Hierarchical Semantic Labeling for Task-Relevant RGB-D Perception

Chenxia Wu, Ian Lenz and Ashutosh Saxena
Department of Computer Science, Cornell University, USA.
Email: \{chenxiawu, ianlenz, asaxena\}@cs.cornell.edu

Abstract—Semantic labeling of RGB-D scenes is very important in enabling robots to perform mobile manipulation tasks, but different tasks may require entirely different sets of labels. For example, when navigating to an object, we may need only a single label denoting its class, but to manipulate it, we might need to identify individual parts. In this work, we present an algorithm that produces hierarchical labelings of a scene, following is-part-of and is-type-of relationships. Our model is based on a Conditional Random Field that relates pixel-wise and pair-wise observations to labels. We encode hierarchical labeling constraints into the model while keeping inference tractable. Our model thus predicts different specificities in labeling based on its confidence—if it is not sure whether an object is Pepsi or Sprite, it will predict soda rather than making an arbitrary choice. In extensive experiments, both offline on standard datasets as well as in online robotic experiments, we show that our model outperforms other state-of-the-art methods in labeling performance as well as in success rate for robotic tasks.

I. INTRODUCTION

Semantic scene labeling is crucial to many robotic tasks, allowing a robot to precisely localize objects, build maps, perform mobile manipulation tasks, and achieve many other goals. In recent work, many algorithms have been developed to produce such a labeling for RGB-D images (e.g., [37, 24, 4, 14]). However, these approaches produce only a flat labeling of a scene, ignoring important relationships between the label classes. In this work, we present an algorithm whose output is a hierarchical labeling of the scene.

These hierarchical labels are very important for a wide range of robotic applications. Segmenting object parts, such as handles, knobs, and buttons, separately from the body of the object is critical to properly afford most household objects. Understanding hierarchical object classes can also enable a robot to make rational substitutions between objects. Consider, for example, the task of fetching a Coke from a fridge (Fig. 1). To open the fridge, the robot must detect and grasp the fridge handle separately from its door. Then, if a Coke is not present in the fridge, it is much more desirable for the robot to return with another soda, such as a Pepsi, than empty-handed.

Using a semantic label hierarchy as shown in Fig. 2 enables these behaviors, which could not be realized using flat labels. When labeling with this hierarchy, each pixel belongs to a series of increasingly-general labels - for example, a pixel of class fridge-handle would also be of classes fridge-door, fridge and electronics. This also allows us to represent uncertainty, using a more general class when the algorithm is not sure which low-level class a pixel should belong to.

Fig. 1: Hierarchical Labels are produced by our algorithm as required for a robotic task. In the above environment, a robot is asked to fetch a Coke. It needs to perform three sub-tasks: navigate to the fridge, open the fridge door, and pick up the Coke (shown in three rows). For navigation, the robot needs to produce a higher-level fridge-door label so that it can approximately navigate close to it. Once it gets closer, producing a more detailed fridge-handle label is necessary. In the last step, the robot cannot detect Coke, so it fetches another soda instead. Such a label hierarchy lets a robot hedge its bets.

Conventional flat labeling approaches [37, 14] might simply be applied by flattening all the classes in the semantic hierarchy, but this sacrifices important information. Meanwhile, image classification approaches using semantic hierarchies [10, 35], which predict only one label for each image, cannot be applied to most robotic tasks that require pixel-level labeling of the entire scene. Properly integrating a semantic hierarchy into the labeling problem is a major challenge, and the main focus of this work.

To this end, we propose a novel approach which uses mixed integer programming to optimize a model isomorphic to a Conditional Random Field (CRF). Our model encodes
We perform extensive evaluation on the NYUD2 dataset as
been put into RGB-D scene understanding recently [45, 37, 38, 12, 8]. Due to the availability of affordable RGB-D sensors, significant effort has been put into RGB-D scene understanding recently [35, 37, 24, 31, 4, 25, 14, 20, 19, 17, 27]. Ren et al. [37] developed Kernel Descriptors, highly useful RGB-D feature, and used the segmentation tree to get contextual information. Gupta et al. [14] generalized 2D gPb-ucm contour detection to 3D, giving more effective segmentation. Koppula et al. [24] and Anand et al. [4] used rich contextual information for semantic labeling of 3D point clouds. Jia et al. [19] interpreted objects in a scene by reasoning about blocks, support, and stability. All these works predict flat labels, which are not applicable to many robotic tasks. Instead, our approach outputs a hierarchical labeling, which aids navigation, object finding and rational target substitution in robotic applications.

