_c$ to filter out the incorrect 3D models from $\boldsymbol{M}_c$. For this, as well as for the other steps in our pipeline, we first bring both sets into a common representation, as described next.

4.1 Rendering of the models

To obtain a common representation for both 2D and 3D data, we summarize each model $M_c^i \in \boldsymbol{M}_c$ via a set of 2D renderings $\boldsymbol{V}_c^i$. These renderings are made from a set of N_v viewing angles, at a fixed elevation in a ring around the object. In our experiments, the renderings were taken directly from the 3D Warehouse search results, which provide $N_v = 36$ views per 3D model. Figure 4 shows an example. Note that we assume that the up-vector for all models is consistent, and is similar to the standard up-vector in 2D images. While this may not be the case for all types of objects, we have found this assumption to hold for most categories that we considered (except 'guitars' and 'helicopters'). Let us stress that we represent each 3D model as *a structured collection* of images. Throughout the paper we will heavily exploit the circular ordering (i.e., views are wrapped around 360°) and the consistent up-direction of the renderings of each 3D model.

Figure 4: *Each 3D model retrieved from the Trimble 3D Warehouse is used to create 36 renderings in different views, by sampling 10 degree rotations at a fixed elevation (only a few shown in this example). We assume that every input model is upright oriented.*

4.2 Feature extraction

The 2D repository consists mostly of photographs, which often have background clutter, whereas the 3D models are rendered against no background. In addition, although some of the models have textures, they are not nearly of photographic quality. Finally, the geometry of the models in the 3D repository is often of poor quality, further increasing the difference in appearance. In order to gain resilience to a large class of transformations, we employ a feature encoding approach. We use two different feature representations, both selected for their individual strengths.

KC-encoded HOG features. The first type of feature is a combination of local histogram of gradients (HOG) [Dalal and Triggs 2005] features, and a Kernel Codebook (KC) encoding [van Gemert et al. 2008] to combine them globally. The success of HOG in many recent object detection and classification approaches (cf., [Felzenszwalb et al. 2008]) and its sensitivity to changes in orientation makes this descriptor especially interesting in our application. For each 8×8 patch, the HOG features yield a 36-dimensional feature vector, which captures local image gradients (see Figure 5).

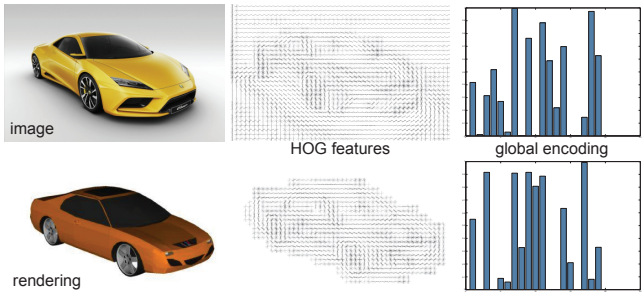

Figure 5: *Left-to-right: Input images, corresponding HOG features, global feature vector encoding using max pooling (K=20 for visualization). Top row: car image from Bing; bottom row: a comparable camera view rendering from Trimble 3D Warehouse.*

We combine the local HOG features into global descriptors using the aforementioned kernel codebook encoding. Specifically, we first perform K-means clustering on all local HOG features of all images in both $\boldsymbol{I}_c$ and $\boldsymbol{V}_c$, resulting in a codebook $\boldsymbol{\mu}_c$ of K visual words for each class c ($K = 800$ in our tests). Then, for a given image $\boldsymbol{x}$ with HOG features $H(\boldsymbol{x})$, each local HOG feature $h \in H(\boldsymbol{x})$ is encoded as a K-dimensional vector, which has non-zero elements only for the closest k ($k = 4$) neighbors of h in $\boldsymbol{\mu}_c$:

$$[KC_c(h)]_i = \begin{cases} \frac{d(h,\mu_c^i)}{\sum_{j \in \mathcal{N}_k} d(h,\mu_c^j)} & \text{if } i \in \mathcal{N}_k(h, \boldsymbol{\mu}_c) \\ 0 & \text{otherwise} \end{cases}$$

where, $\mathcal{N}_k(h, \boldsymbol{\mu}_c)$ are the indices of the k closest neighbors of h in $\boldsymbol{\mu}_c$, and $d(h, \mu)$ is a distance kernel, defined as

$$d(h, \mu) = \exp(-\gamma ||h - \mu||^2).$$

We used $\gamma = 100$ in our experiments. A major advantage of this approach is that the global feature vector will vary more smoothly with small changes in the local feature vectors. This is important when modeling the view classifiers as a smoothly varying family, as discussed in Section 6.2. Finally, we combine these local encodings into a global encoding using max pooling [Jarrett et al. 2009; Boureau et al. 2010]:

$$[KC_c(H(\boldsymbol{x}))]_i = \max_h [KC_c(h)]_i, \quad h \in H(\boldsymbol{x}).$$

This setup results in one 800-dimensional Euclidean feature vector per image. Please note the variation from the original approach [van Gemert et al. 2008], where the local encodings are summed together. Our reasoning here is that the 2D images in $\boldsymbol{I}$ often have background. Using sum pooling, the gradients of the background would always be taken into account, which can adversely affect later image comparison. Our experiments confirm this boost in performance.

CNN features. The second type of feature is extracted using a pre-trained convolutional neural network (CNN) [Krizhevsky et al. 2012; Jia et al. 2014]. This network was trained on ImageNet [Deng et al. 2009] – a dataset of over 10 million images in over 10000 subcategories – and is the state-of-the-art for classification on this particular dataset. We use the final 4096-dimensional fully connected layer from this network as a global image feature. One CNN feature vector represents one image, and is of dimension 4096. Note

that this feature descriptor is expected *not* to discriminate well between views, as it was specifically trained to recognize objects in any configuration. We show and evaluate the difference in performance between the two features in Section 7.

For each object class c, we compute HOG and CNN features of all images in $\boldsymbol{I}_c$ and all rendered views in $\boldsymbol{V}_c$. For any image $\boldsymbol{x}$, we will refer to its KC-encoded HOG features as $\mathrm{HOG}(\boldsymbol{x})$, and to its CNN features as $\mathrm{CNN}(\boldsymbol{x})$. As most of the pipeline is agnostic to the type of feature used, we refer to any feature generically as $\mathrm{F}(\boldsymbol{x})$.

5 3D Model Collection Filtering

Our first observation is that label accuracy of the images in $\boldsymbol{I}_c$ is significantly higher than of the models in $\boldsymbol{M}_c$. Therefore, we capture the characteristic features of the given class in a classifier by using the high accuracy of the images in $\boldsymbol{I}_c$, and then re-sort the models in $\boldsymbol{M}_c$ using this classifier.

By using one of the feature representations $\mathrm{F}(\boldsymbol{x})$ mentioned above, we first create a set of 'positives' from the feature encodings of the in-class images in $\boldsymbol{I}_c$, i.e.,

$$P_c^{\text{filt}} := \mathrm{F}(\boldsymbol{I}_c).$$

Similarly, as negatives, we use the feature encodings of the out-class images, as well as the feature encodings of a sampling of renderings from the out-class models:

$$N_c^{\text{filt}} := \mathrm{F}(\boldsymbol{I}_{C\setminus c}) \cup \mathrm{F}(V_{C\setminus c}^{i,j})$$

where, $V_c^{i,j}$ is a sampling of the j-th rendered view of 3D model i. In our experiments, we take 10 models per class, and 2 views per model, randomly sampled from the set. Empirically, the addition of these out-of-class negatives significantly improved the results. Intuitively, the features in the negatives from rendered 3D models are more likely to appear in the test data, and therefore, force the classifier to find a better tradeoff between the positive features from the natural images and negative features from both images and rendered 3D models.

