Support surface prediction in indoor scenes

Ruiqi Guo
Department of Computer Science
University of Illinois at Urbana-Champaign
guo29@illinois.edu

Derek Hoiem
Department of Computer Science
University of Illinois at Urbana-Champaign
dhoiem@uiuc.edu

Abstract

In this paper, we present an approach to predict the extent and height of supporting surfaces such as tables, chairs, and cabinet tops from a single RGBD image. We define support surfaces to be horizontal, planar surfaces that can physically support objects and humans. Given a RGBD image, our goal is to localize the height and full extent of such surfaces in 3D space. To achieve this, we created a labeling tool and annotated 1449 images with rich, complete 3D scene models in NYU dataset. We extract ground truth from the annotated dataset and develop an algorithm for predicting room layout and heights and extents of support surfaces. A key challenge is that many support surfaces are covered with items or hidden. Our approach infers occluded surfaces using contextual cues and layout priors.

1. Introduction

Knowledge of support surfaces is crucial to understand or interact with a scene. People walk on the floor, sit in chairs, eat on tables, and place objects on desks. Our goal is to infer the heights and extents of support surfaces in the scene from a single RGBD image. The main challenge is that support surfaces have complex multi-layer structures and that much of the scene is hidden from view (Fig. 1). Often, surfaces such as tables are littered with objects which limits the effectiveness of simple plane fitting strategies.

One barrier to our study was lack of a dataset that has complete 3D annotations. We undertook an extensive effort to create full 3D models that correspond to scenes in the NYU (v2) dataset [14], which we expect will be of interest to many other researchers. Our annotations complement the existing detailed 2D object and support relation annotations and can be used for experiments on inferring free space, support surfaces, and 3D object layout.

Another challenge is how to represent support surfaces, which are often scattered and multilayered. For example, a shelf may be stacked on a computer desk that rests on the floor, with a chair pushed under the desk. When initially viewing a scene, we do not know how many surfaces there are or at which heights. Our solution is to infer a set of overhead support maps that indicate the extent of supports at various heights on the floor plan.

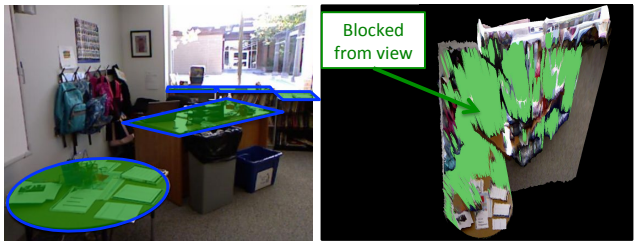


Figure 1: **Challenge.** Our goal is to predict the height and extent of all support surfaces, including occluded portions, from one RGBD image. As shown on the left, these surfaces are often covered with objects and are nearly invisible when they occur near eye-level. As shown on the right, we must perform inference over large portions of the scene that are blocked from view.

We need to specify both the heights and extents of support surfaces, which is difficult due to clutter and occlusion. Our approach is to label visible portions of the scene, project these labels into an overhead view, and refine estimates and infer occluded portions based on scene priors and context. Our approach is analogous to that of Guo and Hoiem [3] who first label an RGB image according to the visible surfaces at each pixel and then infer occluded background labels. See Figure 2 for an overview. We label visible pixels into "floor", "wall", "ceiling" and "object" using the RGBD region classifier from Silberman et al. [14] and then project these pixels into an overhead view using the depth signal. Before projection, the scene is rotated so that walls and floor are axis-aligned. We then predict which heights are likely to contain a support surface based on the normals of visible surfaces and straight lines in the image. One height could contain multiple surfaces such as table and counter tops or the seats of several chairs. For each height, we then predict the extent of support on the floor map using a variety of 2D and 3D features. These estimates are refined using an autocontext [16] approach and shape priors incorporated by matching whole surfaces from the training