**Visual recognition using semantic hierarchies.** Our work is also related to visual recognition using semantic hierarchies [9, 39]. One similar work [10] classified large scale images by optimizing accuracy-specificity trade-offs. Ordonez et al. [35] considered predicting labels that people actually use to name an object. Both of these works targeted web image classification, and so predict a single label for each image denoting the most salient object. For many robotic tasks, we must consider pixel level labeling of multiple objects in a complex scene using a semantic hierarchy.

**Robotic tasks using vision.** There is also a huge body of works using vision algorithms to help perform different robotic tasks [13, 38, 16, 30], such as object grasping [42, 11, 29], navigation [16, 20], trajectory control [44], and activity anticipation [23]. Many works focused on improving SLAM techniques to better depict an environment for planning and navigation [34, 28], such as incremental smoothing and mapping using the Bayes Tree [21], real-time visual SLAM over large-scale environments [46], and object level SLAM [40]. Milford [35], He and Upcroft [15] proposed a place recognition algorithm for mobile robots. Katz and Brock [22] developed interactive segmentation for observing object motion during manipulation. Pangercic et al. [36] built semantic object maps for manipulation tasks for an autonomous service robot. Hinkle and Edwin [18] proposed a technique for functionally classifying objects using features obtained through physical simulations.

III. Overview

The input to our algorithm is a co-registered RGB and Depth image pair $I \in \mathbb{R}^{m \times n \times 3}, D \in \mathbb{R}^{m \times n}$, where $m, n$ are the image height and width. Our goal is to predict the label of each pixel and output the label matrix $L \in \mathbb{C}^{m \times n}$, where $C$ is the set of possible hierarchical semantic labels. We achieve this by mapping a semantic hierarchy graph to the segmentation tree built on the input image. We will first introduce the semantic hierarchy graph and the segmentation tree in this section.

**Semantic hierarchy graph.** For many robotic actions, we need semantic labels at different levels of abstraction rather than a simple object level. Therefore, we consider two types of relations in a semantic hierarchy:

- *Is-part-of.* For some robotic tasks, we need detailed localization of specific object parts. For example, to open a fridge, it is much better to know exactly where the fridge-handle

![Fig. 2: Semantic hierarchy graph. Each node denotes a class and each directed edge denotes a ‘belong to’ relation.](image-url)
is from the labeling rather than to simply guess based on a higher-level fridge-door label.

- **Is-type-of.** Understanding which objects belong to the same higher-level semantic class allows a robot to make rational substitutions between such objects. For example, if the robot is sent to find a Coke but cannot, it could instead return with any soda such as a Pepsi.

We represent this semantic hierarchy by a directed acyclic graph, called a **semantic hierarchy graph**, where the nodes \( C = \{ c_k \} \) represent the possible labels and the edges represent one of aforementioned relations. See Fig. 3 for an example.

**Segmentation tree of the RGB-D image.** We begin by segmenting the image into small segments. This gives us a set of candidate segments \( \{ s_i \} \) to label. If a segment is too small, visual and/or geometric information might be limited; if it is too large, it might straddle a class boundary. We therefore first obtain leaf node over-segmentations using a gPb-ucm approach extended for RGB-D images [14]. Second, we merge the most similar pairs of nodes step-by-step based on a similarity measure (the gPb-ucm boundary value\(^1\)) forming a tree as shown in Fig. 3.

![Segmentation tree](image)

**Fig. 3:** Illustration of segmentation tree. Pixels are grouped into small segments which are then merged to form a segmentation tree.

Note that mapping the semantic hierarchy graph to the segmentation tree is challenging, because both labels and segments are hierarchical rather than flat as in previous works. For example, for a parent segment with two child segments, it is possible to label them with parent-child labels such as labeling the parent as chair and the children as chair-back and chair-base, or to only label the children as two unrelated classes such as TV and cabinet. Thus, we need to take into account appropriate constraints in designing our CRF-based objective function. For many robotic applications, it is also desirable to be able to select the degree of specificity of the produced labels in the semantic hierarchy. Integrating all these desiderata into a parsimonious model is challenging.

**IV. PRELIMINARIES**

Our approach is based on a Conditional Random Field (CRF), modeling the unary terms of, and pair-wise relations between, the segments. We will introduce the unary term and a CRF model to label RGB-D images with flat-labels in this section. We first define the following notations:

- \( c_k \) \( k \)-th label in the semantic hierarchy graph.
- \( s_i \) \( i \)-th segment in the segmentation tree.
- \( y_{ik} \in \{ 0, 1 \} \). If \( s_i \) is labeled with \( c_k \), \( y_{ik} = 1 \), o.w. \( y_{ik} = 0 \).
- \( a_i \) number of pixels in segment \( s_i \).
- \( a_{ik} \) number of pixels of class \( c_k \) in segment \( s_i \).
- \( w_{ik} = a_{ik}/a_i \), fraction of \( c_k \) class pixels in segment \( s_i \).