We use the sets P_c^{filt} and N_c^{filt} to train a standard linear support vector machine (SVM) [Cortes and Vapnik 1995] f_c^{filt} that finds the separating hyperplane with the largest margin between the positives and negatives. This will allow us to give a confidence score that a particular 3D model in $\boldsymbol{M}_c$ actually belongs to class c, by considering the scores of its rendered views on f_c^{filt}.

We apply the trained support vector machine to all views in $\boldsymbol{V}_c$, resulting in a per-view score $f_c^{\text{filt}}(V_c^{i,j})$. For a 3D model, we define the score as the maximum score over all its views:

$$f_c^{\text{filt}}(M_c^i) = \max_j f_c^{\text{filt}}(V_c^{i,j}).$$

An alternative would be to use the average instead of maximum. However, we have found that the distribution of different views in $\boldsymbol{I}_c$ is most often not well-balanced, and thus we cannot expect views in $\boldsymbol{V}_c$ that are not well represented in $\boldsymbol{I}_c$ to have a high score, even when they are views of an in-class model (see Figure 6).

Finally, we sort the models in $\boldsymbol{M}_c$ based on their scores to obtain an ordering on the 3D models based on the confidence of them belonging to class c. In Section 7, we show that this ordering significantly outperforms the default ordering given by Trimble 3D Warehouse or obtained by a direct shape descriptor-based clustering. Finally, the filtered model set is obtained by removing all models with a negative classifier score.

Figure 6: *We re-sort the (top) original ordering of 'cars' from the 3D model collection to obtain a new ordering (bottom), automatically pushing the false positives to the end of the list.*

Figure 7: *The input models do* not *come co-aligned (top). We propose a simple effective method to consistently align them (bottom).*

5.1 3D model alignment

Having filtered the 3D models for each class c, next we organize them in a way that makes joint shape analysis and exploration simple and effective. First, we co-align the shapes into a shared canonical position (see Figure 7). To achieve this, we developed a novel method for joint alignment, which is sensitive to object appearance. Our approach takes as input a set of 3D models that mostly belong to a single category c, with a few potential mis-classified instances, gathered using the method described in the previous section. Our goal then is to find the rotation for each 3D model such that a properly defined global alignment error is minimized. The proposed technique particularly exploits the circular nature of the view space, and hence is both efficient and robust in the presence of mis-classified instances.

In this part we use the same hybrid 2D-3D shape representation as described above. Namely, we represent each 3D model as a collection of 36 images rendered at 10 degree increments of azimuth from a fixed elevation. Recall that this representation assumes that all 3D models have a consistent "up" direction. We summarize each of the rendered views compactly using a Euclidean vector of size K with the chosen feature encoding F. This means that each 3D model is represented as a matrix of size $K \times 36$, where the rows and columns stand for feature encodings and views (camera poses), respectively.

Given a set of matrices $\{M_1, M_2, \dots M_N\}$ of size $K \times 36$ corresponding to N different shapes, our goal is to find a vector V of size N, where each V_i is an integer in the range $[0 \dots 35]$, such that the following error is minimized:

$$E(V) = \sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{k=0}^{35} \left\| M_i^{V_i+k \bmod 36} - M_j^{V_j+k \bmod 36} \right\|_2^2. \quad (1)$$

Here we let M_i^l be the l^{th} column of matrix M_i, and the norm corresponds to the L_2 norm between the corresponding column vectors. Intuitively, the alignment problem amounts to finding the offset V_i for each shape i, such that the feature representation of the $k + V_i$ view of 3D model i corresponds to the representation of $k + V_j$ view of 3D model j for each j and each k, modulo 36. Note that $E(V) = E(G)$ if G is any vector such that $G_i = V_i + c \mod 36$ for any constant c for all i.

We optimize Equation (1) using a simple iterative technique, similar to Iterated Conditional Modes inference of Markov Random Fields, and to congealing [Learned-Miller 2006] with a discrete search space. In particular, we start by considering a random vector V. Then, for each $i \in [1 \dots N]$, we find the minimizer of $E(V)$ with all but i^{th} dimension of V fixed, and update V_i, if necessary. Note that since $V_i \in [0 \dots 35]$ the optimum can be found by direct

inspection. We repeat this procedure, iterating over the different dimensions of V (corresponding to different shapes), until convergence. We then restart this procedure for several (200 in our experiments) initial random vectors V and keep the solution V^{opt} which minimizes the error $E(V)$. In practice, we have noticed that keeping one dimension of V fixed during optimization helps to speed up convergence by reducing the presence of multiple global optima due to the circular nature of the energy function E.

The final vector V^{opt} provides a set of views, one per shape, that are as consistent as possible (see Figures 7 and 12). Note that if we expect V_i^{opt} and V_j^{opt} to correspond for shapes i and j, then views $V_i^{\text{opt}}+k \mod 36$ and $V_j^{\text{opt}}+k \mod 36$ should correspond for any k as well. This simple method is remarkably efficient and works well resulting in an average error of only 5-10° (see Section 7).

6 Image Collection Organization

The output of the method described in the previous section is a set of co-aligned and filtered 3D models with a small number of false positives for each class c. We then use these models to better organize the corresponding image collections. Namely, we use the co-aligned 3D models to estimate both the camera pose and certain geometric properties of objects in the 2D image set $\boldsymbol{I}_c$, which then enables queries such as 'show images of a tall chair from a particular viewpoint'.

Note that we can render the coaligned 3D models from specific viewing angles for each model in $\boldsymbol{M}_f$. Specifically, we put together sets of images (renderings) taken from the same view, and use them as positives in a linear classifier, as described below.

6.1 Camera pose estimation

Suppose we are given a class c and a viewing angle θ. To compute a classifier corresponding to this view, we first construct a set of positive examples by gathering all views of the 3D models in $\boldsymbol{V}_c$ corresponding to θ:

$$P_{\text{view}} := \mathrm{F}(\boldsymbol{V}_c^{\theta}).$$

As negatives, we use the *other* views from the same class:

$$N_{\text{view}} := \mathrm{F}(\boldsymbol{V}_c \setminus \boldsymbol{V}_c^{\theta}).$$

As neighbouring views are often very similar, including them in the negatives decreases the score for positives as well. However we are interested only in the relative scores of the classifiers to decide which view a given 2D image should be assigned to. This overall decrease in classifier score is thus not an issue for our case.

Having trained a linear SVM with these positives and negatives, we run the obtained classifier f_c^{view} on the images in $\boldsymbol{I}_c$. The resulting classifier scores correspond to the confidence of each 2D image being associated with the viewing angle θ. Note that the approach here is similar to the 3D repository filtering from Section 5 – although the modality from which the training data and the test data originate are now switched, the overall classification approach we employ is the same.

We generate one classifier f_c^{view} per viewing angle, yielding N_v different classifiers (36 in our case). Given this set of classifiers, we use them to re-sort the 2D images according to their scores $f_{c,v}^{\text{view}}$, which puts the images in $\boldsymbol{I}_c$ taken from the view v at the top.

