Contextually Guided Semantic Labeling and Search for 3D Point Clouds

Abhishek Anand*, Hema Swetha Koppula*, Thorsten Joachims, Ashutosh Saxena
Department of Computer Science, Cornell University.
{aa755,hema,tj,asaxena}@cs.cornell.edu

***Abstract*— RGB-D cameras, which give an RGB image together with depths, are becoming increasingly popular for robotic perception. In this paper, we address the task of detecting commonly found objects in the 3D point cloud of indoor scenes obtained from such cameras. Our method uses a graphical model that captures various features and contextual relations, including the local visual appearance and shape cues, object co-occurence relationships and geometric relationships. With a large number of object classes and relations, the model's parsimony becomes important and we address that by using multiple types of edge potentials. We train the model using a maximum-margin learning approach. In our experiments over a total of 52 3D scenes of homes and offices (composed from about 550 views), we get a performance of 84.06% and 73.38% in labeling office and home scenes respectively for 17 object classes each. We also present a method for a robot to search for an object using the learned model and the contextual information available from the current labelings of the scene. We applied this algorithm successfully on a mobile robot for the task of finding 12 object classes in 10 different offices and achieved a precision of 97.56% with 78.43% recall.**[1]

I. Introduction

Inexpensive RGB-D sensors that augment an RGB image with depth data have recently become widely available. These cameras are increasingly becoming the de-facto standard for perception for many robots. At the same time, years of research on SLAM (Simultaneous Localization and Mapping) has now made it possible to merge multiple RGB-D images into a single point cloud, easily providing an approximate 3D model of a complete indoor scene (i.e., a room). In this paper, we explore how this move from part-of-scene 2D images to full-scene 3D point clouds can improve the richness of models for object labeling.

In the past, a significant amount of work has been done in semantic labeling of 2D images (Murphy et al., 2003; Heitz and Koller, 2008; Felzenszwalb et al., 2008; Collet et al., 2011; Li et al., 2012). However, a lot of valuable information about the 3D shape and geometric layout of objects is lost when a 2D image is formed from the corresponding 3D world. A classifier that has access to a full 3D model can access important geometric properties in addition to the local shape and appearance of an object. For example, many objects occur in characteristic relative geometric configurations (e.g., a monitor is almost always on a table), and many objects consist of visually distinct parts that occur in a certain relative configuration. More generally, a 3D model makes it possible to reason about a variety of 3D properties such as 3D distances, volume and local convexity.

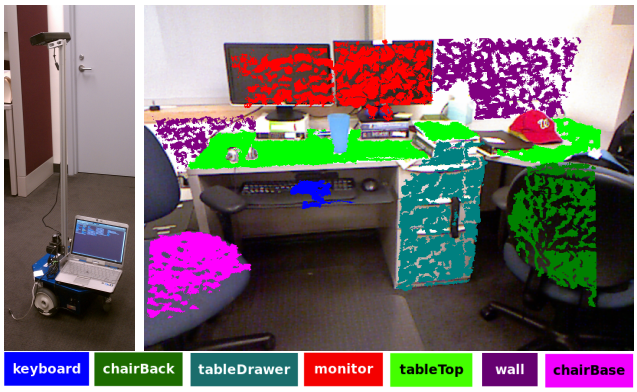

Fig. 1. (Left) Cornell's Blue robot mounted with an RGB-D camera (Microsoft Kinect). (Right) Predicted labeling of a scene.

Some previous works attempt to first infer the 3D structure from 2D images (Saxena et al., 2005; Hoiem et al., 2006; Saxena et al., 2009; Divvala et al., 2009) for improving object detection. However, the inferred 3D structure is not accurate enough to give significant improvement. Another recent work (Xiong and Huber, 2010) considers labeling a scene using a single 3D view (i.e., a 2.5D representation). In our work, we first use SLAM in order to compose multiple views from a Microsoft Kinect RGB-D sensor into one 3D point cloud, providing each RGB pixel with an absolute 3D location in the scene. We then (over-)segment the scene and predict semantic labels for each segment (see Fig. 2). We not only predict coarse classes like in (Xiong and Huber, 2010; Anguelov et al., 2005) (i.e., wall, ground, ceiling, building), but also label individual objects (e.g., printer, keyboard, monitor). Furthermore, we model rich relational information beyond an associative coupling of labels (Anguelov et al., 2005).

In this paper, we propose and evaluate the first model and learning algorithm for scene understanding that exploits rich relational information derived from the full-scene 3D point cloud for object labeling and search. In particular, we propose a graphical model that naturally captures the geometric relationships of a 3D scene. Each 3D segment is associated with a node, and pairwise potentials model the relationships between segments (e.g., co-planarity, convexity,

[1]Parts of this work have been published at (Koppula et al., 2011a,b); those works did not present our contextually-guided search algorithm and the robotic experiments. This submission also includes more details on the algorithm and results.

* indicates equal contribution.

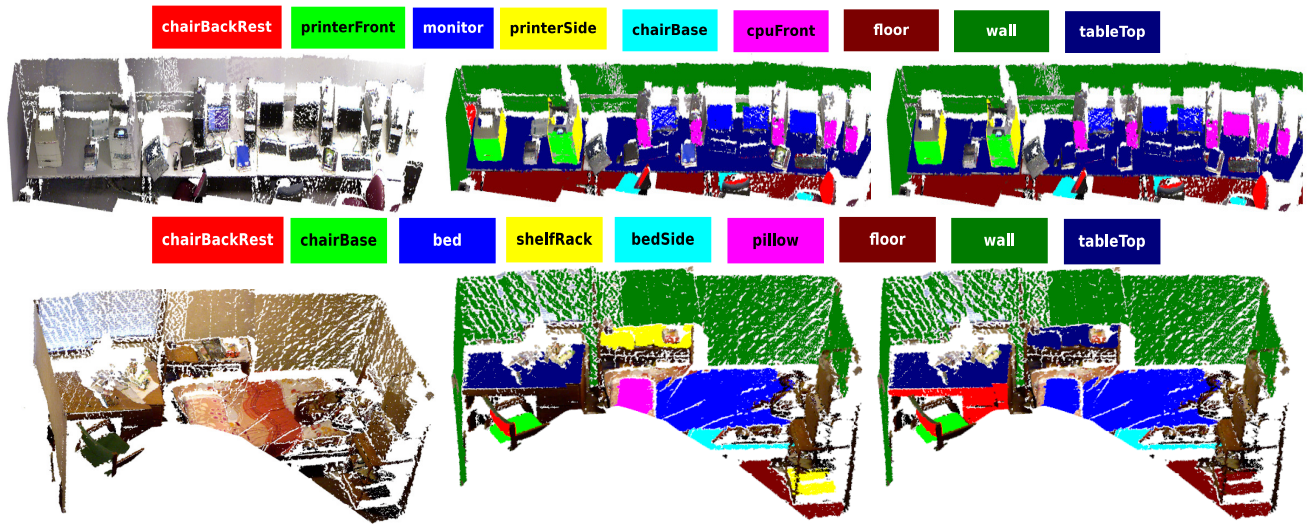

Fig. 2. Office scene (top) and Home (bottom) scene with the corresponding label coloring above the images. The left-most is the original point cloud, the middle is the ground truth labeling and the right most is the point cloud with predicted labels.

visual similarity, object co-occurrences and proximity). The model admits efficient approximate inference (Rother et al., 2007), and we show that it can be trained using a maximum-margin approach (Taskar et al., 2004; Tsochantaridis et al., 2004; Finley and Joachims, 2008) that globally minimizes an upper bound on the training loss. We model both associative and non-associative coupling of labels. With a large number of object classes, the model's parsimony becomes important. Some features are better indicators of label similarity, while other features are better indicators of non-associative relations such as geometric arrangement (e.g., *on-top-of*, *in-front-of*). We therefore model them using appropriate clique potentials rather than using general clique potentials. Our model is thus highly flexible.

We also present an algorithm that uses the contextual information present in a 3D scene to predict where an object can be found. For example, its more likely to find a keyboard on top of a table and in front of a monitor, and find a table drawer between the table top and the floor. A robot can use this information in many ways. The robot can move towards the contextually likely location to obtain a better view of the object, resulting in an increase in detection performance. This is especially helpful for small objects such as a keyboard that often appear as a segment with only a few points in the original view. It also helps when an object is not visible in the current view or occluded—the robot can move to obtain additional views in contextually likely positions of the object.

We extensively evaluate our model and algorithms over a total of 52 scenes of two types: offices and homes. These scenes were built from about 550 views from the Kinect sensor. We considered the problem of labeling each segment in the scene (from a total of about 50 segments per scene) into 27 classes (17 for offices and 17 for homes, with 7 classes in common). Our experiments show that our method, which captures several local cues and contextual properties, achieves an overall performance of 84.06% on office scenes and 73.38% on home scenes.

We also evaluated our labeling and contextual search algorithms on two mobile robots. In particular, in the task of finding 12 objects in 10 cluttered offices, our robot found the objects with 96% precision and 75% recall. Fig. 1 shows Cornell's Blue robot which was used in our experiments and a sample output labeling of an office scene. We have made the videos, data and the full source code (as a ROS and PCL package) available online at: `http://pr.cs.cornell.edu/sceneunderstanding`.

II. Related Work

There is a huge body of work in the area of scene understanding and object recognition from 2D images. Previous works have focussed on several different aspects: designing good local features such as HOG (histogram-of-gradients) (Dalal and Triggs, 2005), bag of words (Csurka et al., 2004), and eigenvectors and eignevalues of the scatter matrix (Lalonde et al., 2006), active vision for robotics (e.g., Jia et al., 2011a), and designing good global (context) features such as GIST features (Torralba, 2003). Collet et al.'s (2011) MOPED framework performs single-image and multi-image object recognition and pose estimation in complex scenes using an algorithm which iteratively estimates groups of features that are likely to belong to the same object through clustering, and then searches for object hypotheses within each of the groups.

However, the above mentioned approaches do not consider the relative arrangement of the parts of an object or of different objects with respect to each other. It has been shown that this contextual information significantly improves the performance of image-based object detectors. A number of works propose models that explicitly capture the relations between different parts of the object, e.g., part-based models (Felzenszwalb et al., 2008), and between different objects in 2D images (Murphy et al., 2003; Heitz and Koller, 2008; Li et al., 2011). However, a lot of valuable information about

the shape and geometric layout of objects is lost when a 2D image is formed from the corresponding 3D world. In some recent works, 3D layout or depths have been used for improving object detection (e.g., Saxena et al., 2005; Hoiem et al., 2006; Saxena et al., 2008, 2009; Hedau et al., 2010; Lee et al., 2010; Leibe et al., 2007; Heitz et al., 2008; Li et al., 2012; Batra and Saxena, 2012). Here a rough 3D scene geometry (e.g., main surfaces in the scene) is inferred from a single 2D image or a stereo video stream, respectively. However, the estimated geometry is not accurate enough to give significant improvements. With 3D data, we can more precisely determine the shape, size and geometric orientation of the objects, and several other properties and therefore capture much stronger context.

