EUROGRAPHICS 2015 / O. Sorkine-Hornung and M. Wimmer
(Guest Editors)

Volume 34 (2015), Number 2

SMARTANNOTATOR
An Interactive Tool for Annotating Indoor RGBD Images

Yu-Shiang Wong[1,2] Hung-Kuo Chu[1] Niloy J. Mitra[2]

[1]National Tsing Hua University, Taiwan [2]University College London

Abstract

RGBD images with high quality annotations, both in the form of geometric (i.e., segmentation) and structural (i.e., how do the segments mutually relate in 3D) information, provide valuable priors for a diverse range of applications in scene understanding and image manipulation. While it is now simple to acquire RGBD images, annotating them, automatically or manually, remains challenging. We present SMARTANNOTATOR*, an interactive system to facilitate annotating raw RGBD images. The system performs the tedious tasks of grouping pixels, creating potential abstracted cuboids, inferring object interactions in 3D, and generates an ordered list of hypotheses. The user simply has to flip through the suggestions for segment labels, finalize a selection, and the system updates the remaining hypotheses. As annotations are finalized, the process becomes simpler with fewer ambiguities to resolve. Moreover, as more scenes are annotated, the system makes better suggestions based on the structural and geometric priors learned from previous annotation sessions. We test the system on a large number of indoor scenes across different users and experimental settings, validate the results on existing benchmark datasets, and report significant improvements over low-level annotation alternatives. (Code and benchmark datasets are publicly available on the project page.)*

1. Introduction

Images with high quality semantic annotations provide rich training data for a variety of supervised and semi-supervised learning algorithms, both in computer graphics and computer vision. For example, in scene understanding, algorithms extract cues from the annotated datasets to learn dominant relationships between object labels and image features. The trained models can then used as *priors* for segmentation, recognition, manipulation, synthesis, etc. Beyond usage for learning models, such annotated datasets also provide qualitative and quantitative groundtruth for evaluating segmentation and labeling algorithms.

Users can generate such high quality semantic annotations by carefully annotate images one at a time. The process is tedious, time-consuming, and prone to errors. State-of-the-art web-based image annotation tools (e.g., LabelMe [RTMF08]) simplify the process by offering easy-to-draw interfaces and facilitating collaborative annotation. Alternately, semi-automatic segmentation (e.g., GrabCut [RKB04]) can potentially ease the burden on the users. Such methods, however, produce only 2D segmentation and object labels, and fail to reveal important 3D relations common across manmade scenes.

Affordable RGBD sensors (e.g., Microsoft Kinect) by providing synchronized color and depth data can resolve the problem. Unfortunately, annotating such images not only inherits the problems of the image setting, but is further complicated by the depth information being noisy. Directly investigating such RGBD images in 3D is frustrating given the partial nature of the data and can be confusing to the user. Not surprisingly, manually annotating such images requires

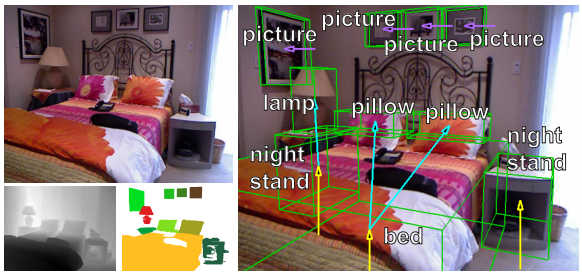


Figure 1: *Given an RGBD image and its image-level segmentations (left), object labels, 3D cuboid abstractions and structure of the scene (right) can be effectively annotated using the* SMARTANNOTATOR *with only a few user selections.*

significant efforts from the user, who has to flip across multiple viewpoints to indicate dimension of each object while imagining missing regions due to occlusion. For example, given a moderate scene with properly annotated object segments and labels, an average time of 5 minutes is reported to annotate only 3D cuboids (see [GH13]). Thus, while properly annotated depth data is valuable, the manual annotation process itself is difficult, and poses a major bottleneck.

In this work, we present SMARTANNOTATOR, an assisted interactive tool to annotate indoor RGBD images. Starting from raw RGBD images and a few user selections, the system outputs both image and scene level segmentation, object labels, 3D cuboid abstraction and structural relationships (e.g., contact, on-top, etc.) of the objects (see Figure 1). This is achieved by combining a novel 2D/3D inference scheme with object annotation such that they mutually assist each other. The system, in the background, generates multiple hypotheses involving computing segmentation, predicting the object labels, and inferring the 3D structure of scene. The user simply supervises the process by providing initial scribbles, progressively accepting suggestions, and in rare cases refining the predicted 3D structures. Thus, the user only selects among the ordered suggestions (e.g., if a shown box is 'bed' versus 'cabinet'), while the system updates its *understanding* of the scene and proposes refined suggestions, both in terms of labels and 3D structures for the remaining objects. At any point the user can 'approve all' to finalize the result. To the best of our knowledge, this is the first system that simultaneously annotates scene level segmentation, infers object labels and discovers 3D structure, all in an interactive setup.

The system works in two phases: a *learning phase* to bootstrap the system using a small set of labeled RGBD images (40 scenes) with properly annotated 3D structures from where the algorithm learns geometric and structural probability models; and, the key *annotating phase* to parse the input RGBD image into a 3D structure followed by reasoning possible support relationships and predicting the labels using the cuboids and learned models, respectively. Both geometric and structural models are exploited in the process, and progressively integrated over different annotation sessions. Hence, the models are progressively enriched as earlier scenes get assimilated as training data, which in turn simplifies the annotation of subsequent RGBD images (see supplementary material and video).

We evaluated our system on a benchmark RGBD dataset (762 indoor scenes with groundtruth annotation) across multiple users. Our system achieves label prediction accuracy of 70+% and 90+% in their frequencies where the target label appears among the top 3 and top 6 suggestions, respectively. The system is faster and more accurate compared to naive low-level tools, and produces high quality abstraction and structure in 3D with very simple user guidance.

Contributions. In summary, our main contributions include:

- an interactive indoor RGBD images annotation tool that combines incremental learning-based label prediction, 3D structure inference and refinements, and user assistance to facilitate annotation process;
- a label prediction algorithm that exploits geometric and structural models learned from the 3D structures of indoor scenes; and
- a context-driven structure refinement that utilizes the learned models and user hints to automatically adjust the dimensions and support relationships of cuboids, and solve occlusion.

2. Related Work

Image annotation. The ability to collect a large amount of annotated images is crucial for applications in computer vision. Russell *et al.* [RTMF08] develop a web-based image annotation tool, called LabelMe, to collect a large dataset of labeled images via an easy-to-use drawing interface. Xiao *et al.* [XOT13] extend the idea to propose a semi-automatic tool for annotating RGBD stream. Guo and Hoiem [GH13] present an interactive tool for annotating RGBD images with 3D structures using cuboids. While such systems expect intensive manual effort and focus on either pixel-level segmentation and their labels, or 3D geometry alone, SMARTANNOTATOR allows the user to accurately annotate not only image-level segmentation and labels, but also object-level 3D geometry and structural relationships. Our system also bears resemblance to the work of Boyko and Funkhouser [BF14], which combines the automatic grouping of similar objects with human intervention to facilitate labeling objects in 3D point cloud of urban landscape.