In some applications we may also need assign a single view to each image. However, we cannot directly compare classifier scores, as SVM classifier scores are not calibrated, i.e., similar scores in $f_{c,i}^{\text{view}}$ and $f_{c,j}^{\text{view}}$ $(i \neq j)$ does not imply similar confidence of the image belonging to view i or j. Thus, we use Platt scaling [Platt 1999] to convert the SVM score of a test image to a probability.

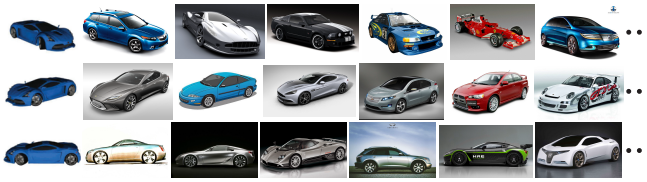

Figure 8: *Camera pose estimation. Here, we show the top results for three different views, one per row, for the 'car' dataset. The icons on the far left indicate the corresponding 3D model view.*

Instead of directly taking the view with the maximum probability, we once again exploit the circular structure of the data. Specifically, given an image and its ground truth view, we expect the score of the classifier pertaining to that specific view to be high, but as neighboring views are similar in appearance, we furthermore expect the neighboring classifiers to score above average as well. For the feature vector of a given image $\boldsymbol{x}$, we take this into account by computing a score for each view v as the weighted sum of the classifier score $f_{c,v}^{\text{view}}(\boldsymbol{x})$ and its neighboring views $f_{c,v\pm r_v}^{\text{view}}(\boldsymbol{x})$, with weights chosen as a Gaussian distribution g with zero mean and unit variance:

$$f_{c,v}^{\text{viewWeighted}}(\boldsymbol{x}) = \sum_{i=-r_v}^{r_v} g(i) f_{c,v+i}^{\text{view}}(\boldsymbol{x}).$$

We then assign each image to the view which has the highest weighted score. We have observed this weighting across neighboring views to improve performance (Figure 8).

Note on regression and non-linearity. Regression seems to be the natural choice for learning the relationship between the viewing angle and the feature vector of an image. However, this approach is made difficult by the inherent non-linearity of the relationship. As the Euclidean distance between feature vectors of two neighboring views is expected to be small, the feature vectors of the 36 views of a given model lie on a loop in feature space. We tried support vector regression with a number of different kernels (radial basis function, hypertangent and polynomial) with a range of different parameters, but the performance was very low in all cases (see Section 7). Although writing a kernel capable of dealing with this specific type of non-linearity may be possible, our approach is particularly appealing due to its simplicity and good performance in practice.

6.2 Modeling of classifier weights

Training one classifier for each of the 36 different views is both costly in practice, and moreover, does not allow classification corresponding to views outside of this fixed set. Intuitively, the weights of the classifiers f_c^{view} will have a regular structure: we expect the weights w_c^i of $f_{c,i}^{\text{view}}$ to be quite similar to the weights of $f_{c,i+1}^{\text{view}}$. We illustrate this by performing a Principal Component Analysis (PCA) on the classifier weights of the 36 classifiers for the class 'car.' Examining the resulting PCA dimensions we note that the first three explain over 80% of the variance in the classifier weights (Figure 9, left). Plotting the coefficients in these dimensions (Figure 9, right) of the computed SVM weights f_c^{view} against the viewing angle shows very smooth and regular structure. This regularity is also present for the classifier biases b_c. We exploit this structure to setup a model of the classifier weights and bias, allowing us to create a one-parameter *family* of classifiers per object class, parameterized by the viewing angle, *without* explicitly training an independent classifier for each angle.

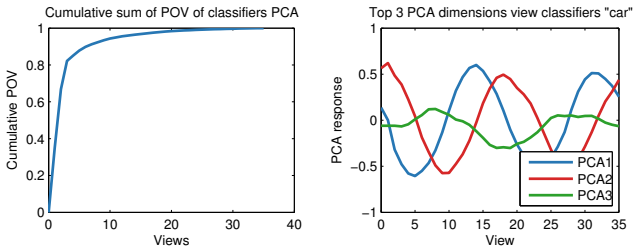

Figure 9: *Left: Cumulative fraction of variance of the PCA dimensions for the view classifier weights of class 'car.' Note that 80% of variance is explained by just 3 PCA dimensions. Right: The top 3 PCA dimensions show regular structure, suggesting that they can be modeled.*

Specifically, we sample N_{sample} of 36 viewing angles, evenly across the circle (every $360/N_{\text{sample}}$ degrees), and perform PCA on the weight vectors of the resulting classifiers (see Section 6.1). For each of the top 3 PCA dimensions $[p_c^i]_{i=1}^3$ and for the bias vector b_c we find interpolating cubic splines $[q_i]_{i=1}^3, q_b$. Then, to compute a classifier for a given view θ, we evaluate the interpolators for θ and reproject to the original space:

$$w_c^\theta = \sum_{i=0}^{3} p_c^i \, q_i(\theta) \quad \text{and} \quad b_c^\theta = q_b(\theta). \tag{2}$$

This yields a classifier $f_{c,\theta}^{\text{viewModeled}}$ that provides a confidence score for a given example of class c as to whether it belongs to view θ. Note that the higher we set N_{sample}, the less data we take into account, and thus the lower the actual training cost. Section 7 describes the effect of varying N_{sample}.

6.3 2D repository re-sorting by shape

At this stage, we have a one-parameter family of classifiers that we use to re-sort the 2D images in $\boldsymbol{I}_c$ for any viewing angle. Having now tackled view variation, we focus on in-class shape attribute variation. Capturing such variations in the image domain allows not only sorting the images by camera pose (view), but also to sort the images within each view by object structure.

To achieve this, we extract, from the filtered models $\boldsymbol{M}_f$ in class c a certain scalar geometric shape attribute $X \in \mathbb{R}$, which is chosen to be trivial to extract in the 3D domain, but challenging in the 2D domain. In our experiments, X is the ratio of the height over the width of the model (we consider the ratio to account for image/model scale variations). Concatenating these X_i for each model in $\boldsymbol{M}_c$ results in a vector $\boldsymbol{X}_c$.

Our goal is to arrive at an estimator that links the property X using the features of the images in $\boldsymbol{I}_c$. We do so by training kernel ridge regressors (KRR) [Saunders et al. 1998] on the feature encodings F of the views in $\boldsymbol{V}_c$. To factor out view variation, we train one regressor r_c^θ per viewing angle θ.

More so than in the previous parts of the pipeline, the stark difference in feature distribution between photographs and renderings significantly limits the regressor's performance on the 2D images when trained on the 3D renderings. To offset this difference between the domains, we use a geodesic flow kernel [Gong et al. 2012] in the KRR. This kernel constructs an implicit feature domain assembling information from the source and target domains (photographs and renderings) and an infinite number of domains interpolating between the two. Our results show a significant performance increase when using such a domain-adaptation technique.

To estimate the value of the attribute $\boldsymbol{X}$ in a given image $I_c^i \in \boldsymbol{I}_c$, we use the regressor corresponding to the assigned view of I_c^i, which we extracted in the previous section, and apply it on the feature encoding of I_c^i:

$$x(I_c^i) = r_c^\theta(\mathrm{F}(I_c^i)).$$

The output of this procedure is a set of estimators of the given geometric attribute, one per each viewing angle. We use these estimators to sort the natural images according to the shape of the objects in them.

