Semantic Labeling of 3D Point Clouds for Indoor Scenes

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

Abstract

Inexpensive RGB-D cameras that give an RGB image together with depth data have become widely available. In this paper, we use this data to build 3D point clouds of full indoor scenes such as an office and address the task of semantic labeling of these 3D point clouds. We propose 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. The model admits efficient approximate inference, and we train it 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, having 2495 segments labeled with 27 object classes), we get a performance of 84.06% in labeling 17 object classes for offices, and 73.38% in labeling 17 object classes for home scenes. Finally, we applied these algorithms successfully on a mobile robot for the task of finding objects in large cluttered rooms.[1]

1 Introduction

Inexpensive RGB-D sensors that augment an RGB image with depth data have recently become widely available. At the same time, years of research on SLAM (Simultaneous Localization and Mapping) now make it possible to reliably merge multiple RGB-D images into a single point cloud, easily providing an approximate 3D model of a complete indoor scene (e.g., 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. 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. 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 easy to reason about a variety of properties, which are based on 3D distances, volume and local convexity.

Some recent works attempt to first infer the geometric layout from 2D images for improving the object detection [12, 14, 28]. However, such a geometric layout is not accurate enough to give significant improvement. Other recent work [35] 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 together 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. 1). We predict not only coarse classes like in [1, 35] (i.e.,

[1] This work was first presented at [16].
* indicates equal contribution.

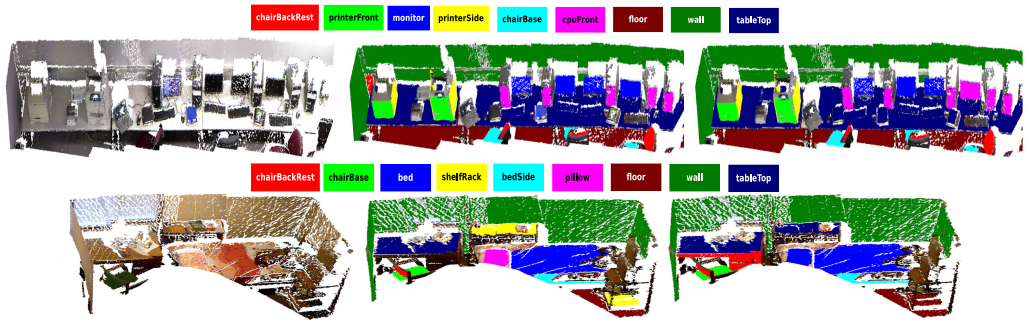

Figure 1: Office scene (top) and Home (bottom) scene with the corresponding label coloring above the images. The left-most is the original point cloud, the middle is the ground truth labeling and the right most is the point cloud with predicted labels.

wall, ground, ceiling, building), but also label individual objects (e.g., printer, keyboard, mouse). Furthermore, we model rich relational information beyond an associative coupling of labels [1].

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. 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, visual similarity, object co-occurrences and proximity). The model admits efficient approximate inference [25], and we show that it can be trained using a maximum-margin approach [7, 31, 34] 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 introduce parsimony in the model by using appropriate clique potentials rather than using general clique potentials. Our model is highly flexible and our software is available as a ROS package at: `http://pr.cs.cornell.edu/sceneunderstanding`

To empirically evaluate our model and algorithms, we perform several experiments over a total of 52 scenes of two types: offices and homes. These scenes were built from about 550 views from the Kinect sensor, and they are also available for public use. We consider labeling each segment (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 consider the problem of labeling 3D segments with multiple attributes meaningful to robotics context (such as small objects that can be manipulated, furniture, etc.). Finally, we successfully applied these algorithms on mobile robots for the task of finding objects in cluttered office scenes.

2 Related Work

There is a huge body of work in the area of scene understanding and object recognition from 2D images. Previous works focus on several different aspects: designing good local features such as HOG (histogram-of-gradients) [5] and bag of words [4], and designing good global (context) features such as GIST features [33]. However, these approaches do not consider the relative arrangement of the parts of the object or of multiple objects with respect to each other. A number of works propose models that explicitly capture the relations between different parts of the object e.g., Pedro et al.'s part-based models [6], and between different objects in 2D images [13, 14]. 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., [11, 12, 14, 20, 21, 22, 27, 28]). 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.

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 [9, 18, 19, 24, 26] and 3D segmentation (e.g., [3]). 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.). [8, 30] capture context by designing node features and [36] do so by stacking layers of classifiers; however these methods do not model the correlation between the labels. Some of these works model some contextual relationships in the learning model itself. For example, [1, 23] use associative Markov networks in order 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. All of these works [1, 23, 29, 30, 36] were designed for outdoor scenes with LIDAR data (without RGB values) and therefore would not apply directly to RGB-D data in indoor environments. Furthermore, these methods only consider very few geometric classes (between three to five classes) in outdoor environments, whereas we consider a large number of object classes for labeling the indoor RGB-D data.