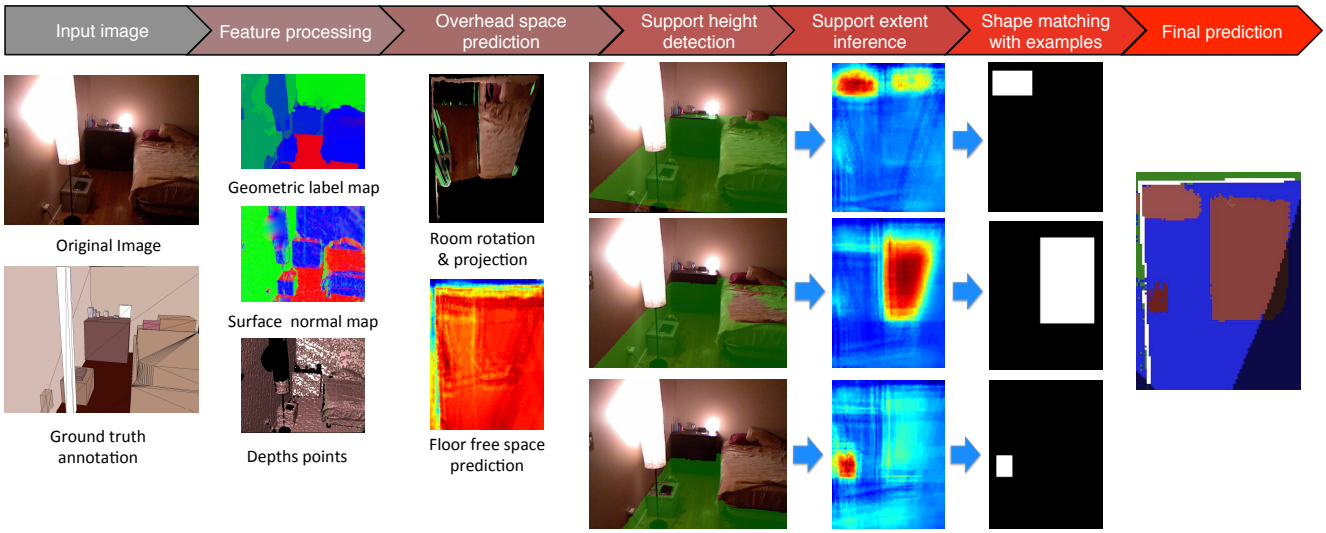


Figure 2: **Approach.** The input is an aligned RGBD image. We first compute features based on the depth image and estimated labels of image plane pixels into "floor", "ceiling", "wall" and "foreground". These features are projected into an overhead view and used to estimate the locations of walls (or floor free space). Next, a large number of horizontal planes are proposed and classified as "support" or "non-support". The horizontal extent of a supporting surface is then estimated for each support plane based on the features at each point and surrounding predictions. Template matching is performed with the footprints of training objects, which provides a more regularized estimate of support extent. In the rightmost image, green pixels are estimated walls, blue pixels are floor, and red pixels are support surfaces, with lighter colors corresponding to greater height. The darkly shaded portion corresponds to parts of the overhead map that are outside the camera's viewing angle. White lines correspond to wall estimates based on simple plane fitting.

set.

Our experiments investigate the accuracy of our estimates of support height and extent, the effects of occlusion due to foreground objects or surfaces above eye-level, and the effectiveness of our contextual inference and shape matching. Our method substantially outperforms a baseline of plane-fitting, but there is still much room for improvement by incorporating object recognition or more structured models of relations among surfaces.

1.1. Related work

Interpreting indoor scenes has recently become an active topic of research. One line of work is to predict boxy layouts from RGB images [6, 7, 10, 8, 5, 12, 11], in which occlusion and lack of 3D sensors is combatted with simple scene models and strong priors. By operating on RGB-D images that provide both color and depth signals, we aim to explore more detailed and complex scene representations. However, the depth signal does not trivialize the problem, since many support surfaces are obscured by clutter or completely hidden.

Our approach directly builds on recent efforts by Silberman et al. [14] and Guo and Hoiem [3]. Silberman et al. created the NYU v2 dataset and proposed algorithms to orient the scene, find major surfaces, segment the image into objects, label regions into "prop", "furniture", "structure", and "floor", and estimate which objects support which others. Our approach incorporates their algorithms for estimating 3D scene orientation and labeling visible pixels into geometric classes. We differ in that we predict the full 3D extent of support surfaces, and we extend their dataset with full 3D annotations of objects using a tool that enables users to markup scenes based on both image data and depths. Guo and Hoiem [3] propose an approach to label both visible and occluded portions of a 2D image. We adopt their basic strategy of labeling visible portions of the scene and inferring occluded portions based on autocontext and shape priors. However, our approach is applied to full 3D scenes, which requires new representations, features, and matching strategies.

Other works reason about support relations between objects and other surfaces. Gupta et al. infer support in outdoor scenes from RGB images to aid in geometric labeling [4], and Silberman et al. [14] infer support relations and types (e.g., supported from the side or from below) for pairs of regions. Our work differs in its aim to infer full 3D extent of support surfaces. Jiang et al. [9] propose an algorithm that trains a robot to place objects based on 3D point clouds. Our work on estimating underlying support surfaces would facilitate object placement and allow more complex interactions. Taylor and Cowley [15] estimate the layout of walls from an RGBD image based on plane fitting and inference in an overhead view but do not address the problem of support surfaces. Finally, our work has some relation to efforts