Incremental learning. Although crowdsourcing is a promising way to collect annotated data, annotations are expensive and error prone due to its labor-intensive nature. To minimize human efforts, active learning aims at requesting manual labeling only on images that are most informative to the classifier [KGUD07, VG11]. Other approaches bootstrap the learning algorithm using a small set of labeled images and automatically crawl data from the Internet to improve the classifiers [LFF10, CSG13]. Instead of minimizing the amount of requested images in annotation, we gather knowledge from previously annotated data to facilitate subsequent annotation sessions.

Indoor scene understanding. Indoor scene understanding can broadly be classified into two categories, namely, scene labeling and modeling. In the context of scene labeling, a large body of work focuses on extracting novel 2D image (e.g., SIFT and HOG) and 3D (e.g., depth image) features for the model learning. Algorithms that combine rich RGBD features from image level segmentation and contextual models have achieved dramatic performance gains in label prediction [KAJS11, SF11, RBF12]. Our scene labeling algorithm reasons on the 3D structure of the scene inferred from

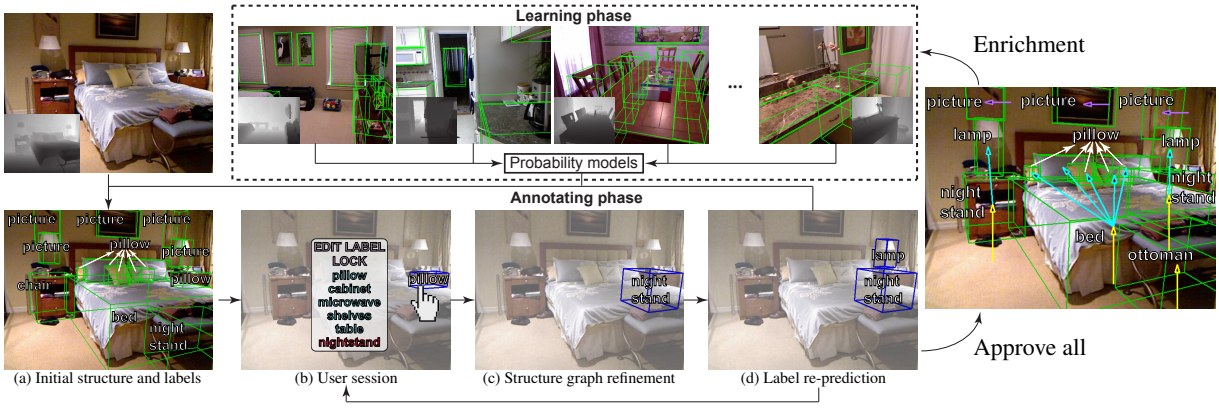


Figure 2: *System overview: Input to the learning phase is a small set of RGBD images with properly annotated labels and 3D structures (highlighted cuboids), based on which the algorithm learns the probability models. In the annotating phase, the system (a) builds the initial 3D structure of an input RGBD image, and predicts object labels using the learned models. (b-d) The user supervises the system by selecting among suggestions (e.g., re-order from 'pillow' to 'nightstand') while the system automatically refines the 3D structure to resolve ambiguity due to occlusion (e.g., the nightstand is refined to stand against the floor and wall) and re-predicts object labels (e.g., object on top of the 'nightstand' is more likely to be a 'lamp' than a 'pillow'). The process iterates until the user approving all the annotated data. The annotated image is shown on the rightmost side and is used to augment the training data.*

the indoor RGBD image. This goes beyond the scope of traditional scene labeling (c.f., [MWZ*13]).

Modeling indoor scene from monocular images has been extensively studied both in computer graphics and computer vision. Most approaches rely on detecting image features or high-level annotation to infer the 3D structure in a simplified setting such as planar popup segments [RT09, SSN09] and cuboids aligned with the dominant axes of the room [GEH10, HHF10]. Recent advancement incorporates geometric and contextual priors learned from 3D models or online furniture catalogs to boost the performance of appearance-based models [DPBF*12, CCPS13, ZZ13]. Zhang *et al.* [ZSTX14] present PanoContext, a system to model a full 3D context model from a panorama, and achieve impressive performance in the task of object detection.

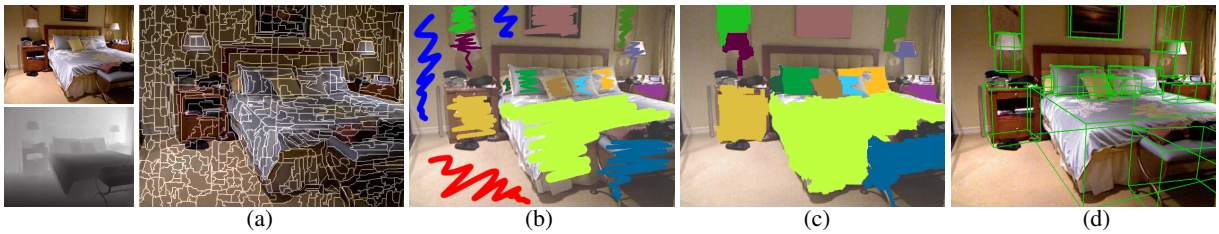
Motivated by the availability of low-cost depth sensors, significant progress has been made in inferring 3D structures from RGBD images. Silberman *et al.* [SHKF12] propose to reason the support surfaces and identify structural classes (e.g., 'Ground', 'Furniture' and 'Prop') for image regions. The work has been extended by Guo and Hoiem [GH13] to infer full 3D extent of support surfaces. However, these methods produce only support hypotheses either at segment level or using planar surfaces. Our approach focuses on predicting labels and inferring 3D geometry and support relationships at the level of full objects, represented as cuboids.

Our system partially overlaps with state-of-the-art methods in label prediction [LFU13] and 3D abstraction [JGSC13, SMZ*14] using RGBD images. Lin *et al.* [LFU13] base the contextual model learned from the detected cuboids to recognize the scene type and object labels. The system performance is sensitive to the quality of initial cuboids, and bad cuboids due to imperfect segmentation and occlusion are simply ignored in their evaluation. They achieve label prediction accuracy of 60%. While our system reports lower prediction accuracy of 47% in the first hit, the performance is dramatically improved to 70+% and 90+% in the top 3 and top 6 hits, respectively, under the interactive setup.

Jia *et al.* [JGSC13] and Shao *et al.* [SMZ*14] focus on the problem of abstracting occluded regions using physical stability. Labeling and in-class priors are not considered. For example, Jia *et al.* [JGSC13] incorporate support and stability inference into the segmentation pipeline to obtain plausible cuboid configuration. Their approach is based on merging segments to improve configuration and does not deal with scenes suffering object occlusion (see Figure 13 in [JGSC13] and Figure 15 in [SMZ*14]). Shao *et al.* [SMZ*14] optimize the arrangement of cuboids by analyzing physical stability to hallucinate the geometry of occluded regions. The system takes 10-20 seconds to converge and produces medium quality 3D structures. In contrast to these automatic methods, our interactive system solves occlusion among objects and produces high quality 3D structures with very simple user guidance.