The most related work to ours is [35], 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 was based on maximizing the pseudo-likelihood resulting in a sub-optimal learning algorithm. In comparison, our basic representation is a 3D segment (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.

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

Before getting into the technical details of the model, the following outlines the properties we aim to capture in our model:

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 (i.e., 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.

3.1 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

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

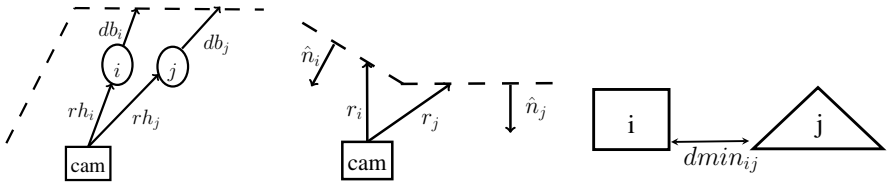

Figure 2: Illustration of a few features. (Left) Features N11 and E9. Segment i is infront of segment j if $\mathbf{rh_i} < \mathbf{rh_j}$. (Middle) Two connected segment i and j are form a convex shape if $(r_i - r_j).\hat{n_i} \geq 0$ and $(r_j - r_i).\hat{n_j} \geq 0$. (Right) Illustrating feature E8.

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 $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$. We describe in Section 3.2 how this graph is derived from the spatial proximity of the segments. 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.

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

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 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 $T_t = \{(k, k) | \forall k = 1..K\}$. And it is modeled by an non-associative edge potential if $T_t = \{(l, k) | \forall l, k = 1..K\}$. Finally, it is modeled by an object-associative edge potential if $T_t = \{(l, k) | \exists \text{object}, \; l, k \in \text{parts(object)}\}$.

Parsimonious model. In our experiments we distinguished between two types of edge feature maps—"object-associative" features $\phi_{oa}(i, j)$ used between classes that are parts of the same object (e.g., "chair base", "chair back" and "chair back rest"), and "non-associative" features $\phi_{na}(i, j)$ that are used between any pair of classes. Examples of features in the object-associative feature map $\phi_{oa}(i, j)$ include similarity in appearance, co-planarity, and convexity—i.e., features that indicate whether two adjacent segments belong to the same class or object. A key reason for distinguishing between object-associative and non-associate features is parsimony of the model. In this parsimonious model (referred to as *svm_mrf_parsimon*), we model object associative features using object-associative edge potentials and non-associative features as non-associative edge potentials. As not all edge features are non-associative, we avoid learning weight vectors for relationships which do not exist. 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.

3.2 Features

Table 1 summarizes the features used in our experiments. $\lambda_{i0}, \lambda_{i1}$ and λ_{i2} are the 3 eigen-values of the scatter matrix computed from the points of segment i in decreasing order. c_i is the centroid of segment i. r_i is the ray vector to the centroid of segment i from the position camera in which 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 $\phi_n(i)$ consist of visual appearance features based on histogram of HSV values and the histogram of gradients (HOG), as well as local shape and geometry features that capture properties such as how planar a segment is, its absolute

[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.

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. 2)	1

Edge features for (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. 2)	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. Diff. in angle with vert.: $\cos^{-1}(n_{iz})$ - $\cos^{-1}(n_{jz})$	1
E8. Dist. between closest points: $\min_{u \in s_i, v \in s_j} d(u, v)$ (Fig. 2)	1
E8. rel. position from camera (in front of/behind). (Fig. 2)	1

Table 1: Node and edge features.

location above ground, and its shape. Some features capture spatial location of an object in the scene (e.g., N9).

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 4. The edge features $\phi_t(i, j)$ (Table 1-right) consist of associative features (E1-E2) based on visual appearance and local shape, as well as non-associative features (E3-E8) that capture the tendencies of two objects to occur in certain configurations.

Note that our features are insensitive to horizontal translation and rotation of the camera. However, our features place a lot of emphasis on the vertical direction because gravity influences the shape and relative positions of objects to a large extent.