**A. Unary term of a segment**

The unary term relates the features of a segment to its label. Kernel descriptors have been proven to be useful features for RGB-D scene labeling [37], so we extract six such descriptors from each segment: gradient, color, local binary pattern, depth gradient, spin, surface normals, and KPCA/self-similarity. The feature vector of segment \( s_i \) is denoted as \( z_i \). We then use the fraction of \( c_k \) class pixels in segment \( s_i \): \( w_{ik} = a_{ik}/a_i \) as a confidence score for \( s_i \) belonging to \( c_k \). Since each pixel belongs to several ground-truth classes in the hierarchy such as chair-back, chair, sittable, furniture, we treat this as a linear regression problem rather than a classification problem as in previous work [37]. In detail, ridge linear regression is used to train the linear prediction function \( w_{ik} = \theta^T z_i \).

**B. Labeling RGB-D Images with Flat Labels**

Previous work by Anand et al. [4] started by dividing the RGB-D image into small segments, with the goal of labeling each segment from a flat label set \( \{ c_k \} \). They then used a CRF to model the unary terms of and pair-wise relations between the segments. Since each segment is allowed to belong to only one class, we have the constraint \( \sum_{c_k} y_{ik} = 1 \). The objective function is as follows:

\[
\max_{y} \sum_{s_i, c_k} y_{ik} \hat{w}_{ik} + \sum_{(s_i, s_j) \in \mathbb{N}_{ck}} y_{ik} y_{jk} \Phi(s_i, s_j),
\]

subject to \( \sum_{c_k} y_{ik} = 1 \) \( \forall s_i \), \( y_{ik} \in \{ 0, 1 \} \).

Here the unary term is \( \hat{w}_{ik} \), the edge term is \( \Phi(s_i, s_j) = \alpha \exp(-\beta gPb(s_i, s_j)) \), in which \( gPb(s_i, s_j) \) is the gPb-ucm boundary weight between \( s_i, s_j \), and \( \alpha, \beta \) are two weighting parameters. The edge term encourages neighboring segments \( (s_i, s_j) \in \mathbb{N} \) with small boundaries to take the same label.

**V. OUR APPROACH**

In this section, we will describe an improved CRF model with constraints which allow labeling over semantic trees using hierarchical labels. We first define the following notations:

- \( \pi(\nu) \) a function that takes a vertex \( \nu \) in a directed graph and returns the set of its ancestors, including itself.
- \( \hat{\pi}(\nu) \) the set of ancestors without \( \nu \) itself: \( \pi(\nu) \setminus \{ \nu \} \).
- \( s_l \) \( l \)-th leaf node segment in the segmentation tree.
- \( H_l \) hierarchical relation graph of the ancestor set \( \pi(s_l) \).
- \( Q_l \) \( l \)-th maximal independent set of graph \( H_l \).
A. Labeling Segmentation Trees with Flat Labels

Now we describe how we label a segmentation tree, where flat segments are merged to form a tree as in Fig. [3] As some segments in the tree overlap, we first need to select which ones to label, and second predict their labels. We achieve this by enforcing that, for each leaf node segment, only one of its ancestors (including itself) is labeled. This is because a pixel can have only one label in a flat labeling scheme while these segments are overlapping. So following constraints are added.

**Non-overlapping constraints (NO-CT).** We replace the sum-to-one constraint \( \sum_{c_i} y_{ik} = 1, \forall s_i \) in Eq. [1] with \( \sum_{s_i \in \pi(s_j), c_k} y_{ik} = 1, \forall s_i \). Since all leaf nodes are considered, every pixel is labeled with exactly one label. We also need to ensure that the area of the child segment vs. the parent segment is accounted for in the objective function. We therefore weight each \( \tilde{w}_{ik} \) by the total number of pixels \( a_i \) of the segment \( s_i \).

The objective function then becomes:

\[
\text{TreeSeg-FlatLabel}_{\Phi}(\tilde{w}, a, \Phi) : \max_y \sum_{s_i, c_k} y_{ik} a_i + \sum_{(s_i, s_j) \in \Pi(c_k)} y_{ik} y_{jk} \Phi(s_i, s_j),
\]

subject to

\[
\sum_{s_i \in \pi(s_j), c_k} y_{ik} = 1, \forall \tilde{s}_i, y_{ik} \in \{0, 1\}.
\]  

B. Labeling Segmentation Trees with Hierarchical Labels

When hierarchical labels are introduced, the following interesting property emerges: even if a child node is labeled, its ancestors can be labeled with its ancestor classes. This complicates the specification of constraints in the model, so we add following hierarchical relation constraints. We summarize our RGB-D hierarchical semantic labeling approach in Alg. [1]

**Hierarchical relation constraints (HR-CT).**

1) **Find hierarchical relations.** We first define a tuple \((s_i, s_j, c_k, c_z)\), called hierarchical relation if it follows \(c_z \in \pi(c_k), s_j \in \pi(s_i)\). This allows the pair of segments \((s_i, s_j)\) to be labeled with \((c_k, c_z)\), respectively, as their order is consistent in both the segmentation tree and the semantic hierarchy graph. All such tuples comprise a set \(\Omega_l\) for each \(\pi(s_i)\).