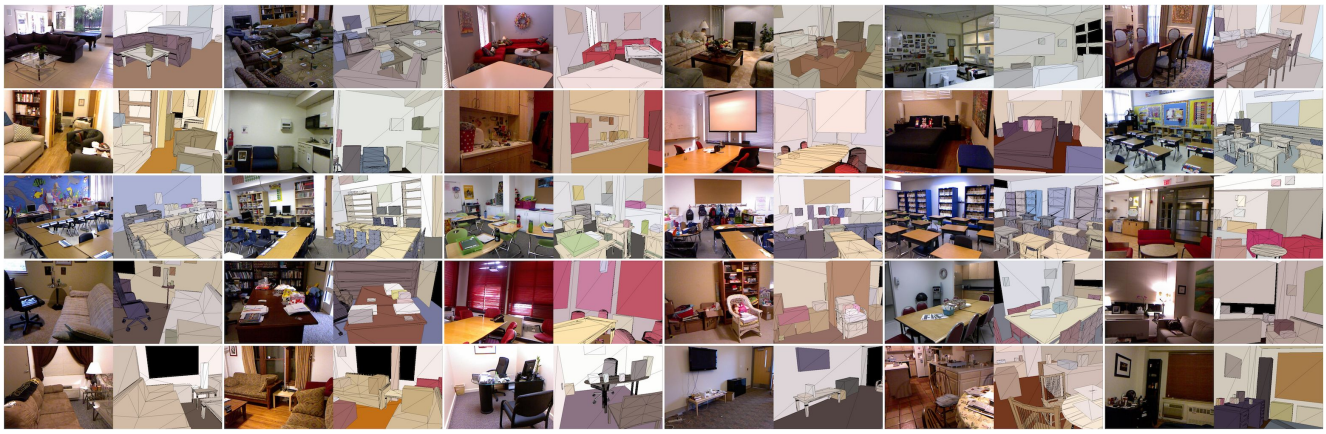


Figure 3: **3D annotation examples.** Our annotations can be used to study estimation of 3D layout of support surfaces (as in this paper), objects, or occupied space.

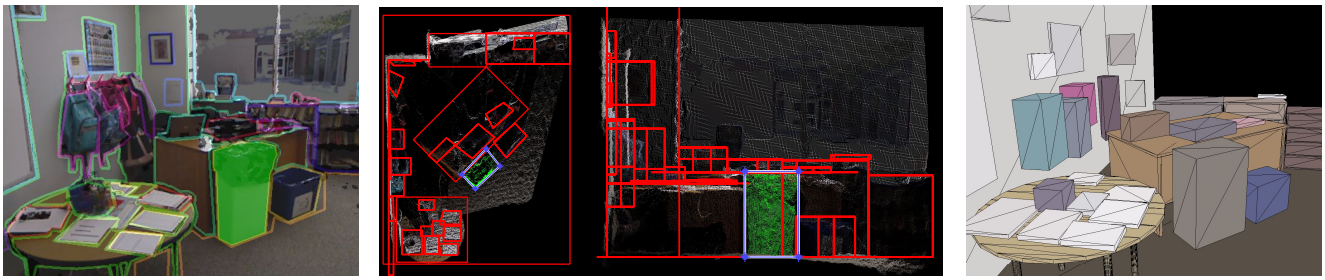


Figure 4: **Annotation tool.** Users annotate the 3D scene based on views of the RGB image (left), an overhead view (left-center), and a horizontal view (right-center), and with the help of 2D object segmentations and automatic initial guesses of the object footprint. The 3D model is shown on the right.

in 3D scene modeling (e.g., [2, 17, 13]). We differ in that we are provided with only one viewpoint, and our goal is to recover extent of underlying support surfaces rather than a full model of visible objects and structures.

1.2. Contributions

This paper offers two main contributions: (1) detailed 3D annotations for the 1449 images of the NYU Kinect dataset that can be used to study 3D reconstruction, free space estimation, and support inference; and (2) a model and approach to recover the extent of support surfaces, which is a fundamental but largely unexplored problem.

2. Creating 3D Annotations for Indoor Scenes

We extend the NYU Depth Dataset, which contains 1449 images of roughly 500 distinct scenes, with complete, detailed 3D models (Fig. 3). Our annotation tool (Figure 4) enables users to model scenes based on both image data and point clouds from the depth sensor while viewing the scene from multiple angles. Our 3D annotations, combined with the existing detailed 2D object annotations, will be useful for exploring a wide variety of scene understanding tasks.

2.1. 3D Scene Model

We label three types of scene elements:

- **Layout structures** include floors, walls and ceilings. Because walls are always perpendicular to the floor, they are modeled as line segments in the overhead view and polygons in the horizontal view. Similarly, ceiling and floors are line segments in the horizontal view and polygons in the overhead view. We also model openings such as doorways and windows on the walls as polygons on the wall planes.

