Learning 3D Part Detection from Sparsely Labeled Data

Ameesh Makadia
Google
New York, NY 10011
makadia@google.com

Mehmet Ersin Yumer
Carnegie Mellon University
Pittsburgh, PA 15213
meyumer@cmu.edu

***Abstract*—For large collections of 3D models, the ability to detect and localize parts of interest is necessary to provide search and visualization enhancements beyond simple high-level categorization. While current 3D labeling approaches rely on learning from fully labeled meshes, such training data is difficult to acquire at scale. In this work we explore learning to detect object parts from sparsely labeled data, *i.e.* we operate under the assumption that for any object part we have only one labeled vertex rather than a full region segmentation. Similarly, we also learn to output a single representative vertex for each detected part. Such localized predictions are useful for applications where visualization is important. Our approach relies heavily on exploiting the spatial configuration of parts on a model to drive the detection. Inspired by structured multi-class object detection models for images, we develop an algorithm that combines independently trained part classifiers with a structured SVM model, and show promising results on real-world textured 3D data.**

I. Introduction

Navigating large online collections of 3D objects is still restricted by limited search capabilities (*e.g.* Yeggi[1] and 3D Warehouse[2] provide a simple text based search, while the latter also provides a similar-shape search based only on global shape similarity). Understanding objects at a part or component level will be critical for the next generation of 3D search, analysis, and visualization. To this end we consider the problem of automatically detecting the parts of a 3D model.

Recent efforts towards 3D mesh labeling (*e.g.* [13], [26]) operate under the strong assumption that fully segmented and labeled meshes are available for training. This requirement makes it difficult to scale the proposed solutions to a real-world setting since collecting this kind of training data requires monumental human effort. We suggest an alternative approach of learning from *sparsely* labeled 3D objects (*e.g.* for a camera model, the presence and location of the lens would be indicated by a single labeled vertex on the mesh). While learning from a minimal labeling presents a unique challenge (since there is no indication as to the shape or scale of the object parts), collecting large training datasets of this form is more realistic since users would have to indicate only a single point per object part rather than providing a complex region boundary.

[1] http://www.yeggi.com/
[2] http://3dwarehouse.sketchup.com

Figure 1. Examples where localized annotations enhance product visualization: left: maybe3d.com, right: dillards.com

To complement our training data, our objective is also to predict highly localized annotations and call out a single representative vertex per detected component. Identifying object components by highlighting a single vertex is important for visualization tasks (see fig. 1). To produce localized annotations, our approach is built upon the observation that the spatial configuration between object parts provides a crucial cue towards their discrimination. For example, while the different types of buttons on an electronic device may not be distinguishable from each other in terms of shape or appearance, their layout on the device may provide the strongest indication of their function. Inspired by the success of contextual layout models applied to object recognition in images, we propose a spatial layout model for 3D object part detection.

In this paper we introduce (to our knowledge) the first textured 3D model part labeling algorithm designed to learn from sparse labels and to predict localized detections. Our approach utilizes a spatial layout model that builds upon contextual models used for multi-class object detection in images, specifically [5] . We present an algorithm that combines independently trained part classifiers with a structured SVM model. Promising experimental results on unique large-scale datasets of real-world textured 3D objects confirm that spatial layout is vital signal for part detection.

A. Related Work

The problem of mesh labeling has largely been approached via segmentation (see [20], [8], [10], [22]). Most similar to our work are the supervised approaches for joint

segmentation and labeling [13], [26]. For example [13] employs a contextual CRF [16] model where the spatial term encodes the smoothness constraint for labels of adjacent mesh faces. These methods require densely labeled training data which limits their application at scale. Furthermore, they cannot be adapted in a straightforward manner to our task since spatial smoothness is not a constraint that can be applied to our sparse labeling setting.

Our problem setting is most similar to that of multi-class object detection in images (the components to be detected in 3D model are analogous to the object classes that can appear in an image). Structured prediction techniques [25], [23], [16] have regularly been utilized for labeling, segmentation, and detection problems in images. The most common approaches treat labeling as learning and inference on graphical models. Neighborhood edges provide for smoothness, and "long range" edges can be used to encode the spatial layout (see [9], [14], [29] among others).

Deformable part models provide an appealing framework for representing the part configurations within an object category [6], [7]. Here object appearance is represented by a single root model and individual part models, and the layout score is determined by the spatial placement of the parts relative to the root. Human-selected vertices in CAD models have been used to train deformable models for pose estimation in images [30]. The spatial layout model presented in [5] utilizes a two-stage approach for multi-class object detection in images. Independently trained sliding window object detectors are combined with a structured prediction step that enforces a learned spatial layout model. Our approach follows this framework: individual 3D object part classifiers are learned from the sparsely labeled training data, and subsequently a structured-SVM model that captures all the pairwise relationships between object parts is trained. In the following sections we detail our approach.

II. 3D PART DETECTION AS STRUCTURED PREDICTION

We focus our attention on labeling 3D models belonging to the same category (in practical applications, the category can be discerned from associated text descriptions or metadata, or reliably determined from automated techniques [19]). In our setting a textured 3D object is represented by a triangulated mesh and an associated image texture map. We let the mesh provide the discretization: each triangle is an independent node that can be positively detected as an object part. For a mesh with n triangles, we select the n triangle centroids to comprise the vertex set X input to the detection task: $X = \left\{x_i \in \mathbb{R}^3 | i = 1, \ldots, n\right\}$.