7 Evaluation

We extensively evaluated the proposed framework on various real world examples. First, we shortly discuss the data sources on which we performed our experiments, including their origins, size, and associated ground truth. Then, we show the results of applying each part of the system on the data, and both discuss and show how we evaluate the various results.

Data collections. Below, we present results for 4 (of 20) classes, namely *airplane*, *bicycle*, *car* and *chair*. For each of these classes, we scraped the top 150 results from a keyword search from Bing and Trimble 3D Warehouse for 2D and 3D data, respectively. As mentioned in Section 4.1, for each model 3D Warehouse provides 36 views, rendered from evenly spaced angles in a circle around the model from a fixed elevation. Results for another 16 classes are included in the supplementary material.

Ground truth. We manually annotated all 3D models as being either a good example of their class (true positive), or a bad example (false positive). In some cases, a model file contained either more than one instance of the class, or contained multiple other objects as well. In those cases, we only counted the model as positive if the model was prominent in the renderings. In the 2D repository, each image was annotated with its ground truth viewing angle (discretized to 10 degree bins to correspond with the renderings of the models). Note that this assumes camera poses level with the ground (fixed up-vector) and a view taken from a similar elevation as the renderings. This assumption breaks down for some classes, such as airplane and helicopter.

7.1 3D repository filtering using 2D

In the first experiment, we filter the 3D models using the 2D images, based on the feature encodings of the 2D images and the renderings of the models (see Section 5). Note that the only user supervision in the whole process is the choice of classes, via a choice of the text query issued to the 2D and 3D search engines (Bing Image Search and Trimble 3D Warehouse search), such as 'car,' 'chair,' 'bicycle,' etc. We tested the setup with both KC-encoded HOG features, as well as with the pretrained CNN features.

Lightfield baseline. Intuitively, models in the same class should be geometrically similar, and are thus expected to be clustered in any standard descriptive feature space. We extracted 3D shape descriptors from all models and then performed unsupervised hierarchical clustering on the feature vectors. The optimal distance criterion for the clustering was found by trial and error on a number of classes and then fixed. Next, we ordered the clusters by size (largest first, smallest last), and within each cluster sorted the models by distance to the cluster centroid (closest first, farthest last). We use this as a baseline to compare our method. We employed a Lightfield descriptor [Chen et al. 2003], which was found to be the most discriminative shape descriptor available [Shilane et al. 2004].

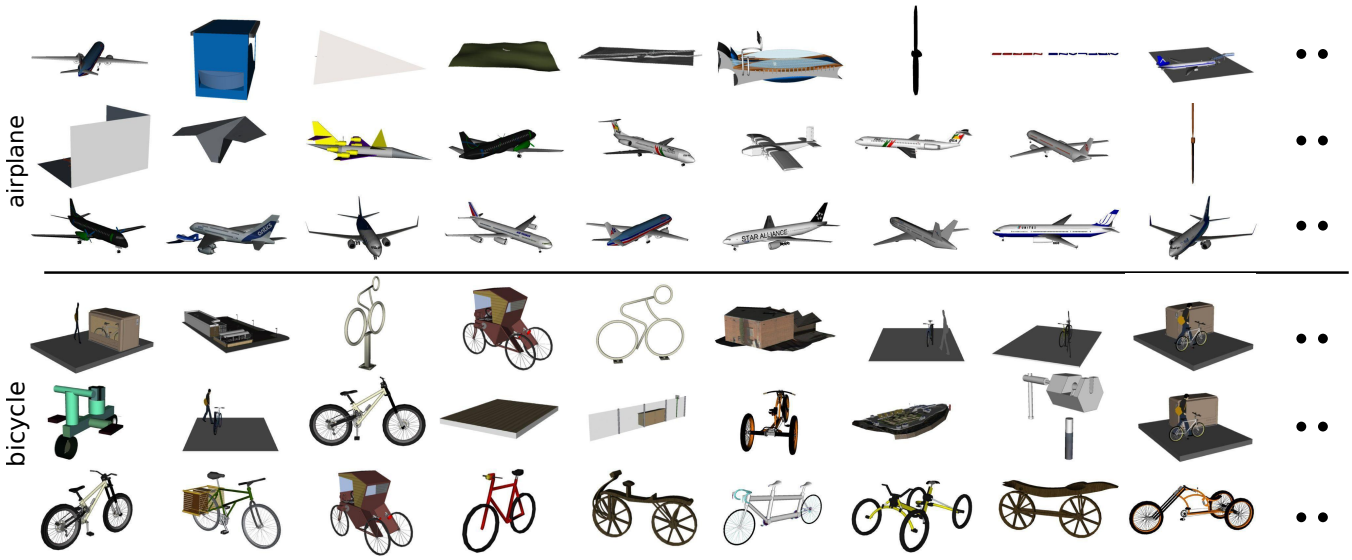

Figure 10: *Re-sorting of model collections for 'airplanes' and 'bicycles.' In each case, top shows original ordering from the 3D Trimble Warehouse, middle shows ordering using Lightfield clustering, while bottom shows* CrossLink *results. See Figure 11 for performance evaluation against manually annotated ground truth. Note that the results are not co-aligned at this stage.*

Qualitative evaluation. Figure 10 shows the results of two keyword searches according to the original ordering in which they were returned by 3D Warehouse, the ordering garnered from the unsupervised clustering of the Lightfield descriptor as mentioned above, as well as the ordering by our algorithm. Note that the original top results returned by 3D Warehouse contain a high number of false positives, even though many true positives do appear later in the set of results. Although the Lightfield clustering does improve results, the presence of many false positives throws it off balance. After reordering using our algorithm, most of the false positives in the top results for both classes have disappeared, having been assigned low scores by the classifier.

Quantitative evaluation. In Figure 11, we show the ROC curves for 4 different classes, for the original ordering from 3D warehouse, the one using the Lightfield descriptor, and our ordering for both feature setups. For all classes, the performance is significantly better than the original ordering. For classes with little shape variation and many true positives, such as 'car', the Lightfield descriptor clustering works well. However for classes with a significant number of false positives, such as 'bicycle' and 'airplane', our method is noticeably better than the Lightfield clustering baseline.

CNN vs. HOG. The CNN based feature outperforms HOG on nearly all classes. This is especially apparent for classes where there is very consistent background in the photographs, airplane and boat (see supplemental material). We believe that the HOG based feature associates the class with the background, which is missing from the renderings. In contrast, CNN based feature does not have this problem.

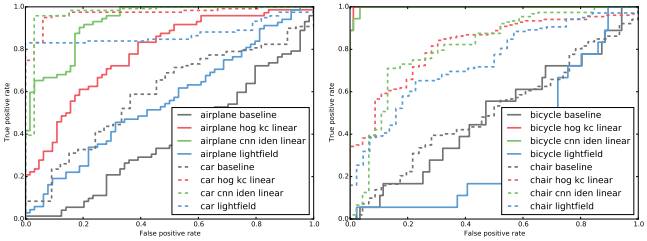

Figure 11: *ROC curves for 4 different classes measuring effect of 3D repository filtering using 2D image information as compared to manually annotated ground truth quality for 3D models. As baselines, we present both the original orderings in the model collections and the ordering based on the Lightfield descriptor.*

7.2 3D model alignment

Having filtered the datasets, we now test the 3D alignment method in image space, as proposed in Section 5.1. We apply our method to the model set of each class. The resulting alignment for one such class is shown in Figure 12. Note that although the original set of models is quite random in its coalignment, our simple image-based algorithm finds the correct alignment for most models.