Visual context can also improve the performance of object detection techniques, by providing cues about an object presence. Perko and Leonardis (2010) provided a method to use contextual features and train a classifier which can predict likely locations of objects, also referred to as the 'focus of attention', for directing object detection. In methods using active recognition, the performance of object detection for robotics is improved by letting the robots take certain actions, such as moving to an optimal location for obtaining a different view of the object, based on measurement related to entropy (e.g., Laporte and Arbel, 2006; Laporte et al., 2004; Denzler and Brown, 2002; Meger et al., 2010; Ma et al., 2011). The goal here is to obtain high performance in less number of actions. Jia et al. (2011b) proposes a path planning method which selects a path where the robot obtains new views of an object, and then these views are used for training the classifiers. However, these methods only use 2D images and do not have advantage of using the rich information present in 3D data. We show that our proposed model captures context which not only helps in labeling the scene but also to infer most contextually likely locations of objects using the objects already found in the scene. This enables the robot to move to contextually likely locations and obtain better views for finding objects and improve the performance of scene labeling.

Some earlier works Gould et al. (2008); Rusu et al. (2008); Dima et al. (2004); Quigley et al. (2009) have tried to combine shape and color information from multiple sensors for tasks such as object and obstacle detection. The recent availability of synchronized videos of both color and depth obtained from RGB-D (Kinect-style) depth cameras, shifted the focus to making use of both visual as well as shape features for object detection (e.g., Lai et al., 2011a,b), 3D segmentation (e.g., Collet Romea et al., 2011), human pose estimation (Ly et al., 2012, e.g.,), and human activity detection (Sung et al., 2012). These methods demonstrate that augmenting visual features with 3D information can enhance object detection in cluttered, real-world environments. However, these works do not make use of the contextual relationships between various objects which have been shown to be useful for tasks such as object detection and scene understanding in 2D images. Our goal is to perform semantic labeling of indoor scenes by modeling and learning several contextual relationships.

There is also some recent work in labeling outdoor scenes obtained from LIDAR data into a few geometric classes (e.g., ground, building, trees, vegetation, etc.). Golovinskiy et al. (2009) capture context by designing node features. Xiong et al. (2011) do so by stacking layers of classifiers; however this models only limited correlation between the labels. An Associative Markov Network is used in (Munoz et al., 2009; Anguelov et al., 2005; Xiao and Quan, 2009) to favor similar labels for nodes in the cliques. However, many relative features between objects are not associative in nature. For example, the relationship *on-top-of* does not hold in between two ground segments, i.e., a ground segment cannot be *on-top-of* another ground segment. Therefore, using an associative Markov network is very restrictive for our problem. Contemporary work by (Shapovalov and Velizhev, 2011) did address this issue by using a cutting plane method to train non-associative Markov network. However, as we later show in our experiments, a fully non-associative Markov Model is not ideal when we have a large number of classes and features. More importantly, many of the works discussed above (e.g., Shapovalov et al., 2010; Shapovalov and Velizhev, 2011; Munoz et al., 2009; Anguelov et al., 2005; Xiong et al., 2011) were designed for outdoor scenes with LIDAR data (without RGB values). Using RGB information together with depths presents some new challenges, such as designing RGB features for 3D segments. Also, since we consider much larger number of classes compared to previous works (17 vs 3-6 for previous works), the learning task is more challenging due to the large number of parameters. We address this by proposing a parsimonious model.

The most related work to ours is (Xiong and Huber, 2010), where they label the planar patches in a point cloud of an indoor scene with four geometric labels (walls, floors, ceilings, clutter). They use a CRF to model geometrical relationships such as orthogonal, parallel, adjacent, and coplanar. The learning method for estimating the parameters in (Douillard et al., 2011; Xiong and Huber, 2010) was based on maximizing the pseudo-likelihood resulting in a sub-optimal learning algorithm. In comparison, our basic representation is 3D segments (as compared to planar patches) and we consider a much larger number of classes (beyond just the geometric classes). We also capture a much richer set of relationships between pairs of objects, and use a principled max-margin learning method to learn the parameters of our model.

III. Approach

We now outline our approach, including the model, its inference methods, and the learning algorithm. Our input is multiple Kinect RGB-D images of a scene (i.e., a room) stitched into a single 3D point cloud using RGBDSLAM.[2] Each such combined point cloud is then over-segmented based on smoothness (i.e., difference in the local surface normals) and continuity of surfaces (i.e., distance between the points). Fig. 3 shows the segmentation output for an office

[2]http://openslam.org/rgbdslam.html

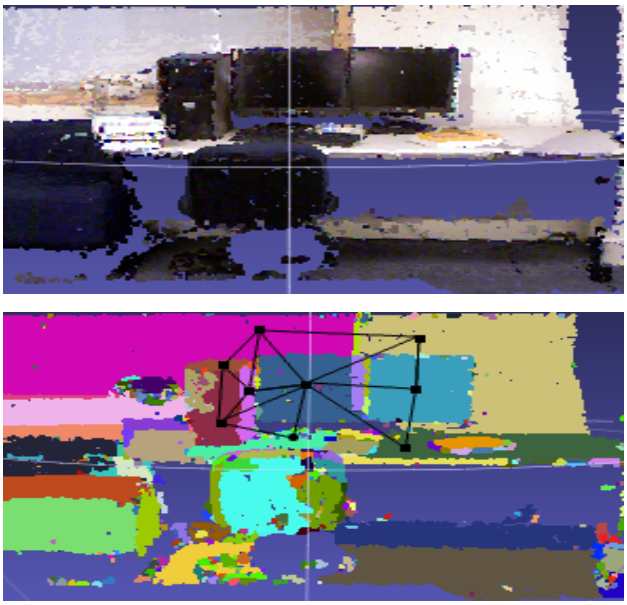

Fig. 3. (Top) Point cloud of an office scene containing 2 monitors and a CPU on a table and a chair beside it. (Bottom) The segmentation output of the point cloud with each segment represented with a different color. The black dots and lines represent the nodes and edges of the undirected graph defined over the segments (for clarity not all nodes and edges are shown).

scene. These segments are the atomic units in our model. Our goal is to label each one of them.

Before getting into the technical details of the model, we first outline the properties we aim to capture in our model below:

Visual appearance. The reasonable success of object detection in 2D images shows that visual appearance is a good indicator for labeling scenes. We therefore model the local color, texture, gradients of intensities, etc. for predicting the labels. In addition, we also model the property that if nearby segments are similar in visual appearance, they are more likely to belong to the same object.

Local shape and geometry. Objects have characteristic shapes—for example, a table is horizontal, a monitor is vertical, a keyboard is uneven, and a sofa is usually smoothly curved. Furthermore, parts of an object often form a convex shape. We compute 3D shape features to capture this.

Geometrical context. Many sets of objects occur in characteristic relative geometric configurations. For example, a monitor is always *on-top-of* a table, chairs are usually found *near* tables, a keyboard is *in-front-of* a monitor. This means that our model needs to capture non-associative relationships (e.g., that neighboring segments differ in their labels in specific patterns).

Note the examples given above are just illustrative. For any particular practical application, there will likely be other properties that could also be included. As demonstrated in the following section, our model is flexible enough to include a wide range of features.

A. Model Formulation

We model the three-dimensional structure of a scene using a model isomorphic to a Markov Random Field with log-linear node and pairwise edge potentials. Given a segmented point cloud $\mathbf{x} = (x_1, ..., x_N)$ consisting of segments x_i, we aim to predict a labeling $\mathbf{y} = (y_1, ..., y_N)$ for the segments. Each segment label y_i is itself a vector of K binary class labels $y_i = (y_i^1, ..., y_i^K)$, with each $y_i^k \in \{0, 1\}$ indicating whether a segment i is a member of class k. Note that multiple y_i^k can be 1 for each segment (e.g., a segment can be both a "chair" and a "movable object"). We use such multi-labelings in our attribute experiments where each segment can have multiple attributes, but not in segment labeling experiments where each segment can have only one label.

For a segmented point cloud $\mathbf{x}$, the prediction $\hat{\mathbf{y}}$ is computed as the argmax of a discriminant function $f_{\mathbf{w}}(\mathbf{x}, \mathbf{y})$ that is parameterized by a vector of weights $\mathbf{w}$.

$$\hat{\mathbf{y}} = \underset{\mathbf{y}}{\operatorname{argmax}}\, f_{\mathbf{w}}(\mathbf{x}, \mathbf{y}) \tag{1}$$

The discriminant function captures the dependencies between segment labels as defined by an undirected graph $(\mathcal{V}, \mathcal{E})$ of vertices $\mathcal{V} = \{1, ..., N\}$ and edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$. This undirected graph is derived from the point cloud by adding a vertex for every segment in the point cloud and adding an edge between vertices based on the spatial proximity of the corresponding segments. In detail, we connect two segments (nodes) i and j by an edge if there exists a point in segment i and a point in segment j which are less than *context_range* distance apart. This captures the closest distance between two segments (as compared to centroid distance between the segments)—we study the effect of context range more in Section V. Fig. 3 shows the graph structure induced by a few segments of an office scene.

Given $(\mathcal{V}, \mathcal{E})$, we define the following discriminant function based on individual segment features $\phi_n(i)$ and edge features $\phi_t(i, j)$ as further described below.

$$\begin{aligned} f_{\mathbf{w}}(\mathbf{x}, \mathbf{y}) = &\sum_{i \in \mathcal{V}} \sum_{k=1}^{K} y_i^k \left[w_n^k \cdot \phi_n(i) \right] \\ &+ \sum_{(i,j) \in \mathcal{E}} \sum_{T_t \in \mathcal{T}} \sum_{(l,k) \in T_t} y_i^l y_j^k \left[w_t^{lk} \cdot \phi_t(i, j) \right] \end{aligned} \tag{2}$$

The node feature map $\phi_n(i)$ describes segment i through a vector of features, and there is one weight vector for each of the K classes. Examples of such features are the ones capturing local visual appearance, shape and geometry. The edge feature maps $\phi_t(i, j)$ describe the relationship between segments i and j. Examples of edge features are the ones capturing similarity in visual appearance and geometric context.[3] There may be multiple types t of edge feature maps $\phi_t(i, j)$, and each type has a graph over the K classes with edges T_t. If T_t contains an edge between classes l and k, then this feature map and a weight vector w_t^{lk} is used to

[3]Even though it is not represented in the notation, note that both the node feature map $\phi_n(i)$ and the edge feature maps $\phi_t(i, j)$ can compute their features based on the full $\mathbf{x}$, not just x_i and x_j.