- **Furniture** objects are common in indoor scenes and tend to have complicated 3D appearance. For example, chairs and sofas cannot be modeled using cubic blocks, and their support surfaces are often not the top part. To accurately model them, we use Google SketchUp models from Google 3D Warehouse repository. We manually select 30 models to represent 6 categories of furniture that are most common: chair, table, desk, bed, bookshelf and sofa. The support surface of each object is labeled by hand on the 3D model.

- **3D extruded models** are used to describe all other ob-

jects. Most clutter objects are small and do not have regular shape, and we model them as polygons extruded from their support surface.

2.2. Preprocessing

The RGBD scenes are first preprocessed to facilitate annotation as well as support surface inference. Because the preprocessing is automatic, we use the same procedure at the start of our inference procedure. We start with the cropped RGBD scene, which corresponds to the area where information from both sensors are available. Then the surface normals are computed at each pixel by fitting local planar surfaces in its neighborhood. These local planar surfaces take into account color information as well, in a manner similar to bilateral filtering, which improves robustness of plane fits compared to using only depth information.

Next, we compute the dominant room orientation by iteratively aligning the surface normals to the x, y and z axes. Initially, surface normals of each point is assigned to the nearest axis. Then we fix the assignment and compute optimal rotation matrix R of the room using SVD, based on all points that are aligned within a given threshold. The point cloud is then rotated using R and we repeat from the alignment step again until the rotation matrix does not change.

2.3. Annotation procedure

If the floor is visible, our system estimates the floor height from the depth information and object labels (from [14]). If necessary, the annotator can correct or specify the floor height by clicking on a scene point and indicating its height above the floor.

The annotator then alternates between labeling in an overhead view (from above the scene looking down at the floor) and horizontal view (from the camera looking at the most frontal plane) to model the scene:

1. The annotator is asked to click to select one region from the 2D annotated image plane by clicking.

2. In the overhead view, the annotator is shown highlighted 3D points and an estimated bounding box that correspond to the object. The annotator can fit a polygon to the footprint of the object.

3. The horizontal view is then shown, and the annotator specifies the vertical height of the object by drawing a line segment at the object's height.

4. Additionally, the annotator can supply more details of the object, such as adding openings to the wall or placing a detailed SketchUp model for furniture. The user can choose the SketchUp model and orientation with a single keystroke.

A major advantage of our annotation tool is that the annotation process can guided by depth values in RGBD image, improving ease and accuracy of layout recovery and object labels. Furthermore, the tool uses existing 2D object label in the image plane (available in the NYU dataset) so that the 3D models are consistent with the 2D annotation. The tool also has a number of handy features that help users annotate quickly including (1) **a snapping feature** that snaps the polygon vertices to neighboring support surfaces or corners, and/or aligns annotations with the room axes; 2) an initial guess of the extent by fitting **default bounding boxes** to depth points included in the region. Therefore the users can often trust the default model configuration and have to edit only when the system's estimate is poor.

Our annotation tool is implemented in Matlab. On average, it takes about 5 minutes to model a scene with more than 10 objects, depending on the complexity of the scene. We recruited three student annotators with little or no previous Matlab experience and obtained thousands of high quality annotations shown in Figure 3.

3. Support surface prediction

In this section, we present an approach (Fig. 2) to predict the vertical height and horizontal extent of support surfaces in a scene. Scene parsing in an overhead view is very different than in the image plane. One challenge is that many locations are blocked from view and do not have observed evidence. On the other hand, since the room directions have been rectified, objects tend to have rectangular shapes in an overhead view. We design features that apply to the overhead view and use spatial context to improve parsing results.

3.1. Preprocessing

We first label pixels in the image plane using Silberman et al.'s algorithm [14] into four geometric categories: "floor", "ceiling", "wall" and "foreground". We then project the labeled pixels into the overhead view using the associated depth signal.

We also construct a voxel map representation of the scene. We first voxelize the scene into a grid and then project the voxels onto the camera plane. The projected depths are then compared to the depth values of the observed scene. All voxels with smaller depth values are observed free space, and the voxels with larger depth values than the scene are not direct observed. The rest of the voxels are outside of the view scope and treated as "don't care".

3.2. Features

We use the following set of features, illustrated in Fig. 5. To detect support heights, we aggregate the features at each candidate height and classify on the aggregated features. After support heights are detected, we extract these features at each height as feature maps for overhead parsing.