2) **Build hierarchical relation graph.** In order to find all the constraints in each ancestor-set \(\pi(s_i)\) considering both the non-overlapping and hierarchical labeling properties, we build an undirected graph \(H_l = (\forall l, E_l)\), called hierarchical relation graph, of which the vertices are all possible assignments: \(V_l = \{y_{ik}, \forall s_i \in \pi(s_i), s_j \in \pi(s_i)\}\) and edges link vertices if they follow the hierarchical relation: \(E_l = \{(y_{ik}, y_{jk}) : \forall(s_i, s_j, c_k, c_z) \in \Omega_l\}\).

3) **Find constraints on the hierarchical relation graph.** Following the hierarchical relation, if two vertices \((y_{ik}, y_{jk})\) on \(H_l\) are linked by an edge, they can be both set to one.

**Algorithm 1** RGB-D Hierarchical Semantic Labeling.

**Input:** RGB and Depth image matrix \(I, D\).

**Output:** Pixel-level label matrix \(L\).

1. Obtain segment set \(\{s_i\}\) by building the segmentation tree on \(I, D\) (Section III);  
2. Extract feature \(z_i\) from each segment \(s_i\) (Section IV-A);  
3. Compute terms \(a_i, \tilde{w}_{ik}, \tilde{r}_k, \Phi\) in \(O_{\text{TreeSeg-HierLabel}}(\tilde{w}, a, \Phi, \Phi)\) Eq. [5];  
4. Obtain ancestor-set \(\pi(s_i)\) for each leaf node \(\tilde{s}_i\);  
5. Find hierarchical relations for each \(\pi(s_i)\):  
   \[\Omega_l = \{(s_i, s_j, c_k, c_z) : c_z \in \pi(c_k), s_j \in \pi(s_i), \forall(s_i, s_j) \in \Omega_l\},\]
   (Section V-B (1));
6. Build hierarchical relation graph \(H_l = (\forall l, E_l)\):
   \[E_l = \{(y_{ik}, y_{jk}) : \forall(s_i, s_j, c_k, c_z) \in \Omega_l\},\]
   (Section V-B (2));
7. Enumerate maximal independent set \(Q_{it}\) on each \(H_l\) (Section V-B (3));
8. Solve \(O_{\text{TreeSeg-HierLabel}}(\tilde{w}, a, \Phi, \Phi)\) Eq. [5] (Section VI);  
9. Label each pixel \(p\) with the most specific label from the set \(\{c_k | p \in s_i & y_{ik} = 1\}\):
   \[L_p = \arg \max_y a_i r_p \text{ subject to } p \in s_i & y_{ik} = 1\].

Otherwise, at most one can be set to one following the non-overlapping constraint. To give efficient and sufficient constraints, we constrain the sum of all \(y_{ik}\) in each maximal independent set (the set of vertices, no pair of which are adjacent) to be not greater than one. The problem then becomes to enumerate all maximal independent sets \(\{Q_{it}, t = 1, \ldots\\}\) of \(H_l\). In practice, we will introduce a parsimonious model (Sec. VII), leading to a sparse graph \(H_l\) thus more efficient constraint-finding. After finding these sets, we add the constraints \(\sum_{y_{ik} \in Q_{it}} y_{ik} \leq 1, \forall(s_i, t)\). To further ensure that all pixels to be labeled, we add the completeness constraints (CM-CT) \(\sum_{s_i \in \pi(s_i)} y_{ik} \geq 1, \forall(s_i, l)\) to ensure at least one segment in each \(\pi(s_i)\) to be labeled.

4) **Overlapping unary correction.** To give even weighting for each pixel, we also modify the unary term for the overlapping pixels when both parent and child segments are labeled. If \(y_{ik}\) and \(y_{jk}\) are both set to 1, \(y_{ik} y_{jk} = 1\), when \((s_i, s_j, c_k, c_z) \in \Omega_l\), we would rather label their overlapping pixels \(a_i\) with the more specific label \(c_k\). So, the summation of the unary term would be \(\tilde{w}_{ik} a_i + y_{jk}(a_j - a_i)\). Then, the objective function relating these two terms changes to \(y_{ik} \tilde{w}_{ik} a_i + y_{jk} \tilde{w}_{jk} a_j - y_{ik} y_{jk} \tilde{w}_{ik} \tilde{w}_{jk} a_i a_j\).