Pairwise consistency. The mean pairwise angular offset (MPAO) of all models before and after alignment is shown in Table 1. This error metric is the expected difference in alignment between two randomly picked models from the set. The 'airplane' class still has a high MPAO after alignment, which is mostly due to the algorithm not being able to distinguish well between flipped versions of the same model, due to their feature encoding similarity. Changing the feature representation could improve performance in this case, and exploring this avenue is left for future work.

Comparison with mesh-based method. For comparison, we ran the filtered model set for two of the classes (laptop and car) through an existing mesh-based alignment method, as described in [Huang et al. 2013a; Averkiou et al. 2015]. Although the accuracy of this method is slightly higher than ours, the method takes significantly longer to run. See supplementary material for quantitative comparison. Note that for both our method and the mesh-based method, we started with the filtered sets of 3D models. The performance decreases significantly for both methods when using the unfiltered model collections.

Effect of filtering. We show the importance of filtering the 3D models before applying the alignment step usign the class 'bicycle.' This class contains very few true positives. As such, without

Figure 12: *Co-alignment of 3D models. (Top) Initial alignment across the 3D models, (bottom) consistently co-aligned 3D models for the 'chair' models. Please refer to Table 1 for detailed error analysis against hand annotated ground truth data.*

prefiltering, the dataset is very noisy, making it difficult to find a consistent coalignment across models. After prefiltering the performance is increased by an order of magnitude. The same can be observed for small data sizes in general, as shown for class 'helicopter.' When only taking 50 models, realigning them in the unfiltered state yields an error 50% higher than when filtering first.

7.3 2D repository view sorting

After filtering and alignment of the 3D repository, we reorder the 2D images from Bing according to view variation (see Section 6). We solely use KC-encoded HOG-based feature for this part, as the CNN-based feature was specifically trained not to respond to changes in viewing angle. Figure 13 shows, for class 'car,' the assigned views for a number of images, as well as the ROC curves for the ordering based on the view classifier scores of three different views. Note that many of the errors are due to the assignment of images to 180 degree flipped views, which are especially prevalent for the side view of the car, but are also present in other views.

Error per viewing angle. Figure 14 visualizes the R-precision (the precision of the classifier at position R, where R is the number of positives available) per viewing angle together with the distribution of viewing angles across the ground truth. Judging only from the ordering of the images themselves, the somewhat high quantative error for some views seems surprising. We have observed that perceptually two views can look very similar, while still being objectively somewhat further apart. For exploration, this ambiguity works in our favor – even with slightly erroneous view alignment, perceptually the ordering of the images makes sense.

Table 1: *Accuracy of 3D alignment. Each value represents the mean pairwise angular offset between models in that specific scenario. For the 'bicycle' class, there are not many models left after filtering, resulting in an underconstrained optimization. This is reflected by the lower number of perfect alignments. 'Airplane' models often have a* 180° *error, resulting in a relatively high MPAO.*

Class	Before	After	Perfect
airplane	82.5°	41°	64%
bicycle	91.2°	7.72°	23%
bicycle, no filtering	91.2°	83.32°	23%
car	90.3°	5.79°	95%
chair	87.4°	0.9°	95%
helicopter	93.1°	17.4°	95%
helicopter, no filt.	93.1°	19.2°	95%
helicopter, no filt. (50 models)	93.1°	30.2°	95%

Comparison with regression. In Figure 15 we show the distribution of error magnitude in degrees for two classes, for both our method as well as a comparative regression method. We ran a support vector regressor using 4 different common kernels (linear, polynomial, hypertangent and radial basis function) on the data, using 10-fold cross validation to find the best parameters (both for the kernel and the regularization parameter). The results shown in the figure have highest performance. Note that this approach does not work nearly as well as our classification approach. Finding a kernel capable of handling the circular nature of the data remains an interesting direction for future work.

Effect of background clutter. Figure 16 shows representative examples of view classification degradation under background clutter in test images. Often these images exhibit strong directional lines, such as a sharp horizon, or the silhouettes of buildings. Dealing more explicitly with such difficulty, for example by incorporating background in the training stage, remains a direction for future work.

7.4 2D view classifier modeling

To avoid the high cost of training many classifiers, we exploit the regular structure of the classifiers' weight vectors and bias, as described in Section 6.2. Figure 17 shows for the class 'car' the Kendall-Tau rank correlation between the original view classifiers, trained as normal, and the modeled classifiers. This statistic measures the correlation of two data orderings, being 1 when the orderings are equal, -1 when they are opposite, and 0 when they are mutually independent. The left-most chart shows that applying PCA to the weight vectors and using only the top 3 dimensions does not change the resultant ordering much, showing that just the top 3 dimensions in feature space are responsible for most of the view classification performance. In the middle chart, we sample only every 30 degrees, reducing the number of classifiers we have to train by a factor of 3. Although the original classification score orderings are not preserved entirely, the performance is still reasonable. Even increasing the step size to 50 degrees does not dwindle performance entirely, although it approaches the limit of what is useful.

Currently, we use cubic interpolation for the estimation of the weight vectors. The curves, as shown in Figure 9 and as we observed for other classes as well, have a clear periodic structure, and could possibly be approximated using a sum of sines model. This would possibly constrain the system more, allowing us to use even sparser sampling of the classifier space. We leave this possibility for future work.

7.5 2D repository shape sorting

We test the method described in Section 6.3 for estimating a certain shape property from images in the 2D repository. First, we extract the ratio of height over width from the set of aligned 3D models. Then, for each viewing angle a KRR is trained using the method described in Section 6.3. To estimate the ratio for a given image from the 2D repository, we apply the KRR corresponding to the image's assigned viewing angle to the features of that image. Applying this

Figure 13: *View-classifying Bing images using classifiers trained using 3D models. We assign a best view-estimate for each Bing image. Here, we show a set of example view estimations for 'car' images. The corresponding ROC curves are computed again ground truth data compiled by manually annotating each of the retrieved Bing images (only a sampling shown here).*

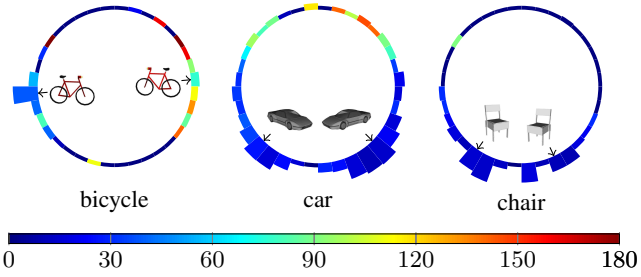

Figure 14: *R-precision of view classification per view shown for three different classes. For each bar, color signifies the R-precision with respect to ground truth, while height corresponds to the number of Bing images with that view as ground truth. Note that the height gives a measure of confidence to the error (i.e., taller bars indicate more statistically significant), and also shows the distribution of views per class. For all classes, this distribution is very biased towards a number of canonical views (e.g., showroom 3/4th views for cars). Tall blue bars indicate perfect results, while short bars of any color can be ignored due to lack of enough data.*

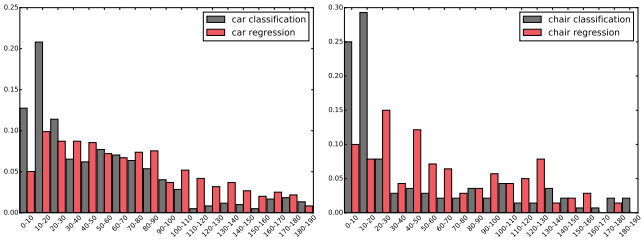

Figure 15: *Histogram of error magnitudes for the 'car' and the 'chair' class, discretized in bins of 10 degrees, for both classification and non-linear regression. Note that our method results in a large concentration of errors in 0-10 and 10-20 bins.*

to all images of each view, we can re-sort the images within a view by ratio. We show a sampling of such results for the class 'chair' in Figure 18. Figure 19 shows for the 3D renderings the Kendall-Tau rank correlation between the ordering by ground truth ratio and the ordering by estimated ratio for each view. Although the score decreases for views from which the width is difficult to judge, the scores across most other views is high.