3.2.1 Computing Predictions

Solving the argmax in Eq. (1) for the discriminant function in Eq. (2) is NP hard. However, its equivalent formulation as the following mixed-integer program has a linear relaxation with several desirable properties.

$$\hat{\mathbf{y}} = \underset{\mathbf{y}}{\operatorname{argmax}} \max_{\mathbf{z}} \sum_{i \in \mathcal{V}} \sum_{k=1}^{K} y_i^k \left[w_n^k \cdot \phi_n(i) \right] + \sum_{(i,j) \in \mathcal{E}} \sum_{T_t \in \mathcal{T}} \sum_{(l,k) \in T_t} z_{ij}^{lk} \left[w_t^{lk} \cdot \phi_t(i, j) \right] \tag{3}$$

$$\forall i, j, l, k: \; z_{ij}^{lk} \leq y_i^l, \quad z_{ij}^{lk} \leq y_j^k, \quad y_i^l + y_j^k \leq z_{ij}^{lk} + 1, \quad z_{ij}^{lk}, y_i^l \in \{0, 1\} \tag{4}$$

Note that the products $y_i^l y_j^k$ have been replaced by auxiliary variables z_{ij}^{lk}. Relaxing the variables z_{ij}^{lk} and y_i^l to the interval $[0, 1]$ leads to 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) [10]. Furthermore, this relaxation can also be solved as a quadratic pseudo-Boolean optimization problem using a graph-cut method [25], which is orders of magnitude faster than using a general purpose LP solver (i.e., 10 sec for labeling a typical scene in our experiments). Therefore, 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* [2, 10]. 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.

Since every segment in our experiments is in exactly one class, we also consider the linear relaxation from above with the additional constraint $\forall i : \sum_{j=1}^{K} y_i^j = 1$. This problem can no longer be solved via graph cuts and is not half-integral. We refer to its solution as $\hat{\mathbf{y}}_{LP}$. Computing $\hat{\mathbf{y}}_{LP}$ for a scene takes 11 minutes on average[4]. Finally, we can also compute the exact mixed integer solution including the additional constraint $\forall i : \sum_{j=1}^{K} y_i^j = 1$ using a general-purpose MIP solver[4]. We set a time limit of 30 minutes for the MIP solver. This takes 18 minutes on average for a scene. All runtimes are for single CPU implementations using 17 classes.

When using this algorithm in practice on new scenes (e.g., during our robotic experiments), objects other than the 27 objects we modeled might be present (e.g., coffee-mugs). So we relax the constraint $\forall i : \sum_{j=1}^{K} y_i^j = 1$ to $\forall i : \sum_{j=1}^{K} y_i^j \leq 1$. This increases precision greatly at the cost of some drop in recall. Also, this relaxed MIP takes lesser time to solve.

3.2.2 Learning Algorithm

We take a large-margin approach to learning the parameter vector $\mathbf{w}$ of Eq. (2) from labeled training examples $(\mathbf{x}_1, \mathbf{y}_1), ..., (\mathbf{x}_n, \mathbf{y}_n)$ [31, 32, 34]. Compared to Conditional Random Field training [17]

[4] http://www.tfinley.net/software/pyglpk/readme.html

using maximum likelihood, this has the advantage that the partition function normalizing Eq. (2) does not need to 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

$$R(h) = \frac{1}{n}\sum_{j=1}^{n}\Delta(\mathbf{y}_j, \hat{\mathbf{y}}_j), \tag{5}$$

where $\hat{\mathbf{y}}_j$ is the optimal solution of Eq. (1) and $\Delta(\mathbf{y}, \hat{\mathbf{y}}) = \sum_{i=1}^{N}\sum_{k=1}^{K}|y_i^k - \hat{y_i^k}|$. 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 [15]:

$$\min_{w,\xi} \quad \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\xi \tag{6}$$

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

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 [15]. 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 need access to an efficient method for computing

$$\bar{\mathbf{y}}_i \quad = \underset{\mathbf{y}\in\{0,0.5,1\}^{N\cdot K}}{\operatorname{argmax}} \left[\mathbf{w}^T\Psi(\mathbf{x}_i, \mathbf{y}) + \Delta(\mathbf{y}_i, \mathbf{y})\right]. \tag{7}$$

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 [7], the value of ξ at the solution is an upper bound on the training error in Eq. (5). Furthermore, [7] observed empirically that the relaxed prediction $\hat{\mathbf{y}}_{cut}$ after training $\mathbf{w}$ via Eq. (6) 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.

4 Experiments

4.1 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 (composed from about 550 views). Each scene was reconstructed from about 8-9 RGB-D views from a 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, 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`.

4.2 Results

Table 2 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. Since our algorithm can only predict one label for each segment, micro precision and recall are same as the percentage of correctly classified segments. Macro precision and recall are respectively the averages of precision and recall for all classes. The optimal C value is determined separately for each of the algorithms by cross-validation.

Figure 1 shows the original point cloud, ground-truth and predicted labels for one office (top) and one home scene (bottom). We see that on majority of the classes we are able to predict the correct

Table 2: **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*	26.23	26.23	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

label. It makes mistakes in some cases and these usually tend to be reasonable, such as a pillow getting confused with the bed, and table-top getting confused with the shelf-rack.

One of our goals is to study the effect of various factors, and therefore we compared different versions of the algorithms 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 2. The *svm_node_only* model is equivalent to the multi-class SVM formulation [15]. Table 2 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 3.1, 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* respectively. 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.

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.