Note that considering these hierarchical relations and constraints,

2To enumerate maximal independent sets of \(H_l\), we first divide \(H_l\) into a subgraph \((\forall l, \emptyset)\), where \(\forall l\) are all isolated vertices in \(H_l\), \(\emptyset\) is the empty edge set, and another subgraph \(H_l = (\forall l, \emptyset)\). Then we enumerate all maximal independent sets \(\{Q_{it}, t = 1, \ldots\\}\) of \(H_l\) by enumerating all cliques of its complementary graph, which is a well-studied problem in graph theory [12] and is solved by the Bron-Kerbosch algorithm [13] in our approach. Finally, \(Q_{it} = Q_{it} \cup \forall l\).
leading to a linear objective which can be solved by a mixed integer programming (MIP) solver \( \Pi \):

\[
\text{OpTreeSeg-HierLabel}_{y,c}(\mathbf{w}, \mathbf{a}, \mathbf{r}, \Phi) : \\
\max \sum_{s_i \in \pi(c_k)} y_{ik} \tilde{w}_{ik} \tilde{r}_{k} a_i - \sum_{s_i, s_j \in \pi(c_k)} \xi_{ij} \tilde{y}_{ij} \tilde{r}_{ij} a_i \\
\text{overlapping correction terms} \\
+ \sum_{(s_i, s_j) \in \mathbb{N}, c_k} \xi_{ij} \Phi(s_i, s_j), \\
\text{edge terms} \\
\text{NO-CT, HR-CT, CM-CT} \\
\text{s.t.} \sum_{y_{ik} \in \mathcal{Q}_{ik}} y_{ik} \leq 1 \forall \bar{s}_i, t, \sum_{s_j \in \pi(s_i), c_k} y_{ik} \geq 1 \forall \bar{s}_i, \\
y_{ik} \in \{0,1\}, \xi_{ij} \tilde{y}_{ij} \tilde{r}_{ij} \in \{0,1\}, \\
(5)
\]

**Parsimonious Model.** We observe that there is some redundancy in the above objective, and introduce a parsimonious model to avoid this.

First, we do not need to consider all possible classes for each segment. Classes with low unary terms \( \tilde{w}_{ik} \tilde{r}_{k} a_i \) can be omitted for \( s_i \). We consider only the top \( \tau \) classes, leaving only \( \tau \) possible \( y_{ik} \) for each \( s_i \).

Second, in constraint-finding, some hierarchical relations \( (s_i, s_j, c_k, c_z) \in \Omega_l \) are mutually exclusive. So we also consider \( \tilde{w}_{ik} \tilde{r}_{k} a_i \) in each hierarchical relation, reducing the number of relations by greedily selecting the top ones with no conflicts. In detail, we first rank all the possible hierarchical relations \( (s_i, s_j, c_k, c_z) \) by the sum of unary terms of each pair \( \tilde{w}_{ik} \tilde{r}_{k} a_i + \tilde{w}_{ij} \tilde{r}_{ij} a_j \), all of which consist a candidate relation list. We select the one with the highest score from the list, link the corresponding edge in graph \( H_l \), and remove all its violating relations from the list. We repeat this selection until no relations remain in the list. As a result, the graph \( H_l \) becomes sparse with many isolated vertices, since only most confident relations are considered.

The most time consuming step in Alg. 1 is to enumerate the maximal independent sets in step 8. In the worst case it is \( O(n_3^3 h_s \tau^3) \), where \( n_3 \) is the number of leaf nodes of and \( h_s \) is the height of the segmentation tree, and \( \tau \) is the number of top considered classes. Though the worst-case running time is non-polynomial, the Bron-Kerbosch algorithm runs much faster in practice \( [3] \). In our experiments on the NYUD2 dataset, it only takes an average of 0.84 and 0.49 seconds per image respectively to find the constraints and optimize the objective using our parsimonious model.

---

3With larger \( \eta \), the relative weight for more specific class: \( (r_1/r_j)^\eta, r_j > r_1 \) is larger, thus prediction is more specific. The prediction is balanced when \( \eta = 0 \).
TABLE I: Average class recall of each class level on NYUD2 dataset.