3. Overview

Annotating an indoor RGBD image using the SMARTANNOTATOR involves two phases. The system first learns and reasons on RGBD data (learning phase) followed by utiliz-



(a) (b) (c) (d)

Figure 3: *Given an input RGBD image, our system generates the 3D abstraction by (a) computing the over-segmentation using color and depth cues; (b) requesting the user to provide rough scribbles on the regions of floor (red), wall (blue) and objects; (c) automatically inferring the image segments via the user's input; and (d) fitting 3D planes and cuboids to the segments of room layout and objects, respectively.*

ing the learned models to assist the user in annotation (annotating phase), as illustrated in Figure 2.

In the *learning phase*, the system takes a small set of annotated indoor RGBD images (40 scenes) as input. These images contain detailed annotation including object segmentation and labels, and a 3D abstraction of the scene comprising of a room layout (e.g., floor and walls) and cuboids, based on which we encode the geometry and structural relationships of objects in a *structure graph* (see Section 4.2). The learning algorithm bootstraps by reasoning on structure graphs of the training images to learn probability models that capture the geometry and structure of the objects. (see Section 5).

In the *annotating phase*, the system takes a novel RGBD image and generates an over-segmentation based on the color and depth cues. With the aid of the user who provides few scribbles on the image, the system automatically groups superpixels into object segments and generates a 3D abstraction of the scene (see Section 4.1). Based on the estimated room layout and cuboids, the geometry and structural relationships of objects are encoded in an initial structure graph (see Section 4.2).

The system then infers a list of suggestions (i.e., labels) for each object by reasoning on the initial structure graph using a joint probabilistic function based on the learned models (see Section 6.1). The control is then passed to the user who is responsible for supervising the system. The user is able to select an object and perform one of the two actions: i) confirm, reorder or override (e.g., typing) the suggestions proposed by the system through a context menu; and ii) modify the support relationships between objects via a drag-and-drop mouse interface (see Section 6.2). In response to every user's action, the system, in background, automatically refines the structure graph to resolve ambiguity and occlusion (see Section 6.3), re-predicts object labels based on the updated structure graph, and then waits for the next user session. The process iterates until the user approves all the annotated data. The user can optionally adjust the dimension of cuboids to improve the fitting accuracy. Then, we progressively get richer models by augmenting the existing dataset with the newly annotated images and perform retraining.

4. Modeling the 3D Structure of Scene

We propose a semi-automatic algorithm that exploits the depth and color cues from RGBD image to model the 3D structure of the scene. Specifically, we model the 3D abstraction of scene using 3D planes and cuboids to represent the room layout (e.g., floor and walls) and objects, respectively. By estimating the relationships among the cuboids and room layout, we encode the geometry and structural relationships of objects in a structure graph, which is further used in models learning and labels prediction algorithms.

4.1. 3D Abstraction

Given a RGBD image, the system starts by computing the surface normal at each pixel and applying a graph-based image segmentation [FH04] to obtain image over-segmentation based on the color and normal cues [SHKF12] (see Figure 3(a)). The user is requested to manually specify the image segments for room layout and objects by scribbling on the regions of floor, walls, and objects. Then the system automatically groups the superpixels that are covered by the scribbles to form the segments for floor, walls, and objects (see Figure 3(b)-(c)).

To extract the geometry of room layout, we fit a 3D plane to each of 'floor' and 'wall' segments by applying a RANSAC method on the corresponding 3D points. To generate 3D cuboids for objects, we assume that all the cuboids are standing up-right against the floor or stacked on top of other cuboids. Thus, we project the 3D points of object segment to the floor as 2D points, calculate a convex hull of 2D projection and fit a line to boundary points using RANSAC. The orientation of 3D cube is determined by two vectors: the direction orthogonal to the line and the normal to the floor. The dimension (or size) of 3D cube is determined by calculating a minimum bounding box that extends to the boundaries of 3D points. Note that we include only 95% of the 3D points in the bounding volume to improve the robustness. An example is shown in Figure 3(d).

4.2. Structure Graph

To facilitate both the learning and annotating processes, we introduce the *structure graph*, a data structure that encodes geometric and structural information of the estimated 3D abstraction. We define a structure graph as a *directed* graph $G := (V,E)$, where two special nodes, $v_f, v_w \in V$, denoting respectively the floor and wall, and each node $v_i \in V - \{v_f, v_w\}$ representing an object. For each object v_i, we denote c_i as its 3D cuboid and r_i as the 2D projection of c_i on the floor. Each directed edge $e_{ij} \in E$ describes a support relationship between node v_i and v_j and is associated to one of the support relationships listed below:

• **Supported by floor:** An object v_i is supported by the floor v_f if the the distance from bottom face of c_i to floor is within a threshold.

• **Supported by wall:** To examine the relationships between a cuboid and wall, we define the 'back' face of a cuboid as the face, excluding top and bottom ones, that is closest to any wall in the room. An object v_i is supported by the wall v_w if the distance and angle between its back face and the nearest wall are within a threshold.

• **Supported by object:** An object v_j is supported by another object v_i if one of the following criteria is met: (i) The distance between the top face of c_i and bottom face of c_j is within a threshold, and the centroid of r_j falls inside r_i or 30% of r_j is contained within r_i; or (ii) c_j is completely contained within c_i. The latter criterion is used to account for the scenario where an object is supported by a non-convex object (e.g., 'pillow' on a 'sofa'). For simplicity, we assume that each object is supported at most by one object. If there are more than one supporting objects, we choose the most probable one that better meets the above criteria.

According to the estimated support relationships, the objects are further classified into four sets, which are V_f (supported by floor), V_w (supported by wall), V_o (supported by object), and V_q (floating objects). Note that estimating the support relationships from the geometric point of view will introduce ambiguity such that two support relationships are met simultaneously (e.g., bed is supported by floor and wall). We currently resolve such ambiguity based on the predefined precedence that the relationship "supported by floor" is over "supported by object," which is over "supported by wall." We use the same distance and angular thresholds across the paper with the default setting of 15 cm and 30°, respectively.

5. Learning Phase

Learning models from informative features plays the key role to assist the user in annotation task. Inspired by previous efforts that extensively exploit geometry and contextual information in scene understanding [HHF10, DPBF*12, SHKF12, LFU13, ZZ13] and synthesis [MSL*11, YYT*11, FSH11, FRS*12], we learn from the training data the probability models that capture geometry, spatial configuration, and support relationships of objects.