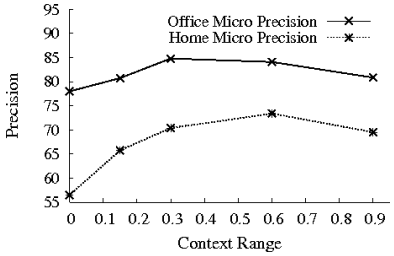

Figure 3: Effect of context range on precision (=recall here).

In order to study this, we compared our *svm_mrf_parsimon* with varying context range for determining the neighborhood (see Figure 3 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 increasing the context range can connect irrelevant objects

with an edge, 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 2, we compare the performance of our full model with a model that was trained and tested on single views of the same scenes. During the comparison, the training folds were consistent with other experiments, however the segmentation of the point clouds was different (because each 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 scenes (e.g., wall, ground) get split up into multiple segments in single views. 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 Table 2 were generated using the MIP solver. We observed that the MIP solver is typically 2-3% more accurate than the LP solver. The graph-cut 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. Since it is orders of magnitude faster, it is ideal for realtime robotic applications.

4.3 Robotic experiments

The ability to label segments is very useful for robotics applications, for example, in detecting objects (so that a robot can find/retrieve an object on request) or for other robotic tasks. We therefore performed two relevant robotic experiments.

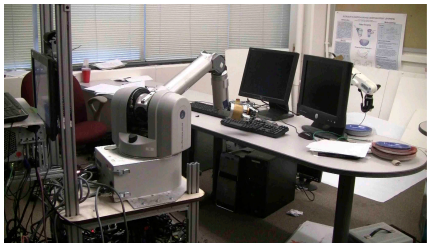

Figure 4: Cornell's POLAR robot using our classifier for detecting a keyboard in a cluttered room.

Attribute Learning: In some robotic tasks, such as robotic grasping, it is not important 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 scenes, giving a total of 10 unique attributes in total, comprised of: *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.

Object Detection: We finally use our algorithm on two mobile robots, mounted with Kinects, for completing the goal of finding objects such as a keyboard in cluttered office scenes. The following video shows our robot successfully finding a keyboard in an office: `http://pr.cs.cornell.edu/sceneunderstanding/`

In conclusion, we have proposed and evaluated the first model and learning algorithm for scene understanding that exploits rich relational information from the full-scene 3D point cloud. We applied this technique to object labeling problem, and studied affects of various factors on a large dataset. Our robotic application shows that such inexpensive RGB-D sensors can be extremely useful for scene understanding for robots. This research was funded in part by NSF Award IIS-0713483.

References

[1] D. Anguelov, B. Taskar, V. Chatalbashev, D. Koller, D. Gupta, G. Heitz, and A. Ng. Discriminative learning of markov random fields for segmentation of 3d scan data. In *CVPR*, 2005.

[2] E. Boros and P. Hammer. Pseudo-boolean optimization. *Dis. Appl. Math.*, 123(1-3):155–225, 2002.

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

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

[5] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005.

[6] P. Felzenszwalb, D. McAllester, and D. Ramanan. A discriminatively trained, multiscale, deformable part model. In *CVPR*, 2008.

[7] T. Finley and T. Joachims. Training structural svms when exact inference is intractable. In *ICML*, 2008.

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

[9] S. Gould, P. Baumstarck, M. Quigley, A. Y. Ng, and D. Koller. Integrating Visual and Range Data for Robotic Object Detection. In *ECCV workshop Multi-camera Multi-modal (M2SFA2)*, 2008.

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

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

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

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

[14] D. Hoiem, A. A. Efros, and M. Hebert. Putting objects in perspective. In *In CVPR*, 2006.

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

[16] H. Koppula, A. Anand, T. Joachims, and A. Saxena. Labeling 3d scenes for personal assistant robots. In *R:SS workshop on RGB-D cameras*, 2011.

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

[18] K. Lai, L. Bo, X. Ren, and D. Fox. A Large-Scale Hierarchical Multi-View RGB-D Object Dataset. In *ICRA*, 2011.

[19] K. Lai, L. Bo, X. Ren, and D. Fox. Sparse Distance Learning for Object Recognition Combining RGB and Depth Information. In *ICRA*, 2011.

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

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

[22] C. Li, A. Kowdle, A. Saxena, and T. Chen. Towards holistic scene understanding: Feedback enabled cascaded classification models. In *NIPS*, 2010.

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

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

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

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

[27] A. Saxena, S. H. Chung, and A. Y. Ng. Learning depth from single monocular images. In *NIPS 18*, 2005.

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

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

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

[31] B. Taskar, V. Chatalbashev, and D. Koller. Learning associative markov networks. In *ICML*. ACM, 2004.

[32] B. Taskar, C. Guestrin, and D. Koller. Max-margin markov networks. In *NIPS*, 2003.

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

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

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

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