Let $\mathcal{D} = \left\{l_1, l_2, \ldots, l_{|\mathcal{D}|}\right\}$ be the dictionary of parts for the category (*e.g.* for the category *camera*, $\mathcal{D}$ ={*lens*, *flash*, *on/off button*, *tripod socket*, ...}). The labels we assign to detected parts will come from $\mathcal{D}$, but since large areas of the object may not belong to any part in the dictionary, we include a background label l_b in the final label set: $\mathcal{Y} = \mathcal{D} \cup \{l_b\}$. A labeling of an object assigns a label from $\mathcal{Y}$ to each vertex: $Y(X) = \{y_i \in \mathcal{Y} | i = 1, \ldots, n\}$ (we will write Y instead of $Y(X)$ for simplicity). At training, we are given models paired with ground truth vertex labels (X, Y). Note, as discussed earlier Y is a *sparse* labeling, meaning for any object part all but one representative vertex within the part region will be labeled as *background* (l_b).

Our task is to predict a labeling Y given a previously unseen model X. The output Y has the same sparsity requirement as the input training data (only one representative vertex should be labeled as non-background per detected part). Any vertex with $y_i \neq l_b$ will be considered a unique positive part detection. For our purposes the representative vertex we are trying to detect should reflect the detections provided at training time. This is motivated by studies that indicate the user-selected vertices we see at training are likely to be good representatives for visualization and identification [3]. Thus, our task boils down to predicting the same type of sparse detections that are provided in the ground truth training data. Let us define the score of an object labeling as

$$S(X, Y) = \sum_{i,j} s_2(x_i, y_i, x_j, y_j) + \sum_i s_1(x_i, y_i) \qquad (1)$$

Intuitively, the score s_1 measures how well the local geometry and appearance of vertex x_i matches that of the training points labeled as y_i. The score s_2 measures how well the relative spatial configuration of vertices x_i and x_j matches that of training vertex pairs labeled with y_i and y_j. The detection task is to compute the best labeling for a 3D object: $\tilde{Y} = \arg\max_Y S(X, Y)$.

Since there exist $|\mathcal{Y}|^n$ labeling variations, we utilize powerful discriminative classifiers to reduce the output space and guide the detection. Let $c_y(x_i)$ be a binary classifier trained for class $y \in \mathcal{D}$, such that $c_y(x_i) > 0$ for a positive classification (see the following section for classifier details). We reduce the set of vertices to be labeled and the corresponding pool of potential labels:

$$X = \{x_i \mid \exists y \in \mathcal{D} \quad s.t. \quad c_y(x_i) > 0\} \qquad (2)$$

$$Y = \{y_i \mid y_i \in \mathcal{Y} \setminus \{y \in \mathcal{D} | c_y(x_i) \leq 0\}\} \qquad (3)$$

Following [5] we utilize a simple linear model for the scoring functions s_1 and s_2. While we have freedom to choose the vertex representation $f(x_i)$, we use the positive classifier detection score as the vertex feature as in [5]: $f(x_i) = [c_{y_i}(x_i),\ 1]^T$ (the 1 is appended to help capture inter-class classifier score biases at training). This definition of $f(x_i)$ assumes only one classifier has fired for vertex x_i. For multiple classifications at the same vertex, it is straightforward to produce an $f(x_i)$ for each positive classifier, but we omit this here to keep the notation simple. We can now write $s_1(x_i, y_i) = w_{y_i}^T f(x_i)$, where $w_{y_i} \in \mathbb{R}^2$ are the weights for class $y_i \in \mathcal{Y}$ to be learned at training.

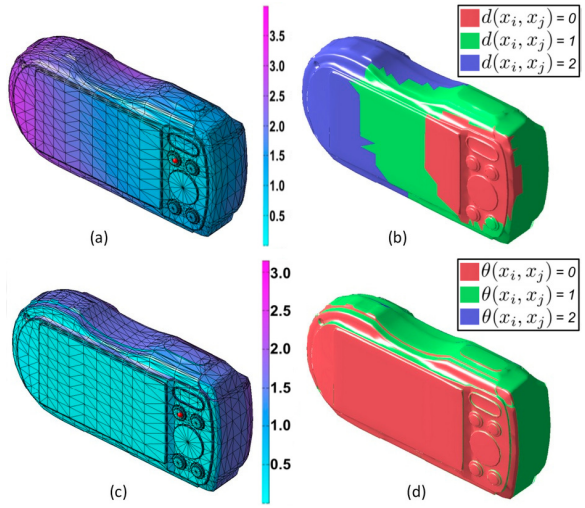

Figure 2. (a) normalized geodesic distance to the selected face, (b) geodesic distance bins, (c) angle between face normals of the selected face and others, (d) angle bins.