5.1. Learning Probability Models

Dataset. The learning process is bootstrapped using 40 indoor RGBD images. These images contain detailed annotation including the object segments and labels, which are manually annotated by the user (e.g., via LabelMe). We extract the 3D abstraction of input scene based on the proposed algorithm (see Section 4.1), and manually refine the 3D structure (i.e., cuboid dimensions and support relationships) to build a baseline training data. Note that such a training data can be as well obtained from any existing dataset where the ground truth structure data are available. Then we convert the 3D abstractions into structure graphs and learn the geometric and structural models via reasoning on the structure graphs with respect to a list of object classes denoted as $L = \{l_1, ...l_k\}$. We further enrich the samples for each object class by randomly selecting 50 samples with text-based information from online furniture and appliances catalogs (e.g., IKEA).

Geometric model. The design of indoor objects is closely related to their functionality for supporting human activities. For example, the bed for sleeping is often large in base and flat, while the bookshelf for storing is typically taller. Hence, the 3D size of object can be a discriminative feature to distinguish different object classes [ZZ13]. We capture the geometric properties of an object using a 2-tuple vector describing the height and area of its 3D cuboid and 2D projection.

According to the presence of contextual relationships, we train two kinds of models using multi-class probability SVM [CL11]. The first one learns a probability model regarding all the object classes using the whole training data. We denote this model as $\mathcal{P}_g(l|v)$, which represents a likelihood of an object v is belong to the class label l. The second model takes into account the contextual relationships of objects. We examine support relationships among object classes based on the input structure graphs and classify the object classes into three categories of being supported by the floor (L_f), supported by walls (L_w), and supported by the object (L_o). Then, we train a probability model for each category, and denote it as $\mathcal{P}^i_g(l|v), i \in \{L_f, L_w, L_o\}$.

Support model. Support relationship has been proved to be a strong cue describing the local structure of an indoor scene [YYT*11, FRS*12, SMZ*14]. For example, pillow and lamp tend to be supported respectively by bed and desk, which are typically placed on the ground. In this work, we are interested in modeling three kinds of support relationships among the objects. Specifically, we define the support model, $\mathcal{P}_s(l|l')$, as the normalized frequency by counting the relationship an object class, $l \in L$, is supported by another object class (or room layout), $l' \in L \cup \{floor, wall\}$, among all the occurrence of (l, l') in the training data.

Spatial model. In interior design, the arrangement of most furniture is dominated by the geometry of room layout. For example, large furniture (e.g., bed and sofa) is usually placed against to wall. To model such a spatial configuration, we consider two configurations of close-to-wall and parallel-to-wall, and define respectively two probability models, $\mathcal{P}_{cw}(l)$ and $\mathcal{P}_{pw}(l)$, as the normalized frequencies of specific configurations that appear in the training data. We define object is close to or parallel to wall if the distance or angle between its back face and the nearest wall is within threshold.

6. Annotating Phase

Given a novel RGBD image, the system constructs the initial structure graph with the aid of the user as described in Section 4, and starts the annotating session by predicting object labels based on the initial structure graph and the learned models. The user then supervises the system by modifying the label or support relationship of a target object, while the system in background automatically refines the structure graph, re-predicts the object labels, and then waits for the next user feedback. The user can 'approve all' to finish the iterative process at any point of time.

6.1. Label Prediction

Given a structure graph, G, we formulate the problem of predicting labels for objects $\{v_1,...v_n\}$ as a maximum a posteriori (MAP) inference problem that aims at finding the most probable assignment of object classes $L^* = \{l_1^*,...,l_n^*\}, l_i^* \in L$ and is defined as:

$$\{L^*\} = \arg\max_L P(L|G)$$

where, $P(L|G)$ is a joint probability function defined on G,

$$P(L|G) = \prod_{v_i \in V} P(v_i) \cdot \prod_{e_{ij} \in E} P(v_j|v_i). \tag{1}$$

By taking $E_p(L|G) = -\log P(L|G)$ and factorizing $P(v_i)$ using contextual relationships, finding the MAP is equal to minimize the energy function,

$$\begin{aligned} E_p(L|G) = &- \sum_{j \in \{f,w,o\}} \sum_{v_i \in V_j} \log(\mathcal{P}_g^{L_j}(l_i|v_i)) - \\ &\sum_{v_i \in V_q} \log(\mathcal{P}_g(l_i|v_i)) - \sum_{e_{ij} \in E} \log(\mathcal{P}_s(l_j|l_i)). \end{aligned} \tag{2}$$

Instead of finding an optimal assignment, we adopt a strategy similar to N-best algorithm [PR11] to predict 6 high-scoring labels for each object. Generating multiple hypotheses is particularly suitable for such annotation task with human in the loop because it would reduce the frequency of tedious typing. Thus, for each object, we evaluate object classes using the potential functions and select class labels with top 6 scores. Objects are processed in a manner that starts from the first-tier objects in G (V_f and V_w), traces down the support hierarchy, and ends at floating objects (V_q).

Predicting on first-tier objects. For each object $v \in V_f \cup V_w$, we define a potential function on an object class, $l \in L$, as

$$\psi_f(v,l) = \begin{cases} \log(\mathcal{P}_g^{L_f}(l|v)) + \log(\mathcal{P}_s(l|floor)) & \text{if } v \in V_f \\ \log(\mathcal{P}_g^{L_w}(l|v)) + \log(\mathcal{P}_s(l|wall)) & \text{if } v \in V_w. \end{cases}$$

The candidate labels are those object classes with top 6 scores among L, evaluated by the potential ψ_f.

Predicting on supported objects. For each object $v \in V_o$, we utilize the 6-best labels from its supporting parent to predict the candidate labels. We define a potential function on an object class, $l \in L$, with a parent label, l_p, as

$$\psi_o(v,l,l_p) = \log(\mathcal{P}_g^o(l|v)) + \log(\mathcal{P}_s(l|l_p)),$$

and the best candidate label is the one with the highest score among L. The 6-best labels are retrieved by iteratively visiting parent candidate labels and selecting the best candidate label using the potential ψ_o.

Predicting on floating objects. After all the objects in support hierarchy are processed, the remaining objects are floating objects (V_q) that have no supporting parent. To infer 6-best labels for each floating object, we simply evaluate object classes using the geometric model, $\mathcal{P}_g$, and select those with top 6 scores.

6.2. User Session

After predicting the object labels, the system enters the user session and waits for the user's feedbacks. In display, the system draws contours of object segments and prints the object label with the highest score. To avoid confusion, a coloring scheme is used to render the contour and label according to the status of object (e.g., idle, selected, processed, etc). The user is able to select an object and performs one of the following operations.

Selecting among suggestions. The user supervises the suggestions from the system by performing the following actions via a context menu:

- **Confirm.** The user clicks the 'Lock' button in the context menu to confirm the label with the top score.
- **Re-order.** The correct label is not on top of the suggestions and the user rectifies the error by selecting a correct one among other candidate labels.
- **Type.** The label prediction is failed and none of the suggestions is correct. The user overrides the suggestions by typing a new label through a dialog.