<table>
<thead>
<tr>
<th></th>
<th>class level0</th>
<th>class level1</th>
<th>class level2</th>
<th>class level3</th>
</tr>
</thead>
<tbody>
<tr>
<td>[37]</td>
<td>23.77</td>
<td>30.52</td>
<td>36.02</td>
<td>41.56</td>
</tr>
<tr>
<td>Ours(bs+bc)</td>
<td>28.96</td>
<td>34.14</td>
<td>41.69</td>
<td>46.29</td>
</tr>
<tr>
<td>Ours(ts+bc)</td>
<td>30.08</td>
<td>36.09</td>
<td>45.96</td>
<td>51.80</td>
</tr>
<tr>
<td>Ours(ts+hc)</td>
<td>33.35</td>
<td>44.46</td>
<td>51.79</td>
<td>61.51</td>
</tr>
</tbody>
</table>

VII. SCENE LABELING EXPERIMENTS

Data. We evaluate our approach on a hierarchically-labeled subset of the NYUD2 dataset [45], which consists of RGB-D images from a wide variety of environments. We manually labeled a subset of 500 images using a hierarchy. We used 20 most common object classes and one background class, and additionally labeled 12 object-part classes and generalized 10 higher level classes. In total, we have 43 classes in the semantic hierarchy. We use the standard split of the NYUD2 dataset, giving 269 training images and 231 test images.

Implementation details. In our experiments, we used six RGB-D kernel descriptors to represent segments for both [37] and our approach. We kept the same setting as in [37] to run their approach: first, we ran gPb-ucm algorithm [5] on both the RGB and depth images separately and linearly combine them to get the gPb-ucm values, then built one segmentation tree by using different values to threshold these values. To make a fair comparison, we also ran the 3D gPm-ucm [14] algorithm to get the gPb-ucm value for both approach [37] and ours. So we denote the approach [37] based on original gPb-ucm as [37] and based on 3D gPb-ucm as [14]+[37].

Evaluation metric. We use three metrics for evaluating scene labeling performance: cumulative pixel accuracy, average information gain and average class recall. We label each scene image at the pixel level and consider it correct to label a pixel with its ground truth label or any of its ancestors, e.g., a pixel of class chair-back is also of class chair. If \( L_p \) is a prediction of a pixel label and \( L^*_p \) is its ground truth leaf node label, the cumulative pixel accuracy over the whole dataset is defined as: \( \sum_{p} I(L_p \in \pi(L^*_p))/n_p \), where \( I(.) \) is an indicator function and \( n_p \) is the number of pixels in the whole dataset, \( \pi(L_p) \) is the set of all possible correct predictions including \( L^*_p \) and all its ancestors in the semantic hierarchy.

With hierarchical labels, an algorithm can always predict the top-level parent classes and get higher performance, e.g., it is easier to label furniture vs table-leg. Therefore, following [10], we evaluate the degree of specificity for prediction. Specifically, we compute the information gain (Eq. 5) of each predicted class as defined earlier and compute the average.

Recall for class \( c \) is defined as: \( \frac{\sum_{p} I(L_p \in ch(c)) \& L_p \in \pi(L^*_p))}{\sum_{p} I(c \in \pi(L^*_p))} \), where \( ch(c) \) represent the class set of all \( c \)'s children plus \( c \) itself in the semantic hierarchy. So, the numerator is the number of correctly predicted pixels for class \( c \), and the denominator is the number of pixels with \( c \) as ground truth label.

Fig. 5: Results on NYUD2 dataset. For the same degree of specificity for prediction (i.e., same information gain, left) and recall (right), our algorithm performs better.

A. Results

We first evaluate the average class recall on four levels of the semantic hierarchy. Table. I summarizes the results. Class level0 contains the base classes, the most specific classes in the tree, e.g. object parts and low-level object classes. Higher levels are obtained by merging nodes in each previous level, leading to more general classes. Fig. 7 shows all classes for each level.

In this experiment, we train and predict labels on the base classes for flat labeling approaches [37],[14]+[37]. For our approach, we train and predict labels using leaf node segments on the base classes (Ours(bs+bc)), the segmentation tree on the base classes (Ours(ts+bc)) and the segmentation tree on the test class level and all classes below them in the semantic hierarchy (Ours(ts+hc)), with \( \eta = 0 \) for balanced prediction. These results reveal a number of interesting points as follows:

- The proposed approach Ours(ts+hc) shows the best results at each level, even though predicting more hierarchical labels is harder than the task of the other compared approaches, which only predict the base classes. This is because our approach effectively considers the mapping of the semantic hierarchy to the segmentation tree.
- Labeling on segmentation trees, e.g. Ours(ts+bc) and Ours(ts+hc), outperform methods labeling on flat segmentations. In [37], they considered hierarchical segmentation by packing all semantic features together in a tree path. However, they still label on the flat leaf node segmentations, losing some visual information.
- Prediction becomes easier when classes are more general. Thus, for tasks where specificity is not strictly required, we can predict more general labels to achieve higher accuracy.