To capture the pairwise spatial relationships between vertices, we choose a straightforward quantization of relative distance and orientation:

$$d(x_i,x_j) = \min\left(2, \lfloor g(x_i,x_j)/\sigma_X \rfloor\right) \quad (4)$$

$$\theta(x_i,x_j) = \min\left(2, \left\lfloor 3\arccos(\hat{x}_i^T \hat{x}_j)/\pi \right\rfloor\right) \quad (5)$$

Here $g(x_i,x_j)$ is the geodesic distance between vertices (which we approximate with shortest path distances), σ_X is a scale normalizer computed as the 15th percentile of all-pairs geodesic distances on the mesh, and $\hat{x}_i$ is the unit surface normal at x_i. Both relative distance and orientation are quantized into one of three values.[3] Fig. 2 shows an example of the quantization. The final binning for spatial layout is given with $\alpha(x_i,x_j) \in \{0,1\}^9$ where

$$\alpha(x_i,x_j)_k = \begin{cases} 1 & \text{if } k == 3d(x_i,x_j) + \theta(x_i,x_j) \\ 0 & \text{otherwise} \end{cases}$$

We define $s_2(x_i,y_i,x_j,y_j) = w_{y_i,y_j}^T \alpha(x_i,x_j)$, where $w_{y_i,y_j} \in \mathbb{R}^9$ are the weights for labels y_i and y_j appearing in the 9 different spatial configurations. Note, the spatial layout terms are symmetric ($\alpha(x_i,x_j) = \alpha(x_j,x_i), w_{y_i,y_j} = w_{y_j,y_i}$). Eq 1 is now:

$$S(X,Y) = \sum_{i,j} w_{y_i,y_j}^T \alpha(x_i,x_j) + \sum_i w_{y_i}^T f(x_i) \quad (6)$$

Note, we constrain all weights involving the background label to be 0 (*i.e.* if $y_i = l_b$ then $w_{y_i} = 0$ and $w_{y_i,y_j} = w_{y_j,y_i} = 0$. This is not strictly required but speeds up inference. In the next section we detail the process of learning classifiers c_{y_i}.

[3] Although it is trivial to generalize this for a finer quantization, we found no benefit in our experiments with additional bins and performance begins to drop beyond 5 bins for each of d, θ (this could be explained by the limited number of pairwise observations at training).

A. *Vertex classification*

The classifiers c_y contribute significantly to our part detection pipeline (*e.g.* we are unable to recover from false negatives from c_y). A crucial input to building robust classifiers is selecting appearance and shape features that can be useful for part discrimination. We adopt the geometry features from [13] which contribute the most for mesh classification: *Spin Images* [12], *3D Shape Contexts* [1], *Curvature*, and *PCA coefficients*. In addition to surface geometry, the appearance of a model provides rich information for discrimination. For any surface point, we orthographically project the model texture along the surface normal direction [28], and compute a Sift [17] descriptor in the rendered image plane. At each vertex the descriptors are extracted at multiple scales and concatenated into a single 426-dimensional vector $\vec{x}$.

We choose to compute c_y as a one-vs-all binary polynomial-kernel SVM classifier [4] trained over descriptors $\vec{x}$ (c_y is the SVM decision score and $c_y > 0$ indicates a positive classification for label y). [4]

B. *Inference*

Generating a final set of detections for a model requires computing $\arg\max_Y S(X,Y)$. The most common approaches to inference on graphical models with cycles are message passing algorithms such as *Loopy Belief Propagation* (LBP) and *Tree Re-Weighted Belief Propagation* [27], [15]. Drawbacks to these approximate inference methods include no guarantees on convergence (LBP) and high computational complexity (*e.g.* for fully connected graphs). A greedy forward search was presented in [5] which showed performance comparable to LBP. In our setting, this greedy search can be described with the following steps: (1) initialize all vertices with the label l_b, (2) select the vertex that when labeled as non-background would increase $S(X,Y)$ by the largest value, (3) repeat step (2) until no score-increasing vertex remains. The biggest advantage of the greedy search is large speedup over the message passing algorithms. However, in our experiments we found this forward search often led to configurations for Y that were quite far from the LBP-generated solution. As an alternative, we introduce a more general greedy search. Our approach follows [5] with the distinction that we also allow deselections (returning a selected vertex to have the label l_b) and vertex-pair toggling. Before presenting the pseudo-code, let $\{(x,y)|y \in \mathcal{D}, c_y(x) > 0\}$ be the set of vertex-label pairs from which we are trying to select our final detections, and let $I_{(x,y)} \in \{-1,1\}$ be an indicator function ($I_{(x,y)} = 1$ implies x has been labeled with y in the final output). The pseudo-code is given in algorithm 1. To keep the pseudocode simple we assume each vertex has only one potential object part label. To handle multiple classifications per vertex, we simply disallow selecting the pair (x,y) if there already

[4] See the supplementary file for more details.