Notes on scalability and performance. So far, we reported evaluations on 20 classes with image and model sets of size 150 per collection. To test scalability, we tested a number of classes on datasets of larger size. (Note that the main bottleneck is to prepare the ground truth for larger sets.) Specifically, we tested the influence of increasing both the size of training data and the number of testing data. The left ROC plot in Figure 20 shows that increas-

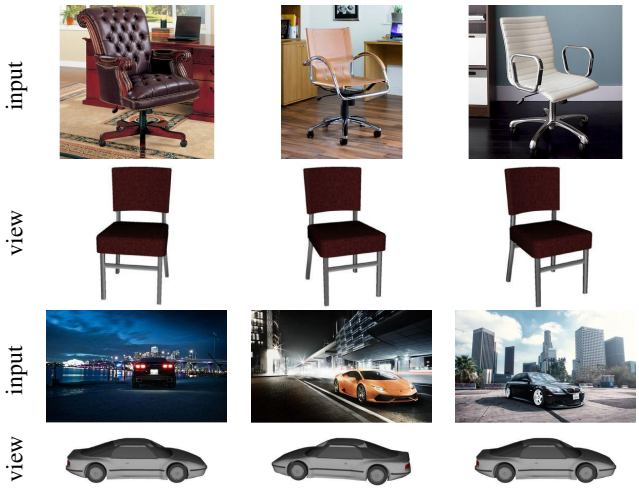

Figure 16: *View classification of images with significant background clutter. The photographs are input to our view classification pipeline, the renderings are the resulting view classifications. In some cases (bottom row) the background clutter leads to misclassifications, whereas in other cases the system handles the difficulty well.*

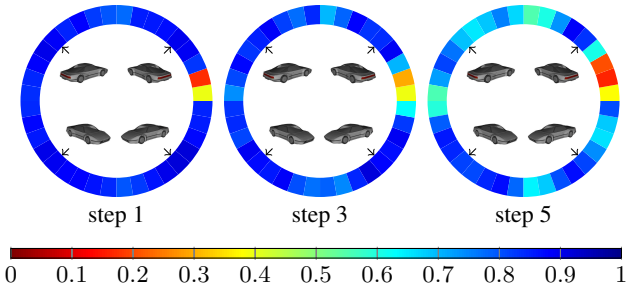

Figure 17: *Performance of modeled SVM classifiers on synthetic data using 3 different sampling densities. A step size of n means we model the first 3 PCA dimensions using every n^{th}th classifier from the set of normally trained classifiers. The color signifies the Kendall-Tau ranking correlation between the modeled SVM of the corresponding view and the respective normally trained SVM. A score of 1 signifies perfect correlation (equal ordering, so zero loss), a score of 0 means uncorrelated orderings. To show the effect of the PCA we also show step size 1 (taking all classifiers).*

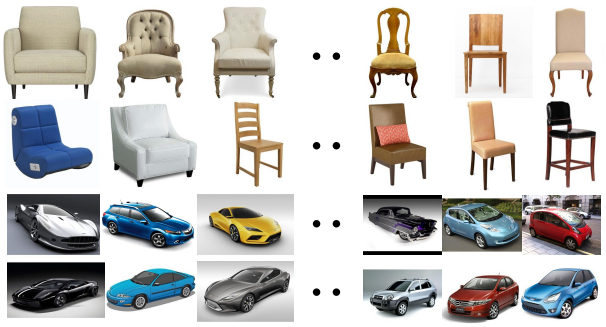

Figure 18: *Attribute-sorted view-classified Bing images for two classes. Each row shows a different view sorting. For each row, as we go from left-to-right, the height-to-width ratio increases, i.e., the objects turn from short-and-wide to tall-and-narrow.*

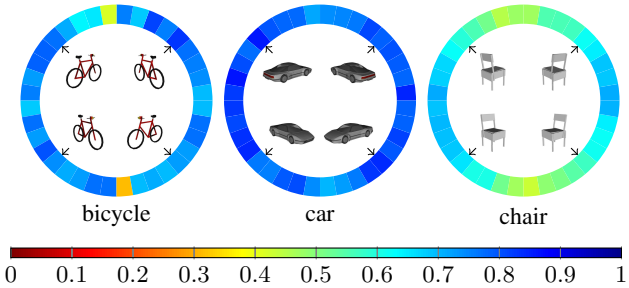

Figure 19: *Accuracy of shape estimation. Color represents the Kendall-Tau rank correlation coefficient for the ordering of the synthetic models according to the shape ground truth versus the estimated shape. A score of 1 indicates perfect correlation (equal ordering); a score of 0 indicates uncorrelated orderings.*

ing the test set to the top 1000 search results does not decrease performance with respect to the original set of 150 (compare with Figure 11). Note that this is not trivial, as we expect the 3D Warehouse search results to increasingly contain more false positives. In contrast, recall that method only relying on 3D data, will perform worse as the fraction of outlier shapes increase. The right plot shows the performance of training with 1000 examples. Note that performance increases only slightly with respect to the original training of only 150 examples.

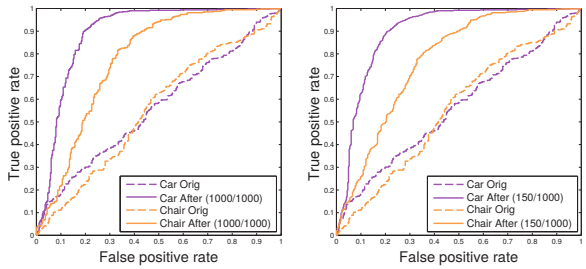

Figure 20: *Left: ROC curve for HOG-based classifier trained on 1000 2D images with the purpose of filtering 1000 3D models. Right: ROC curve for classifier trained on 150 2D images with the purpose of filtering 1000 3D models. Although results improve slightly, very similar performance is obtained with the smaller training set. This shows that our approach scales to large datasets, as only training the classifiers is expensive.*

Our pipeline consists of unoptimized code, with the extraction of the features being the main bottleneck. While timings can be improved in the future, currently in order to process 150 image/model sets, the system takes 1-2 minutes for the image → model direction and 3-4 minutes for the image ← model direction. While linear in complexity with the number of models, the step can be easily be run across multiple threads.

Limitations. We observed two main sources of errors: (i) In case of a class like 'boats,' consistent image background (i.e., water) can easily be learned as a distinguishing feature by view classifier. Although this effect is diminished when using CNN features, it is still an issue. An interesting future direction is to avoid such errors, without explicit background extraction. (ii) In case of a class like 'helicopters,' we observed a consistent difference in the camera pose in the image (looking up to the object) and model renderings (looking horizontally at the object). This leads to higher than usual view estimation error (around 17°).