Modifying the support relationships. The user is able to modify the support relationships estimated by the system using an intuitive mouse interface. To specify a support relationship for an object, the user simply clicks on the object segment to grab the object, drags the object toward its supporting parent and then drops it.

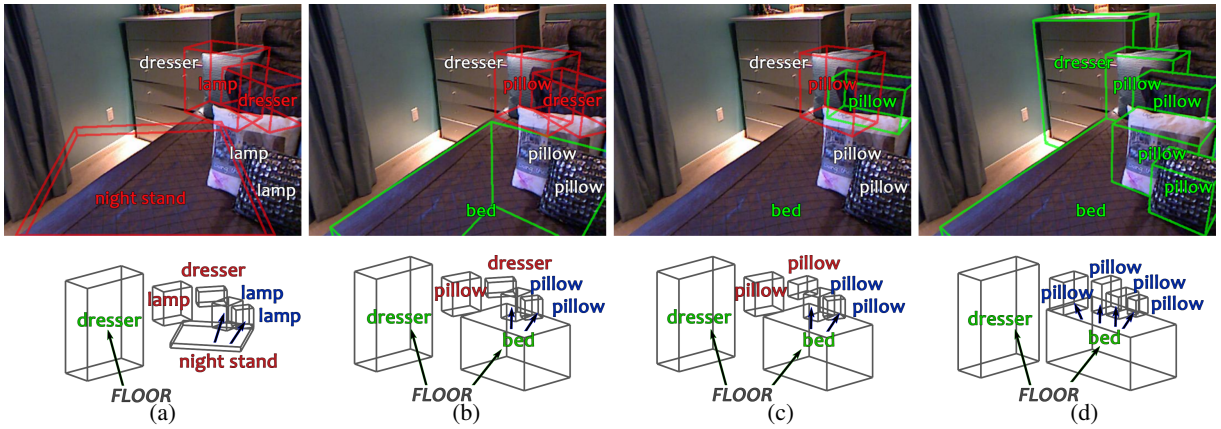


Figure 4: *Structure refinements. (a) The initial 3D structure and predicted labels. (b) The user re-orders from 'nightstand' to 'bed' and the system performs local refinements to improve the dimension and orientation of bed. (c) The user further assists in resolving the ambiguity of a 'pillow' versus 'dresser' and approves all the labels. (d) The system then globally infers the relationships among objects, and expands bed to support two floating pillows.*

Ending the annotation and post-processing. Each time the user performs one of the above actions, the system automatically refines the structure graph, re-predicts the object labels, and enters the next user session. Such an iterative process proceeds until the user clicks the 'Approve all' button to confirm all the annotated data. After that, the user can optionally adjust the dimension and orientation of cuboids to improve the overall cuboid approximation errors. We refer the reader to the supplementary video for such a user session.

6.3. Structure Graph Refinement

The initial 3D structure estimated by the automatic algorithm usually presents artifacts and ambiguities due to the inaccurate object segments, noisy depth data, and occlusion. While both the fully automatic and manual approaches are infeasible and impractical to deal with the problem, our insight lies in exploiting the learned models and prior knowledge (e.g., labels) offered by the user to resolve ambiguities and improve the quality. Thus, our system triggers the automatic structure refinement under the following scenarios: (i) when the user confirms the label of an object, the system performs a *local refinement* to adjust the dimension and orientation of the cuboid; and (ii) when the user approves all to end the annotation, the system first applies local refinement to all the objects, followed by performing a *global refinement* to resolve ambiguities for the floating objects and refine cuboids such that the physical support relationships among cuboids are consistent with the annotated structure. Figure 4 shows an example of such structure refinement. Now we elaborate the algorithm in details.

Local refinement. Given an object v with the label l confirmed by the user, the system adjusts the dimension and orientation of its cuboid based on the learned spatial and support models. Specifically, we propose three kinds of refinements, which are (i) aligning the cuboid to wall, (ii) extruding the back face of cuboid to wall, and (iii) extruding the bottom face of cuboid to floor, and are associated to the probability models, $\mathcal{P}_{pw}(l)$, $\mathcal{P}_{cw}(l)$, and $\mathcal{P}_s(l|floor)$, respectively. To determine whether a refinement should be carried out on the object, we define a potential function as

$$\psi_L(v,l) = f(l) + \phi(v),$$

where $f(l) \in \{\mathcal{P}_{pw}(l), \mathcal{P}_{cw}(l), \mathcal{P}_s(l|floor)\}$, and $\phi(v)$ is a penalty function to avoid the excessive refinement, which returns -1 if the refined cuboid introduces extra intersection with other cuboids and otherwise returns 0. The system checks each refinement in turn and applies the change to the object if $\psi_L > 0.5$. Then the support hierarchy of structure graph is re-estimated based on the updated cuboid.

Global refinement. In this step, the system exploits the annotation (i.e., labels and 3D structure) approved by the user to refine the structure of whole scene. The algorithm runs in two steps: (i) for any pair of objects (v_i, v_j) with the edge $e_{ij} \in E$, the system refines the cuboids, c_i and c_j, in a manner that c_j is physically supported by c_i; and (ii) inferring the support hypotheses for the floating objects (V_q) by inspecting the inter-object relationships based on the learned support model. In step (i), if the supporting parent of object v_j is a floor or wall, we simply extrude the 'bottom' or 'back' face of c_j to the floor or wall, respectively. Otherwise, the system extrudes or shrinks the bottom face of c_j to the top face of c_i, and expands the dimension of c_i in the direction parallel to floor such that the centroid of r_j falls inside r_i.

To generate support hypotheses for a floating object $v \in V_q$ with label l, the system searches nearby objects that are *likely to support* v, and denote them as V_s. Then an object,

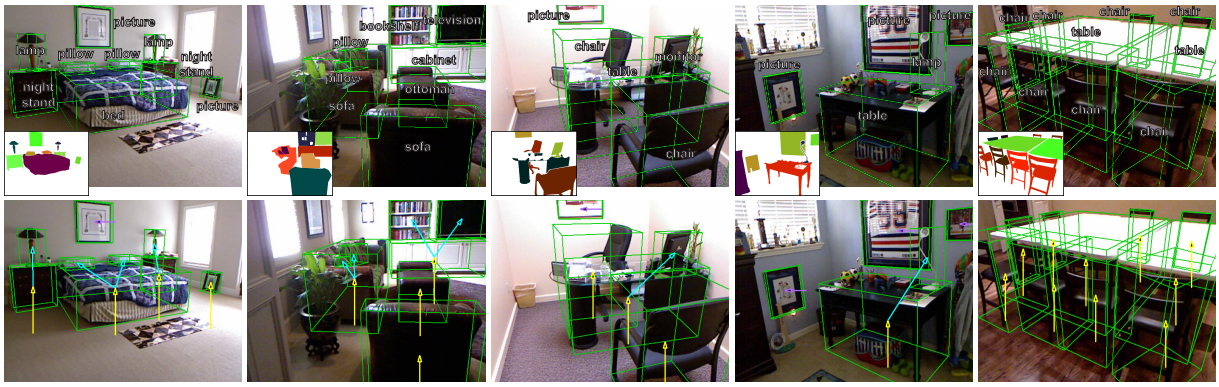


Figure 5: *Five annotated scenes in the experiment where the user only supervised on labeling. The top row shows the annotated object labels, the 3D cuboids and the ground truth segmentation as inset. The inferred structure is shown in the bottom row with yellow, purple and light blue arrows indicating support relationships of objects with floor, wall and other objects, respectively.*

the most probable supporting parent of v, is selected using

$$v^* = \arg\max_{v_i \in V_s} \mathcal{P}_s(l|l_i),$$

and the system refines the cuboids of v and v^* as described in step (i). We define an object v_i is *likely to support* another object v_j if the distance between two objects in both 3D (i.e., the top face of c_i and bottom face of c_j) and 2D (i.e., the projected bounding rectangles, r_i and r_j) are within a threshold.