Algorithm 1 Greedy Inference

Initialize: $I_{(x,y)} = -1$, $\Delta(x,y) = 0$, and $S = 0$
$z(x,y) = s_1(x,y) + \Delta(x,y)$
$(x^*, y^*) = \underset{(x,y)s.t.I_{(x,y)}=-1}{\arg\max} z(x,y)$
if $z(x^*, y^*) \geq 0$ **then** ▷ Select
  $\Delta(x,y) = \Delta(x,y) + 2s_2(x,y,x^*,y^*)$
  $S = S + z(x^*, y^*)$
  $I_{(x^*,y^*)} = 1$
  GOTO step 2
end if
$(x^*, y^*) = \underset{(x,y)s.t.I_{(x,y)}=1}{\arg\min} z(x,y)$
if $z(x^*, y^*) < 0$ **then** ▷ Deselect
  $\Delta(x,y) = \Delta(x,y) - 2s_2(x,y,x^*,y^*)$
  $S = S - z(x^*, y^*)$
  $I_{(x^*,y^*)} = -1$
  GOTO step 2
end if
$z_1(x,y,x',y') = -I_{(x,y)}z(x,y) - I_{(x',y')}z(x',y') + 2I_{(x,y)}I_{(x',y')}s_2(x,y,x',y')$
$(x^*, y^*, x^{*'}, y^{*'}) = \underset{x,y,x',y'}{\arg\max} z_1(x,y,x',y')$
if $z_1(x^*, y^*, x^{*'}, y^{*'}) > 0$ **then** ▷ Toggle
  $S = S + z_1(x^*, y^*, x^{*'}, y^{*'})$
  $\Delta(x,y) = \Delta(x,y) - 2I_{(x^*,y^*)}s_2(x,y,x^*,y^*)$
  $\Delta(x,y) = \Delta(x,y) - 2I_{(x^{*'},y^{*'})}s_2(x,y,x^{*'},y^{*'})$
  $I_{(x^*,y^*)} = -I_{(x^*,y^*)}$
  $I_{(x^{*'},y^{*'})} = -I_{(x^{*'},y^{*'})}$
  GOTO step 3
end if
terminate

exists a selection for that vertex (*i.e.* $I_{(x,y')} = 1, y' \neq y$). At each iteration we prioritize selecting a single vertex-label pair over deselecting and toggling. Intuitively, this algorithm can be viewed as initialization with a greedy forward search followed by *select-deselect-toggle* improvements (note that the deselect and toggle steps can only increase the score).

C. Learning

We now discuss learning the weights w_{y_i,y_j} and w_{y_i} in our scoring function $S(X,Y)$ (eq. 6). We will formulate our task as a regularized learning problem which can be directly optimized with the structured SVM algorithm [25]. We first rewrite our scoring function $S(X,Y)$ with a joint feature map:

$$s_1(x,y) = w_y^T f(x) = w_{s_1}^T f_{s_1}(x,y) \quad (7)$$

$$s_2(x_i,y_i,x_j,y_j) = w_{y_i,y_j}^T \alpha(x_i,x_j) \quad (8)$$

$$= w_{s_2}^T \alpha_{s_2}(x_i,y_i,x_j,y_j) \quad (9)$$

Letting d_f be the dimensionality of $f(x)$ ($d_f = 2$), then $w_{s_1} \in \mathbb{R}^{(d_f * |\mathcal{Y}|)}$ (each of the $|\mathcal{Y}|$ blocks of d_f dimensions corresponds to one of the w_y). Similarly $f_{s_1}(x,y) \in \mathbb{R}^{(d_f * |\mathcal{Y}|)}$ with values of $f(x)$ in the dimensions corresponding to label y. Letting K be the dimensionality of $\alpha(x_i,x_j)$ ($K = 9$), then $w_{s_2} \in \mathbb{R}^{(K|\mathcal{Y}|^2)}$ (each of the $|\mathcal{Y}|^2$ blocks of K dimensions corresponds to one of the w_{y_i,y_j}). Similarly, $\alpha_{s_2}(x_i,y_i,x_j,y_j) \in \mathbb{R}^{(K|\mathcal{Y}|^2)}$ with values of $\alpha(x_i,x_j)$ in the dimensions corresponding to (y_i,y_j). We now have

$$S(X,Y) = w^T \Psi(X,Y), \quad (10)$$

$$w = \begin{bmatrix} w_{s_1} \\ w_{s_2} \end{bmatrix}, \Psi(X,Y) = \begin{bmatrix} \sum_i f_{s_1}(x_i,y_i) \\ \sum_{i,j} \alpha_{s_2}(x_i,y_i,x_j,y_j) \end{bmatrix}$$

Letting N be the number of ground-truth labeled training examples (X_i, Y_i), the regularized learning of w with margin-rescaling structured SVM [25] is:

$$\min_{w,\xi} \frac{1}{2}\|w\|^2 + \frac{C}{N}\sum_{i=1}^{N}\xi_i$$
$$\text{s.t.} \forall i, \forall Y' : w^T(\Psi(X_i,Y_i) - \Psi(X_i,Y')) \geq L(X_i,Y_i,Y') - \xi_i \quad (11)$$

We select a decomposable loss function $L(X,Y,Y') = \sum_i l(x_i,y_i,y_i')$ where

$$l(x_i,y_i,y_i') = \begin{cases} 1 & \text{if } y_i' \neq l_b \wedge \nexists x_j \in V(x_i) \text{ s.t. } y_j = y_i' \\ 1 & \text{if } y_i' = l_b \wedge y_i' \neq y_i \\ 0 & \text{otherwise} \end{cases}$$

This loss penalizes detections if they are outside the neighborhood of the ground truth location ($V(x)$). In our experiments the neighborhood radius is set to the median distance between all adjacent vertices (*e.g.* for a uniform mesh $V(x)$ would contain x and the one-ring neighbors of x). This loss is more forgiving than a hamming loss, and in our case necessary since we cannot reasonably expect to select exact vertices for detection.