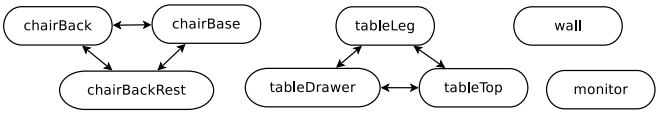

Fig. 4. Dependency graph for object-associative features. It also contains self-loops (which we do not show here for clarity).

model the dependencies between classes l and k. If the edge is not present in T_t, then $\phi_t(i,j)$ is not used.

We say that a type t of edge features is modeled by an associative edge potential if the corresponding graph only has self loops, i.e. $T_t = \{(k,k)|\forall k = 1..K\}$. And it is modeled by an non-associative edge potential if the corresponding graph is a fully connected graph, i.e. $T_t = \{(l,k)|\forall l,k = 1..K\}$.

Parsimonious model. Most of the works on 2D images only used associative edge potentials (e.g., (Szeliski et al., 2008)), where the idea is that visual similarity is usually an indicator of whether two segments (or pixels) belong to same class. As discussed before, using only associative edge potentials is very restrictive for the task of object labeling and by using non-associative edge potentials we are able to model the dependencies between different objects. Examples of such dependencies are geometric arrangements such as *on-top-of*; we usually find monitor, keyboard etc. *on-top-of* a table, rather than a table *on-top-of* a table. However, modeling every edge feature via a non-associative edge potential will need K^2 parameters per edge feature. Therefore, the number of parameters becomes very large with increase in the number of classes and the number of edge features. Even though, given sufficient data, a non-associative clique potential is general enough to learn associative relationships, this generality comes at an increased cost of training time and memory requirements when the number of classes is large.

We make the observation that not all features need to be modeled using non-associative edge potentials that relate every pair of classes. As described above, certain features such as the visual similarity features indicate when the two segments belong to the same class or parts of the same object, where as other geometric features capture relations between any pair of classes. A key reason for distinguishing between object-associative and non-associative features is parsimony of the model. As not all edge features are non-associative, we avoid learning weight vectors for relationships which do not exist.

Based on this observation we propose our parsimonious model (referred to as *svm_mrf_parsimon*) which partitions edge features into two types—object-associative features (T_{oa}), such as visual similarity, coplanarity and convexity, which usually indicate that two segments belong to the same object, and non-associative features (T_{na}), such as relative geometric arrangement (e.g., *on-top-of*, *in-front-of*), which can capture characteristic configurations under which pairs of different objects occur. We model the object-associative features using object-associative potentials which capture only the dependencies between parts of the same object. The graph for this type of features contains only edges between parts of same object and self loops as shown in Fig. 4. Formally, $T_{oa} = \{(l,k)|\exists object,\ l,k \in \text{parts(object)}\}$. The non-associative features are modeled as non-associative edge potentials. Note that $|T_{na}| >> |T_{oa}|$ since, in practice, the number of parts of an objects is much less than K. Due to this, the model we learn with both type of edge features will have much lesser number of parameters compared to a model learnt with all edge features as non-associative features. We show the performance of modeling the edge features using various types of edge potentials in Section V.

TABLE I

NODE AND EDGE FEATURES.

Node features for segment i.

Description	Count
Visual Appearance	**48**
N1. Histogram of HSV color values	14
N2. Average HSV color values	3
N3. Average of HOG features of the blocks in image spanned by the points of a segment	31
Local Shape and Geometry	**8**
N4. linearness (λ_{i0} - λ_{i1}), planarness (λ_{i1} - λ_{i2})	2
N5. Scatter: λ_{i0}	1
N6. Vertical component of the normal: $\hat{n}_{iz}$	1
N7. Vertical position of centroid: c_{iz}	1
N8. Vert. and Hor. extent of bounding box	2
N9. Dist. from the scene boundary (Fig. 5)	1

Edge features for edge (segment i, segment j).

Description	Count
Visual Appearance (associative)	**3**
E1. Difference of avg HSV color values	3
Local Shape and Geometry (associative)	**2**
E2. Coplanarity and convexity (Fig. 5)	2
Geometric context (non-associative)	**6**
E3. Horizontal distance b/w centroids.	1
E4. Vertical Displacement b/w centroids: $(c_{iz} - c_{jz})$	1
E5. Angle between normals (Dot product): $\hat{n}_i \cdot \hat{n}_j$	1
E6. Difference in angle with vertical: $\cos^{-1}(n_{iz})$ - $\cos^{-1}(n_{jz})$	1
E8. Distance between closest points: $\min_{u \in s_i, v \in s_j} d(u,v)$ (Fig. 5)	1
E9. Relative position from camera (*in-front-of*/*behind*). (Fig. 5)	1

B. Features

The various properties of objects and relations between them are captured in our model with the help of various features that we compute from the segmented point clouds. Our model uses two types of features – the node features and the edge features. Since, the robot knows the height and orientation of its Kinect sensor, we align all the point clouds so that the z-axis is vertical and the ground is at zero height for computing the features. Our features are insensitive to horizontal translation and rotation of the camera, but they place a lot of emphasis on the vertical direction because gravity influences the shape and relative positions of objects to a large extent. Note that unlike images, there is no ambiguity of scale in the input point could and all distance measurements are in meters.

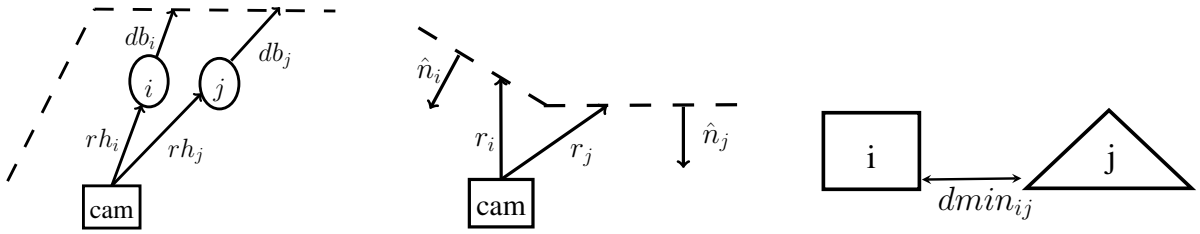

Fig. 5. Illustration of a few features. (Left) Features N9 and E9. Segment i is *in-front-of* segment j if $\mathbf{rh_i} < \mathbf{rh_j}$. (Middle) Two connected segment i and j form a convex shape if $(\mathbf{r_i} - \mathbf{r_j}).\hat{\mathbf{n}}_\mathbf{i} \geq \mathbf{0}$ and $(\mathbf{r_j} - \mathbf{r_i}).\hat{\mathbf{n}}_\mathbf{j} \geq \mathbf{0}$. (Right) Illustrating feature E8.

Table I summarizes the features used in our experiments. $\lambda_{i0}, \lambda_{i1}$ and λ_{i2} are the first three eigenvalues of the scatter matrix computed from the points of segment i. s_i is the set of points in segment i and c_i is the centroid of segment i. r_i is the ray vector to the centroid of segment i from the camera position when it was captured. rh_i is the projection of r_i on horizontal plane. $\hat{n}_i$ is the unit normal of segment i which points towards the camera ($r_i.\hat{n}_i < 0$). The node features $\phi_n(i)$ (Table I-top) consist of visual appearance features based on histogram of HSV values and the histogram of gradients (HOG) (Dalal and Triggs, 2005), as well as local shape and geometry features that capture properties such as how planar a segment is, its absolute location above ground, and its shape. The local shape features commonly used in the spectral analysis of point clouds (N4-N5) are given by $\lambda_{i2} - \lambda_{i1}$ (linearness), $\lambda_{i2} - \lambda_{i1}$ (planarness) and λ_{i0} (scatterness). N9 captures the tendency of some objects to be near the scene boundaries (walls), e.g., shelf, table, etc.

The edge features $\phi_t(i, j)$ (Table I-bottom) consist of associative features (E1-E2) based on visual appearance and local shape, as well as non-associative features (E3-E9) that capture the tendencies of two objects to occur in certain configurations. The local shape features include coplanarity and convexity. Coplanarity is defined as:

$$Coplanarity(s_i, s_j) = \begin{cases} -1 & abs(\hat{n}_i \cdot \hat{n}_j) \leq \cos\alpha \\ 1/d_{ij} & \text{otherwise} \end{cases}$$

where $d_{ij} = abs((r_i - r_j) \cdot \hat{n_i})$ is the distance between centroids in the direction of the normal. Coplanarity only makes sense if the planes are almost parallel. So we use a tolerance angle α (which was set to 30 degrees). The value of coplanarity feature is inversely proportional to the distance between the segments when they are parallel, a large value when the segments are coplanar and -1 when they are not parallel. Convexity determines if the two adjoined segments form a convex surface and is defined as:

$$Convexity(s_i, s_j) = \begin{cases} 1 & (dmin_{ij} < \tau) \wedge [[(\hat{n}_i \cdot \overrightarrow{d_{ij}} \leq 0) \\ & \wedge (\hat{n}_j \cdot \overrightarrow{d_{ji}} \leq 0)] \vee \hat{n}_i \cdot \hat{n}_j \leq \cos\alpha] \\ 0 & \text{otherwise} \end{cases}$$

where $dmin_{ij}$ is the minimum distance between segments s_i and s_j, $\overrightarrow{d_{ij}} = (r_j - r_i)$, is the displacement vector from r_i to r_j and $\overrightarrow{d_{ji}} = -\overrightarrow{d_{ij}}$. τ and α are the tolerance values for minimum distance and the angle respectively. The geometric configuration features include features that capture relationships such as *on-top-of* and *in-front-of*. These are encoded by the vertical distance between centroids (E4), which is positive if segment i is *on-top-of* segment j and negative otherwise, and the relative distance from the camera (E9), which is positive if segment i is *in-front-of* segment j and negative otherwise.

Finally, all the features are binned using a cumulative binning strategy. Each feature instance is represented by n binary values (each corresponding to a bin), where n is the number of bins. n thersholds are computed where i^{th} one is the $\frac{100i}{n}{}^{th}$ percentile of values of that features in the dataset. The value corresponding to the i^{th} bin is set to 1 if the feature value for that instance lies in the range $[min, th_i)$, where th_i is the i^{th} threshold. This gives us a new set of binary features which are then used for learning the model and during inference. Binning helps in capturing various non-linear functions of features and hence, significantly improves prediction accuracies. In our experiments we use 10 bins for every non-binary feature.