To further evaluate the labeling performance using our semantic hierarchy, we plot average information gain vs. accuracy curves (Fig. 5[5] left) and average class recall vs. accuracy curves (Fig. 5[5] right) by varying the degree of specificity for prediction parameter \( \eta \). We compare approaches [37], [14]+[37] and our approaches using single ucm tree (Ours(ucm)), single normal tree (Ours(normal)), single semantic tree (Ours(category)) and using all three trees (Ours(full)). For the flat labeling approach [37], [14]+[37], we treat each class in the hierarchy as an arbitrary class without considering the hierarchy and train a one-vs-all SVM as in [37]. From these results, we can see that our approaches outperform the flat labeling approaches by a large margin, since the semantic hierarchy is considered. For the same de-
Fig. 6: Some samples of the results on NYD2 dataset (small areas are not shown with label names for clarity). In the first row, sofa back is labeled correctly since semantic hierarchy (sofa, sofa back) is considered. In the second row, our algorithm labeled the higher level classes desk, basic construction instead of desk surface, wall to avoid possible mistakes with the help of semantic hierarchy.

Fig. 7: Multi-level confusion matrix of our final results on NYUD2 dataset. From left to right, the confusion matrix zooms in to see more specific results in the next level below. In each confusion matrix, the red border square gives the classes merged in the next level up.

gree of specificity, our algorithms give higher accuracy. Using multiple segmentation trees also improves the performance.

We give two visual examples of labeling results in Fig. 6. In the first example, we can see that our algorithm yields a better labeling because semantic hierarchical relations such as (sofa, sofa back) are considered. The second example shows that the hierarchical labeling can use higher level classes to avoid possible mistakes, such as using desk or basic construction rather than desk surface or wall.

To further study the labeling results of our algorithm, we illustrate a multi-level confusion matrix in Fig. 7. We can see that some between-class labeling errors occur within one general class such as sofa, chair, stood on, most of which vanish in the next-higher level. However, some classes are hard to discriminate at any levels, such as door and wall, door and hanging. Our algorithm performed poorly for the background class others as it contains large variations in visual appearance.

VIII. ROBOTIC EXPERIMENTS

We evaluated our approach on three robotic tasks: object search, retrieval, and placement. We used a PR2 robot equipped with a Microsoft Kinect as our robotic platform. Table II shows a summary of the results, listing the perception accuracy (‘perc’) and end-to-end execution (‘exec’) separately.

A. Object Search Experiments

Here the goal for the robot is to locate a particular object in a room by moving around [37]. We compare our approach to [37]. For repeatable experiments, we pre-recorded a search tree at 20 discrete locations, each with a corresponding RGB-D frame (not in the training set).

We ran four separate trials for each algorithm, with the goal of searching for a chair back, fridge handle, mug handle, and baseball. To evaluate performance, the robot takes a fixed number of steps, and then reports the location at which it had the highest confidence of finding the given object. We score the algorithm’s performance based on the overlap ratio of the reported and ground-truth pixels of the target class for that frame, i.e., $|p_d \cap p_g| / |p_d \cup p_g|$, where $p_d, p_g$ are the detected object pixels and ground-truth object pixels.

Fig. 10 shows that for any fixed number of steps, our algorithm was able to outperform the approach from [37] for this task. Our algorithm was able to achieve an average overlap ratio of 0.4 after only 6 steps, while [37] took 15, showing that our approach does a better job of informing the search.

5Experimental setup details: The robot moves in discrete steps through the room, effectively moving through a search tree spanning the room. At each node in the tree, it turns to face each potential next location to move to, recording and labeling an RGB-D image for each. The robot will then move to the next location with the highest confidence score for containing the target object. If there are no unvisited neighboring locations, or this score is below some threshold, the robot will instead backtrack.
We define this as some other descendant of a class’s parent in the semantic hierarchy - for example, *Pepsi* is a substitute for *Coke* because both have the parent class *soda*. A flat labeling scheme is incapable of determining such substitutes, and will report failure if the target class is not found.

From Table II we can see that our algorithm achieves a very high rate of success for the complex drink-retrieval task shown in Fig. 8. Even though this task requires three separate phases of perception, our perception algorithm failed only once in ten trials, giving a 90% perception success rate. One more execution failure was due to the fridge door swinging closed before the robot could hold it open, giving an 80% overall success rate. Results for the bowl retrieval experiment were similar.

TABLE II: Robotic experiment results. Success rates for perception (‘perch’) and actual robotic execution (‘exec’) of each task.