To optimize eq. 11 we utilize the cutting plane method [11], which during optimization requires the computation of the "most violated" constraint:

$$\arg\max_{Y'} L(X,Y,Y') - w^T(\Psi(X,Y) - \Psi(X,Y')) = \arg\max_{Y'} L(X,Y,Y') + w^T\Psi(X,Y') \quad (12)$$

The contribution of the decomposable loss function above can be folded into the weight vector. This means algorithm 1 can be modified trivially to also provide a solution for eq. 12. Having defined the joint feature map Ψ, loss function L, and approximate inference for the most violated constraint, the cutting plane optimization for eq. 11 is performed with the implementation provided in SVM$^{\text{struct}}$ (svmlight.joachims.org).

D. Part multiplicity

To complement the detection algorithm presented above, we will utilize observations on object part multiplicity to further constrain our predictions. An intra-object multiplicity constraint makes practical sense for 3D objects (*e.g.* it is not unreasonable to limit the number of lenses on a camera, or legs on a human). The multiplicity of label y' in training

Figure 3. Sample models and human annotations from the cameras set (top) and video recorders set (bottom).

example n is defined as $\tau_{n,y'} = \sum_i \delta(y_i == y')$. The upper bound multiplicity for label y' is set as the max of all training examples: $\tau(y') = \max\{\tau_{n,y'} | n = 1, \ldots, N\}$ (we set $\tau(l_b) = \infty$). For any 3D model in which we apply our part detection algorithm, we require $\sum_i \delta(y_i == y') \leq \tau(y')$. This requirement is easily applied in our greedy inference (sec II-B). At each iteration, we only consider a vertex-label pair (x, y') for selection (either during a forward or toggle step) if the result of the selection will not violate $\tau(y')$. Additionally, we typically see convergence speedups during inference since at each iteration the constraints τ greatly reduce the set of valid vertex-label pairs.

III. Datasets

We obtained two datasets from maybe3d.com. The *cameras* set contains 360 models (298 train, 62 test, 34 part labels). The *video recorders* set contains 64 models (50 train, 14 test, 12 part labels). [4] This dataset presents an interesting challenge due to many part types that are very small and difficult to detect and distinguish (*e.g.* buttons, connectors).

IV. Experiments

To be evaluated as a true positive, a part detection must pass a proximity test: (1) it must be within a small distance of the matching part's ground truth location (within neighborhood $V()$ as defined earlier) and (2) it must not be closer to any other part's ground truth location. Furthermore, since we aim for a sparse labeling, at most one detection per object part can be a true positive (if multiple detections for the same part pass the proximity test all but the closest will be evaluated as false positives).

Descriptors and classifiers It is important to know how well our choice of descriptors and classifiers perform independent of the structured prediction. This will also provide a reasonable baseline for evaluating the structured prediction. The SVM classifier is compared against two reasonable alternatives: (1) a k-NN classifier implemented simply as a majority vote of the $k = 5$ neighbors, and (2) JointBoost [24] which has shown state-of-the-art performance for classifying shape descriptors [13]. Table I shows that the SVM classifiers are a reasonable choice for c_y. To better understand the contribution of the descriptors, we observe the relative performance

	5-NN	JointBoost [24]	SVM
MAP	0.36	0.39	**0.42**

Table I
Mean per-label AP for classifiers on the cameras.

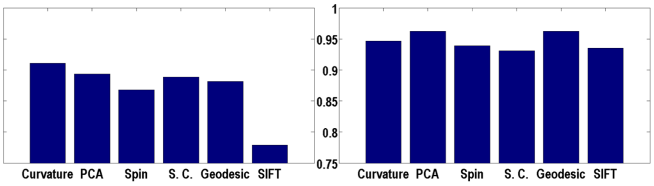

Figure 4. Relative performance of classifier when omitting one feature type. Left: cameras, right: video recorders.

of the classifiers when using a subset of the full feature ensemble. Fig. 4 shows the relative performance when one feature type is omitted from the ensemble. Interestingly, leaving out the appearance descriptor hurts the most for the cameras set, but the contribution is more uniform across feature types for video recorders.

Inference. We denote I_1 as the greedy forward search inference algorithm of [5], and I_2 our proposed generalized greedy inference (algorithm 1). Table II shows the mean inference score over camera test models for both algorithms compared to LBP. We see that I_2 generates scores closer to LBP than I_1, and for the cameras set I_1 had improved the score over I_2 for a full 25% of the models.

Structured prediction. Let SVM_{I_1} refer to the structured prediction algorithm using inference I_1 (no multiplicity constraints), $SVM^c_{I_1}$ for the same algorithm with multiplicity constraints, and $SVM^c_{I_2}$ for our proposed inference I_2 with multiplicity constraints. Table III shows the mean per-label AP performance for both datasets. Performance improves significantly when adding any structured prediction layer to the base SVM classifiers. As expected from table II we see small but consistent improvement when choosing I_2 over I_1, and similarly the multiplicity constraints have a consistently positive impact.