1. **Observed upward planar surfaces** indicate if the surface at this voxel has a normal pointing upwards (along

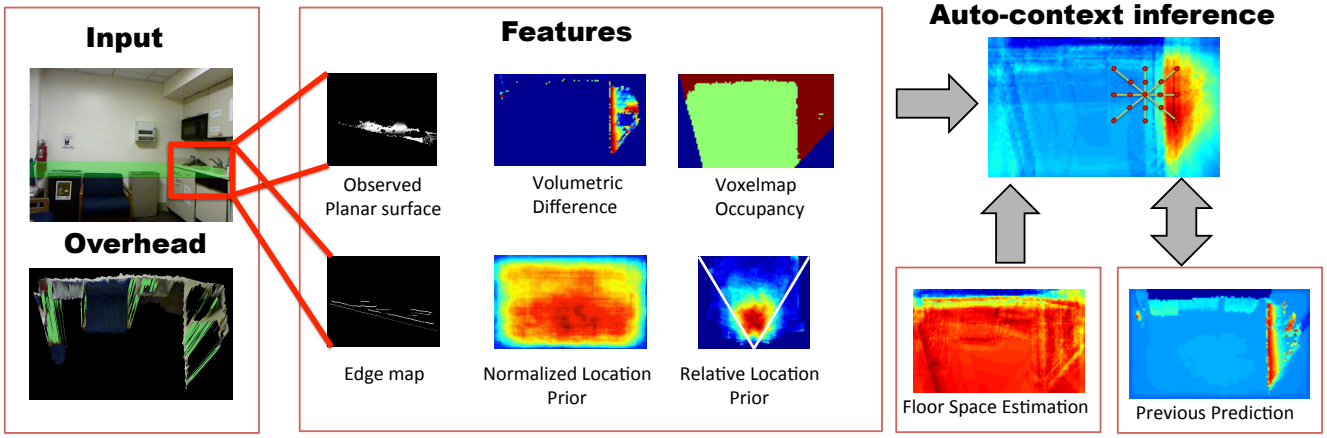


Figure 5: **Overhead scene parsing.** The feature set we used in our support surface prediction: observed up-pointing points, 3D geometric labels and edgemaps are computed in the image plane and then projected. Volumetric difference, occupancy and location/view prior are directly computed on the overhead grid. Auto context inference is applied to each predict support height.

the Y axis).

2. **Observed geometric labels** The mean and max likelihood in each cell, after projecting to an overhead view.

3. **Edgemap** We first detect straight line segments in the RGB image.Then, the edgemap is a binary map of whether the cell contains a edge segment. This follows the intuition that horizontal straight lines often occur at the edges of support surfaces.

4. **Voxel occupancy** is the amount of observed free space voxels near the near the surface height. If a voxel is observed to be free space, it cannot be a part of a support surface.

5. **Volumetric difference** is the difference of number of free space voxels above the height at this location subtracted by the number of the number of free space voxels below it. Support surfaces often have more air above them than below.

6. **Support height prior** is the spatial distribution of the vertical height of support planes.

7. **Normalized location prior** is the spatial prior on the overhead view of where support surfaces are in the training scenes, normalized by the scale of the scene.

8. **Relative location prior** is the unnormalized spatial prior, rotated to the viewpoint of the camera.

3.3. Predicting support height

We first predict the support heights. Our intuition is that at the height of a support plane, we are likely to see: (1) observed surfaces with upward normals; (2) a difference in the voxel occupancy above and below the plane; and (3) observed 3D points near the height. Also, we use an estimated prior, since some heights are more likely than others. We extract the aggregated features of {1,2,3,4,5,6} from the list of the previous subsection for each height of the proposed plane. Then we estimate an empirical log-likelihood for each feature for both "support" or "non-support" heights. We transform the features into log-likelihood in this way and use a linear SVM to classify each height of the plane into "support" or "non-support" categories. After classification, we perform non-maximum suppression to remove planes within a height tolerance of ϵ. ϵ is set to 0.15m because the average depth is approximately 3m and the measurement error of the Kinect is empirically measured as $depth * 0.05$.

3.4. Estimating floor and support surface extent

We estimate the extent of the support surface(s) at each detected height. First, we divide the overhead view into grid cells. For cells on the slice of the space at that support height, we compute features {1,2,3,4,5,7,8} of subsection 3.2. We use a linear classifier to predict whether each cell is part of the floor or outside of the room. Likewise, for each support plane height, we classify each grid cell in the overhead view as being a support surface at that height or not.

Support surfaces have spatial context. Therefore, in addition to prediction from directly observed features, we also use surrounding context from the same height and support surface predictions of the heights below. Context is especially important to predict surfaces that are occluded or near eye-level. Similar to Guo and Hoiem [3], we use an auto-context procedure to infer occluded surfaces.

The auto-context process iteratively aggregates the fea-

tures from the neighboring regions in a large template and applies a discriminative classifier (linear SVM in our case) to predict labels for each grid cell. Then, the algorithm repeats by adding the output of previous predictions into the feature set to re-predict until the prediction does not change. We first compute features at each overhead cell on the overhead view as in the previous subsection and apply auto-context. Figure 5 visualizes this iterative procedure.