C. Computing Predictions

Our goal is to label each segment in the segmented point cloud with the most appropriate semantic label. This is achieved by finding the label assignment which maximizes the value of the discriminant function in Eq. (2). Given the learned parameters of the model and the features computed from the segmented point cloud, we need to solve the argmax in Eq. (1) for the discriminant function in Eq. (2). This is NP hard. It can be equivalently formulated as the following mixed-integer program.

$$\begin{aligned} \hat{\mathbf{y}} = \underset{\mathbf{y}}{\text{argmax}} \max_{\mathbf{z}} & \sum_{i \in \mathcal{V}} \sum_{k=1}^{K} y_i^k \left[w_n^k \cdot \phi_n(i) \right] \\ & + \sum_{(i,j) \in \mathcal{E}} \sum_{T_t \in \mathcal{T}} \sum_{(l,k) \in T_t} z_{ij}^{lk} \left[w_t^{lk} \cdot \phi_t(i,j) \right] \qquad (3) \\ \forall i, j, l, k: \; & z_{ij}^{lk} \leq y_i^l, z_{ij}^{lk} \leq y_j^k, \quad y_i^l + y_j^k \leq z_{ij}^{lk} + 1 \\ & z_{ij}^{lk}, y_i^l \in \{0, 1\} \qquad (4) \\ & \forall i : \sum_{j=1}^{K} y_i^j = 1 \qquad (5) \end{aligned}$$

Note that the products $y_i^l y_j^k$ have been replaced by auxiliary variables z_{ij}^{lk}. We can compute the exact mixed integer

solution using a general-purpose MIP solver.[4] We use this method for inference in our offline object labeling experiments (described in Section V-C) and set a time limit of 30 minutes for the MIP solver. This takes 18 minutes on average for a full-scene point cloud and 2 minutes on average for a single-view point cloud. All runtimes are for single CPU implementations using 17 classes.

However, when using our algorithm on labeling new scenes (e.g., during our robotic experiments), objects other than the 27 objects we modeled might be present. Forcing the model to predict one of the 27 objects for all segments would result in wrong predictions for every segment of an unseen class. Therefore, we relax the constraints $\forall i : \sum_{j=1}^{K} y_i^j = 1$ in Eq. (5) to $\forall i : \sum_{j=1}^{K} y_i^j \leq 1$, which allows a segment to remain unlabeled. This increases precision significantly at the cost of some drop in recall. Also, this relaxed MIP only takes 30 seconds on an average for a single-view point cloud.

We further relax the problem by removing the constraints in Eq. (5) and let the variables z_{ij}^{lk} and y_i^l take values in the interval $[0, 1]$. This results in a linear program that can be shown to always have half-integral solutions (i.e., y_i^l only take values $\{0, 0.5, 1\}$ at the solution) (Hammer et al., 1984). Furthermore, this relaxation can also be solved as a QPBO (Quadratic Pseudo-Boolean Optimization) problem using a graph-cut method[5] (Rother et al., 2007), which is orders of magnitude faster than using a general purpose LP solver. The graph-cut method takes less than ***0.05 seconds*** for labeling any full-scene or single-view point cloud. We refer to the solution of this relaxation as $\hat{\mathbf{y}}_{cut}$.

The relaxation solution $\hat{\mathbf{y}}_{cut}$ has an interesting property called *Persistence* (Boros and Hammer, 2002; Hammer et al., 1984). Persistence says that any segment for which the value of y_i^l is integral in $\hat{\mathbf{y}}_{cut}$ (i.e., does not take value 0.5) is labeled just like it would be in the optimal mixed-integer solution. Note that in all relaxations, we say that the node i is predicted as label l iff $y_i^l = 1$.

In Section V, Table III shows that the solution $\hat{\mathbf{y}}_{cut}$, despite being a relaxation, achieves similar results on all metrics for both full-scene and single-view point clouds, and is orders of magnitude faster to compute.

D. Learning Algorithm

We take a large-margin approach to learning the parameter vector $\mathbf{w}$ of Eq. (2) from labeled training examples $(\mathbf{x}_1, \mathbf{y}_1), ..., (\mathbf{x}_n, \mathbf{y}_n)$ (Taskar et al., 2003, 2004; Tsochantaridis et al., 2004). Compared to Conditional Random Field training (Lafferty et al., 2001) using maximum likelihood, this has the advantage that the partition function need not be computed, and that the training problem can be formulated as a convex program for which efficient algorithms exist.

Our method optimizes a regularized upper bound on the training error

$$R(h) = \frac{1}{n} \sum_{j=1}^{n} \Delta(\mathbf{y}_j, \hat{\mathbf{y}}_j), \tag{6}$$

where $\hat{\mathbf{y}}_j$ is the optimal solution of Eq. (1), $\Delta(\mathbf{y}, \hat{\mathbf{y}}) = \sum_{i=1}^{N} \sum_{k=1}^{K} |y_i^k - \hat{y_i^k}|$, and h is the function mapping the input $\mathbf{x}$ to an output $\mathbf{y}$. In our training problem, the function h parameterized by w is $h_w(\mathbf{x}) = \hat{\mathbf{y}}$. To simplify notation, note that Eq. (3) can be equivalently written as $\mathbf{w}^T \Psi(\mathbf{x}, \mathbf{y})$ by appropriately stacking the w_n^k and w_t^{lk} into $\mathbf{w}$ and the $y_i^k \phi_n(k)$ and $z_{ij}^{lk} \phi_t(l, k)$ into $\Psi(\mathbf{x}, \mathbf{y})$, where each z_{ij}^{lk} is consistent with Eq. (4) given $\mathbf{y}$. Training can then be formulated as the following convex quadratic program (Joachims et al., 2009):

$$\begin{aligned} \min_{w,\xi} \quad & \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\xi \\ s.t. \quad & \forall \bar{\mathbf{y}}_1, ..., \bar{\mathbf{y}}_n \in \{0, 0.5, 1\}^{N \cdot K} : \\ & \frac{1}{n}\mathbf{w}^T \sum_{i=1}^{n} [\Psi(\mathbf{x}_i, \mathbf{y}_i) - \Psi(\mathbf{x}_i, \bar{\mathbf{y}}_i)] \geq \Delta(\mathbf{y}_i, \bar{\mathbf{y}}_i) - \xi \end{aligned} \tag{7}$$

While the number of constraints in this quadratic program is exponential in n, N, and K, it can nevertheless be solved efficiently using the cutting-plane algorithm for training structural SVMs (Joachims et al., 2009). The algorithm maintains a working set of constraints, and it can be shown to provide an ϵ-accurate solution after adding at most $O(R^2C/\epsilon)$ constraints (ignoring log terms). The algorithm merely needs access to an efficient method for computing

$$\bar{\mathbf{y}}_i \quad = \quad \underset{\mathbf{y} \in \{0, 0.5, 1\}^{N \cdot K}}{\operatorname{argmax}} \left[\mathbf{w}^T \Psi(\mathbf{x}_i, \mathbf{y}) + \Delta(\mathbf{y}_i, \mathbf{y}) \right]. \tag{8}$$

Due to the structure of $\Delta(.,.)$, this problem is identical to the relaxed prediction problem in Eqs. (3)-(4) and can be solved efficiently using graph cuts.

Since our training problem is an overgenerating formulation as defined in (Finley and Joachims, 2008), the value of ξ at the solution is an upper bound on the training error in Eq. (6). Furthermore, Finley and Joachims (2008) observed empirically that the relaxed prediction $\hat{\mathbf{y}}_{cut}$ after training $\mathbf{w}$ via Eq. (7) is typically largely integral, meaning that most labels y_i^k of the relaxed solution are the same as the optimal mixed-integer solution due to persistence. We made the same observation in our experiments as well: specifically, the average percentage of variables per example that are labeled with integral values is 98.54%.

IV. Contextual Search

In robotic tasks such as of finding objects, a robot might not find the object it was looking for in its current view. The object might be far away, or occluded, or even out of view. In this section, we propose a method which determines the optimal location for the robot to move for finding the object it was looking for. Our method uses the context from the objects already identified in the robot's current view.

Formally, the goal is to find the 3D location where the desired object is most likely to be found, given a (partially) labeled point cloud. This can be done by sampling

[4] http://www.gnu.org/software/glpk

[5] http://www.cs.ucl.ac.uk/staff/V.Kolmogorov/software/QPBO-v1.3.src.tar.gz

3D locations in the current view and using the learned discriminant function to compare the chances of finding the desired object at those locations. We generate these samples by discretizing the 3D bounding box of the labeled point cloud and considering the centre of each grid as a sample. We considered 1000 equally spaced samples in our experiments.

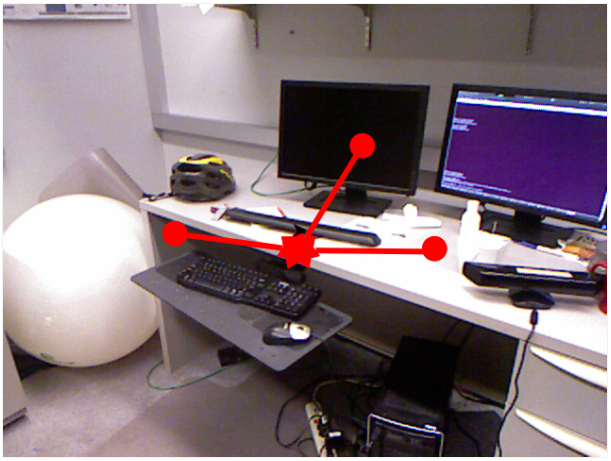

Fig. 6. Figure illustrating our method for contextual search. The location of the hallucinated segment is shown by a red star and its neighbors are denoted using red edges. In this particular scene the hallucinated keyboard segment is connected to monitor, table-top and table-leg segments. The keyboard and tray segments were left unlabeled by the algorithm, so they are not connected.

A. Augmented Graph

To evaluate the score of a sample (l_x, l_y, l_z) we first generate a graph on which the discriminant function can be applied. For this, we first take the graph of labeled segments (labeled using our algorithm) and augment it with a node h corresponding to a hallucinated segment x_h at location (l_x, l_y, l_z). We then add edges between this node, h, and the nearby nodes (labeled segments) that are within $context\_range$ distance from the sampled location. We denote these neighboring segments of the hallucinated segment by $Nbr(x_h)$. Suppose we are looking for an (missing) object class k, we label the newly added node as class k, i.e., the k^{th} element of y_h is 1 and rest are 0. Fig. 6 illustrates one such augmentation step. We can now apply our discriminant function to this augmented graph. Note that each of our sampled locations will give us a unique corresponding augmented graph.