The high mean per-label AP scores in table III are a misleading indicator of the difficulty of this dataset. This is in part due to a number of labels which appear relatively infrequently in the test set but have high performance. Another perspective is obtained by measuring AP over all detections. In order to compare detections across labels, we update our scoring to use an approximation of the marginal

	I_1 [5]	I_2	LBP [18]
mean score	3.99	4.29	4.32

Table II
Average inference scores for different techniques.

	SVM	SVM_{I_1}	$SVM^c_{I_1}$	$SVM^c_{I_2}$
cameras	0.42	0.55	0.61	**0.67**
video recorders	0.73	0.81	0.86	**0.88**

Table III
MEAN PER-LABEL AP PERFORMANCE.

	SVM_{I_1}	$SVM^c_{I_1}$	$SVM^c_{I_2}$
cameras	0.46	0.55	**0.57**
video recorders	0.52	0.61	**0.63**

Table IV
MEAN PER-3D MODEL AP PERFORMANCE.

posterior probabilities (see [5] for details). Note this scoring is only used for performance evaluation as it is not required for our algorithm to detect parts. Table IV shows the mean AP per 3D object (after scores have been merged across classes). Although the performance trends here are similar to table III, the absolute performance is significantly lower. Going one step further, fig. 5 shows the PR curves over all detections in the test set. The performance trends are generally the same as earlier with the exception that the multiplicity constraints hinder performance at the higher-recall segment of the curves. More importantly, these PR curves accurately highlight the challenging nature of the sparse labeling problem. Typical detection results for both datasets are visualized in fig. 6. The zoomed portions of the images show examples where a spatial layout model can resolve many detections and false positives correctly.

Test-vs-train dissimilarity. To better understand the limits of our method, we propose an extreme test that increases the variation between the training and test sets. Although the low performance of the k-NN classifier (see table I) indicates there is already quite a bit of variation, we go

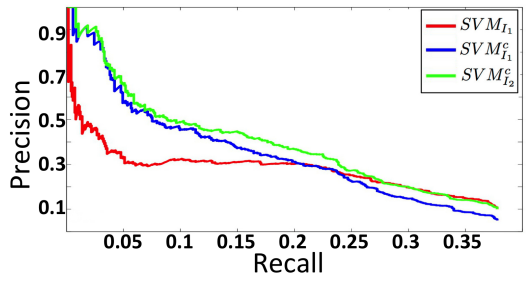

Figure 5. Precision-recall curves for the cameras set.

	SVM	SVM_{I_1}	$SVM^c_{I_1}$	$SVM^c_{I_2}$
cameras (brand)	0.17	0.20	0.21	**0.22**
cameras (resolution)	0.35	0.42	0.45	**0.47**

Table V
MEAN PER-LABEL AP FOR DIFFERENT TEST SCENARIOS.

	SVM	$SVM^c_{I_2}$	CRF [13]
per-label AP	0.51	0.54	-
segment match rate	0.66	0.73	86.3

Table VI
FIRST ROW: MEAN PERFORMANCE OVER ALL 8 CATEGORIES. SECOND ROW: LABELING ACCURACY RATE MEASURED BY INTERSECTION WITH PART SEGMENTATION.

further by creating mutually exclusive partitions based on camera brand (*e.g.* for brand *Nikon*, train on all non-*Nikon* models and test only on *Nikon* models). The average results over 8 camera brand names are shown in the top row of table V. Although the overall performance drops expectedly when emphasizing the difference between train and test sets, there are still elements of the spatial configuration that can be exploited as we still see sizeable improvements when applying the spatial layout models. [4]

Mesh resolution variability. Table V (second row) evaluates sensitivity to mesh resolution. In this experiment we create a spatially non-uniform mesh alteration by subdividing each mesh triangle in the training set into three smaller ones with probability $p = 0.35$. To increase the challenge we subdivide triangles in the test set at a different rate ($p = 0.15$). Despite these alterations, the performance trends remain consistent (compare with table III). However, the relative changes indicate the feature descriptors are more robust to resolution than the spatial layout model. [4]

Densely labeled models. While our approach is designed for realistic textured objects, we perform additional experiments with data where dense part segmentations are available for evaluation. A portion of the Princeton shape dataset [21] (PSB) has been densely labeled by [2]. We choose 8 categories which each contain 20 models (split into 14 train and 6 test). To simulate a sparse labeling we select the vertex at the center of each part segment (this is driven by the observation in [3] that "segment centeredness" is key factor when humans are tasked with selecting points on a 3D surface). Results averaged over all 8 categories are shown in the top row of table VI. The challenges here are (1) limited training data (this reinforces our motivation for working with sparse annotations which are easy to collect at scale), and (2) articulated shapes. In such a setting, accurately learning the spatial layout model is difficult and the results confirm this as there is only a small gain from our model. Fig. 6 shows detections from two of the articulated categories, Ant and Armadillo. After applying the spatial layout inference $SVM^c_{I_2}$ there are still multiple hand detections on the Armadillo hand (top row). This is a failure case where the spatial layout model may not have learned the relative distance weights for hand parts accurately.