3.5. Retrieving full support surfaces using template matching

Although spatial context helps to fill in the occluded part in the overhead view, it does not fully exploit the shape prior such as objects often being rectangular. To do this, we first aggregate the templates from the training set, and then we match them with the probability map at each support height using convolution. We assign a positive score to locations that are support surfaces on both the prediction and the template; we assign a heuristic score to each of the template according to:

$$Score(T,P) = \sum_{i \in T} c_1(p_i - t)I(p_i > t) - c_2(t - p_i)I(p_i < t)$$

where P is the probability map, T is the template and t is the threshold that corresponds to 0.5 recall. c_1 is set to 10 and c_2 is set to 1. This equation encourages the template to overlap with high probability area in the probability map and penalizes the overlap with the freespace. We do not assign any score to locations that are out of the field of view.

The template matching uses convolution to compute matching scores and can be done efficiently using the Fast Fourier Transform. For scenes with complicated support surface layout, it is often the case that there is not a single perfect match, but there exist multiple good partial matches. We iteratively match a portion of the scene to a training template, set the probability of the matched portion to zero, and find another matched template, until no more matches can be found with score above a fixed threshold.

Compared to holistic scene matching strategies, such as Satkin et al. [11], our matching scheme is more flexible because it allows translation and partial matches. Like Satkin et al. [11], our matching is independent of viewpoint because scenes are first rectified by orientation.

4. Experiments

To verify the effectiveness of our proposed method, we evaluate our support surface prediction (both heights and extent) by comparing to the ground truth support surfaces extracted from our annotated dataset.

4.1. Generating ground truth

We defined categories of objects that are capable of support, such as table, sofa, shelves, cabinet etc. The top or supporting surface of objects from these categories are considered as support surfaces unless they are within d of the ceiling or another support surface. Some surfaces maybe too close to each other, such as the case when a cabinet is directly underneath a countertop. In these cases, we will set the one below to be "don't care". Again, ϵ is empirical error of Kinect over the dataset, which is set to 0.15m.

For prediction, we project the depth points to the overhead view and quantize the projected area into a grid with a spacing of 0.03m, we perform prediction and evaluation based on this grid. The ground truth floor space is the union of the annotated area with the observed area, subtracted by the area that has been occluded by walls from the viewpoint. The areas that are out of the scope are marked as "don't care".

To make evaluation less sensitive to noise in localization, we make the area around boundary of support surface within a thickness of ϵ to be "don't care". As in floor space, we also do not evaluate the area that is out of the field of view. In all, there are 5495 support surfaces in 1449 RGBD images, so on average 3.79 support surfaces per scene. In those support surfaces, 5095 are below the camera while 400 are above it.

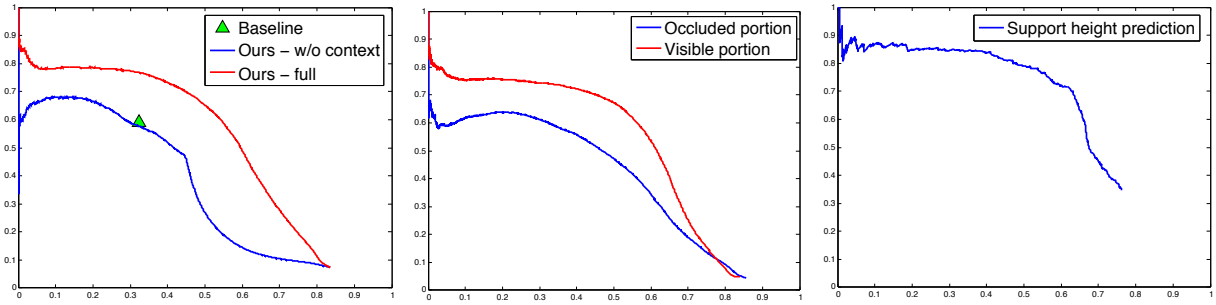
4.2. Experiment details

We use the training/testing split from the Silberman et al. paper [14] with 795 training images and 654 testing images. For training support height classifier, we propose a plane at each vertical height with a spacing of 0.01m. The negative examples are the ones are far away from any ground truth heights by a threshold of ϵ. The positive examples are the annotated heights. We use a linear SVM to train and predict on support heights. At testing time, we also propose and classify all vertical heights with 0.01m spacing and then apply non-maximum suppression.

In auto-context, we used a template of sparse points with a distance of 0m, 0.03m, 0.12m, 0.27m, 0.48m, and 0.75m away from the center pixel distributed in a star shape. This large template helps model the long-range contextual interactions. We repeat the auto-context for three iterations before it terminates. To do template matching, we first aggregate the support surface configurations from training scenes, and obtain a total of 3057 templates. For the extent probability map we predict at each support height, we retrieve the top 10 templates with the highest matching scores. The iterative matching process terminates when there is no matching above 20, in the unit of grid cells, which is equivalent to $0.18m^2$ in the physical world.