8 Exploring Image and 3D Model Collections

We now describe how to use the output of the jointly analyzed image and 3D model collections for multi-modal data exploration. Figure 21 shows the user interface (see also accompanying video).

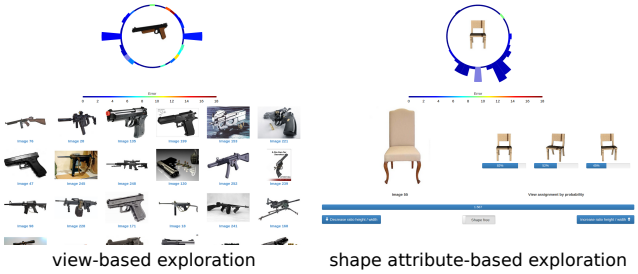

Figure 21: *The view-shape refactored image collections and the filtered and consistently coaligned model collections enable novel exploration possibilities. (Left) User selects a view by posing the model icon in the view-dial, while the system retrieves the top rated images for the indicated view. (Right) Any selected image can be used to probe for other images in a comparable view or with a comparable shape attribute.*

CROSSLINK produces filtered and coaligned model collections, and image collections resorted by view-shape attribute axes. The user

can then select a 3D model (shown as an icon) and use the provided view-dial to interactively pose a view (i.e., vary azimuthal angle) as the system retrieves the top rated images for the selected view. The height of the bars indicates the number of images in that view, while the color indicates confidence in the view estimates. The user can click on any image to further probe the confidence in its view estimate. More interestingly, the user can ask for images (from the same view) of objects with higher/lower shape attribute values. This interaction makes use of the discovered links between the two data repositories. Note that while this mode is very natural using our view-attribute reordered images, performing comparable actions using the raw image and/or model collections would be cumbersome and very difficult using existing query interfaces.

9 Conclusions and Future Work

We presented a framework for joint processing of image and 3D model collections that exploits the strengths of each data modality to improve tasks in the other. As a key difference to standard image/shape analysis approaches, we investigated how to factor out *both* shape and (camera) pose variations across such collections, and thus reveal their underlying structure. Our proposed framework is easy to scale, and does not attempt to explicitly compute point- or patch-level correspondences, or background segmentation on the image. Technically, we modeled how pose variation manifests as image-space feature variation, and then demonstrated how to factor out such variations to reveal consistent shape attribute-based reordering on images across multiple views. Finally, we extensively evaluated our framework to demonstrate that cross-domain processing not only results in cleaner and more consistent image and 3D model search, but also enables novel exploration possibilities.

While we presented a first framework to jointly exploit correlations across across image and 3D model collections, there are many exciting and important questions that need to be investigated: (i) A natural next step will be to investigate how 2-parameter view variations (i.e., include altitude variations) beyond one parameter view variations as investigated here. (ii) The domain adaptation technique we used for shape attribute regression can also be used for the other parts of the pipeline. As performance in these other parts was good without this extra layer, we did not perform any domain adaptation there. Performance might still increase by using the geodesic flow kernels throughout the pipeline. (iii) The shape attribute regressor currently does not take into account the circular nature of the data – as with the model alignment, realizing that renderings from very similar views will have similar regressor weights could be used as an extra regularizer. Note that this regularization would need to be carefully combined with the geodesic flow kernel. (iv) Finally, as a long term goal, we would expect to use such cross-domain connections along with advances in material modeling and semantic links [Yumer et al. 2015] to eventually unify image and 3D model collection, and thus be able to naturally transition between the two representations.

Acknowledgements

We thank the reviewers for their comments and suggestions for improving the paper. Special thanks to Melinos Averkiou for helping to compare with mesh based co-alignment algorithms and Ersin Yumer for providing 3D data. The authors are grateful to Leonidas Guibas, Thomas Funkhouser, John Shawe-Taylor, Aron Monszpart, James Hennessey, Clément Godard and Peter Hedman for helpful discussion and encouragement. This work was supported in part by ERC Starting Grant SmartGeometry (StG-2013-335373), Microsoft Research through its PhD Scholarship Programme, Marie Curie CIG, Marie-Curie CIG-334283-HRGP, a CNRS chaire d'excellence, and chaire Jean Marjoulet from École polytechnique.

References

AUBRY, M., MATURANA, D., EFROS, A. A., RUSSELL, B., AND SIVIC, J. 2014. Seeing 3D chairs: exemplar part-based 2d-3d alignment using a large dataset of CAD models. In *IEEE CVPR*.

AUBRY, M., RUSSELL, B. C., AND SIVIC, J. 2014. Painting-to-3D model alignment via discriminative visual elements. *ACM TOG 33*, 2, 14:1–14:14.

AVERKIOU, M., KIM, V., ZHENG, Y., AND MITRA, N. J. 2014. Shapesynth: Parameterizing model collections for coupled shape exploration and synthesis. *CGF Eurographics*.

AVERKIOU, M., KIM, V. G., AND MITRA, N. J. 2015. Autocorrelation descriptor for efficient co-alignment of 3d shape collections. *CGF*.

BLANZ, V., AND VETTER, T. 1999. A morphable model for the synthesis of 3d faces. In *Proc. SIGGRAPH*, 187–194.

BOUREAU, Y.-L., PONCE, J., AND LECUN, Y. 2010. A theoretical analysis of feature pooling in visual recognition. In *Proc. ICML*, 111–118.

CHEN, D.-Y., TIAN, X.-P., SHEN, Y.-T., AND OUHYOUNG, M. 2003. On visual similarity based 3d model retrieval. In *Computer graphics forum*, vol. 22, Wiley Online Library, 223–232.

CORSINI, M., DELLEPIANE, M., PONCHIO, F., AND SCOPIGNO, R. 2009. Image-to-geometry registration: a mutual information method exploiting illumination-related geometric properties. In *CGF*, vol. 28, 1755–1764.

CORTES, C., AND VAPNIK, V. 1995. Support-vector networks. *Machine learning 20*, 3, 273–297.

DALAL, N., AND TRIGGS, B. 2005. Histograms of oriented gradients for human detection. In *IEEE CVPR*, vol. 1, IEEE, 886–893.

DAMBREVILLE, S., SANDHU, R., YEZZI, A., AND TANNENBAUM, A. 2008. Robust 3d pose estimation and efficient 2d region-based segmentation from a 3d shape prior. In *ECCV*. 169–182.

DATTA, R., JOSHI, D., LI, J., AND WANG, J. Z. 2008. Image retrieval: Ideas, influences, and trends of the new age. *ACM Comput. Surv. 40*, 2, 5:1–5:60.

DENG, J., DONG, W., SOCHER, R., LI, L.-J., LI, K., AND FEI-FEI, L. 2009. Imagenet: A large-scale hierarchical image database. In *IEEE CVPR*, IEEE, 248–255.

EITZ, M., RICHTER, R., BOUBEKEUR, T., HILDEBRAND, K., AND ALEXA, M. 2012. Sketch-based shape retrieval. *ACM SIGGRAPH 31*, 4, 31:1–31:10.

EVGENIOU, T., AND PONTIL, M. 2004. Regularized multi–task learning. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM, 109–117.

FELZENSZWALB, P., MCALLESTER, D., AND RAMANAN, D. 2008. A discriminatively trained, multiscale, deformable part model. In *IEEE CVPR*, IEEE, 1–8.