7. Experiment and Evaluation

We extensively evaluated the performance of the system by conducting the experimental studying that (i) evaluated the performance on the learning and label prediction algorithms; (ii) compared with a naive annotation tool in terms of efficiency and quality of annotation; and (iii) evaluated the sensitivity of the system to different initial segmentation conditions. We compared the quality of annotated 3D structure with respect to ground truth data by calculating the approximation of the cuboid dimensions, and validating the support relationships using the precision, recall and F-measure.

Dataset and ground truth. We tested the system on the benchmark NYU2 RGBD dataset with images from 7 scene classes, including 'bedroom', 'kitchen', 'living room', 'bathroom', 'dining room', 'office', and 'home office'. Among these scenes, we manually processed the object classes by merging similar classes and discarding unfrequent ones. As a result, we obtained 762 scenes and 24 object classes, excluding the 'floor', 'wall' and 'ceiling' classes. To collect the ground truth data, we took the annotated object segments and labels from the NYU2 RGBD dataset, and extracted the 3D cuboids and support relationships from the dataset of Guo and Hoiem [GH13] to serve as the ground truth.

Evaluation metrics. Given an annotated scene, we compared the quality of structure graph to the corresponding ground truth by calculating how well the cuboid dimensions is approximated, and validating how well the support relationships are recovered. We employed the metric defined in the work of Shao *et al.* [SMZ*14] (see the Equation 6 therein) to calculate weighed L_1 norms that compare the extents of cuboid dimensions to the ground truth. To validate the quality of inferred structure, we calculated the precision-recall (*PR*) ratio, which captures the ratio of correct edges among respectively the target and ground truth graph, and converted it to the F-measure ($F_1 = 200\mathbf{PR}/(\mathbf{P+R})$).

7.1. Performance of Learning and Labeling

We randomly picked 40 scenes to bootstrap the learning process, and designed an incremental learning scheme by organizing the remaining 722 scenes into 16 trials such that each trial contains around 280 objects. We recruited 16 users with no prior knowledge about our system and requested each user to annotate a trial using the system. Each user was given a tutorial with 6 scenes to get used to the system interface and the flow of annotation process. To control the quality of image segments, we assumed that the segments of floor, walls and objects are given as inputs (e.g., using the ground truth). Thus the timing starts from inferring the initial 3D abstraction using the image segments and predicting object labels. During the annotating phase, the user is only allowed to perform the 'Confirm,' 'Re-order,' and 'Type' actions through context menu. Figure 5 shows 5 annotated scenes in the experiment (refer to supplementary material for a complete annotated dataset).

We evaluated the performance on: (i) the average timing of annotating a scene; (ii) the frequency of manual 'Type' action performed by the user; and (iii) the accuracy of label prediction. We computed the ratio of 'Top-1-Hit', 'Top-3-Hit', and 'Top-6-Hit', which indicate the frequency of the correct object label appearing respectively at the top, among

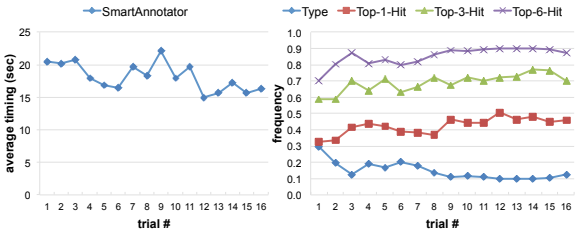


Figure 6: *Performance on learning and labeling. (Left) The average annotating time using our system, excluding the timing in specifying the image segments. (Right) The frequency of the manual action ('Type') performed by the user and the accuracy of label prediction ('Top-N-Hit').*

the top 3, and the top 6 of suggestions. As shown in Figure 6, our system achieves high label prediction accuracy in Top-3-Hit ($\sim$ 70%) and Top-6-Hit ($\sim$ 90%), meaning that only a small portion of objects ($\sim$ 10%) require to type in the labels. Hence, it takes only 18 seconds in average to annotate a scene using our system. One thing worth noting is that while the system bootstraps from weaker models and results in a lower prediction accuracy and higher 'Type' ratio in the earlier trials, the overall performance improves as more scenes are processed as shown in the climbing 'Top-N-Hit' curves.

We also evaluated the performance in a class-specific basis. In Table 1, we show the top 4 and bottom 4 object classes in a list sorted by the prediction accuracy in 'Top-1-Hit', and the corresponding class that is most easily confused by the system. We found that the performance usually drops in the objects that are frequently occluded in the indoor scene (e.g., 'nightstand' is occluded by 'bed'). In addition, a 'television' is frequently confused as a 'picture' due to its features of being flat in shape and hanging on the wall. Incorporating extra image-based features (e.g., texture, HOG, etc) might potentially improve the discriminating power of our models.

User experiences. We asked the user to comment the system after finishing the trial. In general, most users agreed that it is quite handy to annotate object labels using the SMARTANNOTATOR. They found it is particular useful and save a lot of time to identify the label from a short list of suggestions, especially when the answer frequently appears in the top 3

	samples	Top-1-Hit (%)	Top confusion (%)
picture	576	85.9	chair (3.4)
pillow	505	76.0	picture (17.4)
chair	833	65.2	picture (13.1)
bed	235	43.4	cabinet (18.7)
nightstand	98	0.0	chair (69.4)
television	43	0.0	picture (37.2)
dresser	42	0.0	chair (59.5)
ottoman	28	0.0	chair (57.1)

Table 1: *Class-specific performance evaluation.*

		SMARTANNOTATOR	Naive Tool
avg. timing (sec.)	label	13	17
	supp.	4	9
supp. quality (P/R(%), F_1)	init.	86.9 / 76.4, 81.3	50.0 / 78.4, 61.1
	label	92.7 / 92.7, 92.7	-
geo. quality (approx. err.)	init.	0.28	0.44
	label	0.26	-
	label+supp.	0.25	-

Table 2: *Comparing with naive annotation tool.*

suggestions. In addition, they also appreciated the favour offered by the system that propagates the change in an object's label to correct the labels of other objects.