B. Discriminant Function and Inference

We use the discriminant function from Eq. (2) to compute the score of each augmented graph. The optimal sample (location) for finding an instance of class k, $OL(k)$, is the one with highest value of the discriminant function applied to its corresponding augmented graph. Note that only the location-dependent terms in the discriminant function for the newly added node and edges can effect its value, since the rest of the terms are same for every sample. Thus, we can compute $OL(k)$ very efficiently.

We denote the sum of all terms of the discriminant function which do not depend on the location (l_x, l_y, l_z) by a constant ρ. We denote to the label of a node j in the original labeled graph (before augmentation) as $label(j)$. Let $\phi'_n(h)$ denote the features of the hallucinated node h, which only depend on its location (l_x, l_y, l_z). Similarly, let $\phi'_t(h, j)$ denote the features of the edge (h, j) which only depend on location of the hallucinated node. Let w'_n and w'_t denote the projection of node and edge weight vectors respectively that contain only the components corresponding to the location-dependent features. Now we can formally define the optimal location as:

$$OL(k) = \underset{x_h \in samples}{\operatorname{argmax}} \; f_{\mathbf{w}}(\mathbf{x} \cup x_h, \mathbf{y} \cup y_h) \tag{9}$$

where,

$$f_{\mathbf{w}}(\mathbf{x} \cup x_h, \mathbf{y} \cup y_h) = w'^{k}_{n} \cdot \phi'_n(h) + \sum_{(j) \in Nbr(x_h)} \sum_{T_t \in \mathcal{T}} \Big[w'^{k\ label(j)}_{t} \cdot \phi'_t(h, j) + w'^{label(j)\ k}_{t} \cdot \phi'_t(j, h) \Big] + \rho$$

Once the optimal location $OL(k)$ is found by solving Eq. (9), the robot moves close to this location to find objects as described in our contextually-guided object detection experiments presented in Section V-E.

V. Experiments

In this section, we first describe our dataset and present results of the scene labeling task on this dataset. We present a detailed analysis of various factors such as the effect of the 3D features, the effect of adding context, the effect of the presence of unlabeled segments, etc.

We then present the results on robotic experiments on attribute learning and contextually-guided object detection. The object detection experiments were performed using two robots: Cornell's Blue and POLAR robot, shown in Fig. 7 and Fig. 8 respectively. These robots consist of a mobile base (Erratic and Segway Omni 50 respectively) and were mounted with Kinect sensors as shown in respective figures. (For more details on the hardware specification of the POLAR robot, please see Jiang et al., 2012.) We used ROS libraries[6] to program these robots.

A. Data

We consider labeling object segments in full 3D scene (as compared to 2.5D data from a single view). For this purpose, we collected data of 24 office and 28 home scenes from a total of about 550 views. Each scene was reconstructed from about 8-9 RGB-D views from a Microsoft Kinect sensor and contains about one million colored points. We first over-segment the 3D scene (as described earlier) to obtain the atomic units of our representation. For training, we manually labeled the segments, and we selected the labels which were present in a minimum of 5 scenes in the dataset. Specifically,

[6] http://www.ros.org

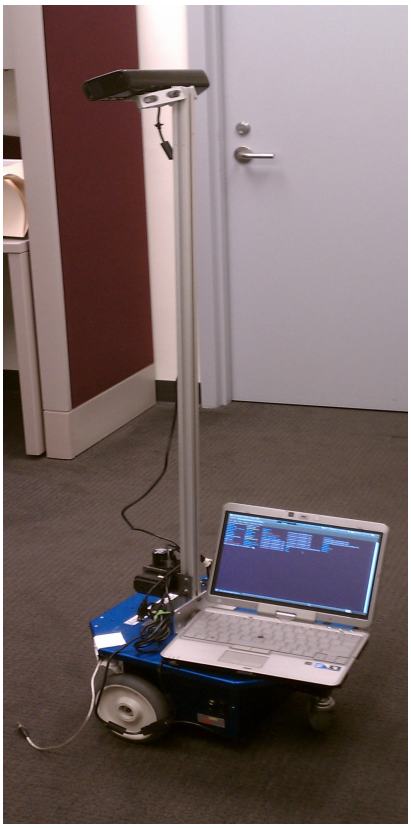

Fig. 7. Cornell's Blue Robot. It consists of a mobile base mounted with a Kinect sensor.

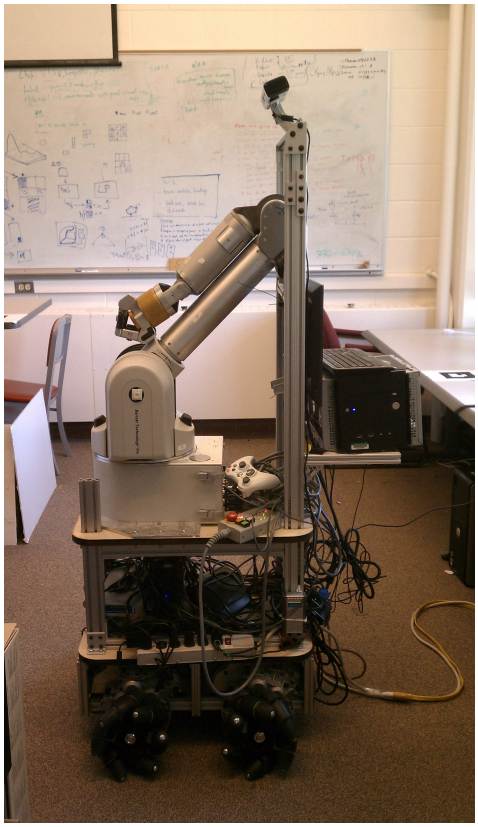

Fig. 8. Cornell's POLAR Robot. It consists of a omni-directional mobile base with a robotic arm and Kinect sensor mounted on the top.

the office labels are: {*wall, floor, tableTop, tableDrawer, tableLeg, chairBackRest, chairBase, chairBack, monitor, printerFront, printerSide keyboard, cpuTop, cpuFront, cpuSide, book, paper*}, and the home labels are: {*wall, floor, tableTop, tableDrawer, tableLeg, chairBackRest, chairBase, sofaBase, sofaArm, sofaBackRest, bed, bedSide, quilt, pillow, shelfRack, laptop, book*}. This gave us a total of 1108 labeled segments in the office scenes and 1387 segments in the home scenes. Often one object may be divided into multiple segments because of over-segmentation. We have made this data available at: `http://pr.cs.cornell.edu/sceneunderstanding/data/data.php`.

B. Segmentation

For segmenting the point cloud, we use a region growing algorithm similar to the Euclidean clustering in the Point-Cloud Library (PCL).[7] It randomly picks a seed point and grows it into a segment. New points are added to an existing segment if their distance to the closest point in the segment is less than a threshold and the local normals calculated at these points are at an angle less than a threshold. We also made the distance threshold proportional to the distance from camera because points far from the camera have more noisy depth estimates. In detail, we used a distance threshold of $0.1d$, where d is the distance of the candidate point from camera, and an angle threshold of $30°$. We observed that these thresholds slightly over-segmented almost all scenes into object-parts of desired granularity. Such a simple approach would not perfectly segment all objects, but we find that our learning method is quite robust to such noisy segmentation.

C. Object Labeling Results

In this subsection, we report the results of offline labeling experiments. Table II shows the results, performed using 4-fold cross-validation and averaging performance across the folds for the models trained separately on home and office datasets. We use both the macro and micro averaging to aggregate precision and recall over various classes. Let $pred_i(k)$ denote that the i^{th} segment was predicted to be of class k, and $gt_i(k)$ denote that the ground truth label of the i^{th} segment was class k. We now formally define the following metrics used in literature.

$$
\begin{aligned}
precision(k) &= \frac{|\{i : (pred_i(k) \wedge gt_i(k))\}|}{|\{i : pred_i(k)\}|} \\
recall(k) &= \frac{|\{i : (pred_i(k) \wedge gt_i(k))\}|}{|\{i : gt_i(k)\}|} \\
macro\_precision &= \frac{\sum_{k=1}^{K} precision(k)}{K} \\
macro\_recall &= \frac{\sum_{k=1}^{K} recall(k)}{K} \\
micro\_precision &= \frac{\sum_{k=1}^{K} |\{i : (pred_i(k) \wedge gt_i(k))\}|}{\sum_{k=1}^{K} |\{i : pred_i(k)\}|} \\
micro\_recall &= \frac{\sum_{k=1}^{K} |\{i : (pred_i(k) \wedge gt_i(k))\}|}{\sum_{k=1}^{K} |\{i : gt_i(k)\}|}
\end{aligned}
$$

[7] `http://www.pointclouds.org/documentation/tutorials/cluster_extraction.php`

TABLE II

Learning experiment statistics. THE TABLE SHOWS AVERAGE MICRO PRECISION/RECALL, AND AVERAGE MACRO PRECISION AND RECALL FOR HOME AND OFFICE SCENES.