GONG, B., SHI, Y., SHA, F., AND GRAUMAN, K. 2012. Geodesic flow kernel for unsupervised domain adaptation. In *IEEE CVPR*, IEEE, 2066–2073.

HUANG, Q.-X., SU, H., AND GUIBAS, L. 2013. Fine-grained semi-supervised labeling of large shape collections. *ACM SIGGRAPH Asia 32*, 6, 190:1–190:10.

HUANG, S.-S., SHAMIR, A., SHEN, C.-H., ZHANG, H., SHEFFER, A., HU, S.-M., AND COHEN-OR, D. 2013. Qualitative organization of collections of shapes via quartet analysis. *ACM SIGGRAPH 32*, 4 (July), 71:1–71:10.

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

JARRETT, K., KAVUKCUOGLU, K., RANZATO, M., AND LECUN, Y. 2009. What is the best multi-stage architecture for object recognition? In *IEEE ICCV*, 2146–2153.

JIA, Y., SHELHAMER, E., DONAHUE, J., KARAYEV, S., LONG, J., GIRSHICK, R., GUADARRAMA, S., AND DARRELL, T. 2014. Caffe: Convolutional architecture for fast feature embedding. In *Proc. ACM ICM*, ACM, 675–678.

KIM, V. G., LI, W., MITRA, N. J., CHAUDHURI, S., DIVERDI, S., AND FUNKHOUSER, T. 2013. Learning part-based templates from large collections of 3d shapes. *ACM SIGGRAPH 32*, 4.

KLEIMAN, Y., FISH, N., LANIR, J., AND COHEN-OR, D. 2013. Dynamic maps for exploring and browsing shapes. In *Proc. SGP*, 187–196.

KRIZHEVSKY, A., SUTSKEVER, I., AND HINTON, G. E. 2012. Imagenet classification with deep convolutional neural networks. In *NIPS*, 1097–1105.

LEARNED-MILLER, E. G. 2006. Data driven image models through continuous joint alignment. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 28*, 2, 236–250.

LI, Y., ZHENG, Q., SHARF, A., COHEN-OR, D., CHEN, B., AND MITRA, N. J. 2011. 2d-3d fusion for layer decomposition of urban facades. In *IEEE ICCV*.

LI, B., ET AL. 2015. A comparison of 3d shape retrieval methods based on a large-scale benchmark supporting multimodal queries. *Computer Vision and Image Understanding 131*, 0, 1 – 27.

MASCI, J., BRONSTEIN, M. M., BRONSTEIN, A. M., AND SCHMIDHUBER, J. 2014. Multimodal similarity-preserving hashing. *IEEE PAMI 36*, 4, 824–830.

MIN, P., KAZHDAN, M., AND FUNKHOUSER, T. 2004. A comparison of text and shape matching for retrieval of online 3d models. In *Research and Advanced Technology for Digital Libraries*. Springer, 209–220.

MITRA, N. J., WAND, M., ZHANG, H., COHEN-OR, D., AND BOKELOH, M. 2013. Structure-aware shape processing. In *EUROGRAPHICS State-of-the-art Report*.

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.

PEREIRA, COSTA, J., COVIELLO, E., DOYLE, G., RASIWASIA, N., LANCKRIET, G., LEVY, R., AND VASCONCELOS, N. 2014. On the role of correlation and abstraction in cross-modal multimedia retrieval. *IEEE PAMI 36*, 3, 521–535.

PLATT, J. C. 1999. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *Advances in large margin classifiers*, Citeseer.

PRISACARIU, V. A., AND REID, I. D. 2012. Pwp3d: Real-time segmentation and tracking of 3d objects. *IJCV 98*, 3, 335–354.

RAFIEE, G., DLAY, S., AND WOO, W. 2010. A review of content-based image retrieval. In *Communication Systems Networks and Digital Signal Processing (CSNDSP), 2010 7th International Symposium on*, 775–779.

RUSSAKOVSKY, O., DENG, J., SU, H., KRAUSE, J., SATHEESH, S., MA, S., HUANG, Z., KARPATHY, A., KHOSLA, A., BERNSTEIN, M., BERG, A. C., AND FEI-FEI, L., 2014. ImageNet Large Scale Visual Recognition Challenge.

RUSSELL, B. C., SIVIC, J., PONCE, J., AND DESSALES, H. 2011. Automatic alignment of paintings and photographs depicting a 3d scene. In *3dRR ICCV Workshop*, 545–552.

SAUNDERS, C., GAMMERMAN, A., AND VOVK, V. 1998. Ridge regression learning algorithm in dual variables. In *Proc. ICML*, Morgan Kaufmann, 515–521.

SHILANE, P., MIN, P., KAZHDAN, M., AND FUNKHOUSER, T. 2004. The princeton shape benchmark. In *Shape modeling applications, 2004. Proceedings*, IEEE, 167–178.

SHRIVASTAVA, A., MALISIEWICZ, T., GUPTA, A., AND EFROS, A. A. 2011. Data-driven visual similarity for cross-domain image matching. *ACM SIGGRAPH Asia 30*, 6.

SU, H., HUANG, Q., MITRA, N. J., LI, Y., AND GUIBAS, L. 2014. Estimating image depth using shape collections. *ACM SIGGRAPH*.

VAN GEMERT, J. C., GEUSEBROEK, J.-M., VEENMAN, C. J., AND SMEULDERS, A. W. 2008. Kernel codebooks for scene categorization. In *ECCV*. Springer, 696–709.

VICENTE, S., CARREIRA, J., AGAPITO, L., AND BATISTA, J. 2014. Reconstructing pascal voc. In *IEEE CVPR*.

WANG, Y., GONG, M., WANG, T., COHEN-OR, D., ZHANG, H., AND CHEN, B. 2013. Projective analysis for 3d shape segmentation. *ACM SIGGRAPH Asia 32*, 6, 192:1–192:12.

WESTON, J., BENGIO, S., AND USUNIER, N. 2011. Wsabie: Scaling up to large vocabulary image annotation. In *Proc. IJCAI*.

XU, K., LI, H., ZHANG, H., COHEN-OR, D., XIONG, Y., AND CHENG, Z.-Q. 2010. Style-content separation by anisotropic part scales. In *ACM SIGGRAPH Asia*, 184:1–184:10.

XU, K., ZHENG, H., ZHANG, H., COHEN-OR, D., LIU, L., AND XIONG, Y. 2011. Photo-inspired model-driven 3d object modeling. *ACM SIGGRAPH 30*, 4, 80:1–80:10.

XU, K., MA, R., ZHANG, H., ZHU, C., SHAMIR, A., COHEN-OR, D., AND HUANG, H. 2014. Organizing heterogeneous scene collection through contextual focal points. *ACM SIGGRAPH 33*, 4.

YUMER, M. E., AND KARA, L. B. 2014. Co-constrained handles for deformation in shape collections. *ACM SIGGRAPH Asia*.

YUMER, M. E., CHAUDHURI, S., HODGINS, J. K., AND KARA, L. B. 2015. Semantic shape editing using deformation handles. *ACM SIGGRAPH 34*.

ZHANG, L., AND RUI, Y. 2013. Image searchfrom thousands to billions in 20 years. *ACM TOMCCAP 9*, 1s, 36.

ZHU, J.-Y., LEE, Y. J., AND EFROS, A. A. 2014. Averageexplorer: Interactive exploration and alignment of visual data collections. *ACM SIGGRAPH 33*, 4, 160:1–160:11.