Performance of probability models. To validate the effectiveness of geometric and support models in label prediction, we evaluate the performance in prediction accuracy using two model settings, with and without considering the support model. We use half of the dataset for training and the rest for testing. The statistics show that the prediction accuracy increases from 35% (using geometric model only) to 40% (using geometry and support models), demonstrating that using the contextual relationships does improve performance.

7.2. Comparing with Naive Annotation Tool

To prove the effectiveness of our system in facilitating the annotation process, we compared the system with a naive low-level annotation tool using 50 testing scenes with pre-annotated image segments (e.g., using the ground truth). We implemented a naive tool such that the user can type in the object label via a dialog and specify the support relationships using the same mouse interface as ours. To generate a baseline 3D structure in the naive system, we fitted a cuboid to each object segment a using the principal component analysis, and inferred the initial support relationships based on the proximity among cuboids. We used 380 training scenes in the learning phase of our system. Three out of the 16 users in Section 7.1 were recruited to annotate scenes using both systems and were given a similar training process for the naive tool. The annotation process runs in two stages that (i) the user first performs the object labeling task, followed by (ii) manually refining the support relationships.

We compared both systems in terms of the timing and the quality of initial 3D structure. We also evaluated how well the initial 3D structure is refined in our system after the user completing the object labeling task and refining the support relationships. The results can be found in Table 2. We can see that our system not only outperforms the naive one in the quality of initial 3D structure, but also obtains the performance gains in accuracy of support relationships (F_1 : 81.3 $\rightarrow$ 92.7) and improvement of cuboid dimensions ($\sim$ 22%). Thus, in both the object labeling ('label') and support refinement ('supp.') stages, the required manual efforts are only marginal in our system, resulting in the performance

boost in timings when comparing to the naive tool. Moreover, the low quality 3D cuboids generated by the naive system indicates that extra manual effort is expected to further refine the geometry of cuboids, which is a tedious and time consuming task as reported in [GH13].

7.3. Sensitivity to Object Segmentation

We evaluated how the performance of our system varies under different qualities of object segments. To this end, we combined the object contours of the ground truth data with different initial image segmentation. Specifically, we used a parameter, k, in the image segmentation algorithm [FH04] to control the size of superpixels. Note that the bigger the value of k, the larger the size of the superpixels. Then, the object segments were computed by grouping superpixels that have more than 60% of the area lying inside the object contours. We tested different settings of k on 50 scenes and recruited three out of the 16 users in the Section 7.1 to annotate the scenes using our system in a two-stage manner as described in the Section 7.2. As shown in Table 3, the performance of our system is stable under various object segmentations. Note that although the quality of initial 3D structure is only mediocre due to the imperfect object segments, the system effectively improves the accuracy of support relationships ($F_1 : 74.0 \rightarrow 90.6$) and cuboid approximation ($\sim 27\%$) after the annotating process. As a result, the system generates 3D structures with comparable quality to those using the ground truth object segments (see Table 2).

	k=5	k=20	k=100
supp. quality (P/R(%), F_1)			
init.	67.6 / 84.0, 74.9	66.7 / 85.1, 74.8	65.3 / 81.0, 72.3
label	91.1 / 92.1, 91.6	91.1 / 91.6, 91.4	88.6 / 89.1, 88.9
geo. quality (approx. err.)			
init.	0.37	0.37	0.36
label	0.29	0.34	0.28
label+supp.	0.29	0.28	0.27

Table 3: *Sensitivity to object segmentation.*

7.4. Performance

User scribbling. We tested our scribbling interface using a dataset of 50 scenes, with each scene contains 2-6 objects. We recruited two users who are given 6 training scenes to specify the image segments using our scribbling interface. On the average, users took $\sim$ 1 minute per scene. This indicates that although our semi-automatic approach prevents the user from processing at pixel-level, prescribing meaningful image segmentation still poses a bottleneck.

Timings. The automatic algorithm for fitting 3D planes and cuboids to respectively room layout and objects is efficient. Once the image segments are generated, it took less than a second to generate the 3D abstraction of a moderate scene with 4 objects using the unoptimized codes. Please note that the timing is proportional to the scene complexity.

8. Conclusion

We present SMARTANNOTATOR, an interactive system to facilitate annotating indoor RGBD images. The system performs the tedious tasks of predicting labels, inferring 3D structure, and comes up with various hypotheses, while the user only has to flip through a list of suggestions for object labels and marginally refine the hypotheses. The performance of system is validated through an extensive experimental studying, which demonstrates that our system outperforms a naive low-level tool in both the efficiency and quality of annotation. We plan to release both the system and source codes to public in the future.

Limitations and future work: Since we assume all the cuboids are standing up-right against the floor or stacked on top of other cuboids, the cuboid fitting algorithm can fail when the quality of floor segments is poor in a scene (e.g., barely visible or occluded). One possible solution is to offer an interface that allows the user to manually specify the floor orientation and height in 3D. In addition, objects with complex non-convex shape are only crudely captured using cuboids. A sophisticated 3D representation (e.g., composing using multiple cuboids) is necessary to capture more accurate geometric and structural information. While the accuracy of label prediction drops in the 'Top-1-Hit' ratio using our simple probabilistic models, we plan to exploit extra appearance-based features (e.g., SIFT and HOG) and contextual relationships (e.g., relative position and orientation between two objects) to further improve the overall performance. One worthy exploring direction is to integrate a repetition detector that detects multiple object instances (e.g., pillows and chairs) in 2D [CZM*10] or 3D [BF14, GTB14] to accelerate the annotation. Finally, a natural extension would be a fully automatic annotator by progressively confirming the top label of most confident object. For an initial result, see Figure 7.

Acknowledgements

We are grateful to the anonymous reviewers for their comments and suggestions; all the participants of the user study for their time; and Gerardo Figueroa for the video narration. The project was supported in part by the Ministry of Science and Technology of Taiwan (102-2221-E-007-055-MY3 and 103-2221-E-007-065-MY3), the Marie Curie Career Integration Grant 303541, the ERC Starting Grant SmartGeometry (StG-2013- 335373), and gifts from Adobe Research.