		Office Scenes			Home Scenes		
		micro	macro		micro	macro	
features	algorithm	P/R	Precision	Recall	P/R	Precision	Recall
None	*max_class*	26.33	26.33	5.88	29.38	29.38	5.88
Image Only	*svm_node_only*	46.67	35.73	31.67	38.00	15.03	14.50
Shape Only	*svm_node_only*	75.36	64.56	60.88	56.25	35.90	36.52
Image+Shape	*svm_node_only*	77.97	69.44	66.23	56.50	37.18	34.73
Image+Shape & context	*single_frames*	84.32	77.84	68.12	69.13	47.84	43.62
Image+Shape & context	*svm_mrf_assoc*	75.94	63.89	61.79	62.50	44.65	38.34
Image+Shape & context	*svm_mrf_nonassoc*	81.45	76.79	70.07	72.38	57.82	53.62
Image+Shape & context	*svm_mrf_parsimon*	84.06	80.52	72.64	73.38	56.81	54.80

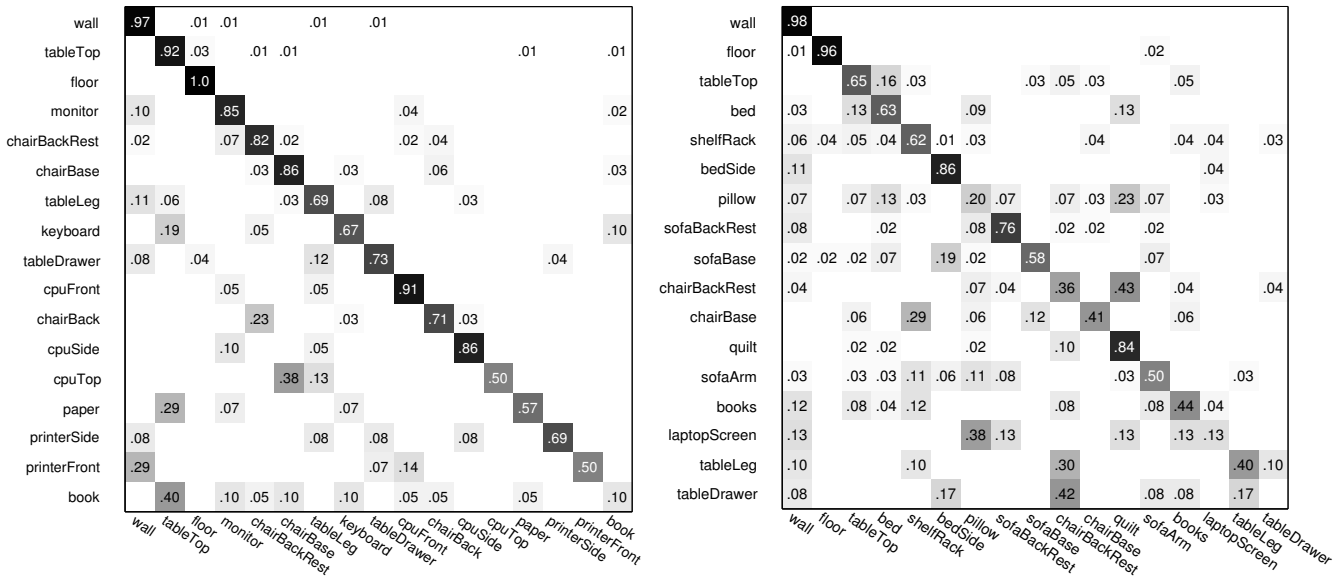

Fig. 9. Confusion Matrix on office dataset (left) and home dataset (right) with *svm_mrf_parsimon* trained on Shape and Image features.

where, for any set S, $|S|$ denotes its size.

In these experiments, prediction is done using an MIP solver with the constraint that a segment can have exactly one label ($\forall i : \sum_{j=1}^{K} y_i^j = 1$). So, micro precision and recall are same as the percentage of correctly classified segments. The optimal C value is determined separately for each of the algorithms by cross-validation.

Fig. 2 shows the original point cloud, ground-truth and predicted labels for one office (top) and one home scene (bottom). Fig. 9 show the confusion matrices for office and home scenes on the left and right respectively. On majority of the classes our model predicts the correct label as can be seen from the strong diagonal in the confusion matrices. Some of the mistakes are reasonable, such as a pillow getting confused with the quilt in homes. They often have similar location and texture. In offices, books placed on table-tops sometimes get confused with the table-top.

One of our goals is to study the effect of various factors, and therefore we compared our algorithm with various settings. We discuss them in the following.

Do Image and Point Cloud Features Capture Complimentary Information? The RGB-D data contains both image and depth information, and enables us to compute a wide variety of features. In this experiment, we compare the two kinds of features: Image (RGB) and Shape (Point Cloud) features. To show the effect of the features independent of the effect of context, we only use the node potentials from our model, referred to as *svm_node_only* in Table II. The *svm_node_only* model is equivalent to the multi-class SVM formulation (Joachims et al., 2009). Table II shows that Shape features are more effective compared to the Image, and the combination works better on both precision and recall. This indicates that the two types of features offer complementary information and their combination is better for our classification task.

How Important is Context? Using our *svm_mrf_parsimon* model as described in Section III-A, we show significant improvements in the performance over using *svm_node_only* model on both datasets. In office scenes, the micro precision increased by 6.09% over the best *svm_node_only* model that does not use any context. In home scenes the increase is much higher, 16.88%.

The type of contextual relations we capture depend on the type of edge potentials we model. To study this, we compared our method with models using only associative or only non-associative edge potentials referred to as *svm_mrf_assoc* and *svm_mrf_nonassoc*. We observed that modeling all edge features using associative potentials is poor compared to our full model. In fact, using only associative potentials showed a drop in performance compared to *svm_node_only* model on the office dataset. This indicates it is important to capture the relations between regions having different labels. Our *svm_mrf_nonassoc* model does so, by modeling all edge features using non-associative potentials, which can favor or disfavor labels of different classes for nearby segments. It gives higher precision and recall compared to *svm_node_only* and *svm_mrf_assoc*. This shows that modeling using non-associative potentials is a better choice for our labeling problem.

However, not all the edge features are non-associative in nature, modeling them using only non-associative potentials could be an overkill (each non-associative feature adds K^2 more parameters to be learnt). Therefore using our *svm_mrf_parsimon* model to model these relations achieves higher performance in both datasets.

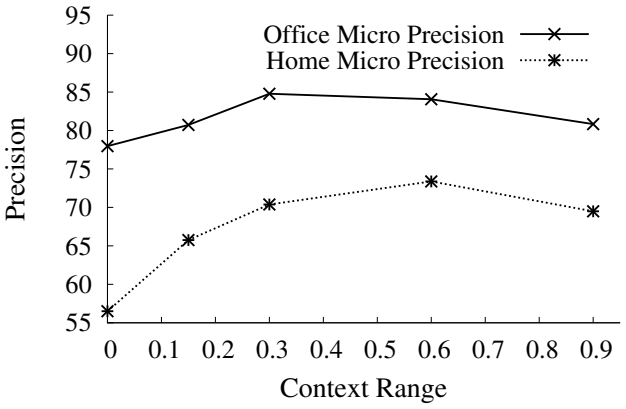

Fig. 10. Effect of context range on precision (=recall here).

How Large should the Context Range be? Context relationships of different objects can be meaningful for different spatial distances. This range may vary depending on the environment as well. For example, in an office, keyboard and monitor go together, but they may have little relation with a sofa that is slightly farther away. In a house, sofa and table may go together even if they are farther away.

In order to study this, we compared our *svm_mrf_parsimon* with varying $context\_range$ for determining the neighborhood (see Figure 10 for average micro precision vs range plot). Note that the $context\_range$ is determined from the boundary of one segment to the boundary of the other, and hence it is somewhat independent of the size of the object. We note that increasing the $context\_range$ increases the performance to some level, and then it drops slightly. We attribute this to the fact that with increasing the $context\_range$, irrelevant objects may get an edge in the graph, and with limited training data, spurious relationships may be learned. We observe that the optimal $context\_range$ for office scenes is around 0.3 meters and 0.6 meters for home scenes.

How does a Full 3D Model Compare to a 2.5D Model? In Table II, we compare the performance of our full model with a model that was trained and tested on single-view point clouds of the same scene. During the comparison, the training folds were consistent with other experiments, however the segmentation of this point cloud was different (because the input point cloud is from a single view). This makes the micro precision values meaningless because the distribution of labels is not same for the two cases. In particular, many large object in a scene (e.g., wall, ground) get split up into multiple segments in single-view point clouds. We observed that the macro precision and recall are higher when multiple views are combined to form the scene. We attribute the improvement in macro precision and recall to the fact that larger scenes have more context, and models are more complete because of multiple views.

What is the effect of the inference method? The results for *svm_mrf* algorithms in Table II were generated using the MIP solver. The QPBO algorithm however, gives a higher precision and lower recall on both datasets. For example, on office data, the graphcut inference for our *svm_mrf_parsimon* gave a micro precision of 90.25 and micro recall of 61.74. Here, the micro precision and recall are not same as some of the segments might not get any label.

What is the effect of having different granularity when defining the object classes? In our experiments, we have considered class labels at object-part level, e.g., classes *chairBase*, *chairBack* and *chairBackRest* which are parts of a chair. We think that such finer knowledge of objects is important for many robotic applications. For example, if a robot is asked to arrange chairs in a room, knowing the chair parts can help determine the chair's orientation. Also, labeling parts of objects gives our learning algorithm an opportunity to exploit relationships between different parts of an object. In order to analyze the performance gain obtained by considering object-part level labeling, we compare our method with one trained on object level classes. With 10 object level classes in office scenes : {*wall, table, chair, floor, cpu, book, paper, keyboard, printer and monitor*}, we observe a drop in performance, obtaining a micro precision/recall of 83.62%, macro precision of 76.89% and recall of 69.81%.

D. Attribute Labeling Results

In some robotic tasks, such as robotic grasping, it is not necessary to know the exact object category, but just knowing a few attributes of an object may be useful. For example, if a robot has to clean a floor, it would help if it knows which objects it can move and which it cannot. If it has to place an object, it should place them on horizontal surfaces, preferably where humans do not sit. With this motivation we have designed 8 attributes, each for the home and office

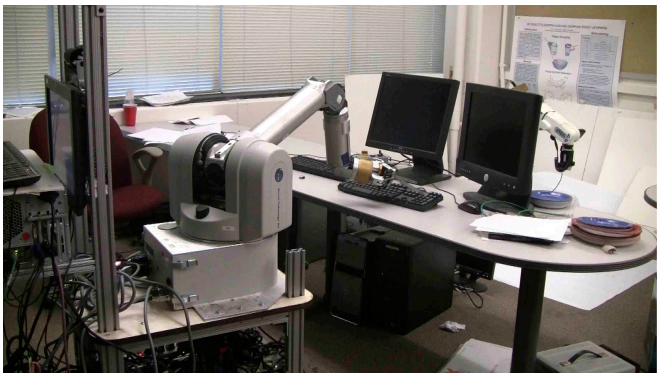

Fig. 11. Cornell's POLAR (PersOnaL Assistant Robot) using our classifier for detecting a keyboard in a cluttered room.

scenes, giving a total of 10 unique attributes. They are: {*wall, floor, flat-horizontal-surfaces, furniture, fabric, heavy, seating-areas, small-objects, table-top-objects, electronics*}. Note that each segment in the point cloud can have multiple attributes and therefore we can learn these attributes using our model which naturally allows multiple labels per segment. We compute the precision and recall over the attributes by counting how many attributes were correctly inferred. In home scenes we obtained a precision of 83.12% and 70.03% recall, and in the office scenes we obtain 87.92% precision and 71.93% recall.

E. Robotic Experiments

The ability to label scenes is very useful for robotics applications, such as of finding/retrieving an object on request. As described in Section III-C, in a detection scenario there can be some segments not belonging to the object classes we consider. Table III shows the results of running our inference algorithms for detection scenario on the offline office dataset when considering all segments, including those belonging to classes other than the 17 mentioned earlier. The solution of the relaxed MIP (described in Section III-C) gives us high precision (89.87% for micro, and 82.21% for macro), but low recall (55.36% for micro, and 35.25% for macro). The $\hat{\mathbf{y}}_{cut}$ solution, computed using the graph-cut method, also achieves comparable accuracy (see line 3 of Table III) and is very fast to compute (takes less than 0.05 second per scene). Therefore, if the robot finds an object it is likely correct, but the robot may not find all the objects easily. This is where our contextual search algorithm (described in Section IV) becomes useful.