Additionally, we evaluate how frequently our detections land in the correct segment (bottom row of table VI). Even if detections are not localized precisely they are frequently

capturing the correct segment on the model. We implement the CRF model of [13] which has shown state of the art results on this dataset, and the labeling recognition rate (i.e. per-triangle label accuracy) is reported in the bottom row of table VI. Although it is an apples-to-oranges comparison with our accuracy, this does give idea of the gap between learning from sparsely vs densely labeled models. [4]

V. Conclusion

This work presents a novel approach for a part-based understanding of 3D objects that exploits the spatial layout of parts. It is scalable for real-world applications since it requires only sparsely labeled models for training. Experimentals show that a spatial configuration model can dramatically improve detection results when the individual parts may not be easily distinguishable independently. Moreover, this performance gain is still observable under extreme scenarios (minimal similarity between the test and train sets and mesh resolution variation). From the experiments on data where dense labeling is available, we see the two biggest limitations of our work are learning from limited data and modeling articulated shapes. It is promising to see that learning from sparse labels produces an understanding of object parts that is not far from what can be achieved with full segmentations. A future direction of this work will be to investigate how spatial layout can be utilized to address other challenges in 3D model analysis, such as fine-grained categorization.

References

[1] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *Pattern Anal. Mach. Intell.*, 24(4):509–522, Apr. 2002. 3

[2] X. Chen, A. Golovinskiy, and T. Funkhouser. A benchmark for 3D mesh segmentation. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 28(3), Aug. 2009. 6

[3] X. Chen, A. Saparov, B. Pang, and T. Funkhouser. Schelling points on 3d surface meshes. *ACM Transactions on Graphics (TOG)*, 31(4):29, 2012. 2, 6

[4] C. Cortes and V. Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995. 3

[5] C. Desai, D. Ramanan, and C. C. Fowlkes. Discriminative models for multi-class object layout. *International journal of computer vision*, 95(1):1–12, 2011. 1, 2, 3, 5, 6

[6] P. Felzenszwalb, D. McAllester, and D. Ramanan. A discriminatively trained, multiscale, deformable part model. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008. 2

[7] P. F. Felzenszwalb and D. P. Huttenlocher. Pictorial structures for object recognition. *International Journal of Computer Vision*, 61(1):55–79, 2005. 2

[8] A. Golovinskiy and T. Funkhouser. Consistent segmentation of 3d models. *Computers & Graphics*, 33(3):262–269, 2009. 1

[9] X. He, R. S. Zemel, and M. A. Carreira-Perpiñán. Multiscale conditional random fields for image labeling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 695–703, 2004. 2

[10] Q. Huang, V. Koltun, and L. Guibas. Joint shape segmentation with linear programming. In *ACM Transactions on Graphics*, volume 30, page 125, 2011. 1

[11] T. Joachims, T. Finley, and C.-N. Yu. Cutting-plane training of structural svms. *Machine Learning*, 77(1):27–59, 2009. 4

[12] A. Johnson. *Spin-Images: A Representation for 3-D Surface Matching*. PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, August 1997. 3

[13] E. Kalogerakis, A. Hertzmann, and K. Singh. Learning 3D Mesh Segmentation and Labeling. *ACM Transactions on Graphics*, 29(3), 2010. 1, 2, 3, 5, 6, 7

[14] P. Kohli, L. Ladický, and P. H. Torr. Robust higher order potentials for enforcing label consistency. *Int. J. Comput. Vision*, 82(3):302–324, May 2009. 2

[15] V. Kolmogorov. Convergent tree-reweighted message passing for energy minimization. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(10):1568–1583, 2006. 3

[16] J. D. Lafferty, A. McCallum, and F. C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the ICML*, pages 282–289, 2001. 2

[17] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, Nov. 2004. 3

[18] Meltzer. http://www.cs.huji.ac.il/ talyam/inference.html. 5

[19] R. Ohbuchi, K. Osada, T. Furuya, and T. Banno. Salient local visual features for shape-based 3d model retrieval. In *In SMI*, 2008. 2

[20] A. Shamir. A survey on mesh segmentation techniques. In *Computer graphics forum*, volume 27, pages 1539–1556. Wiley Online Library, 2008. 1

[21] P. Shilane, P. Min, M. Kazhdan, and T. Funkhouser. The Princeton shape benchmark. In *Shape Modeling International*, 2004. 6

[22] O. Sidi, O. van Kaick, Y. Kleiman, H. Zhang, and D. Cohen-Or. Unsupervised co-segmentation of a set of shapes via descriptor-space spectral clustering. In *ACM Transactions on Graphics (TOG)*, volume 30, page 126. ACM, 2011. 1

[23] B. Taskar, C. Guestrin, and D. Koller. Max-margin markov networks. In *Advances in Neural Information Processing Systems*. MIT Press, Cambridge, MA, 2004. 2

[24] A. Torralba, K. P. Murphy, and W. T. Freeman. Sharing visual features for multiclass and multiview object detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(5):854–869, 2007. 5

[25] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. *J. Mach. Learn. Res.*, 6:1453–1484, Dec. 2005. 2, 4