References

[BF14] BOYKO A., FUNKHOUSER T.: Cheaper by the dozen: Group annotation of 3D data. In *UIST* (Oct. 2014). 2, 10

[CCPS13] CHOI W., CHAO Y.-W., PANTOFARU C., SAVARESE S.: Understanding indoor scenes using 3d geometric phrases. In *IEEE CVPR* (2013), pp. 33–40. 3

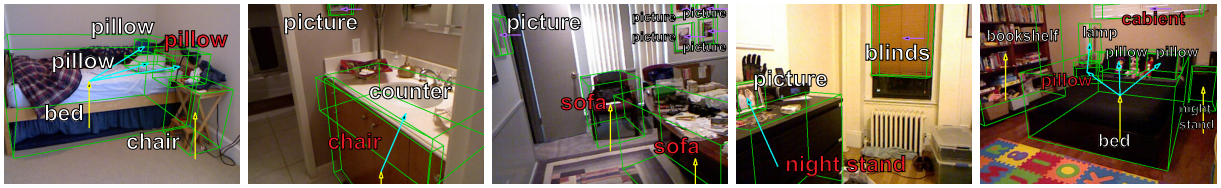


Figure 7: *Fully automatic annotation. The incorrect label prediction is drawn in red.*

[CL11] CHANG C.-C., LIN C.-J.: Libsvm: A library for support vector machines. *ACM Trans. Intell. Syst. Technol. 2*, 3 (2011), 27:1–27:27. 5

[CSG13] CHEN X., SHRIVASTAVA A., GUPTA A.: Neil: Extracting visual knowledge from web data. In *IEEE ICCV* (2013). 2

[CZM*10] CHENG M.-M., ZHANG F.-L., MITRA N. J., HUANG X., HU S.-M.: Repfinder: Finding approximately repeated scene elements for image editing. *ACM Trans. Graph. (Proc. SIGGRAPH) 29*, 4 (2010), 83:1–8. 10

[DPBF*12] DEL PERO L., BOWDISH J., FRIED D., KERMGARD B., HARTLEY E., BARNARD K.: Bayesian geometric modeling of indoor scenes. In *IEEE CVPR* (2012), pp. 2719–2726. 3, 5

[FH04] FELZENSZWALB P. F., HUTTENLOCHER D. P.: Efficient graph-based image segmentation. *IJCV* (2004). 4, 10

[FRS*12] FISHER M., RITCHIE D., SAVVA M., FUNKHOUSER T., HANRAHAN P.: Example-based synthesis of 3D object arrangements. *ACM Trans. Graph. (Proc. SIGGRAPH Asia) 31*, 6 (2012), 135:1–135:11. 5

[FSH11] FISHER M., SAVVA M., HANRAHAN P.: Characterizing structural relationships in scenes using graph kernels. *ACM Trans. Graph. (Proc. SIGGRAPH) 30*, 4 (2011), 34. 5

[GEH10] GUPTA A., EFROS A. A., HEBERT M.: Blocks world revisited: Image understanding using qualitative geometry and mechanics. In *ECCV* (2010). 3

[GH13] GUO R., HOIEM D.: Support surface prediction in indoor scenes. *IEEE ICCV* (2013). 2, 3, 8, 10

[GTB14] GUY E., THIERY J.-M., BOUBEKEUR T.: Simselect: similarity-based selection for 3d surfaces. *Comp. Graphics Forum (Proc. EUROGRAPHICS) 33*, 2 (2014), 165–173. 10

[HHF10] HEDAU V., HOIEM D., FORSYTH D.: Thinking inside the box: Using appearance models and context based on room geometry. In *ECCV* (2010). 3, 5

[JGSC13] JIA Z., GALLAGHER A., SAXENA A., CHEN T.: 3D-based reasoning with blocks, support, and stability. *IEEE CVPR* (2013). 3

[KAJS11] KOPPULA H. S., ANAND A., JOACHIMS T., SAXENA A.: Semantic labeling of 3d point clouds for indoor scenes. In *NIPS* (2011), pp. 244–252. 2

[KGUD07] KAPOOR A., GRAUMAN K., URTASUN R., DARRELL T.: Active learning with gaussian processes for object categorization. In *IEEE ICCV* (2007), pp. 1–8. 2

[LFF10] LI L.-J., FEI-FEI L.: Optimol: automatic online picture collection via incremental model learning. *IJCV 88*, 2 (2010), 147–168. 2

[LFU13] LIN D., FIDLER S., URTASUN R.: Holistic scene understanding for 3D object detection with rgbd cameras. *IEEE ICCV* (2013). 3, 5

[MSL*11] MERRELL P., SCHKUFZA E., LI Z., AGRAWALA M., KOLTUN V.: Interactive furniture layout using interior design guidelines. *ACM Trans. Graph. (Proc. SIGGRAPH) 30*, 4 (2011), 87:1–87:10. 5

[MWZ*13] MITRA N. J., WAND M., ZHANG H., COHEN-OR D., BOKELOH M.: Structure-aware shape processing. In *EUROGRAPHICS State-of-the-art Report* (2013). 3

[PR11] PARK D., RAMANAN D.: N-best maximal decoders for part models. In *IEEE ICCV* (2011), pp. 2627–2634. 6

[RBF12] REN X., BO L., FOX D.: RGB-(D) scene labeling: Features and algorithms. In *IEEE CVPR* (2012). 2

[RKB04] ROTHER C., KOLMOGOROV V., BLAKE A.: "grabcut": Interactive foreground extraction using iterated graph cuts. *ACM Trans. Graph. (Proc. SIGGRAPH) 23*, 3 (2004), 309–314. 1

[RT09] RUSSELL B. C., TORRALBA A.: Building a database of 3D scenes from user annotations. In *IEEE CVPR* (2009). 3

[RTMF08] RUSSELL B. C., TORRALBA A., MURPHY K. P., FREEMAN W. T.: LabelMe: A database and web-based tool for image annotation. *IJCV* (2008). 1, 2

[SF11] SILBERMAN N., FERGUS R.: Indoor scene segmentation using a structured light sensor. In *IEEE ICCV* (2011). 2

[SHKF12] SILBERMAN N., HOIEM D., KOHLI P., FERGUS R.: Indoor segmentation and support inference from RGBD images. In *ECCV* (2012). 3, 4, 5

[SMZ*14] SHAO T., MONSZPART A., ZHENG Y., KOO B., XU W., ZHOU K., MITRA N. J.: Imagining the unseen: Stability-based cuboid arrangements for scene understanding. *ACM Trans. Graph. (Proc. SIGGRAPH Asia) 33*, 6 (2014), 209:1–209:11. 3, 5, 8

[SSN09] SAXENA A., SUN M., NG A. Y.: Make3D: Learning 3D scene structure from a single still image. *IEEE Trans. Pattern Anal. Mach. Intell. 31*, 5 (2009), 824–840. 3

[VG11] VIJAYANARASIMHAN S., GRAUMAN K.: Large-scale live active learning: Training object detectors with crawled data and crowds. In *IEEE CVPR* (2011), pp. 1449–1456. 2

[XOT13] XIAO J., OWENS A., TORRALBA A.: SUN3D: A database of big spaces reconstructed using sfm and object labels. In *IEEE ICCV* (2013). 2

[YYT*11] YU L.-F., YEUNG S.-K., TANG C.-K., TERZOPOULOS D., CHAN T. F., OSHER S. J.: Make it home: Automatic optimization of furniture arrangement. *ACM Trans. Graph. (Proc. SIGGRAPH) 30*, 4 (2011), 86:1–86:12. 5

[ZSTX14] ZHANG Y., SONG S., TAN P., XIAO J.: Panocontext: A whole-room 3d context model for panoramic scene understanding. In *ECCV* (2014). 3

[ZZ13] ZHAO Y., ZHU S.-C.: Scene parsing by integrating function, geometry and appearance models. *IEEE CVPR* (2013). 3, 5