4.3. Quantitative results

We evaluate accuracy of support extent prediction with precision-recall curves. The ground truth consists of a set of known support pixels at some floor position and vertical height and a set of "don't care" pixels at the edges of annotated surfaces or out of the field of view. Our predic-



(a) (b) (c)

Figure 6: **Quantitative Evaluation** on support surface prediction. (a) PR curve of support extent prediction with and without auto-context, and the baseline accuracy. (b) Prediction on visible and occluded portion of the surface extent. (c) PR curve of support height prediction, we are recalling a maximum of 76% of the support plane heights.

tions consist of a set of confidence-valued support pixels. Computation of precision-recall is similar to that for object detection, as in the PASCAL VOC challenge [1]. The most confident predicted pixel at the same overhead position and within ϵ height of ground truth support pixel is labeled positive (or "don't care" if that is the ground truth label). All other pixels are labeled as negative so that duplicate detections are penalized. Because there are many points per image in the dataset, we sample 1000 pixels from each image, so that each image has the same weight. Precision-recall curves are computed based on samples accumulated over the 654 test images.

Figure 6 shows the results of our quantitative analysis. For support heights, we plot the the precision recall curve (Figure 6(a)) for detecting individual support plane heights and was able to recall 76% of the support planes. The average precision on planes below the camera is 0.621, while planes above it is only 0.081, which confirms the intuition that detecting planes above the eye-sight is very hard. For support extent prediction we compare to a baseline of plane-fitting, based on the Silberman et al. [14] code for plane segmentation. Specifically, their method computes 3D normals and uses a RANSAC procedure to sample points, propose 3D planes. Similar planes are merged, and sufficiently large remaining planes are kept. Finally, a graph cut procedure assigns pixels to planes using both 3D data and color models. We directly use the pixel labeled plane mask from the released code and further post-process the result by selecting planes that face up and are likely to be "foreground" from semantic labeling result. The baseline does not have confidences, so it is a single point on the curve. In Figure 6(a), we see that our method outperforms the baseline by 17% precision at the same recall level or 13% recall at the same precision. The gains due to use of autocontext and shape-fitting is large, with most of the gain due to autocontext. The improvement due to shape-fitting affects qualitative results more than quantitative. In Figure 6(b), we compare performance for occluded support surfaces to unoccluded (visible) ones. In qualitative results, we show predictions that have confidence greater than the value corresponding to the 0.5 recall threshold.

4.4. Qualitative results

In Figure 7, we display the visualization of support surface prediction in the overhead view. Although the scenes are cluttered and challenging, we are able to predict most of the support surface extents, even if they are severely occluded (kitchen counter in first image on the top left) or partly out of view (the chair in first top right image). Furthermore, the transferred support planes can give us a rough estimation of individual support objects.

However, there are cases where the support surfaces are hard to find because they are not obvious or not directly observed. For example, in the cabinet-tops in the bottom-left image are above the camera height. The seat of the chair in the bottom right image is out of the field of view and is hard to detect.

5. Conclusions

We propose 3D annotations for the NYU dataset and an algorithm to find support planes and determine their extent in an overhead view. Our quantitative and qualitative results show that our prediction is accurate for nearby and visible support surfaces, but surfaces that are distant or near or above eye-level still present a major challenge. Because many surfaces are occluded, contextual reasoning is necessary to achieve good performance. Methods to improve detection of support planes above eye-level (not directly visible) and to recognize objects and use categorical information are two fruitful directions for future work. In addition, our dataset enables researchers to study problems of occupancy (or free space), estimation of navigable paths, 3D reconstruction, and other spatial understanding tasks.