TABLE III

PRECISION AND RECALL FOR DETECTION EXPERIMENTS IN OFFICE SCENES (OFFLINE, SINGLE-VIEW).

Algorithm	Micro-precision	Micro-recall	Macro-precision	Macro-recall
max_class	22.64	22.64	22.64	5.88
svm_mrf_parsimon	89.87	55.36	82.21	35.25
svm_mrf_parsimon w/ QPBO, $\hat{y}_{cut}$	87.41	56.82	82.95	38.14

TABLE IV

CLASS-WISE PRECISION RECALL FOR ROBOTIC EXPERIMENTS USING CONTEXTUAL SEARCH.

class	# instances	precision	recall
Wall	10	100	100
Table Top	10	100	100
Table Leg	10	71	50
Table Drawer	7	100	71
Chair Backrest	10	100	100
Chair Base	10	100	100
Chair Back	8	100	88
Monitor	9	100	100
Keyboard	9	100	78
CPU	8	50	11
Printer	1	100	100
Paper	10	100	22
Overall Micro	102	96	75
Overall Macro		93	77

In order to evaluate our contextual search algorithm, we test our approach on our Blue robot for the task of finding 12 object classes located in 10 office scenes. The robot starts from a pre-determined location in an office and searches for a given list of objects in the room. The goal of the robot is to find at least one instance for each of the object classes it is asked to search for. Since, the RGB-D sensor has a narrow field-of-view (57 degrees horizontally), the robot first scans the room by turning a fixed angle each time. It labels the point cloud it obtains in each view and saves the labeled point clouds.

Next, for all the object classes it did not find, it computes the contextually likely locations for the objects using the algorithm described in Section IV. Using the the inferred locations, the robot moves in that direction in order to get better view of the objects. Fig. 12 shows the experiment run in one office scene. The first column shows the Blue robot and the scene in which it is asked to find the objects and the rest of the columns show the sequence of colored images corresponding to the labeled point clouds. The first two point clouds were obtained when the robot was scanning the room from a distance, and the last two are obtained after inferring the contextually likely locations of the objects not found and moving closer to these locations. Table IV shows the precision and recall of finding the 12 object classes in our robotic experiments.

To evaluate our contextual search algorithm, we present both qualitative and quantitative results. Fig. 13 qualitatively shows the predictions for finding monitor, keyboard, tableDrawer and paper in a frame in which they were not found. These heatmaps are generated in 3D, but for visualization purposes they are aligned to the original RGB image in Fig. 13. As can be seen form Fig. 13, a monitor is predicted to be most likely found above the table and a tableDrawer is likely to be found under the table. However, it does not do a great job for paper, and this may be because we had very few examples for this class in our training set.

Fig. 14 shows that our algorithm can also be used on a robot to find objects even when they are occluded. Clearly, it predicts that a keyboard is likely to be found in front of the monitor and at about the same height as the tableTop.

Fig. 12. (Left): The robot in an office scene. (Columns 2-5): Sequence of colored images corresponding to the labeled point clouds generated by the robot during the object detection experiment.

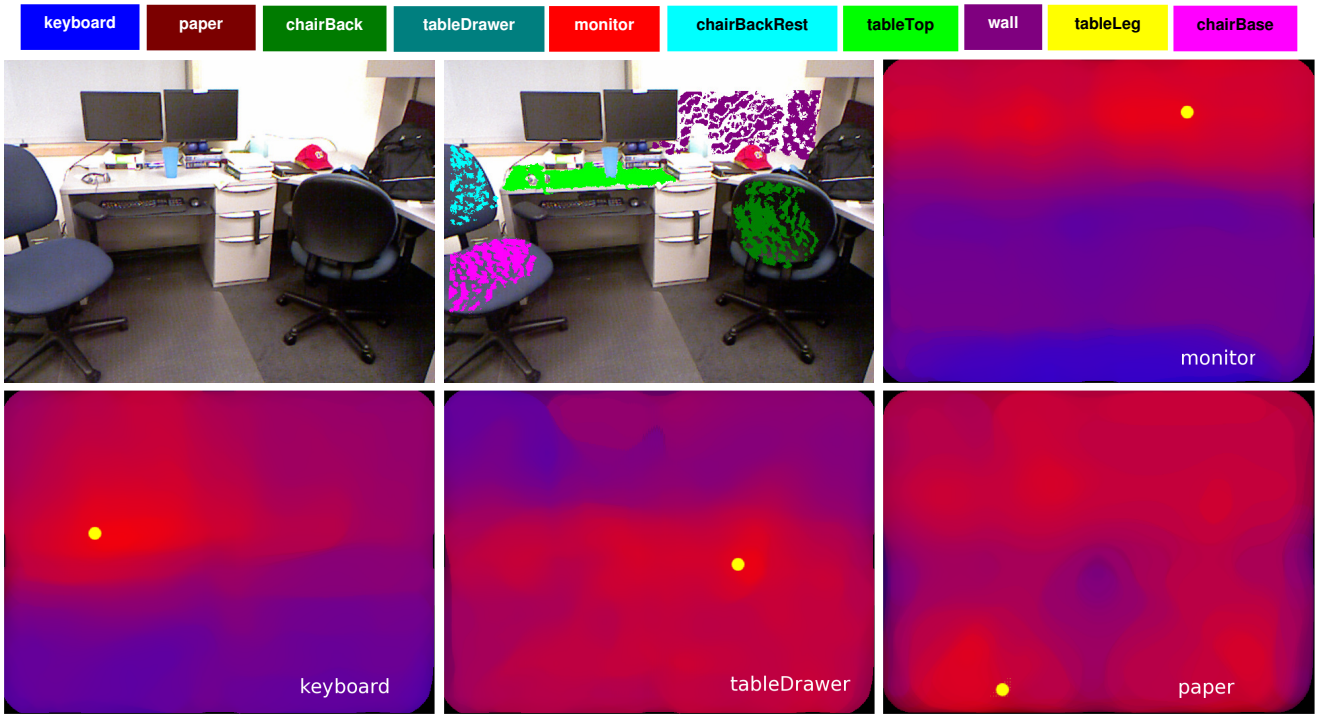

Fig. 13. (Top left) Original image. (Top mid) Inferred labels color-coded using the legend shown above. (Top right, bottom left, bottom mid and bottom right): Contextually likely positions for finding monitor, keyboard, tableDrawer and paper respectively. In these heatmaps, red indicates that the target object is more likely to be found there. The circular yellow dot indicates the most likely location.

To quantitatively evaluate the predictions, we collected 10 frames where a keyboard was not detected, but other objects such as table and monitor were detected. We then applied our contextual search algorithm to find the optimal 3D locations for finding a keyboard. For each of these scenes, we computed the minimum distance of the actual keyboard-points to the inferred optimal location. As a baseline, if the predictor always predicted midpoints of the scene as the probable location of the keyboard, the max, mean and median values over 10 scenes were 113.5cm, 32.6cm and 27.2cm respectively. Using our method, we get 36.3cm, 17.5cm and 15.9cm respectively. We found that this helps the robot in finding the objects with only a few moves.

We have the code available as a ROS and PCL package. Code, datasets as well as videos showing our robots finding objects using our algorithm are available at: `http://pr.cs.cornell.edu/sceneunderstanding/`.

VI. Conclusion

In conclusion, we have proposed and evaluated the first model and learning algorithm for semantic labeling that exploits rich relational information in full-scene 3D point clouds. Our method captures various features and contextual relations, including the local visual appearance and shape cues, object co-occurence relationships and geometric relationships. We showed how visual and shape features can be modeled parsimoniously when the number of classes is large. We also presented an algorithm to infer contextually likely locations for the desired objects given the current labelings in the scene. We tested our method on a large offline dataset, as well as on the task of mobile robots finding objects in cluttered scenes.

VII. Acknowledgements

We thank Gaurab Basu, Yun Jiang, Jason Yosinski and Dave Kotfis for help with the robotic experiments. This research was funded in part by Microsoft Faculty Fellowship and Alfred P. Sloan Research Fellowship to one of us (Saxena), and by NSF Award IIS-0713483.

References

Anguelov, D., Taskar, B., Chatalbashev, V., Koller, D., Gupta, D., Heitz, G., and Ng, A. (2005). Discriminative learning