<table>
<thead>
<tr>
<th></th>
<th>Search @20 steps perc exec</th>
<th>Retrieval Soda perc exec</th>
<th>Bowl perc exec</th>
<th>Placement Cushion perc exec</th>
<th>Average perc exec</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hierar. (ours)</td>
<td>64</td>
<td>38</td>
<td>80</td>
<td>100</td>
<td>87.90</td>
</tr>
<tr>
<td>Flat</td>
<td>44</td>
<td>33</td>
<td>80</td>
<td>50</td>
<td>47.5</td>
</tr>
</tbody>
</table>

At long distances, neither ours nor the baseline labeling algorithms were able to distinguish the handle from the door of the fridge, but our hierarchy informed the robot that the handle was part of the door. The flat labeling approach, meanwhile, lacked this information and simply failed if it could not identify the handle. In fact, the robot was only able to open the fridge 50% of the times using flat labels. Once opened, it could not identify proper substitutes if the desired drink was not present, leading to a mere 33% perception success rate.

C. Object Placement Experiments

We also performed a series of experiments in which the robot’s goal was object placement rather than retrieval. In particular, we considered the task of placing a cushion on a *sofa*, or on some other *sittable* object such as a *chair* if a *sofa* is not present. In every experiment performed, our algorithm was able to successfully locate the sofa, or a substitute if there was no sofa. One example of the robot successfully placing a cushion is shown in Fig. 9. By contrast, when using a flat labeling approach, the robot did not understand to place the cushion on another *sittable* surface if the sofa was not present, and thus succeeded only in the 50% of cases.

IX. CONCLUSION

Objects in human environments can be classified into a meaningful hierarchy, both because these objects are composed of parts (e.g. fridge-fridge door-fridge handle) and because of different levels of abstraction (e.g. drink-soda-Coke). Modeling this is very important in enabling a robot to perform many tasks in these environments. In this work, we developed an approach to labeling a segmentation tree with such hierarchical semantic labels. We presented a model based on a Conditional Random Field which incorporated several constraints to allow labeling using this hierarchy. Our model allows for different levels of specificity in labeling, while still remaining tractable for inference. We showed that our method outperforms state-of-the-art scene labeling approaches on a standard dataset (NYUD2), and demonstrated its use on several robotic tasks.
REFERENCES


tually guided semantic labeling and search for three-dimensional

detection and hierarchical image segmentation. *PAMI*, 33(5):

Search in the real world: Active visual object search based on


Imagenet: A large-scale hierarchical image database. In *CVPR*,
2009.

Optimizing accuracy-specificity trade-offs in large scale visual

exploiting shape adaptability and environmental constraints.
In *ICROS*, 2013.


to detect loop closure from range data. In *ICRA*, 2009.

labeling of 3d point clouds for indoor scenes. In *NIPS*,
2011.


[16] E. Herbst, X. Ren, and D. Fox. RGB-D flow: Dense 3-d motion

[17] E. Herbst, P. Henry, and D. Fox. Toward online 3-d object

[18] L. Hinkle and E. Herbst. Predicting object functionality using


F. Dellaert. iSAM2: Incremental smoothing and mapping using

[22] D. Katz and O. Brock. Interactive segmentation of articulated
objects in 3d. In *Workshop on Mobile Manipulation at ICRA*,
2011.

[23] H. Koppula and A. Saxena. Anticipating human activities using


activities and object affordances from rgb-d videos. *IJRR*, 32

[26] D. Kottas and S. Roumeliotis. Exploiting urban scenes for

[27] K. Lai, L. Bo, and D. Fox. Unsupervised feature learning for

[28] M. T. Lazaro, L. M. Paz, P. Pimientos, J. A. Castellanos, and
In *IROS*, 2013.


[31] M. Madry, C. H. Ek, R. Detry, K. Hang, and D. Kragic. Im-
proving generalization for 3d object categorization with global

[32] T. Malisiewicz and A. A. Efros. Improving spatial support for

[33] M. Milford. Vision-based place recognition: how low can you

[34] J. Neira, A. Davison, and J. Leonard. Guest editorial special
issue on visual SLAM. *Robotics, IEEE Transactions on*, 24(5):

From large scale image categorization to entry-level categories.
In *ICCV*, 2013.

object maps for robotic housework - representation, acquisition

[37] X. Ren, L. Bo, and D. Fox. RGB-D scene labeling: Features

[38] M. Ruhnke, L. Bo, D. Fox, and W. Burgard. Compact RGB-D


Kelly, and A. J. Davison. SLAM++: Simultaneous localisation

[41] A. Saxena, S. Chung, and A. Ng. Learning depth from single

[42] A. Saxena, J. Driemeyer, and A. Ng. Robotic grasping of novel

[43] A. Saxena, M. Sun, and A. Ng. Make3d: Learning 3d scene

[44] S. Shen, Y. Mulgaonkar, N. Michael, and V. Kumar. Vision-
based state estimation and trajectory control towards high-speed

segmentation and support inference from RGB-D images.
In *ECCV*, 2012.

Donald. Robust real-time visual odometry for dense RGB-D