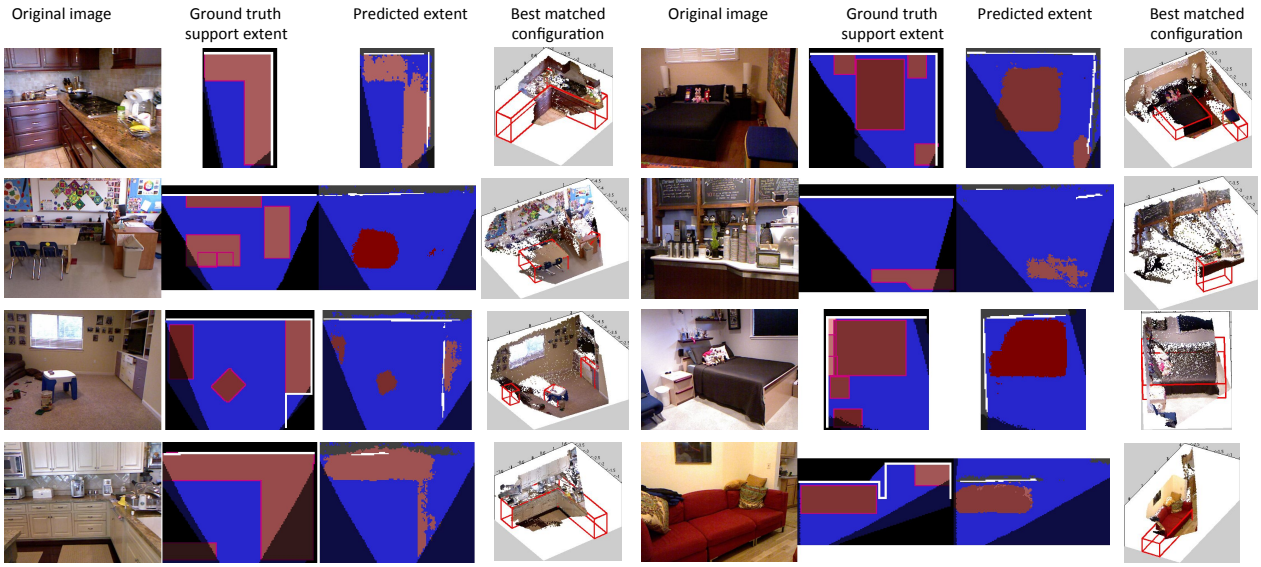


Figure 7: **Overhead visualization.** Green and blue and red areas are estimated walls, floor and support surfaces respectively. The brighter colors of support surfaces indicate higher vertical heights relative to the floor. Dark areas are out of the field of view. The first and second column shows in the original scene and its corresponding ground truth support surfaces. The third column shows our final prediction, by thresholding the probability map at the threshold which correspond to 0.5 recall. The fourth column shows the most confidently matched surface configuration.

6. Acknowledgements

This research is supported in part by ONR MURI grant N000141010934, NSF award IIS 0904209. We also thank Scott Satkin, Vincent Delaitre and Maheen Rashid who gave help and valuable suggestions to our scene annotation tools, as well as our annotators: Gaston Gerchkovich, Shuxin Yu, Sujay Bhobe and Xiang Li.

References

[1] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2010 Results. http://www.pascal-network.org/challenges/VOC/voc2010/workshop/index.html. 7

[2] Y. Furukawa, B. Curless, S. M. Seitz, and R. Szeliski. Reconstructing building interiors from images. In *ICCV*, pages 80–87, 2009. 3

[3] R. Guo and D. Hoiem. Beyond the line of sight: Labeling the underlying surfaces. In *ECCV*, pages 761–774, 2012. 1, 2, 5

[4] A. Gupta, A. A. Efros, and M. Hebert. Blocks world revisited: Image understanding using qualitative geometry and mechanics. In *ECCV*, 2010. 2

[5] A. Gupta, S. Satkin, A. A. Efros, and M. Hebert. From 3d scene geometry to human workspace. In *CVPR*, 2011. 2

[6] V. Hedau, D. Hoiem, and D. Forsyth. Recovering the spatial layout of cluttered rooms. In *ICCV*, 2009. 2

[7] V. Hedau, D. Hoiem, and D. Forsyth. Thinking inside the box: Using appearance models and context based on room geometry. In *ECCV*, 2010. 2

[8] V. Hedau, D. Hoiem, and D. A. Forsyth. Recovering free space of indoor scenes from a single image. In *CVPR*, pages 2807–2814, 2012. 2

[9] Y. Jiang, M. Lim, C. Zheng, and A. Saxena. Learning to place new objects in a scene. *I. J. Robotic Res.*, 31(9):1021–1043, 2012. 2

[10] D. C. Lee, M. Hebert, and T. Kanade. Geometric reasoning for single image structure recovery. In *CVPR*, 2009. 2

[11] S. Satkin, J. Lin, and M. Hebert. Data-driven scene understanding from 3D models. In *BMVC*, 2012. 2, 6

[12] A. G. Schwing, T. Hazan, M. Pollefeys, and R. Urtasun. Efficient structured prediction for 3d indoor scene understanding. In *CVPR*, pages 2815–2822, 2012. 2

[13] T. Shao, W. Xu, K. Zhou, J. Wang, D. Li, and B. Guo. An interactive approach to semantic modeling of indoor scenes with an rgbd camera. *ACM Trans. Graph.*, 31(6):136, 2012. 3

[14] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, pages 746–760, 2012. 1, 2, 4, 6, 7

[15] C. J. Taylor and A. Cowley. Parsing indoor scenes using rgb-d imagery. In *Robotics: Science and Systems*, 2012. 2

[16] Z. Tu and X. Bai. Auto-context and its application to high-level vision tasks and 3d brain image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 32(10):1744–1757, 2010. 1

[17] J. Xiao and Y. Furukawa. Reconstructing the world's museums. In *ECCV (1)*, pages 668–681, 2012. 3