[26] O. Van Kaick, A. Tagliasacchi, O. Sidi, H. Zhang, D. Cohen-Or, L. Wolf, and G. Hamarneh. Prior knowledge for part correspondence. In *Computer Graphics Forum*, volume 30, pages 553–562. Wiley Online Library, 2011. 1, 2

[27] Y. Weiss and W. T. Freeman. Correctness of belief propagation in gaussian graphical models of arbitrary topology. *Neural computation*, 13(10):2173–2200, 2001. 3

[28] C. Wu, B. Clipp, X. Li, J.-M. Frahm, and M. Pollefeys. 3d model matching with viewpoint-invariant patches (vip). *2012 IEEE Conference on Computer Vision and Pattern Recognition*, 2008. 3

[29] Y. Zhang and T. Chen. Efficient inference for fully-connected crfs with stationarity. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 582–589, Washington, DC, USA, 2012. IEEE Computer Society. 2

[30] M. Z. Zia, M. Stark, B. Schiele, and K. Schindler. Detailed 3d representations for object recognition and modeling. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(11):2608–2623, 2013. 2

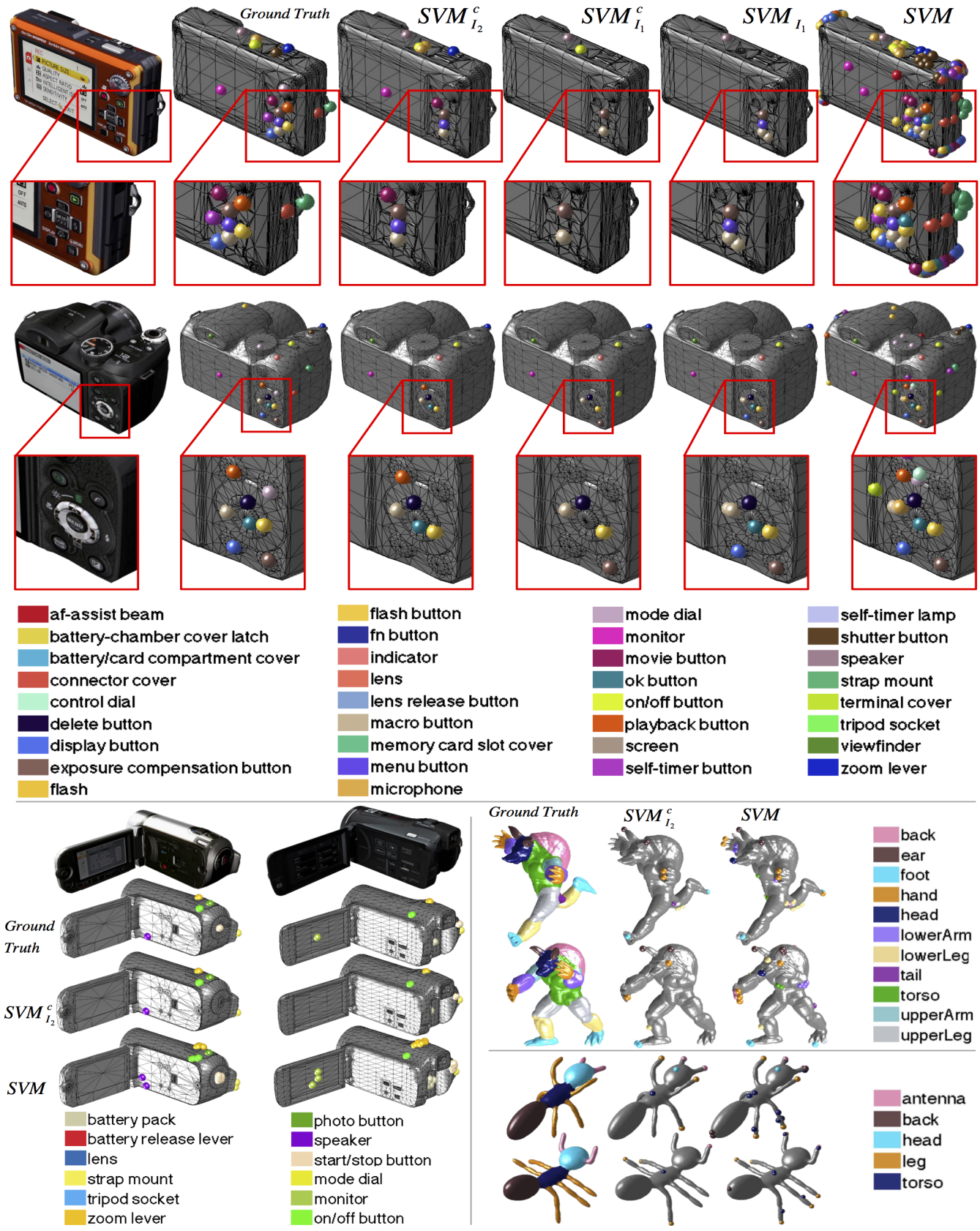

Figure 6. Ground truth sparse labels and detection results of the SVM baseline and our spatial layout models. Top: cameras set. Bottom left: video recorders set. Bottom right: examples from two of the articulated categories in the PSB set (Armadillo on top, Ant on bottom). For more clarity please zoom and view in color.