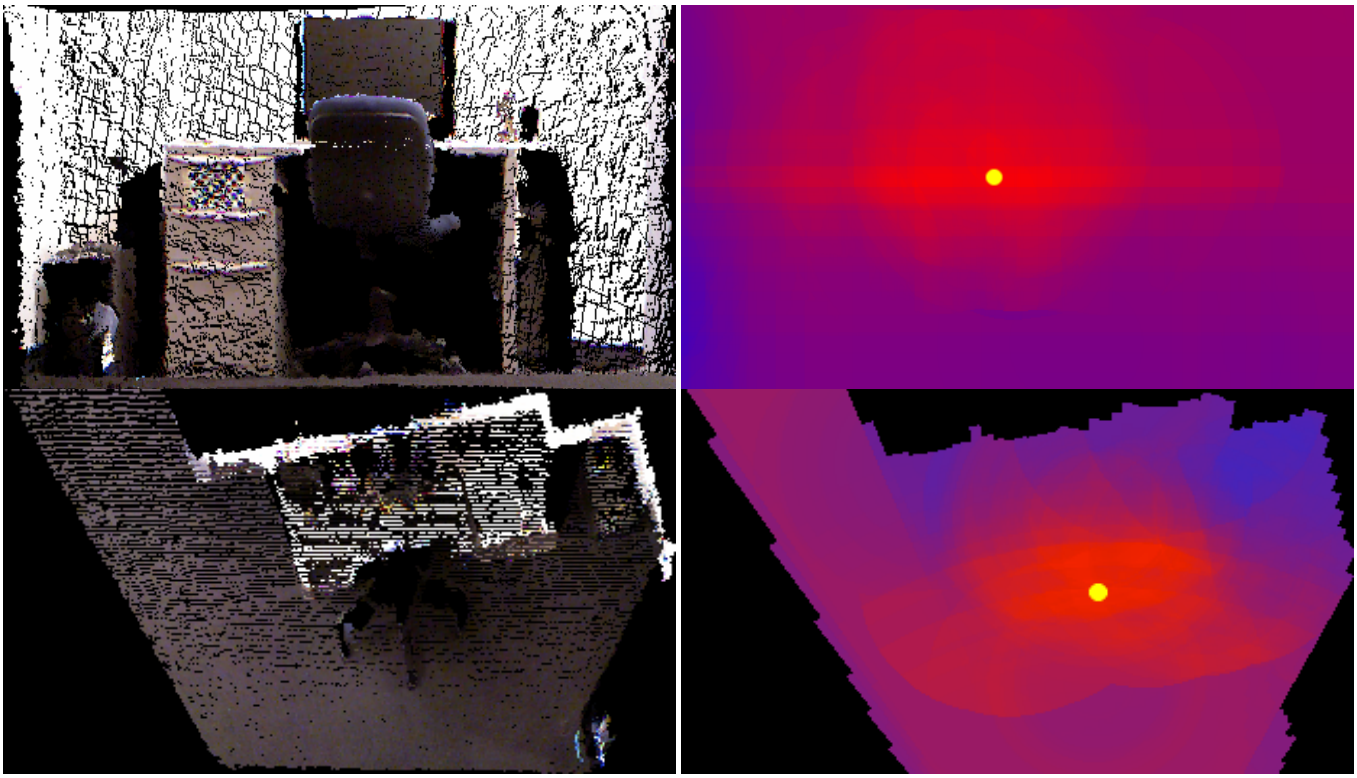

Fig. 14. (Left column): Front and top views(orthographic projections) of an office scene in which a keyboard is occluded. (Right column): Corresponding heatmaps where red indicates that the keyboard is more likely to be found there. The yellow dot indicates the most likely location.

of markov random fields for segmentation of 3d scan data. In *CVPR*.

Batra, D. and Saxena, A. (2012). Learning the right model: Efficient max-margin learning in laplacian crfs. In *CVPR*.

Boros, E. and Hammer, P. (2002). Pseudo-boolean optimization. *Discrete Applied Mathematics*, 123(1-3):155–225.

Collet, A., Martinez, M., and Srinivasa, S. S. (2011). The moped framework: Object recognition and pose estimation for manipulation. *IJRR*.

Collet Romea, A., Srinivasa, S., and Hebert, M. (2011). Structure discovery in multi-modal data : a region-based approach. In *ICRA*.

Csurka, G., Dance, C., Fan, L., Willamowski, J., and Bray, C. (2004). Visual categorization with bags of keypoints. In *Workshop on statistical learning in computer vision, ECCV*.

Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. In *CVPR*.

Denzler, J. and Brown, C. (2002). Information theoretic sensor data selection for active object recognition and state estimation. *IEEE PAMI*, 24(2):145 –157.

Dima, C., Vandapel, N., and Hebert, M. (2004). Classifier fusion for outdoor obstacle detection. In *ICRA*.

Divvala, S., Hoiem, D., Hays, J., Efros, A., and Hebert, M. (2009). An empirical study of context in object detection. In *CVPR*.

Douillard, B., Fox, D., Ramos, F., and Durrant-Whyte, H. (2011). Classification and semantic mapping of urban environments. *IJRR*, 30(1):5–32.

Felzenszwalb, P., McAllester, D., and Ramanan, D. (2008). A discriminatively trained, multiscale, deformable part model. In *CVPR*.

Finley, T. and Joachims, T. (2008). Training structural svms when exact inference is intractable. In *ICML*.

Golovinskiy, A., Kim, V. G., and Funkhouser, T. (2009). Shape-based recognition of 3d point clouds in urban environments. In *ICCV*.

Gould, S., Baumstarck, P., Quigley, M., Ng, A. Y., and Koller, D. (2008). Integrating visual and range data for robotic object detection. In *ECCV workshop Multi-camera Multi-modal (M2SFA2)*.

Hammer, P., Hansen, P., and Simeone, B. (1984). Roof duality, complementation and persistency in quadratic 0–1 optimization. *Mathematical Programming*, 28(2):121–155.

Hedau, V., Hoiem, D., and Forsyth, D. (2010). Thinking inside the box: Using appearance models and context based on room geometry. In *ECCV*.

Heitz, G., Gould, S., Saxena, A., and Koller, D. (2008). Cascaded classification models: Combining models for holistic scene understanding. In *NIPS*.

Heitz, G. and Koller, D. (2008). Learning spatial context: Using stuff to find things. In *ECCV*.

Hoiem, D., Efros, A. A., and Hebert, M. (2006). Putting objects in perspective. In *CVPR*.

Jia, Z., Saxena, A., and Chen, T. (2011a). Robotic object

detection: Learning to improve the classifiers using sparse graphs for path planning. In *IJCAI*.

Jia, Z., Saxena, A., and Chen, T. (2011b). Robotic object detection: Learning to improve the classifiers using sparse graphs for path planning. In *IJCAI*, pages 2072–2078.

Jiang, Y., Lim, M., Zheng, C., and Saxena, A. (2012). Learning to place new objects in a scene. *IJRR*.

Joachims, T., Finley, T., and Yu, C. (2009). Cutting-plane training of structural SVMs. *Machine Learning*, 77(1):27–59.

Koppula, H., Anand, A., Joachims, T., and Saxena, A. (2011a). Semantic labeling of 3d point clouds for indoor scenes. In *NIPS*.

Koppula, H. S., Anand, A., Joachims, T., and Saxena, A. (2011b). Labeling 3d scenes for personal assistant robots. In *RSS workshop RGB-D: Advanced Reasoning with Depth Cameras*.

Lafferty, J. D., McCallum, A., and Pereira, F. C. N. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*.

Lai, K., Bo, L., Ren, X., and Fox, D. (2011a). A large-scale hierarchical multi-view rgb-d object dataset. In *ICRA*.

Lai, K., Bo, L., Ren, X., and Fox, D. (2011b). Sparse distance learning for object recognition combining rgb and depth information. In *ICRA*.

Lalonde, J.-F., Vandapel, N., Huber, D. F., and Hebert, M. (2006). Natural terrain classification using three-dimensional ladar data for ground robot mobility. *J. Field Robotics*, pages 839–861.

Laporte, C. and Arbel, T. (2006). Efficient discriminant viewpoint selection for active bayesian recognition. *IJCV*, 68:267–287.

Laporte, C., Brooks, R., and Arbel, T. (2004). A fast discriminant approach to active object recognition and pose estimation. In *ICPR*.

Lee, D. C., Gupta, A., Hebert, M., and Kanade, T. (2010). Estimating spatial layout of rooms using volumetric reasoning about objects and surfaces. In *NIPS*.

Leibe, B., Cornelis, N., Cornelis, K., and Gool, L. V. (2007). Dynamic 3d scene analysis from a moving vehicle. In *CVPR*.

Li, C., Kowdle, A., Saxena, A., and Chen, T. (2012). Towards holistic scene understanding: Feedback enabled cascaded classification models. *IEEE PAMI*, 34(7):1394–1408.

Li, C., Saxena, A., and Chen, T. (2011). θ-mrf: Capturing spatial and semantic structure in the parameters for scene understanding. In *NIPS*.

Ly, D., Saxena, A., and Lipson, H. (2012). Co-evolutionary predictors for kinematic pose inference from rgbd images. In *GECCO*.

Ma, J., Chung, T. H., and Burdick, J. (2011). A probabilistic framework for object search with 6-dof pose estimation. *IJRR*, 30(10):1209–1228.

Meger, D., Gupta, A., and Little, J. (2010). Viewpoint detection models for sequential embodied object category recognition. In *ICRA*.

Munoz, D., Vandapel, N., and Hebert, M. (2009). Onboard contextual classification of 3-d point clouds with learned high-order markov random fields. In *ICRA*.

Murphy, K. P., Torralba, A., and Freeman, W. T. (2003). Using the forest to see the trees: a graphical model relating features, objects and scenes. *NIPS*.

Perko, R. and Leonardis, A. (2010). A framework for visual-context-aware object detection in still images. *Computer Vision and Image Understanding*, 114:700–711.

Quigley, M., Batra, S., Gould, S., Klingbeil, E., Le, Q. V., Wellman, A., and Ng, A. Y. (2009). High-accuracy 3d sensing for mobile manipulation: Improving object detection and door opening. In *ICRA*.

Rother, C., Kolmogorov, V., Lempitsky, V., and Szummer, M. (2007). Optimizing binary mrfs via extended roof duality. In *CVPR*.

Rusu, R. B., Marton, Z. C., Blodow, N., Dolha, M., and Beetz, M. (2008). Towards 3d point cloud based object maps for household environments. *Robot. Auton. Syst.*, 56:927–941.

Saxena, A., Chung, S., and Ng, A. (2005). Learning depth from single monocular images. In *NIPS*.

Saxena, A., Chung, S., and Ng, A. (2008). 3-d depth reconstruction from a single still image. *IJCV*, 76(1):53–69.

Saxena, A., Sun, M., and Ng, A. Y. (2009). Make3d: Learning 3d scene structure from a single still image. *IEEE PAMI*, 31(5):824–840.

Shapovalov, R. and Velizhev, A. (2011). Cutting-plane training of non-associative markov network for 3d point cloud segmentation. In *3DIMPVT*.

Shapovalov, R., Velizhev, A., and Barinova, O. (2010). Non-associative markov networks for 3d point cloud classification. In *ISPRS Commission III symposium - PCV 2010*.

Sung, J., Ponce, C., Selman, B., and Saxena, A. (2012). Unstructured human activity detection from rgbd images. In *ICRA*.

Szeliski, R., Zabih, R., Scharstein, D., Veksler, O., Kolmogorov, V., Agarwala, A., Tappen, M., and Rother, C. (2008). A comparative study of energy minimization methods for markov random fields with smoothness-based priors. *IEEE PAMI*, pages 1068–1080.

Taskar, B., Chatalbashev, V., and Koller, D. (2004). Learning associative markov networks. In *ICML*.

Taskar, B., Guestrin, C., and Koller, D. (2003). Max-margin markov networks. In *NIPS*.

Torralba, A. (2003). Contextual priming for object detection. *IJCV*, 53(2):169–191.

Tsochantaridis, I., Hofmann, T., Joachims, T., and Altun, Y. (2004). Support vector machine learning for interdependent and structured output spaces. In *ICML*.

Xiao, J. and Quan, L. (2009). Multiple view semantic segmentation for street view images. In *ICCV*.

Xiong, X. and Huber, D. (2010). Using context to create semantic 3d models of indoor environments. In *BMVC*.

Xiong, X., Munoz, D., Bagnell, J. A., and Hebert, M. (2011). 3-d scene analysis via sequenced predictions over points and regions. In *ICRA*.